République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Batna 2 – Mostefa Ben Boulaïd
Faculté de Technologie
Département de Génie industriel

Thèse

Préparée au sein du Laboratoire d'Automatique et Productique (LAP)

Présentée pour l'obtention du diplôme de :
Doctorat en Sciences en Génie industriel
Option : Génie industriel

Sous le Thème :

Contribution à la résolution des problèmes d'ordonnancement en temps réel dans un système de production de type Job Shop

Présentée par :

HADRI Abdelkader

Devant le jury composé de :

Mr. MOUSS Mohamed Djamel	Prof	Université de Batna 2	Président
Mr. SMADI Hacene	Prof	Université de Batna 2	Rapporteur
Mr. MELIANI Sidi Mohammed	Prof	Université de Tlemcen	Examinateur
Melle. GHOMRI Latifa	Prof	Université de Tlemcen	Examinateur
Mr. SENOUSSI Ahmed	MCA	Université de Batna 2	Examinateur
Mr. MAHDAOUI Rafik	MCA	Université de Khenchela	Examinateur

2022

Remerciements

Je remercie avant tout ALLAH le tout puissant, qui m'a donné la force et la patience pour l'accomplissement de ce travail.

Je tiens ensuite à exprimer ma respectueuse gratitude à Monsieur **Bougloula Aimed Edden** pour son orientation, supervision, sa patience, sa rigueur scientifique et ses remarques judicieuses. Le travail sous sa supervision a été une expérience de confiance.

J'adresse aussi mes sincères remerciements au professeur **Smadi Hassan,** Directeur de cette thèse pour l'intérêt qu'il a toujours porté à mes travaux, pour son soutien et ses conseils durant toutes les années que j'ai passé en génie Industriel.

Je remercie également les membres du jury pour l'intérêt qu'ils ont porté à ce mémoire en acceptant d'en être examinateurs.

Je tiens à remercier particulièrement Monsieur **Belkaid Fayçal** pour son aide qu'il a apportée à la thèse, sa disponibilité et ses conseils pertinents.

Merci à mes proches, mes collègues et tous mes amis de Tlemcen et de Batna

Enfin, je remercie vivement **mes parents et ma femme** de leur encouragement, patience et soutien tout au long de mes études. Les mots ne seront jamais assez explicites pour exprimer mes remerciements et ma gratitude. C'est à vous qu'est dédiée cette thèse.

Table des matières

Table des matières

Liste des figures

Liste des Tableaux

Liste des notations

M_k :	La $k^{ième}$ machine
J_j :	Le $j^{ièm}$ job
P_{jk} :	Processing time de job J_j sur la machine M_k
O_j^k :	Opération du job J_j qui sera effectuée sur la machine M_k
nb_j :	Nombres d'opérations de job J_j
np_m :	Nombre de positions de la machine M_m
$Qdem_j^c$:	Quantité de la ressource c demandée par le job J_j
$Qarr_t^c$:	Quantité de la ressource c arrivée dans le temps t
$Tarr_t^c$:	Le temps d'arrivé de la ressource c
T_{jk} :	Le début de traitement de job J_j sur la machine M_k
F_k^p :	La fin de la position p de la machine M_k
Qst_k^{pc}:	La quantité totale demandée de la ressource c jusqu'à la fin de la position p de la machine M_k
C_{max} :	Le makespan
T_j^k :	Temps de déplacement du job J_j vers la machine M_k
$C_j^{k\prime}$:	La fin de traitement du job J_j sur la machine $M_{k\prime}$
D_k: :	La disponibilité de la machine M_k
D_j^k :	La disponibilité du job J_j qui sera exécuté sur la machine M_k
DRC_j^k :	La disponibilité des ressources consommables du job J_j dans la machine M_k
M_g :	La dernière machine de la gamme opératoire du job J_j
STT_j :	La somme des temps de traitement du job J_j
STD_j :	La somme de temps de déplacement du job J_j
Rtd_j:	Rapport du temps de déplacement sur le temps de traitement du job J_j
$RtdRC_j$:	Le produit de la quantité demandée des ressources et Rtd_j
$Rtdcum_j^k$:	Rapport des temps cumulés de Rtd_j

Liste des notations

$Début_P$:	Début de la panne
fin_P :	Fin de la panne
μ :	Taux de réparation du convoyer
δ :	Facteur qui peut prendre des valeurs aléatoires $\in [0,1]$
$C_{max}(Init)$:	Le makespan initiale trouvé dans la phase hors-ligne
$C_{max}(Réo)$:	Le makespan trouvé après le réordonnancement

Introduction générale

L'accroissement et le changement des exigences du marché en terme réduction des délais de fabrication et de diminution des prix ont met une grande pression sur les entreprises industrielles. C'est pourquoi, une entreprise doit constamment chercher à posséder d'un système de production très efficace qui lui garantit sa productivité et sa rentabilité. En effet les systèmes de production flexibles sont des systèmes capables de s'adapter à une possible évolution de l'environnement et de présenter une diversité de produits avec différentes séquences. Ils peuvent changer rapidement de produits et de séquences de produits sans perdre leur productivité.

Une bonne gestion d'un système de production flexible représente une nécessité préoccupante pour les responsables des entreprises. Cette qualité en gestion a une forte relation avec les méthodes et les techniques d'organisation et d'exploitation que ces entreprises utilisent pour gérer leurs ressources. L'ordonnancement de la production présente un outil très important dans la gestion de production, il permet de définir le planning de l'utilisation des ressources (consommables et/ou renouvelables) en déterminant à quel instant on doit traiter quelle tâche avec quelle ressource tout en optimisant un ou plusieurs critères.

En revanche, pour les systèmes de production flexibles FMS qui peuvent changer rapidement de produits et de séquences de produits sans perdre leur productivité, l'ordonnancement prévisionnel limite les capacités de ces systèmes et ne permet pas de répondre aux attentes de leurs responsables (SOUIER, 2012). Alors proposer un système de gestion souple, capable de les piloter en temps réel, permettant de prendre dynamiquement et à très court terme les décisions d'allocation et d'ordonnancement des opérations en fonction de l'état réel du système de production devint l'une des conditions nécessaires pour tirer pleinement partie de la flexibilité offerte par ces systèmes.

Plusieurs types de perturbations (qui peuvent être internes comme pannes de machines, changement de temps d'exécution des jobs et/ou externe comme l'arrivée de nouveaux jobs, manque de matières premières…) empêchent le déroulement normal des travaux dans l'environnement réel des systèmes de production. Les planificateurs de production, dans ce cas, doivent non seulement générer des ordonnancements de haute qualité, mais également réagir rapidement afin de réviser leurs plannings de manière rentable et rapide. Dans ce cadre notre thèse s'intéresse à l'étude du problème d'ordonnancement temps réel d'un atelier job shop avec la présence de la panne du système de transport.

Nous abordons dans cette thèse en premier temps, le problème job shop dans leur contexte statique et avec deux contraintes additionnelles. Ensuite, nous nous intéressons au problème de ré-

ordonnancement en temps réel qui consiste à établir un nouvel ordonnancement des jobs dans le cas d'apparition de la panne de système de transport. Une méthode d'ordonnancement/ré-ordonnancement est proposé afin de trouver le meilleur ordonnancement qui minimise le coût générer par cet évènement perturbant. De ce fait notre thèse s'articule autour de cinq chapitre :

Le premier chapitre est consacré à la présentation générale des systèmes de production et plus particulièrement des systèmes flexibles de production en mettant l'accent sur ses caractéristiques, ses différents types, ainsi que les configurations qui existent de ce type de système. Tandis que dans le deuxième chapitre, nous présenterons tous d'abord, un aperçue général sur les problèmes d'ordonnancement, ces caractéristiques et ces différents types. Ensuite nous donnerons une description du problème d'ordonnancement job shop qui fait la base de notre étude. Enfin nous exposerons quelques méthodes de résolution qui peuvent être utilisées pour résoudre ces problèmes.

Dans le troisième chapitre nous dévoilerons notre première contribution qui concerne le problème d'ordonnancement job shop avec contrainte de ressources non renouvelables. Nous présentons d'abord le système étudier qui est le système flexible iCIM 3000. Puis nous présentons le modèle mathématique développé, l'approche algorithmes génétiques et les heuristiques utilisées pour résoudre ce problème dont l'objectif est de minimiser le temps d'exécution maximal (Makespan). Pour évaluer la performance des méthodes utilisée, plusieurs expériences avec des tailles différentes ont été présenté.

Dans le quatrième chapitre nous allons présenter le problème job shop avec deux contraintes ; contrainte de ressources consommables et contrainte de de transport. L'étude dans cette partie est également focalisée sur le système iCIM 3000. L'objectif visé dans ce chapitre, consiste à trouver les meilleurs séquencements des pièces (jobs) sur les quatre machines qui minimise le temps total d'achèvement des toutes les opérations (Makespen) en tenant compte en même temps le déplacement des pièces et la disponibilité des ressources consommables

Le cinquième chapitre est dédié entièrement à la deuxième contribution qui concerne problème de réordonnancement en temps réel (en linge). Ce réordonnancement consiste à établir un nouvel ordonnancement des jobs dans le cas d'apparition de panne de système de transport. Une méthode d'ordonnancement/ré-ordonnancement basée sur les règles de priorité est présenté qui consiste à trouver le meilleur ordonnancement qui minimise le coût générer par cet évènement perturbant.

Enfin, une conclusion des résultats obtenus et des perspectives de travail sont présentées à la fin ce manuscrit.

Chapitre I : Les Systèmes de Production Flexible

I.1 Introduction

La souplesse du système flexible de fabrication apporte des grands avantages dans la fabrication réelle des produits, car le processus est conçu pour que plusieurs produits soient utilisés sur différentes machines dans une seule installation de fabrication, ce qui permet une plus grande croissance et stabilité avec plus de diversité dans la production (Heragu & Kusiak, 1988). Cette technologie relativement nouvelle a été introduite pour améliorer l'efficacité d'un atelier de job shop tout en conservant sa flexibilité (Blazewicz, et al., 2019).

Dans ce premier chapitre nous nous intéressons à la présentation générale des systèmes de production et plus particulièrement des systèmes flexibles de production en mettant l'accent sur ses caractéristiques, ses différents types, ainsi que les configurations qui existent de ce type de système.

I.2 Les systèmes de production

I.2.1 Définition

Un système de production (ou de fabrication) est l'ensemble des équipements, des personnes, des informations, des processus et des procédures organisés permettant d'assurer la fonction de fabrication d'une entreprise (Groover, 2016) (Badiru, Ibidapo-Obe, & Ayeni, 2018). Les opérations assurées par ce type de système consistent, en exploitant les ressources disponibles, à transformer les matières premières (entrées) en produits finis (sorties) en apportant des modifications sur la forme, la structure, le volume et/ou l'apparence de ces matières (Merchichi, 2016).

Que ce soit la nature ou le type de la fonction assuré par les systèmes de production, ces derniers se divisent généralement en deux parties principales (Groover, 2016) :

- Les installations physiques : Les installations physiques comprennent l'ensemble des équipements (machines) concernées par la fabrication, les moyens de manutention, les endroits et systèmes de stockage et l'usine dans lesquelles sont

situées. Cette partie du système consiste à convertir les matières premières ou composantes en produits finis.

- Le système de pilotages : Il s'agit des procédures utilisées par l'entreprise pour gérer la production et résoudre les problèmes techniques et logistiques. L'objectif de ces procédures est de commander les matériaux, faire avancer le travail dans l'usine et s'assurer que les produits répondent aux normes de qualité.

I.2.2 Les typologies des systèmes de production

La totalité des systèmes de production convergent vers la même fonction, c'est de combiner des facteurs de production (capital, travail) et les transformer selon un processus organisé pour créer des biens ou des services. Cette convergence n'empêche pas l'existence de plusieurs variétés des systèmes de production, de ce fait, de nombreuses classifications des systèmes de productions sont alors présentées dans la littérature pour distinguer ces systèmes. Nous présentons ici l'une de ces classifications qui permet de définir le contexte de notre travail c'est la classification basée la nature et le volume des flux physiques.

Dans cette classification, les systèmes de production sont classifiés à base de la quantité des produits à fabriquer et la nature du flux des matières dans le système, en trois grandes classes qui peuvent être aussi divisées en d'autres sous-classes comme suit (Artigues, 1997) (Courtois, Martin-Bonnefous, & Pillet, 2003):

I.2.2.1 Les systèmes travaillant en continu ou en série

La caractéristique principale de ces systèmes est la circulation des matières en flux continu. Les ressources sont organisées sous forme d'une chaine où les matières premières doivent passer sur toutes ces ressources. Ce type de systèmes concerne surtout les industries dites de « *process* » dont la production nécessite la manipulation d'une grande quantité de produit ou d'une famille de produits (ex : la production des matières liquides ou gazeuses). Avec cette catégorie d'installation on dit que l'on est en présence d'un atelier à flux (**Flow shop**).

I.2.2.2 Les systèmes à flux discret

Ce modèle de systèmes représente le caractère essentiel des entreprises manufacturières. Les produits de quantités petites ou moyennes, peuvent être distingués individuellement et les stocks temporaires entre les postes de travail sont souvent utilisés. Le flux (ou le chemin) des produits est conditionné par l'enchaînement des tâches à réaliser afin de concrétiser ces produits. Cette configuration de systèmes fait la référence à l'atelier à tâches (**Job Shop**).

I.2.2.3 Les systèmes à flux hybride ou discontinu : Ces systèmes se situent entre les deux types de systèmes précédents. Selon le type d'hybridation, deux configurations peuvent être distinguées dans cette classe ;

- Les deux types de systèmes (continu et discret) sont couplés : Dans ce type, il est généralement remarqué l'existence des parties discrètes qui sont intégrées dans le système continu, comme il est le cas de production de la semoule où la production est continue tout en ayant un conditionnement discret des produits à la fin du système.
- Les deux aspects continu et discret cohabitent : La spécification de cette classe est que dans le même système de production, les traitements sont continus mais effectués par lots. L'exemple représentant ce cas est la production des boissons.

I.2.3 La fonction ordonnancement dans la gestion des systèmes de production

Le pilotage ou la gestion des systèmes de production c'est une fonction qui vise à établir une organisation efficace dans l'espace et dans le temps de toutes les activités relatives à la production afin d'atteindre les objectifs de l'entreprise (Giard, 2003). Cette fonction consiste à rechercher des solutions à l'ensemble de problèmes liés à la production tels que la gestion des données, la planification, le contrôle et le suivi de production, la gestion des stocks, l'ordonnancement, etc. En effet, plusieurs types de décisions avec différentes gravités sont possibles à être mis en place. Ces décisions sont généralement élaborées dans trois niveaux hiérarchiques distincts ; stratégique, tactique et opérationnel.

Au niveau opérationnel, la fonction ordonnancement consiste à convertir les décisions de fabrication définies par le programme directeur de production (PdP) en instructions d'exécution à court terme. Les décisions prises par cette fonction, sont destinées à piloter et contrôler l'activité des postes de travail dans l'atelier. En effet, cette partie de gestion de la production peut être décomposée en trois sous-fonctions (Javel G. , 2010)

Élaboration des ordres de fabrication : qui consiste à traduire les informations du programme directeur de production (PdP) en instructions qui déterminent la réalisation des opérations de fabrication ;

Élaboration du planning d'atelier : Cette tâche consiste, en fonction de ces ordres de fabrication et de la disponibilité des ressources consommables (matières premières,

composants) et non consommables (postes de travail), à établir le calendrier prévisionnel de fabrication. Cela revient à transformer les prévisions de fabrication à court terme en ordres d'exécution à très court terme ;

Lancement et suivi : Cette tâche consiste à distribuer aux postes de travail les documents nécessaires à la bonne exécution des fabrications, à suivre l'exécution des fabrications et à résoudre les problèmes générés au cours de la production.

I.3 Les systèmes de production Flexibles

I.3.1 Définition

Dans la littérature plusieurs définitions du système de production flexible (SPF, FMS) ont été proposées, mais sans une définition standard acceptée pour ce terme. La plupart des auteurs donnent des définitions basées sur la composition matérielle du système, tandis que d'autres définissent les FMS relativement à leurs performances et capacités fonctionnelles. D'autres auteurs préfèrent une définition qui intègre les deux aspects fonctionnel et structurel (SOUIER, 2012). Nous présentons ici quelques définitions parmi celles exposées dans la littérature.

Shivanand et al. (Shivanand, Benal, & Koti, 2006) voient que le FMS est un ensemble de machines interconnectées par un système de transport. Ce système sert à transporter les pièces (produits) vers les machines sur des palettes ou d'autres unités d'interface de tel sorte que la synchronisation des pièces-machines est précis, rapide et automatique. Dans cette catégorie, un ordinateur central contrôle à la fois les machines et le système de transport.

Talavage, J. (Talavage, 1987) considère qu'un système de production est dite flexible si ce système est capable de traiter plusieurs types de pièce simultanément et automatiquement avec des machines qui sont capables d'accepter et réaliser des opérations sur des pièces dans n'importe quelle séquence. De ce fait, les machines du système doivent, l'hors d'usinage d'une pièce, disposer des outils pour usiner la pièce suivante ainsi que des instructions nécessaires pour faire un simple changement d'outils sans intervention humaine.

Une définition plus globale est plus précise est donné par Tempelmeier et al. (Tempelmeier & Kuhn, 1993) qui voient que le FMS est un système de production composé d'un ensemble de machines à commande numérique identiques et / ou complémentaires qui sont connectées via un système de transport automatisé. Chaque processus dans le FMS est contrôlé par un

ordinateur dédié. Cela est souvent intégré dans un vaste réseau hiérarchique d'ordinateurs. Un FMS est capable de traiter des pièces dans des séquences arbitraires avec des délais de configuration négligeables entre les opérations.

I.3.2 Les éléments caractérisant le FMS

D'après les définitions présentées précédemment, quatre composants peuvent caractériser un système flexible de production (FMS) ; les stations de travail, le système de stockage, le système de transport et le système de contrôle. Une description très détaillée de ces composants a été présentée par Sari (Sari, 2003).

I.3.2.1 Les stations de travail

Une station de travail ou un poste de travail, comme il était défini dans (Javel G. , 2010), est une machine ou un endroit aménagé spécifiquement par/dans lequel on peut exécuter une ou plusieurs opérations de production. Ces opérations peuvent être des opérations de fabrication, d'assemblage, de contrôle, de chargement et/ou de déchargement, etc. En effet, selon Shivanand et al. (Shivanand, Benal, & Koti, 2006), dans un système flexible de production, plusieurs types de stations de travail peuvent être distingués.

- Les machines à commande numérique : sont des machines-outils contrôlées par des commandes numériques via un ordinateur. Ces machines sont destinées généralement à effectuer des opérations multiples d'usinages tel que ; le tournage, le fraisage, etc.

- Les stations de chargement et de déchargement : Ces stations sont des endroits placés en amant ou en ava le des autres postes de travail et ils sont réservés pour le chargement et /ou le déchargement des produits. Ces deux opérations sont effectuées à l'aide des systèmes de manutention installés dans ces stations.

- Les stations d'assemblage : Ce type de stations est réservé spécifiquement pour les opérations d'assemblage, non seulement des matières première (avec des stations situées généralement en avale) mais aussi des produits semi finis (avec des stations situées au cours du système). Les opérations d'assemblage dans ces stations sont réalisées soit à l'aide d'une machine d'assemblage soit à l'aide des robots manipulateurs.

- Les stations d'inspection (de contrôle) : sont des espaces dans lesquels des opérations de teste des produits sont effectuées. Ces stations peuvent possédés des machines de test, des robots, des caméras, etc.
- Etc.

La figure suivante présente un exemple d'un système de production flexible avec neuf postes de travail. Ce système est destiné à la fabrication des tiges pour les turbocompresseurs (EMAG, 2020). En effet, selon le type des opérations nécessaires à la réalisation des tiges, les neuf stations sont destinées chacune, à accomplir une fonction bien déterminée.

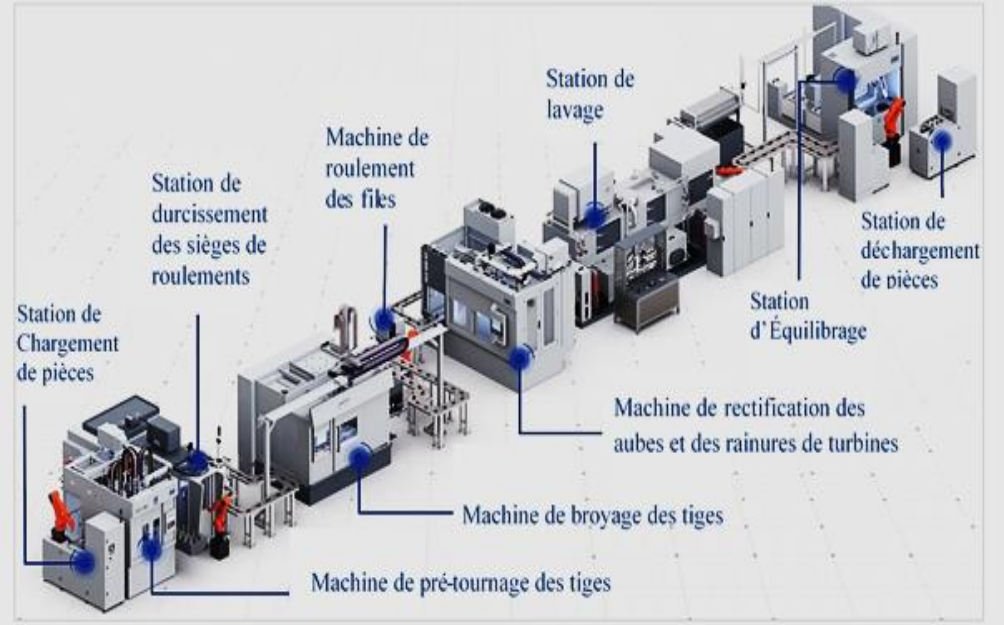


Figure I.1. Système de fabrication des arbres pour le turbocompresseur (EMAG, 2020)

I.3.2.2 Les systèmes de manutention et de transport

Les systèmes de manutention et de transport sont des systèmes qui ont pour objectif de déplacer les matières (les pièces) à traiter, entre les différentes stations. Ces systèmes sont souvent des systèmes automatisés qui fonctionnent en cohérence avec les postes de travail. En effet, la fonction de transport des matières peut être assurée par plusieurs moyens selon le type et de l'organisation du FMS (Eloundou J. , 2016) :

- Les convoyeurs : Un convoyeur est un mécanisme qui permet de déplacer une charge isolée (cartons, bacs, sacs, etc.) ou de produits en vrac (terre, poudre, aliments, etc.) de façon continue sur un trajet prédéterminé. Selon le trajet sur lequel les pièces sont transportées plusieurs types de convoyeur sont distingués ;

convoyeur à bande, convoyeur à résonance, convoyeur à chaînes, convoyeur à pas de pèlerin, etc. (Québec, 1996)

- Des véhicules auto guidés : Les AGV sont des robots qui se déplacent de façon autonome sans l'intervention humaine. Le déplacement de ces moyens s'effectué par la détection d'une piste, tracée dans ou sur le sol, fil émetteur d'ondes enfoui, de rail métallique au sol ou de fil électrique noyé au sol.
- Les transpalettes : les transpalettes sont des chariots hydrauliques, manuels ou électriques servant au déplacement de palettes de manutention.
- Des robots manipulateurs : Un bras manipulateur est un bras d'un robot généralement programmable, avec des fonctions similaires à un bras humain. Les liens de ce manipulateur sont reliés par des axes permettant, soit de mouvement de rotation (comme dans un robot articulé) et/ou de translation (linéaire) de déplacement (…)
- Etc.

I.3.2.3 Les systèmes de stockage

Les systèmes de stockage sont des systèmes dédiés au stockage des matières premières, des produits semi-finis et des produits finis. Ces systèmes contiennent les aires de stockages (des emplacements de stockage) et les systèmes de chargement/déchargement. D'après Kouloughli (Kouloughli, 2013), les systèmes de stockage les plus élaborés sont les systèmes automatisés de stockage/déstockage (Automated Storage Retrieval Systems AS/RS). Ces systèmes sont constitués de magasins de stockage (racks), de machines S/R de stockage/ déstockage et d'un convoyeur reliant les machines S/R et les points d'entrée/sortie de l'aire de stockage.

I.3.1.4 Les systèmes de contrôle et de communication.

L'unité la plus importante qui distingue le système FMS est le système de contrôle et de communication. Sari (Sari, 2003) présente trois composants principaux du système de contrôle ;

- Les calculateurs, sont des ordinateurs à base de microprocesseurs dotés de mémoire et de puissances de calculs plus ou moins grandes.
- Les logiciels, développés pour chaque type de commande, sont le cœur du système de contrôle, leurs puissances dépend de degré de performances du système ;
- Les bases de données, les systèmes de transmission d'information et les capteurs, représentent la troisième entité.

I.3.3 Les différents types du FMS

De nombreuses classifications des systèmes de production flexible ont été proposé selon plusieurs critères. Dans ce cadre nous pouvons citer par exemple le critère de classification liée au nombre de machines à commande numérique et leur agencement dans le système. On peut citer la classification évoquée dans (Tetzlaff, 1990) qui a été divisé en cinq types ;

- La cellule de fabrication flexible (*Flexible manufacturing cell*) : c'est une installation qui se compose de plusieurs machines CNC. Cette cellule est généralement dotée des stocks de pièces, des changeurs d'outils et des changeurs de palettes.
- Système d'usinage flexible (*Flexible machining system*) : c'est un système qui se compose d'une collection de cellules de fabrication flexibles reliées par un système de transport automatisé. Dans ce type, il n'y a aucune restriction sur le flux de pièces, c'est-à-dire que chaque pièce peut se déplacer d'une cellule vers n'importe quelle autre cellule.
- Ligne de transfert flexible (*Flexible transfer line*): Dans les lignes de transfert flexibles, les machines-outils sont disposées selon le principe de l'atelier flow shop. Le flux de différents types de pièces à travers le système suit un chemin unique donné.
- Multi-ligne de transfert flexible (*Flexible transfer multi-line*) : se compose de plusieurs lignes de transfert flexibles interconnectées. Il peut être considéré comme un mélange d'un système d'usinage flexible pur et d'une ligne de transfert flexible.
- Systèmes de fabrication flexibles (*Flexible manufacturing systems*) : Tous les systèmes qui se situent entre les deux extrêmes des systèmes d'usinage flexibles et des lignes de transfert flexibles sont considérés comme des systèmes de fabrication flexibles.

Un autre critère, basé sur les modèles des flux de pièces à traiter, a été retenu pour définir une classification des FMSs. Selon ce critère, Shivanand et al. (Shivanand, Benal, & Koti, 2006) ont classé les FMS en cinq classes :

- FMS séquentiel (*Sequential FMS*) : le FMS séquentiel fabrique un lot de pièces d'un seul type, puis la planification et la préparation sont effectuées pour le lot de type suivant à fabriquer. Il fonctionne comme une petite ligne de transfert flexible par lots.
- FMS aléatoire (*Random FMS*) : Ce type de système fabrique n'importe quel mélange aléatoire de types de pièces à un moment donné.
- FMS dédié (*Dedicated FMS*) : Ce système fabrique continuellement et pendant de longues périodes, le même mélange, mais limité, de types de lots de pièces.

- FMS conçu (*Engineered FM*): le FMS conçu est un cas particulier de type de système précédent, il fabrique le même mélange de types de pièces mais pendant toute sa durée de vie.
- FMS modulaire (*Modular FMS*) : c'est un hôte FMS sophistiqué, il permet à l'utilisateur d'étendre ses capacités (capacités du FMS) de manière progressive à l'un des quatre types précédents.

I.3.4 Les différentes configurations (layouts) du FMS

La configuration du système flexible de production se représente par la manière dont le système est structuré. En d'autres termes, la configuration d'un FMS détermine la position des composants (machines, transporteurs, …) les uns par-rapport aux autres. En réalité, plusieurs factor sont pris en compte lors de détermination et le choix d'une configuration d'un FMS. Parmi eux on peut noter l'espace de l'atelier, le type de produit à traiter, le type de système de manutention et de transport, etc. De ce fait ils existent plusieurs configurations de FMS ;

I.3.4.1 La configuration en ligne

La configuration en ligne ou la configuration sur une seule ligne (rangée) est une configuration largement mise en œuvre dans FMS, dans laquelle les machines sont disposées le long d'une ligne droite. Un dispositif de manutention de pièces est installé tout en long du système, il sert à déplaces les produits d'une machine à l'autre d'une manière directe. (Satheesh Kumar, Asokan, Kumanan, & Varma, 2008).

La figure suivante présente un exemple de système FMS qui se compose de trois machines disposées sous une configuration en ligne. Les pièces doivent suivre le même chemin de déplacement pour être traiter sur les déférentes machines. En effet, ce déplacement dans un chemin unique ne désigne pas forcément que les pièces ont les mêmes étapes et les mêmes opérations de fabrication.

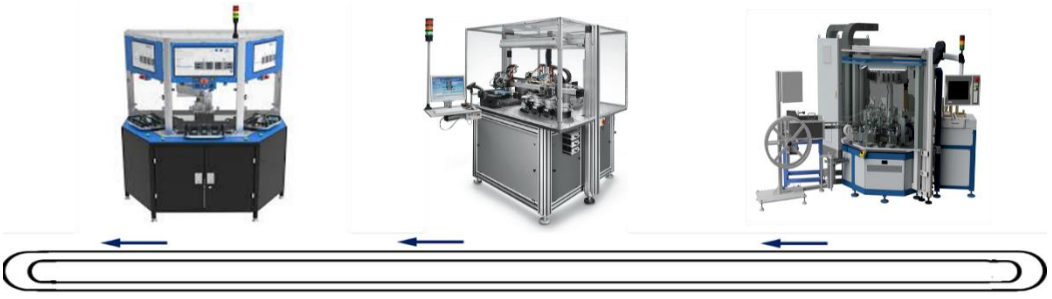


Figure I.2. Configuration en ligne

I.3.4.2 La configuration en boucle (Loop)

La configuration en boucle est l'une des configurations les plus couramment utilisées de l'installation des FMSs. Sous cette configuration, le système FMS se compose de plusieurs machines reliées par un système de manutention de pièces sous forme circulaire (Rifai, Dawal, Zuhdi, Aoyama, & Case, 2016). Comme il est présenté dans le système de la figure I.4, les pièces se déplacent généralement dans une seule direction autour de la boucle, avec la possibilité de s'arrêter et d'être transférées vers n'importe quelle station. Les stations de chargement et de déchargement sont généralement situées à l'extrémité de la boucle. (Shivanand, Benal, & Koti, 2006).

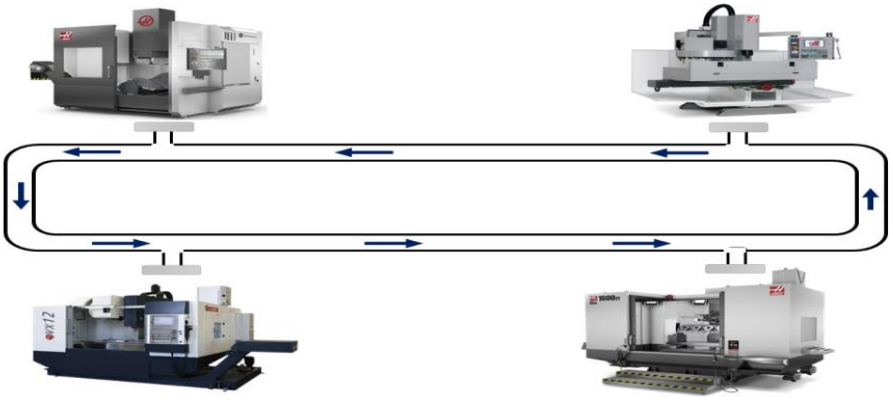


Figure I.3. Configuration en boucle

I.3.4.3. La configuration en échelle

La configuration en échelle (figure I.5) s'agit d'une boucle avec des barreaux sur lesquels se trouvent les postes de travail. Les échelons augmentent le nombre de possibilités de passage d'une machine à une autre et évitent d'avoir recours à un système de manutention secondaire. Cette configuration réduit la distance moyenne de déplacement et minimise l'encombrement du système de manutention en réduisant ainsi le temps de transport entre les stations (Suniya, 2013).

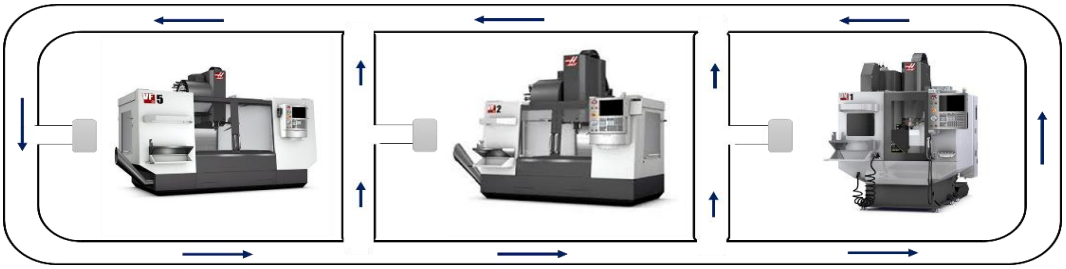


Figure I.4. Configuration en échelle

I.3.4.4 La configuration centrée sur le robot

La cellule centrée sur le robot est une forme relativement nouvelle de système flexible dans laquelle un ou plusieurs robots sont utilisés comme système de manutention (la figure I.3.) Les robots industriels peuvent être équipés de pinces qui les rendent bien adaptés à la manutention de pièces rotatives (Kaushal, Vardhan, & Rajput, 2016).

Figure I.5. Configuration centrée sur le robot

I.3.5 La flexibilité dans les FMS

Selon Erschler et al. (Erschler & De Terssac, 1988), la flexibilité d'un objet ou d'un système est une propriété qui recouvre généralement deux aspects complémentaires mais distincts ; un aspect interne lié à une capacité de changement et de reconfiguration à une variété d'états possibles et un aspect externe lié à une capacité d'adaptation à des modifications de l'environnement. Quant à Nagarur (Nagarur, 1992) il a défini la flexibilité des systèmes de production par la capacité de ces systèmes à s'adapter rapidement à des changements de facteurs clefs tels que le produit, le process, la charge et les pannes des machines

Dans l'industrie, nombreux systèmes de fabrication ont été nommés FMS mais en réalité certains de ces installations sont appelés FMS simplement parce qu'ils contiennent une certaine automatisation comme par exemple les lignes de transfert, les systèmes contenant uniquement un système de stockage/déstockage automatisés et ceux qui ne contiennent que plusieurs machines à commande numérique CNC (Browne, Dubois, Rathmill, Sethi, & Stecke, 1984). Dans ce contexte, plusieurs dimensions de la flexibilité dans les systèmes de production ont été définies afin de bien cerner sur ce qui constitue exactement un FMS (Javel

G. , 2010) (Eloundou J. , 2016) (Gupta & Goyal, 1989) (Eyers, Potter, Gosling, & Naim, 2018) :

I.3.5.1 La flexibilité des machines

La flexibilité des machines se traduit par la capacité d'apporter des modifications nécessaires pour produire un ensemble donné de types de pièces. C'est en effet, la capabilité de ces machines à effectuer plusieurs variétés d'opérations. Cette flexibilité peut être mesurée par le temps nécessaire pour remplacer une coupe usée ou outil cassé, le temps de changement d'outil afin de produire un sous-ensemble différent des types de pièces donnés, et le temps nécessaire pour assembler ou monter de nouveaux équipements requis.

En réalité, ce type de flexibilité permet aux systèmes d'apporter une capacité de produire des variétés de pièces de types différents, d'augmenter le taux d'utilisation de leurs machines, …

I.3.5.2 La flexibilité des outils de manutention

Les moyens de manutention jouent un rôle primordial dans les systèmes flexibles de production. En effet, la flexibilité des moyens de manutention se résume dans la capacité de ces moyens à transporter les différentes pièces de manière efficiente à travers le système de production lors d'une phase de production. Les objectifs de cette flexibilité sont l'augmentation de la disponibilité des machines, l'augmentation du taux d'utilisation des machines, la réduction des temps de production et par conséquent l'augmentation du rendement du système de production manufacturier.

I.3.5.3 La flexibilité des opérations

C'est la capacité d'intervertir ou de remplacer les opérations qui permettent la fabrication d'une pièce. En effet, cela signifie qu'il n'existe pas de contraintes de précédence entre toutes les opérations de fabrication d'une pièce. Les objectifs de la flexibilité des opérations sont l'amélioration de la disponibilité des machines, et de leur taux d'utilisation, la possibilité de poursuivre la production même si une machine est défaillante et la facilitation de l'ordonnancement en temps réel des pièces.

I.3.5.5 La flexibilité de routages des produits

C'est la capacité d'un système de production à prendre en compte les pannes des machines qui surviennent lors de la production et de continuer tout de même à produire. Cette possibilité existe si un type de pièce peut être traité par plusieurs chemin, ou si chaque opération peut être effectuée sur plusieurs machines. Ce niveau de flexibilité ne

peut être atteint que si l'on possède une certaine flexibilité des machines et des machines à commande numériques possédant plusieurs outils.

I.3.5.6 La flexibilité du volume de produits

La flexibilité du volume de produits dans un système de production se traduit par la capacité d'utiliser ce système de manière rentable afin de réaliser différents volumes de produits. Le degré d'automatisation élevé introduit dans le système augmente cette flexibilité, qui peut être mesurée par, à quel point de volumes de pièces, le système étant toujours exploité de manière rentable.

I.3.5.7 La flexibilité des processus de fabrication

La flexibilité des processus de fabrication est définie par l'ensemble des pièces qu'un système peut produire sans apporter des modifications majeures. Cette souplesse en production réside dans le fait que chaque pièce peut être usinée individuellement, et pas nécessairement en lots. L'évaluation de ce type de capacité peut être faite par le nombre de types de pièces qui peuvent être traitées simultanément sans utiliser de lots.

L'objectif d'une telle flexibilité est la diminution de la taille des lots de production et des stocks, l'augmentation des ressources et la minimisation de la duplication de machines. Pour mettre en place la flexibilité des processus de fabrication, le système doit posséder une flexibilité des machines, une flexibilité des opérations significative, et une flexibilité des moyens de manutention.

I.4 Conclusion

Les objectifs de l'intégration de flexibilités dans les systèmes de production sont multiples, traitement d'une variété de produits dans la même installation, réduction de temps de passage des produits entre les postes de travail, diminution des stocks, équilibrage de charge de travail des machines, etc. En revanche, ces avantages et importances liées à ce type de systèmes sont associés, malheureusement, par un défi major consistant à gérer et à optimiser ces derniers, notamment la résolution des problèmes d'ordonnancement.

Chapitre II : Ordonnancement dans les systèmes de production flexibles

II.1 Introduction

La planification et l'ordonnancement sont des processus décisionnels utilisés régulièrement dans de nombreuses industries manufacturières et de services. Ces formes de prise de décision jouent un rôle important dans l'approvisionnement et la production, dans le transport et la distribution, ainsi que dans le traitement et la communication de l'information.

Dans ce chapitre, nous présenterons tous d'abord, un aperçue général sur les problèmes d'ordonnancement, ces caractéristiques et ces différents types. Ensuite nous donnerons une description du problème d'ordonnancement job shop qui fait la base de notre étude. Enfin nous exposerons quelques méthodes de résolution qui peuvent être utilisées pour résoudre ces problèmes.

II.2 L'ordonnancement

II.2.1 Définition

La notion de l'ordonnancement est très vaste et globale qui touche plusieurs domaines dans notre vie. De ce fait, plusieurs définitions de cette notion ont été proposées dans la littérature dont, l'aspect commun entre ces définitions est la notion d'affecter des tâches aux ressources en cherchant à optimiser un certain objectif.

Selon (Artigues, 1997) , un problème d'ordonnancement est un problème d'optimisation combinatoire qui constitué d'un ensemble de tâches, d'un ensemble de ressources, d'un ensemble d'objectifs et d'un ensemble de contrainte. La solution à ce problème est un ordonnancement qui donne des indications sur les dates de début des tâches ou qui propose une relation d'ordre total ou partiel sur la séquence de réalisation des tâches.

Dans (Esquirol, Lopez, & Lopez, 1999) et (Lopez & Roubellat, 2001), les auteurs ont défini ce problème comme suit : « *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement,...) et de contraintes portant sur l'utilisation et la disponibilité de ressources requises* »

Azem, S. (Azem, 2010) a défini le problème d'ordonnancement comme étant un ensemble de jobs à réaliser sur un ensemble de ressources de sorte qu'une fonction objectif soit optimisée

II.2.2 Eléments de problème d'ordonnancement

D'après les définitions présentées précédemment, nous pouvons distinguer quatre éléments primordiaux par lesquels n'importe quel problème d'ordonnancement se caractérise.

II.2.2.1 Les tâches

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début t_i ou de fin c_i, dont la réalisation est caractérisée par une durée P_i et par l'intensité a_i^k avec laquelle elle consomme certains moyens k ou ressources. Cette intensité est supposée comme constante durant l'exécution de la tâche (Lopez & Roubellat, 2001).

En effet, dans certains problèmes d'ordonnancement l'exécution des tâches peut accepter des interruptions. Alors, les tâches sont dites préemptives Dans d'autres problèmes cette interruption est non permise, les tâche donc sont non préemptives. En plus, les tâches ou les opérations se caractérisent par d'autres variables comme ; la date de disponibilité r_i qui correspond à la date avant laquelle la tâche ne peut pas être commencée, et la date d'achèvement souhaitée (ou exigée) d_i qui correspond à la date après laquelle la tâche est en retard. La Figure suivante donne une représentation de la tâche en désignant ses principales caractéristiques

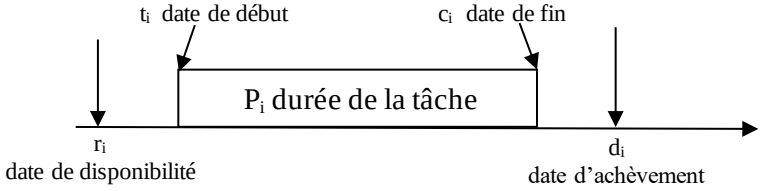


Figure II.1. Caractéristiques d'une tâche **(Maache, 2011)**

II.2.2.2 Les ressources

« Une ressource k est un moyen technique ou humain requis pour la réalisation d'une tâche et elle est disponible en quantité limitée, sa capacité c_k est supposé constante » (Esquirol, Lopez, & Lopez, 1999). Ce moyen nécessaire à la réalisation des tâches, peut être renouvelable c'est à dire qu'il peut être utilisée de nouveau une fois l'opération de traitement est terminée, comme c'est le cas des opérateurs, des machines, des outillages, etc. Mais il peut aussi ne pas l'être lorsqu'on parle de ressource consommable telle que la matière première, les produits d'entretien ou encore les budgets.

Une ressource est de plus doublement contrainte si son utilisation instantanée et sa consommation globale sont limitées. Si une ressource ne peut exécuter qu'une seule tâche à la fois elle est dite disjonctive (ou non partageable) comme c'est le cas pour une machine-à-outil

ou un robot manipulateur. On parle de ressource cumulative (ou partageable) dans le cas d'une ressource qui peut traiter plusieurs tâches simultanément ou le cas de plusieurs ressources qui peuvent être utilisées pour la même tâche (Marmier, 2007).

II.2.2.3 Les contraintes

Les contraintes expriment les restrictions sur les valeurs que peuvent prendre certaines variables. Dans les problèmes d'ordonnancement, deux types de contraintes sont généralement distinguées : les contraintes *temporelles* et les contraintes *de ressources*.

- Les contraintes temporelles : Ces contraintes comprennent les restrictions sur le temps alloué aux tâches (délai de livraison par exemple) ou la durée totale d'un ordonnancement, les contraintes d'antériorité ou de précédence relatives au positionnent des tâches les unes par rapport aux autres, les contraintes de calendrier liées aux plages horaires de travail, etc.

- Les contraintes de ressources : Ces contraintes sont généralement liées à la capacité et/ou la disponibilité des ressources. De point de vis capacité des ressources, deux types de contraintes, selon la nature de ressources utilisées, peuvent être distingués; les contraintes liées à la nature disjonctive où les ressources ne peuvent être utilisées que par une tâche à la fois et les contraintes liées à la nature cumulative dont les ressources peuvent être utilisées par plusieurs tâches simultanément (Esquirol, Lopez, & Lopez, 1999), (Billaut & Roubellat, 1996). De point de vis disponibilité, les contraintes de ressources sont liées à l'existence de ressources au moment d'utilisation (ressources consommables par exemple) et/ou la possibilité de les utiliser (c'est-à-dire pas de panne au moment d'utilisation).

II.2.2.4 Les objectifs et les critères d'évaluation

La résolution de n'importe quel problème d'ordonnancement consiste souvent à chercher, soit de minimiser, soit de maximiser un critère qui correspond à une amélioration suivant au moins l'un des trois facteurs : coût, qualité ou délais. En effet, plusieurs classes d'objectifs concernant un ordonnancement sont distinguées (Esquirol, Lopez, & Lopez, 1999):

- Les objectifs liés au temps : Dans ce type d'objectifs, on trouve par exemple la minimisation du temps total d'exécution (Makespan), la minimisation du temps moyen d'achèvement des tâches, la minimisation des retards par rapport aux dates de livraison, etc.

- Les objectifs liés aux ressources : La maximisation de la charge d'une ressource, la minimisation de la consommation des ressources consommables ou la minimisation du nombre de ressources nécessaires pour réaliser un ensemble de tâches sont des objectifs de ce type.

- Les objectifs liés au coût : Ces objectifs sont généralement ; la minimisation des coûts de lancement, de production, de stockage, de transport, consommation d'énergie, etc.

II.2.3 Classification des problèmes d'ordonnancement

Les problèmes d'ordonnancement représentent les problèmes les plus étudiés par les chercheurs dans le domaine de la recherche opérationnelle. Ces problèmes sont présents dans tous les secteurs d'activités de l'économie depuis l'informatique jusqu'à l'industrie manufacturière. De ce fait, une grande nécessité de classification des ces problèmes s'impose pour aider les chercheurs qui veulent aborder un tel problème d'ordonnancement. Plusieurs classifications des problèmes d'ordonnancement sont présentées dans la littérature, des classifications basées sur le nombre de machines dans l'atelier, d'autre classifications sont basées sur le passage des jobs sur les machines et d'autre sont basées sur les critères à optimiser, etc.

La classification standard la plus populaire est la classification proposée par Graham et al (Graham, Lawler, Lenstra, & Kan, 1979). Dans cette classification les problèmes d'ordonnancement sont distingués par la notation $\boldsymbol{\alpha/\beta/\gamma}$, où le premier paramètre ($\boldsymbol{\alpha}$) décrit l'organisation de l'atelier et le nombre de machines de quelles il est constitué. Le deuxième paramètre ($\boldsymbol{\beta}$) définit les caractéristiques des travaux à réaliser et les contraintes associées. Le dernier paramètre ($\boldsymbol{\gamma}$) représente le critère d'optimisation sur lequel l'ordonnancent sera évalué. Cette notation permet de présenter de manière précise le problème d'ordonnancement à traiter. Le tableau suivant décrit les principales valeurs de ces trois paramètres.

Tableau II.1. Les paramètres de problème d'ordonnancement **(Yahouni, 2019)**

Paramètres	Notation	Description
$\boldsymbol{\alpha_1}$	J F FH O …	Job shop Flow shop Flow shop hybrid Open shop …
$\boldsymbol{\alpha_2}$	$\{m\}$	Problème à m machines

β	r_i d_i prec pmtn …	Contrainte de dates de début au plus tôt des opérations Contrainte de date d'échéances des opérations Contrainte de précédences entre opérations Préemption des opérations autorisé ….
γ	$C_{max} = \max_{i\in\{0..n\}} C_i$ $\sum w_i C_i$ $T_{max} = \max_{i\in\{0..n\}} (C_i - d_i, 0)$ $. \sum w_i T_i$ $\sum T_i$ $L_{max} = \max_{i\in\{0..n\}} (C_i - d_i)$ $\sum w_i L_i$ $\sum \lvert L_I \rvert$ $\sum U_i = \lvert J_i \rvert \; if \; C_i > d_i$ - …	Makespan : la plus grande date de fin La date de fin pondéré des travaux Le retard maximum des travaux Le retard pondéré des travaux Le retard total Le retard algébrique maximum des travaux Le retard algébrique pondéré des travaux Le retard total absolu Le nombre de travaux en retard

II.2.4 Caractéristiques générales des ordonnancements

La résolution d'un problème d'ordonnancement consiste à trouver un ordonnancement (ordre) faisable qui détermine à la fois l'affectation de l'ensemble des tâches sur l'ensemble de ressources et l'ordre chronologique (temporelle) de l'exécution de ces tâches de manière à satisfaire un ou plusieurs critères. En effet, un ensemble de propriétés particulières peuvent être distinguées pour les solutions d'ordonnancement trouvées.

II.2.4.1 Ordonnancement admissible

L'admissibilité d'ordonnancement signifie la faisabilité de ce dernier vis à vis les contraintes imposées dans le problème étudié. C'est-à-dire, on dit qu'un ordonnancement est admissible si l'affectation et l'ordre des tâches définis par cet ordonnancement respectent toutes les contraintes du problème comme les dates limites, les précédences, la disponibilité des ressources, etc. La figure.II.2 présente un exemple d'ordonnancement admissible d'un problème à deux jobs et deux machines. Dans cet exemple une seule contrainte est imposée c'est la restriction sur la capacité des machines (chaque machine traite un seul job à la fois).

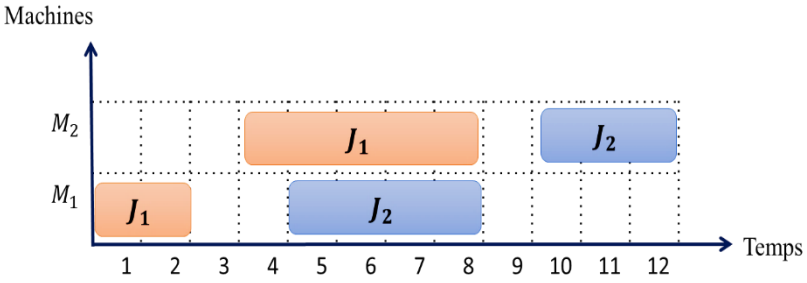


Figure II.2. Ordonnancement admissible

En effet, il est bien clair, comme il est présenté dans la figure II.2, que la solution définie par un ordonnancement admissible n'est pas toujours une solution optimale. Dans ce cas, une amélioration est souvent nécessaire pour atteindre la solution optimale ou au moins proche de l'optimale

II.2.4.2 Ordonnancement semi actif

Un ordonnancement semi-actif est un ordonnancement admissible dans le quel toutes les opérations sont calées à gauche de telle sorte qu'aucune opération ne commence avant sa date de début au plus tôt.

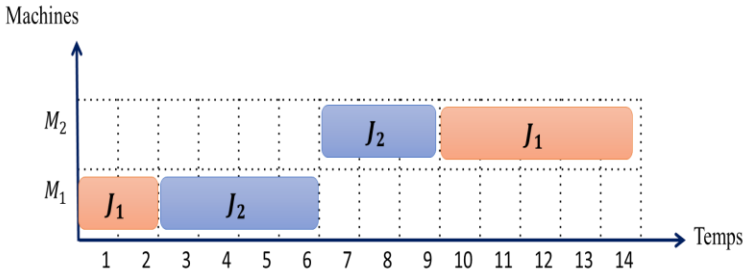


Figure II.3. Ordonnancement admissible semi actif

II.2.4.3 Ordonnancement actif

Dans un ordonnancement actif aucun glissement à gauche local ou global n'est possible. Aucune tâche ne peut commencer plus tôt sans reporter le début d'une autre (Taghezout, 2011), l'ordonnancement actif de l'exemple est présenté dans la figure II.4.

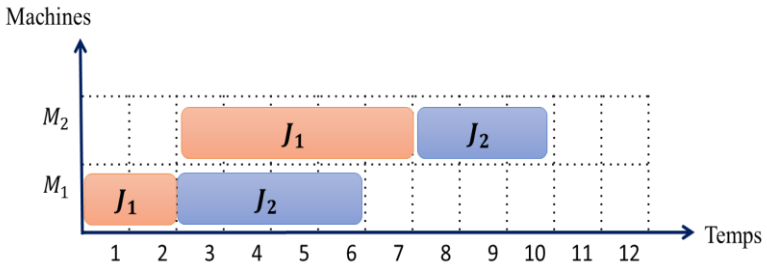


Figure II.4. Ordonnancement admissible actif

En effet, il est noté qu'un ordonnancement actif est aussi un ordonnancement semi-actif et pour un problème de job shop, la solution optimale présente un ordonnancement actif (Yahouni, 2019).

II.2.5 Problèmes d'ordonnancement d'atelier

Les problèmes d'ordonnancement d'atelier sont des problèmes qui concernent un type spécifique d'atelier. Vue la diversité des ateliers existants dans le monde industriel, plusieurs classes de problème d'ordonnancement ont été identifiés et analysés. Cependant, la majorité de ces classes ne sortent pas de l'une des trois grandes classes ; flow shop, job shop et open shop, qui peuvent être également devisée en sous autres classes.

II.2.5.1 Problèmes d'atelier flow shop

Dans un atelier flow shop (ou atelier à cheminement unique), les machines sont installées en série pour former la ligne de production (figure II.5). Le passage des produits à travers ces machines s'effectue dans un ordre unique. C'est-à-dire que tous les produits doivent visiter toutes les machines et avec le même enchainement.

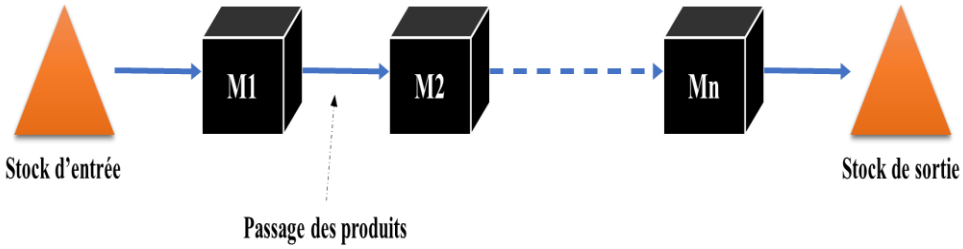


Figure II.5. Atelier Flow shop

Dans cette catégorie d'ordonnancement, nous pouvons distinguer quatre types de Flow shop (Hassam, 2012);

- Flow shop pur : La caractéristique principale de ce type est que les temps de traitement des tâches sont tous positifs.
- Flow shop généralisé : Dans ce type de flow shop, si une tâche ne doit pas s'exécuter sur une machine particulière, le temps opératoire pour cette machine est considéré nuls.
- Flow shop de permutation : C'est un flow shop dans lequel l'ordre de passage des produits ne se change pas entre les postes (ou les machines), c'est-à-dire que le

séquencement d'exécution des produits sur la première machine est le même sur la deuxième et sur toutes les autres machines.

- Flow shop hybride : C'est un atelier flow shop où les produits doivent visiter des étages composés d'une ou plusieurs machines plus tôt que des machines uniques. En effet, les machines appartenant au même étage, ne sont pas forcément identiques et disposées en parallèle. Les produits visitent successivement l'ensemble des étages et ne doivent passer que par une seule machine par étage.

II.2.5.2 Problèmes d'atelier job shop

L'atelier job shop (ou l'ateliers à cheminements multiples) est un atelier dans lequel les produits sont réalisés selon des gammes opératoires spécifiques. C'est-à-dire que le passage de chaque produit à travers les machines n'est pas le même pour tous les produits (figure II.6) et suit les contraintes de précédence prédéfinies pour ce produit. Dans ce cas, plusieurs changements d'outils sont à envisager.

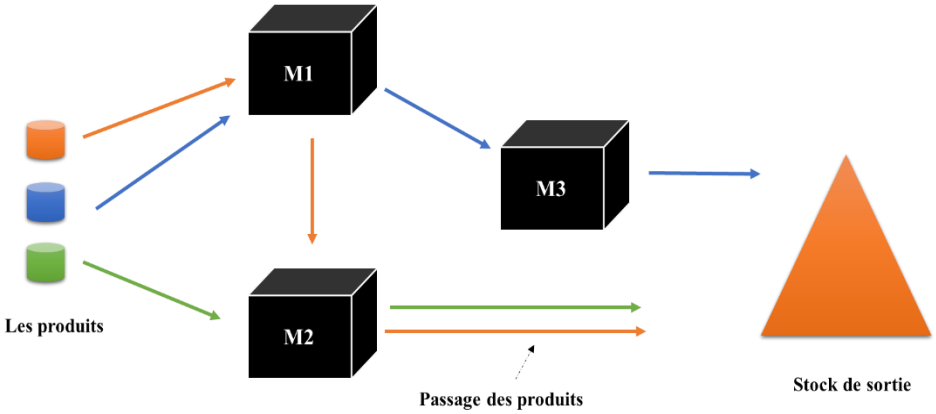


Figure II.6. Atelier Job shop

II.2.5.3 Problèmes d'atelier open shop

Cet atelier est plus généralisé, car dans ce type aucun ordre de passage n'est imposé, dans ce cas l'acheminement de toutes les opérations est multiple et libre, c'est-à-dire ces opérations peuvent être exécutées dans n'importe quel ordre.

Plusieurs extensions et cas particuliers de ces types de problèmes ont été définis dans la littérature, dont l'objectif est de rapprocher beaucoup plus aux caractéristiques réelles des systèmes de production (Pinedo M. , 2016), (Belkaid, 2014). Nous pouvant citer dans ce cadre :

- Les problèmes « à une seul machine », un cas particulier du flow shop où les tâches se réalisent avec une seule opération et qui nécessitent la même machine
- Les problèmes « à machines parallèles » dans lequel la réalisation des produits consiste en une seule opération et chaque opération dispose d'un ensemble de machines parallèles, mais n'en nécessite qu'une pour pouvoir être réalisée.
- Les problèmes « job shop flexible » est une autre classe particulière de ce type où les machines sont capables de réaliser plusieurs types d'opérations, dans ce cas, une opération peut être affectée à n'importe quelles machines pouvant effectuer cette opération.

L'étude menue dans cette thèse examine un problème d'ordonnancement dans les ateliers de type job shop. De ce fait, nous allons présenter, dans la section suivante, plus de détail sur la définition et les caractéristiques de ce problème.

II.3 Problème d'ordonnancement job shop

II.3.1 Définition

Le problème job shop reste parmi les problèmes d'optimisation les plus difficiles de la gestion de la production, sa difficulté réside dans le fait que chaque job a son propre chemin et son propre nombre de machines qu'il doit visiter. La complexité de ce type de problème augmente très considérablement avec l'augmentation du nombre de machines et/ou du nombre des jobs.

Selon Kuhpfahl (Kuhpfahl, 2015), le problème d'ordonnancement job shop consiste à réaliser un ensemble de n jobs (travails) sur un ensemble de m machines. Chaque job est composé d'un ensemble d'opérations qui doivent être exécutées successivement sur des machines spécifiques pour chacune. Les jobs ne s'exécutent pas sur toutes les machines et de plus n'ont pas le même ordre d'exécution. En effet chaque job emprunte le chemin qui lui est propre. L'objectif de l'ordonnancement est de trouver un planning qui détermine le séquencement de traitement de ces n jobs sur les m machines sous réserve de respecter les contraintes du problème tel que :

- Le traitement de la $i^{ième}$ opération du job j prend une durée positive $P_{ij} > 0$.
- Chaque opération doit être traitée une seule fois.
- La préemption n'est pas autorisée pendant le traitement des opérations.
- Le traitement des opérations ne doit pas se chevaucher.

- Il n'y a pas de temps de configuration dépendant de la machine ou de la séquence.
- Les machines doivent toujours être disponibles.

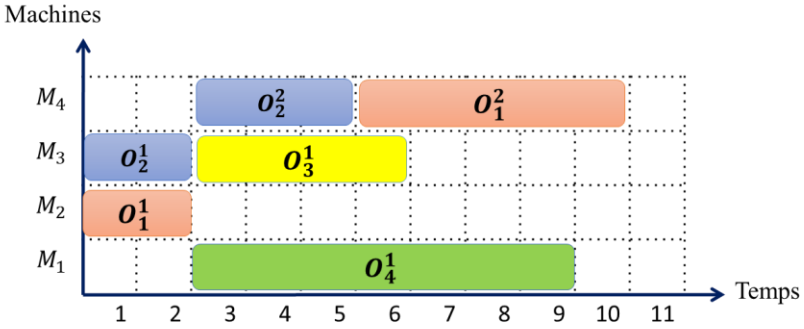
II.3.2 Représentations d'un problème d'ordonnancement job shop

Deux techniques de représentations du problème d'ordonnancement sont souvent rencontrées dans la littérature, la première est le diagramme de Gantt avec lequel on utilise l'échelle de temps pour représenter l'ordre des opérations et la deuxième est le graphe Potentiel-Tâches sur lequel les opérations sont présentées dans des nœuds liés les unes avec autres par des flèches.

II.3.2.1 Le diagramme de Gantt

Le diagramme de Gantt, appelé aussi diagramme à barres, du nom de son développeur Henry Gantt, est devenu le moyen le plus simple et célèbre pour représenter une solution d'un ordonnancement. Dans ce graphique et pour un ordonnancement job shop, on visualise l'enchaînement des opérations sur deux axes ; l'axe des abscisses représente le temps et l'axe des ordonnées représente les machines. Sur chaque ligne horizontale qui correspond à une machine, on met la séquence des opérations à effectuer sur cette machine. Chaque opération est représentée par une barre dont sa longueur est proportionnelle à sa durée. A la fin, on obtient sur le diagramme l'enchaînement de opérations sur chacune des machines, avec les dates de début et de fin de chaque tâche.

La figure suivante présente un exemple de diagramme de Gantt. L'ordre des opérations montré par ce diagramme défini l'une des solutions possibles d'un problème job shop avec quatre jobs à exécuter sur quatre machines. Dans cet exemple il est clair que la contrainte de présidence entre opérations du même job est définie par le séquencement temporelle de ces opérations (aucun chevauchement entre opérations du même jobs) et la contrainte de capacité de ressources est définie par l'affectation d'une seule opération à la même machine en même temps (aucun chevauchement entre opérations sur la même machine)



Figue II.7. Diagramme de Gantt pour un problème 4x4

II.3.2.2 Le graphe Potentiel-Tâches

Cet outil graphique a été développé par Roy et Susmann (Roy & Sussmann, 1964) grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à évènements discrets. Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les contraintes par des arcs, comme il est montré dans la Figure II.8. En plus, les arcs peuvent être de deux types ; les arcs conjonctifs (les flèches continues) illustrant les contraintes de précédence et indiquant les durées des opérations et les arcs disjonctifs (les flèches discontinues) indiquant les contraintes de partage de ressources.

Dans l'exemple présenté par la Figure II.8, S et P sont des étapes qui représentent successivement le début et la fin de l'ordonnancement, par contre les numéros associés aux arcs représente les durés des tâches.

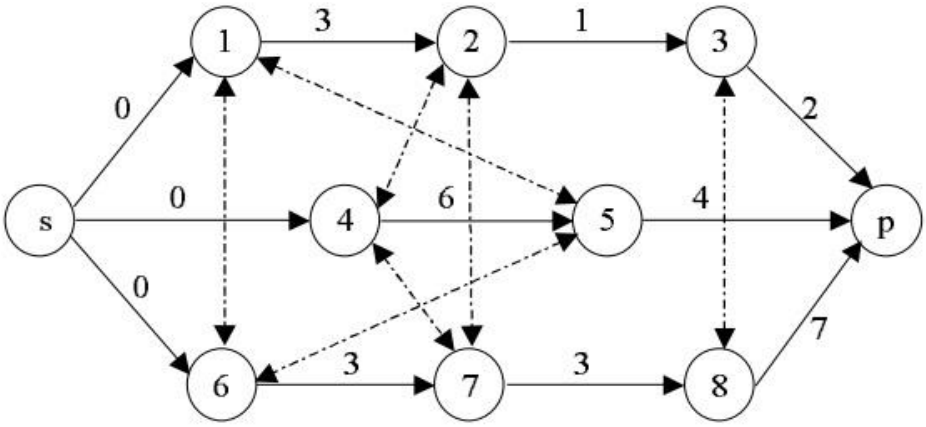


Figure II.8. Graphe Potentiel-Tâches

II.4 Méthodes de résolution

Les problèmes d'ordonnancement comme d'autres problèmes d'optimisation combinatoire, sont généralement résolus en utilisant deux approches différentes ; les méthodes exactes et les méthodes approchées. Les méthodes exactes consistent à trouver une solution optimale du problème, mais dans un temps non borné, de ce fait elles sont généralement appliquées à des problèmes de petites tailles. Par contre, les méthodes approchées ne donnent pas forcément une solution optimale, mais elles sont très performantes pour trouver une solution admissible généralement de bonne qualité dans un temps acceptable. Elles sont appliquées à des problèmes d'ordonnancement plus complexes.

Dans cette section, l'objectif n'est pas d'exposer toutes les méthodes de résolution des problèmes d'ordonnancement définies dans la littérature, mais de présenter quelques méthodes

populaires, notamment les méthodes, que nous avons utilisé pour résoudre notre problématique (voir la deuxième et la troisième partie de ce manuscrit).

II.4.1 Les méthodes exactes

Les méthodes exactes sont des méthodes qui fournissent, à partir des données du problème, une solution optimale en se basant sur des lois bien déterminées (Boukef, 2009). Ces méthodes permettent d'avoir des solutions vis-à-vis un certain critère donné, sous des conditions bien particulières. Dans ce contexte nous trouvons par exemple la méthode la plus utilisée dans la résolution des problèmes d'optimisation, la méthode par séparation et évaluation (branch and bound) (Pinedo, 2012) (Pinedo M. , 2016). D'autres méthodes tel que la programmation linéaire ou la programmation dynamique sont également utilisées pour trouver des solution optimales (Bronson & Naadimuthu, 1997).

L'utilisation de ce type de méthodes s'avère particulièrement intéressant dans les cas des problèmes de petites tailles. Par contre plus le problème est grand plus le recoure à ces méthodes est limité, ceci est à cause de l'accroissement exponentielle du temps du calcul. Les méthodes approchées sont alors les méthodes les plus utilisées quand les méthodes optimales ne permettent pas de résoudre le problème en un temps acceptable. En effet, les méthodes qui fournissent des solutions optimales sont plusieurs et multiples. Parmi ces méthodes nous pouvons citer les plus utilisées ; la programmation linéaire, la programmation dynamique, la procédure par séparation et évaluation (Branch and Bound).

II.4.1.1 La programmation linéaire

La programmation linéaire est une technique mathématique qui consiste à déterminer le meilleur résultat ou la meilleure solution possible à partir d'un ensemble donné de paramètres ou d'une liste d'exigences. Un problème de programmation linéaire est caractérisé, par des fonctions linéaires des variables ; l'objectif est linéaire dans les inconnues, et les contraintes sont également, des égalités linéaires ou des inégalités linéaires dans les inconnues (Luenberger, 2016). En effet, trois grandes familles de programmes linéaires sont distinguées ; la programmation linéaire simples, la programmation linéaire en nombres entiers, et la programmation linéaire mixtes.

Dans le premier type, les variables utilisées pour la modélisation, sont dites continues. La résolution du problème dans ce cas est souvent effectuée par deux algorithmes de résolution ; la méthode du simplex et la méthode des points intérieurs.

Avec le deuxième type de programmation, la modélisation des problèmes implique l'intégration des variables entières dans le modèle mathématique. En revanche, l'inconvénient

majeur de ce type de programmation peut être le grand nombre de contraintes et de variables requises. En effet, la majorité des problèmes d'ordonnancement, qui sont NP-difficiles, ne peuvent pas être résolus d'une façon optimale par des programmes mathématiques en nombres entiers (Belkaid, 2014).

La programmation linéaire mixtes inclue les deux types de variables continues et en nombres entiers. Cette catégorie de programmation ressemble fortement à la programmation linéaire en nombre entiers de point de vue résolution (Larabi, 2010).

Plusieurs travaux ont été présenté dans la littérature dans lesquels les auteurs ont fait le recours à la programmation linéaire pour résoudre leurs problèmes d'ordonnancement. Nous pouvons citer, dans ce cadre, quelque travaux récents : (Meng, Zhang, Ren, Zhang, & Lv, 2020), (Lunardi, Birgin, Laborie, Ronconi, & Voos, 2020), (Meng, Zhang, Shao, & Ren, 2019), (Zhang & Wang, 2018) et (Karimi, Ardalan, Naderi, & Mohammadi, 2017).

II.4.1.2 La programmation dynamique

La programmation dynamique est considérée comme une méthode d'énumération permettant l'investigation des solutions exactes pour les problèmes d'optimisation combinatoires. Les origines de cette méthode reviennent aux année cinquante plus précisément les travaux de Bellman (Bellman, 1957). Cette méthode est applicable à tout problème qui accepte une décomposition en une séquence de sous-problèmes plus simples de telle manière que la solution de l'un de ces sous-problèmes dépend de la solution de celui qui le précède. La méthode repose sur le principe d'optimalité de Bellman 2, ce qui permet de décrire le critère sous la forme d'une relation de récurrence reliant deux niveaux consécutifs (Benkalai, 2018).

II.4.1.3 La procédure par séparation et évaluation (Branch and Bound)

La procédure par séparation et évaluation ou en anglais « Branch and Bound » représente l'une des méthodes les plus utilisées à l'investigation des solutions optimales pour les problèmes combinatoires. Cette méthode fondée par Dantzig et al (Dantzig, Fulkerson, & Johnson, 1954), se base sur le principe d'énumération implicite de toutes les solutions possibles d'un problème d'optimisation. Cette énumération des solutions s'effectuée selon des étapes successives dans lesquelles un ensemble des branches sont établies permettant de décomposer le problème en plusieurs sous-problèmes qui sont décomposées à leur tour en sous-sous-problèmes ainsi de suite. Ce processus peut se visualiser sous forme d'un arbre avec des nœuds, où le nœud principal représente le problème à résoudre et les nœuds feuilles symbolisent les alternatives possibles.

L'utilisation de la procédure « Branch and Bond » repose sur deux concepts ; la séparation qui consiste à fractionner le problème principal sous un ensemble de problèmes plus réduites et l'évaluation qui consiste, selon un critère prédéfini qui détermine la borne inférieure d'une solution, à expertiser les solutions partielles trouvées et d'éliminer par conséquent celles qui ne conduisent pas à des solutions optimales. Cette propriété permet d'éviter d'une manière significative, la numération totale de toutes les solutions possibles qui sert à minimiser le temps de calcul. En revanche, cette méthode présente un inconvénient majeur lorsque nous traitons des problèmes de grande taille, donc un nombre très important de branchements doit être évalué et analysé.

II.4.2 Les heuristiques

Contrairement aux méthodes précédentes, les méthodes approchées (ou approximatives) sont des méthodes qui cherchent des solutions acceptables pour des problèmes d'optimisation combinatoires complexes, mais dans un temps raisonnable. La performance de ces méthodes est souvent mesurée par l'écart entre la valeur de la solution trouvée et la valeur de la solution optimale si elle est calculable (Laribi, 2018). Trois classes de méthodes sont distinguées de cette catégorie de techniques ; les heuristiques de décomposition, les heuristiques de construction et les méta-heuristiques (Belkaid, 2014), (Larabi, 2010).

II.4.2.1 Les heuristiques de décomposition

Les heuristiques de ce genre, consistent à apporter des solutions approchées à des problèmes d'optimisation complexes en les divisant en plusieurs sous problèmes. Un ensemble de familles des heuristiques de décomposition sont présentées dans (Azem, 2010) :

- La décomposition hiérarchique proposée par Erscher et al. (Erschler J. F., 1986). Pour cette technique la répartition du problème s'effectuée en plusieurs niveaux. Dans chaque niveau des décisions sont prises qui deviennent par la suite des contraintes pour les niveaux inférieurs.
- La décomposition temporelle c'est une heuristique qui consiste à ordonnancer une partie d'opérations disponibles avant une date et inclure les autres dans une séquence partielle (Portmann, 1988). Les opérations restantes et les opérations qui deviennent disponibles après sont regroupées puis ordonnancées ensemble, et ainsi de suite.
- La décomposition spatiale proposée par Portmann (Portmann, 1988). Contrairement aux autres types de décomposition, pour cette heuristique, l'atelier de travail celui qui sera divisé en plusieurs sous-ateliers avec un minimum de mouvements entre eux.

La résolution du problème dans ce cas, consiste à ordonnancer les opérations dans chaque atelier et de coordonner par la suite la totalité.

II.4.2.2 Les heuristiques de construction

Les heuristiques de construction sont des méthodes basées sur le principe itératif dans la recherche des solutions. Ce principe consiste à introduire des stratégies simples de décisions qui permettent de construire une solution proche de l'optimale dans un temps de calcul raisonnable (Meziane, 2018). Dans ce contexte, nombreuses règles ont été proposées en utilisant différentes méthodes ou en tenant compte de certaines situations pratiques et la plupart d'entre elles ont obtenu des résultats relativement bons dans des scénarios expérimentaux (Kuhpfahl, 2015).

En revanche, les méthodes les plus couramment utilisées de ce type, sont les algorithmes de liste. Ces algorithmes de recherche se base sur le principe de trier la liste des opérations à exécuter selon une technique de décision appelée règle de priorité telle que la règle SPT, EDD FIFO, LIFO, etc. (Azem, 2010) .

II.4.2.3. Les règles de priorité (dispatching rules)

Ces méthodes permettent de choisir parmi une liste de tâches, en attente d'exécution devant une machine, la prochaine tâche qui va être exécutée. Ce choix est basé essentiellement sur des critères qui dépendent uniquement des tâches (critères locaux) ou qui prennent en compte tout ou une partie de l'état du système (critères globaux) (Hassam, 2012). La recherche sur les règles de priorité ne cesse de continuer depuis plusieurs décennies jusqu'à ce jour. De nombreuses règles ont été élaborées et développées dans la littérature, Parmi les règles les plus utilisées on trouve :

- La règle SPT (Shortest Processing Time) : Pour cette règle le choix est basé sur l'ordre croissant de durées opératoires des tâches ;

- La règle LPT (Longest Processing Time) : Au contraire à la précédente, cette technique donne la priorité aux tâches ayant des durées plus élevées ;

- La règle EDD (Earliest Due Date) : Selon cette règle, lorsqu'une machine devient disponible, la tâche dont la date d'échéance est la plus proche, sera sélectionné pour être traitée ensuite.
- La règle FIFO (First In First Out) : l'ordre de passage selon cette règle est défini par l'ordre de disponibilité, la première tâche disponible est la première à être exécutée.

- La règle LIFO (Last In Last Out) : L'utilisation de cette règle consiste à classer les tâches qui viennent en dernier dans la première position.

Les règles de priorité sont des règles qui déterminent les priorités de passage entre les travaux en attente de traitement. Chaque fois qu'une machine devient disponible, une règle de priorité examine les travaux en attente et sélectionne celui avec la plus haute priorité (selon le critère choisi). Ces règles ont montré leurs efficacités dans plusieurs situations de problème d'ordonnancement, surtouts concernant la minimisation des temps de calcul. Cependant, les résultats fournis par ces méthodes restent dans la plupart du temps, non optimales, pour cela des améliorations de ces techniques ont été élaborées afin de se rapprocher beaucoup plus à la solution exacte. En effet, des approches qui combinent, par agrégation, différentes règles de priorité ont été développées comme la concaténation de l'algorithme de Johnson (Johnson, 1954) avec la règle FAM (First Available Machine) pour résoudre le problème Flow-Shop hybride à deux étages. Récemment, d'autre approches d'extension de ces règles ont été proposé afin de résoudre des problèmes d'ordonnancement avec d'autres objectifs dont nous citons par exemple le travail de Xiong et al (Xiong, Fan, Jiang, & Li, 2017) qui examine une analyse basée sur la simulation des règles de répartition pour l'ordonnancement job shop dynamique avec livraison par lots en tenant compte de la contrainte de priorité technique étendue.

Selon la nature des règles de priorité et les paramètres pris en compte par ces dernières, nombreuses classifications ont été définies. Hassam, dans sa thèse (Hassam, 2012), a présenté un ensemble de classifications basant sur différents critères telles que la classification basée sur la portée des règles (locales ou globales), la classification basée sur la complexité des règles (simples ou composées) et la classification basée sur des paramètres pris en compte par la règle (le temps opératoire, les délais, le coût…).

II.4.3 Les méta-heuristiques

Les méta-heuristiques sont des algorithmes d'optimisation qui représentent un excellent moyen pour fournir des solutions efficaces et réalisables dans des délais acceptables avec peu d'informations sur l'espace de recherche. Elles sont devenues parmi les techniques les plus étudiées ces dernières années. Plusieurs ouvrages récents ont été établis couvrant les notions de ces méthodes notamment les deux livre (Bozorg-Haddad, Solgi, & Loáiciga, 2017) et (Gendreau, 2019).

En effet, les méta-heuristiques sont des stratégies de recherche itératives de haut niveau, destinées à l'exploitation de l'espace de solutions par l'utilisation de différentes techniques. Ces

méthodes n'essayent pas d'examiner toutes les solutions possibles, mais de trouver dans un temps raisonnable une solution satisfaisante par investigation intelligente de l'espace en s'appuyant sur deux principes ; la diversification et l'intensification (Blum & Roli, 2003). La diversification désigne l'exploration de l'espace de recherche pour générer des solutions différentes sur le problème à optimiser à partir d'une solution générée aléatoirement ou bien choisie judicieusement. Quant à l'intensification, elle désigne l'exploitation de l'expérience de recherche accumulée, ce qui conduit le processus de recherche à se concentrer sur les zones jugées déjà encourageantes.

Malgré que les deux principes diversification et intensification caractérisent toutes les méthodes méta-heuristiques, chaque méta-heuristique possède sa propre stratégie de recherche. En effet, plusieurs classifications ont été donc proposées dans la littérature. Parmi ces classifications, nous trouvons la classification la plus répondue qui classifie les méta-heuristiques selon le nombre de solutions de départ.

II.4.3.1 Méta-heuristique à une solution

Les méta-heuristiques de ce type sont basés sur la notion de voisinage qui représente un ensemble de solutions obtenues à partir d'une solution donnée en effectuant un certain nombre de transformations.

a. Le Recuit Simulé

Le recuit simulé est une méthode d'optimisation inspirée d'un processus utilisé en métallurgie. Ce processus consiste à chauffer un métal à une température très élevée, puis à le refroidir très lentement, cela donne une certaine solidification au métal en utilisant une quantité d'énergie minimale. En effet, la méthode du recuit simulé adapte ce processus pour la résolution des problèmes d'optimisation dont l'état du métal représente une solution donnée, et son équilibre thermodynamique est la valeur de la fonction objectif de cette solution. Le passage d'un état du métal à un autre correspond au passage d'une solution à une solution voisine.

Le principe de cette méthode repose sur une procédure itérative qui commence par une température initiale élevée T et une solution initiale qui peut être choisie aléatoirement et qui génère à chaque itération une autre solution par une modification élémentaire de manière aléatoire de la solution courante, ce qui entraîne une variation de l'énergie (ΔE) du système donc de la fonction objectif.

b. La recherche tabou

La méthode de recherche tabou, théorisée par Glover (Glover, 1986), est une technique développée et destinée à la résolution des problèmes d'optimisation discrets et qui peut être utiliser même pour les problèmes continues. Son principe est inspiré de fonctionnement de la mémoire humaine. Avec un processus itératif, cette méthode utilise la notion de mémoire dans la politique d'exploration de voisinage.

Le principe de base de la méthode tabou est de partir d'une solution initiale qui peut être construite par une heuristique ou générée aléatoirement, de converger vers la meilleure solution en exécutant à chaque itération un mouvement dans l'espace de recherche remplaçant à chaque itération la solution courante par la meilleure solution trouvée dans son voisinage. Cependant, une mémoire est utilisée au cours de la recherche afin de conserver les informations sur les solutions déjà visitées.

II.4.3.2 Méta-heuristique à plusieurs solutions (à base de population)

Les méta-heuristique à plusieurs solutions ou à base de population sont des méthodes itératives qui, contrairement aux méthodes précédentes, manipulent un groupe de solutions réalisable (dite population) à chaque itération du processus de recherche au lieu d'une seule solution. Nous pouvons trouver dans cette catégorie les méthodes les plus utilisées : Les algorithmes génétiques et les algorithmes par essaim particulaire.

a. Les algorithmes génétiques

Les algorithmes génétiques sont des algorithmes évolutionnaires proposé dans les années 1975 par Holland (Holland, 1992). Ils s'inspirent leur principe du comportement d'évolution biologique des espèces. Ce principe de l'évolution de l'organisme repose sur une série d'améliorations afin que l'espèce s'adapte au mieux dans son environnement. L'objectif de ces algorithmes est d'optimiser une fonction objectif appelée "fitness" en travaillant sur un ensemble de solutions admissibles, appelé "population" qui va être évoluée par la suite. Chaque solution de la population est représentée par un individu ou chromosome. Ce dernier est constitué d'un ensemble d'éléments, appelés "gènes", qui peuvent prendre plusieurs valeurs, appelées "allèles".

La population de départ ou initiale est, généralement, choisie soit aléatoirement, soit par des heuristiques ou de mélange de solutions aléatoires et heuristiques. Cette population doit être diversifiée pour que l'algorithme ne tombe pas dans un optimum local où plusieurs individus générés sont semblables. Les algorithmes génétiques génèrent de nouveaux individus, de telle

sorte qu'ils soient plus performants que leurs prédécesseurs. Le processus d'amélioration des individus s'effectue par utilisation d'opérateurs génétiques, qui sont la sélection, le croisement et la mutation (Chaari, 2011).

Les algorithmes génétiques sont des algorithmes itératifs. En commençant par la génération d'une population initiale des individus. Une méthode de sélection sera effectuée par la suite pour sélectionnons les individus en calculant leurs fitness. Les individus sélectionnés seront manipulés par un opérateur de croisement qui les choisit selon une probabilité de croisement. Leurs résultats peuvent être mutés par un opérateur de mutation avec une probabilité de mutation donnée. Les deux opérations de croisement et mutation permettent de générer des nouveaux individus pour une nouvelle population et qui ont la forte probabilité d'être plus forts que ceux de la génération précédente. Les individus générés dans la phase de recombinaison seront insérés dans la nouvelle population après avoir évaluer leurs valeurs de la fonction objectif. L'arrêt des itérations est conditionné par un critère donné.

Les étapes principales d'un algorithme génétique sont :

- Codage et création de la population initiale : C'est la première étape de l'algorithme qui consiste à transformer les données réelles du problème traité en une représentation (code) utilisée par l'algorithme génétique, selon cette représentation un ensemble d'individus sont ensuite générés pour créer la première population (population initiale). La génération de ces individus s'effectuée soit aléatoirement soit par une heuristique spécifique. Ces deux opérations sont très importantes et très critiques car, le type de codage choisi influe d'une manière significative sur l'exécution des étapes suivante notamment le croisement et la mutation, ainsi que sur l'augmentation du temps nécessaire à la réalisation de ces deux dernières, ce qui influe directement sur le bon déroulement de l'algorithme génétique et de sa convergence vers la bonne solution. Plusieurs méthodes sont utilisées pour identifier le codage des individus, on distingue : le codage binaire, le codage réel, codage de permutation, etc. (Laribi, 2018).

- Evaluation et sélection des individus : Cette étape consiste à mesurer les performances des individus de la population afin de sélectionner les plus forts c'est-à-dire, les individus qu'ont la capacité de survivre. Cette capacité est liée à une valeur de poids (fitness)qui peut prendre différents critères selon le problème étudié. Les chromosomes sélectionnés sont eux qui vont être ajuster et modifier dans la phase de croisement et mutation.

- Croisement : Le croisement est la phase de diversification de la population initiale. Elle consiste à assurer la reproduction des individus pour créer des nouvelles solutions, en ajustant sur les composantes des chromosomes. A partir de deux individus parents, le croisement permet de générer un ou deux autres individus enfants avec une probabilité donnée appelée probabilité de croisement. Cette étape de croisement s'effectuer par plusieurs opérateurs qui déterminent le nombre des gènes à croiser et la manière dont cette opération sera réalisée. Plusieurs opérateurs de croisement sont proposés dans la littérature, les plus connues sont : le croisement par un seul point et le croisement par deux points, (Nearchou, 2004) (Kebabla, 2008).

- Mutation : La mutation assure l'aspect aléatoire à l'exploration de l'espace. Alors afin d'éviter que l'algorithme génétique converge vers des optimums locaux, une opération de mutation est très nécessaire. Cette tâche consiste à perturber les chromosomes générés en effectuant des changements au niveau de certains gènes. Le changement des gènes ou la mutation se réalise avec une probabilité (généralement faible) s'appelle taux de mutation. Dans ce cadre, nombreuses méthodes ont été développées telles que : Mutation par échange, mutation par insertion, mutation par inversion, etc.
- Insertion : à chaque fois que les étapes précédentes ont été achevées, dans une itération, les nouveaux individus doivent être insérés dans la population précédente pour formuler une nouvelle population. Cette opération suit généralement l'une des deux stratégies : la première est le changement de la totalité des individus de la population par les nouveaux individus ou la deuxième stratégie qui consiste à remplacer uniquement quelques-uns.
- Critère d'arrêt : Le critère d'arrêt c'est un factor qui détermine quand le processus génétique arrête d'évoluer. Ce critère peut être de nature diverse comme ; la valeur du temps de calcul, le nombre des itérations effectuées, la valeur de fitness, etc.

b. Les algorithmes par essaim particulaire

Ce type de métaheuristiques sont des techniques d'optimisation à population. Ces algorithmes sont destinés à la résolution des problèmes d'optimisation complexes. Ils sont développés par James Kennedy et Russell en 1995 (Eberhart & Kennedy, 1995). Le principe de base de ces techniques est inespéré de la manière de voyages collectifs de certains animaux évoluant en essaim, tels que les bancs de poissons et les oiseaux migrateurs, etc. La spécificité du comportement de ces essaims est la collaboration et l'interactions entre individus au sein du groupe pour atteindre un objectif donné. D'une manière pareille, et contrairement aux

algorithmes génétiques où l'évolution est basée sur l'élimination des individus faibles, les algorithmes par essaim particulaire se basent sur la coopération entre les individus pour avoir les évoluer.

Le processus des algorithmes d'essaims particulaires qui s'inspire du comportement des essaimes comme les oiseaux, consiste à initialiser une population initiale (*essaim*) de solutions aléatoires appelé *particules*. Chaque solution ou particule possède une distance par rapport à la solution qui peut être évaluée afin de mettre à jour les meilleures positions connues. Avec une vitesse bien déterminé cette particule se déplace dans l'espace de recherche d'une manière itérative et en suivant la position qui représente la meilleure position réalisée par la particule elle-même (best local) et celle trouvée par toutes ses voisins (best global).

II.5 Conclusion

Dans ce chapitre nous avons aborder la notion des problèmes d'ordonnancement dans les systèmes flexibles de production, nous avons commencé par donner des définitions générales sur l'ordonnancement et ces caractéristiques. Ensuite nous avons présenté les différents types de problème d'ordonnancement d'atelier dont nous avons détaillé le problème de job shop par la suite. Les méthodes de résolution des problème d'ordonnancement qui se divisent en trois grandes catégories ; exactes, heuristiques et méta-heuristique ont été présentées à la fin de ce chapitre.

Chapitre III : Ordonnancement job shop avec ressources consommables.

III.1 Introduction

Les problèmes d'ordonnancement dans les ateliers flexibles de type job shop constituent sûrement pour les gestionnaires des entreprises une des difficultés les plus importantes dans leurs systèmes de pilotage de la production. En effet, c'est à ce niveau que les caractéristiques réelles multiples et complexes des ateliers doivent être prises en compte.

Plusieurs travaux ont été réalisés afin de résoudre ce type de problème dont la plupart de ces travaux considèrent que l'exécution des jobs ne nécessite que des ressources renouvelables (Machines, outils ...) mais en réalité plusieurs types de ressources non renouvelables (consommables) peuvent être utilisés dans la réalisation d'un tel ou tel job comme les composants, l'énergie, les ressources financières, etc.

Dans ce chapitre nous nous intéressant au problème d'ordonnancement job shop avec contrainte de disponibilité de ressources non renouvelables. En effet, un modèle mathématique a été développé pour modéliser le problème et une métaheuristique basée sur des algorithmes génétiques et plusieurs heuristiques ont été utilisées pour résoudre ce problème dont l'objectif est de minimiser le temps d'exécution maximal (Makespan). Ces heuristiques sont proposées à la base des règles de priorité comme la règle SPT, LPT, etc. Pour évaluer la performance des méthodes utilisée, plusieurs expériences avec des tailles différentes ont été faites. Les résultats obtenus sont ensuite comparés avec des résultats obtenus en utilisant méthode exacte basée sur la programmation mathématique sur des problèmes de taille réduite.

III.2 Présentation du modèle FMS étudié

Le modèle FMS étudié est représenté par le système flexible de fabrication iCIM 3000 situé dans le Laboratoire de productique (MELT) à l'Université de Tlemcen. L'iCIM 3000 est l'une des dernières solutions pertinentes proposées par Festo Didactic pour la formation des étudiants et aussi pour répondre aux besoins scientifiques des centres de recherche intéressés par la fabrication. Il est censé jouer un rôle important dans l'illustration de sujets complexes tels que la logistique de production et la planification des séquences dans les systèmes de fabrication flexibles (Delgado Sobrino, Košťál 2, & Vavruška, 2014). Ce système, illustré dans la figure III.1, est composé de machines et d'équipements tels que :

1) Système de stockage / déstockage automatique (AS/RS), c'est le stock principal de toutes les matières utilisées pendant la production dans le système iCIM, c'est-à-dire les matières premières, les produits semi-finis ainsi que les produits finis,

2) Système de convoyage (Convoyeur à Palettes), qui est utilisé pour transférer les produits du stock vers les stations et / ou inversement,

3) Machine de fraisage 105 avec robot d'alimentation flexible, sur laquelle des trous de différents diamètres peuvent être réalisés.

4) Machine de tours 105 avec robot d'alimentation flexible responsable de la production des pièces rotatives spécifiques.

5) Station de qualité et de manutention (QH 200), qui est responsable des tests des pièces et de l'alimentation manuelle du système en palettes.

6) Cellules d'assemblage robotisées flexibles (FAC), la fonction de cette station est d'assembler des produits à partir de semi-produits créés par les machines CNC ou d'autres produits stockés dans la station AS/RS.

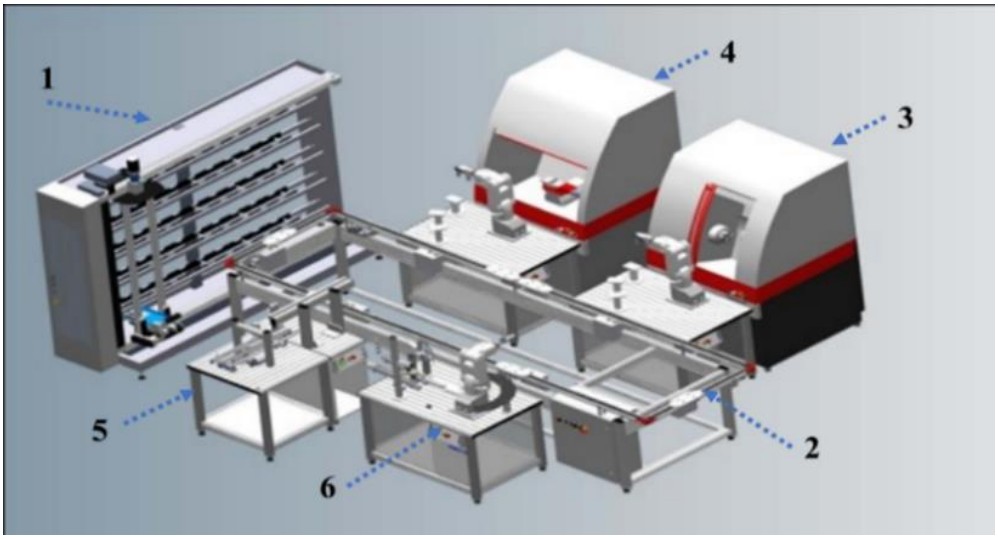
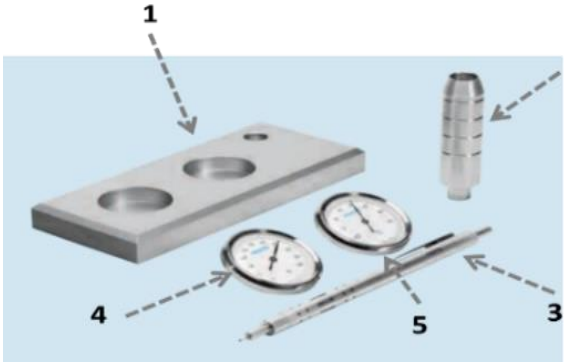


Figure III.1. La configuration 3D du système iCIM 3000 **(Delgado Sobrino & Košťál, 2013)**

Le produit fini réalisé et monté par le système iCIM 3000 est un ensemble de bureau (figure III.2). Ce produit se compose de cinq parties : plaque de base, porte-stylo, stylo, thermomètre utilisé pour la mesure de la température, et enfin un hygromètre utilisé pour la mesure de l'humidité.



(1): Plaque de base
(2): Porte-stylo
(3): Stylo
(4): Thermomètre

Figure III.2. Le produit fini réalisé par iCIM 3000

En fonction du nombre et du type de composants à assembler ainsi que du nombre de trous dans la plaque de base, le système permet de produire plusieurs variantes de ce produit. Chaque type de produit a un flux de matériaux spécifique qui contient une structure typique des opérations à effectuer. Le tableau III.1 montre un exemple de gammes d'opérations spécifiques pour la réalisation de trois produits différents ; A, B et C.

Tableau III.1. Exemple de routages de trois produits

Produit	***Déscription***	***Routage***
A	Plaque de base avec porte-stylo, stylo et thermomètre.	Milling + Testing + Assembly
B	Plaque de base avec porte-stylo et stylo.	Tour + Assembly
C	Plaque de base avec thermomètre et hygromètre.	Milling + Tour +Testing + Assembly

Le problème étudié ici est représenté comme un problème d'ordonnancement de job shop avec des contraintes de disponibilité de ressources non renouvelables. Dans ce problème, on suppose qu'un ensemble de *n* jobs (pièces) $N=\{J_1, J_1, \dots . J_n\}$ doit être exécuté, selon sa propre gamme de fabrication, sur les quatre machines (stations) $M=\{M_1, M_1, M_3, M_4\}$ dont O_j^k représente l'opération du job J_j qui sera exécuté sur la machine M_k pendant un temps de traitement P_{jk}. La machine M_4 représente la station d'assemblage sur laquelle chaque job consomme une quantité $Qdem_j^c$ de composants de type *c*, qui sont stockés dans un stock tampon à capacité limitée. L'arrivée des composants dans ce stock est supposée être périodique, c'est-à-dire qu'ils arrivent en même quantité à chaque unité de temps.

L'objectif est de minimiser le temps d'exécution maximum (makepan). Selon la notation établie par (Graham, Lawler, Lenstra, & Kan, 1979), le problème peut être défini comme suit : $J4/NR/C_{max}$. Le problème que nous traitons ici, repose sur un ensemble des hypothèses qui sont les suivantes :

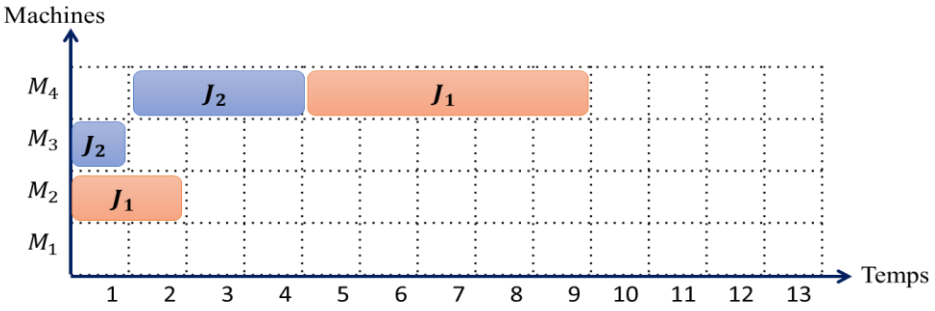
- Chaque job a son propre acheminement, qui est connu avant le début de la production.
- Le temps de traitement des pièces est connu au début et il comprend le temps de changement d'outil, le temps de transport des pièces, le temps de réglage et les temps d'usinage.
- Chaque machine peut traiter, sans interruption, une seule pièce à la fois.
- Le problème de blocage n'est pas pris en compte, c'est-à-dire que chaque job sera déplacé directement et sans retard pour être traité dans la machine suivante (si celle-ci est disponible) une fois qu'il est traité dans la machine précédente.
- Le traitement des pièces ne peut être effectué que sur une seule machine à la fois.
- Pour qu'un job commence son traitement sur la machine d'assemblage il faut que toute la quantité de ressources qu'il va consommer soit disponible dans le stock de la machine.

Le problème se pose lorsque on veut connaître le meilleur enchaînement des jobs qui donne le minimum de Makespan tout en respectant la disponibilité des ressources à consommer. Pour bien expliquer ce problème nous prenons l'exemple illustratif suivant : Nous considérons deux jobs représentés par les données du tableau III.2. Dans cet exemple, les deux jobs J_1 et J_2 nécessitent, respectivement, cinq et trois unités de même ressource, pour qui soient traités. L'arrivée de cette ressource est limitée par une seule entité chaque instant donné.

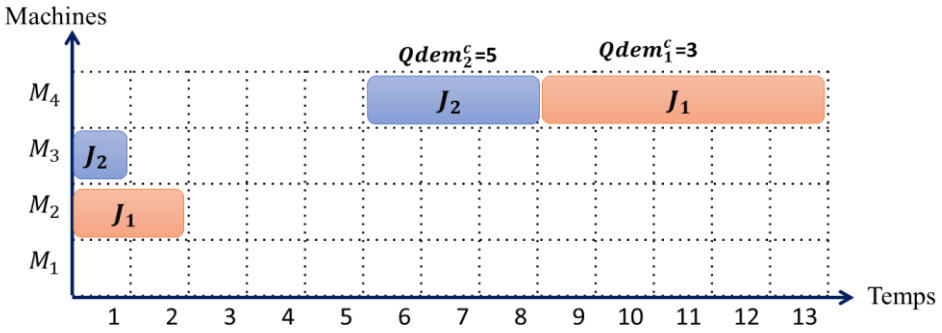
Tableau III.2. Exemple d'un problème de deux jobs

Jobs	***Machine (temps de traitement)***		$Qdem_j^c$
J_1	$M_2(2)$	$M_4(5)$	3
J_2	$M_3(1)$	$M_4(3)$	5

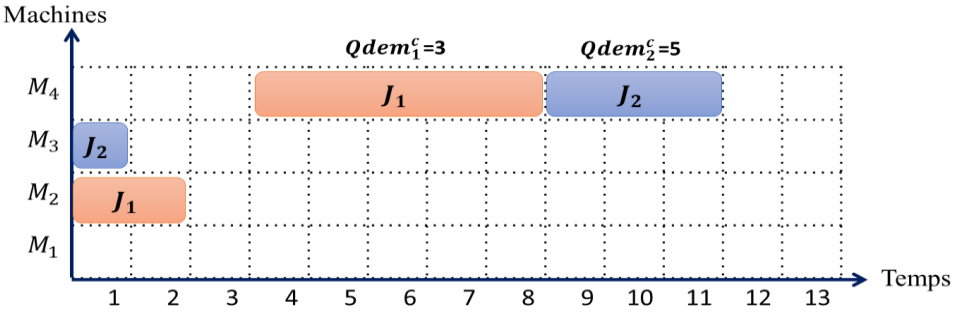
La solution optimale du problème considéré dans cet exemple, sous sa forme classique c'est-à-dire sans ressources consommables, est donnée par l'ordonnancement de la figure III.3(a). Ce séquencement des jobs, ne donne pas nécessairement une solution optimale lorsque l'on considère la contrainte de disponibilité des ressources consommables (CR), ce qui est bien noté dans les figures III.3 (b et c) où l'ordre optimal est totalement différent. Cet exemple montre que, dans le cas où la contrainte des ressources consommables est considérée dans un problème classique JSP, alors il ne suffit pas de trouver un ordre optimal des jobs pour le problème sans contrainte de ressources et de le conserver comme solution optimale pour le problème avec ressource. Ce qui confond ce qu'a été déclaré dans (Toker, Kondakci, & Erkip, 1994).



(a) Solution optimale sans ressources consommables



(b) Solution non optimale avec ressources consommables



(c) Solution optimale avec ressources consommables

Figure III.3. Solutions obtenues avec et sans ressources consommables

III.3 État de l'art

Depuis plusieurs années, nombreux travaux sont focalisés sur la résolution de problème d'ordonnancement job shop dans un contexte de ressources non consommables. Dans (Zhou, Cherkassky, Baldwin, & Olson, 1991), les auteurs ont proposé l'utilisation de réseaux neuronaux comme méthode de résolution. Ils ont montré comment simplifier ce problème NP-complet en un simple réseau neuronal. Le travail effectué dans (Caumond A. , 2006) était également orienté vers le même problème, mais en tenant compte de plusieurs types de contraintes comme les décalages temporels, le transport et d'autres contraintes supplémentaires.

Récemment, des séries d'ouvrages ont abordé le problème de job shop dans sa version classique avec ou sans contraintes. En fait, ce problème reste à ce jour parmi les préoccupations

réelles des chercheurs compte tenu de son importance cruciale. Par exemple, Dabaha et al (Dabaha, Bendjoudi, AitZaib, El-Bazc, & Taboudjemat, 2018) ont présenté une nouvelle approche basée sur la méthode Branch-and-Bound pour résoudre le problème du job shop avec blocage. En outre, Nagata et al (Nagata & Ono, 2018) visent à améliorer le makespan en proposant une méthode basée sur la recherche locale. Gong et al (Gong, Deng, Chiong, Gong, & Huang, 2019) ont proposé un algorithme mimétique efficace (EMA) pour résoudre le problème d'ordonnancement multi-objectifs de l'atelier job shop. Un algorithme méta-heuristique a été proposé dans (Deng, et al., 2020) pour résoudre le problème de job shop avec la contrainte sans attente.

La contrainte des ressources non renouvelables représente un nouveau défi pour les chercheurs dans le domaine de l'ordonnancement. Dans l'ordonnancement des projets, (Muritiba, Rodrigues, & da Costa, 2018) propose un algorithme de liaison de chemin pour le problème d'ordonnancement des projets à ressources limitées en mode multiple. Les auteurs cherchent à obtenir le meilleur calendrier qui minimise la durée totale du projet avec l'existence de ressources renouvelables et non renouvelables. Dans (Tabrizi, Ghaderi, & Haji-Yakhchali, 2019) et pour le même objectif (minimiser l'achèvement du projet), un modèle de programmation mixte est développé pour présenter la planification simultanée de l'ordonnancement du projet et du problème d'approvisionnement en matériel.

Certaines études ont porté sur les problèmes de machines uniques avec des ressources consommables. Toker et al (Toker, Kondakci, & Erkip, 1991) étudient un problème d'ordonnancement à une seule machine avec une seule ressource financière. Les auteurs montrent que ce problème est équivalent à un atelier de flow shop composé de deux machines sans ressources financières. Ils ont appliqué l'algorithme de Johnson pour minimiser le Makespan.

Belkaid et al. (Belkaid, Yalaoui, & Sari, 2016) (Belkaid, Yalaoui, & Sari, 2016), dans leurs travaux, ont traité le problème de l'ordonnancement de la production limité par les ressources consommables dans un environnement de machines parallèles.

Le premier travail dans le domaine de l'ordonnancement job shop introduisant la contrainte des ressources consommables est le travail effectué dans (Grabowski & Janiak, 1987). Dans ce travail, le temps de traitement de chaque opération est représenté par une fonction de la quantité requise de la ressource non renouvelable. Afin de résoudre ce problème, les auteurs ont proposé un algorithme basé sur la théorie des graphes disjonctifs ainsi qu'une technique de branchement

et de liaison. Dans le même contexte, Toker et al (Toker, Kondakci, & Erkip, 1994), ont développé un algorithme approximatif et ont introduit deux bornes inférieures pour résoudre le problème de *n* jobs sur *m* machines avec des besoins en ressources non renouvelables.

L'énergie est un autre type de ressources non renouvelables qui est pris en compte dans plusieurs travaux récents sur les problèmes d'ordonnancement job shop. En fait, dans la plupart des travaux, et contrairement à notre travail, la consommation d'énergie n'a pas été considérée comme une contrainte, mais plutôt comme un objectif à optimiser. Dans ce contexte, nous pouvons citer le travail de Salido et al (Salido, Escamilla, Barber, Giret, & Tang, 2016) dans le quel les auteurs ont traité ce problème en s'appuyant sur trois objectifs : l'efficacité énergétique, la robustesse et le makespan.

Kemmoé et al (Kemmoé, Lamy, & Tchernev, 2015) ont étudié le problème du job shop avec la contrainte d'un seuil de consommation d'énergie à ne pas dépasser. Un modèle linéaire a été proposé pour résoudre simultanément la restriction de puissance et les objectifs temporels traditionnels du problème de job shop. Quant à Gondran et al (Gondran, Kemmoé-Tchomté, Lamy, & Tchernev, Février, 2016), ils ont traité un problème de job shop sous la contrainte des pics de consommation d'énergie. Ils ont proposé d'utiliser une méthode métaheuristique pour trouver une solution de front de Pareto basée sur la quantité d'énergie instantanée nécessaire à l'ordonnancement ainsi que sur la durée totale (makepan). Dans le travail de Meng et al (Meng, Zhang, Shao, & Ren, 2019), les auteurs ont étudié un problème de job shop flexible avec l'objectif de minimiser la consommation totale d'énergie. Dans ce travail, un nouveau modèle de programmation linéaire en nombres entiers mixtes (MILP) est proposé. Ce modèle est basé sur deux variables essentielles : la variable temps de disponibilité et la variable énergie d'inactivité.

En parlant de méta-heuristique, les algorithmes génétiques ont été largement utilisés, grâce à leurs efficacités, pour résoudre les problèmes PN-difficiles. Par exemple, Seidgar et al (Seidgar, Rad, & Shafaei, 2017) ont proposé un algorithme génétique et une nouvelle évolution auto-adaptée pour résoudre les problèmes de l'atelier de flow shop d'assemblage en deux étapes en tenant compte de la panne des machines. Dans les travaux de Rashid et al (Rashid, Arani, Hoseini, & Omran, 2018), un algorithme génétique a été proposé pour obtenir des solutions acceptables au problème de localisation du détaillant avec une demande stochastique et une prise en compte du temps de service. Kumar et al (Kumar, Kumar, & Sharma, 2019) ont traité le problème du Flow Shop (FSP) en s'appuyant sur trois contraintes : l'intervalle de panne, le temps de transport et le poids des jobs. Les auteurs ont utilisé de nouveaux paramètres dans leur algorithme afin de résoudre le FSP avec ces trois contraintes.

Les règles priorités sont également appliquées dans de nombreux domaines de la production et de la logistique afin de résoudre les problèmes d'ordonnancement, en particulier dans l'environnement des ateliers de job shop (Đurasević & Jakobović, 2018). Deux nouvelles heuristiques, appelées NEHMSWG et NEHMS-PS, ont été proposées par Rajendran et al (Rajendran, Rajendran, & Leisten, 2017) afin de résoudre le problème de l'ordonnancement flow-shop de permutation en atelier. Les performances de ces heuristiques ont été étudiées et les résultats obtenus révèlent que les règles de priorité proposées sont simples et efficaces, et améliorent les solutions pour de nombreux cas de problèmes. Dans (Ozturk, Bahadir, & Teymourifar, 2019), les auteurs ont décrit deux nouvelles approches pour extraire les règles de priorité pour les problèmes d'ordonnancement dynamique multi-objectifs en atelier job shop en utilisant la simulation et la programmation de l'expression des gènes.

Cette revue de la littérature nous a permis de constater que peu de travaux se concentrent sur les problèmes d'ordonnancement de job shop en présence de ressources consommables. Cette partie de travail, en revanche, met l'accent sur cet aspect en examinant la disponibilité des ressources consommables dans un JSP. En effet, nous proposons, pour résoudre le problème mentionné ci-dessus, un modèle mathématique, une méta-heuristique basée sur les algorithmes génétiques et quatre heuristiques (basées sur des règles de priorité). Rappelons que l'objectif est de minimiser le temps d'exécution maximum (makepan).

III.4 Le modèle mathématique

La modélisation mathématique plus précisément la programmation linière, est l'un des outils les plus puissants et les plus utilisés dans la résolution des problèmes d'ordonnancement, notamment les problèmes de job shop. Depuis des plusieurs années jusqu'à ce jour, déverses travaux en ordonnancement job shop, ont été basés sur un modèle mathématique pour représenter et résoudre leurs problèmes étudiés nous citons par exemple des travaux récents (Wang, Yang, & Wang, 2019), (Pei, Zhang, Zheng, & Wan, 2020), et (Ambrogio, Guido, Palaia, & Filice, 2020). Le modèle mathématique est un moyen très important et très efficace, il a comme objectif, soit de l'utilisé pour résoudre d'une manière exacte les problèmes d'ordonnancement de taille industriels, soit pour établir une série de règles de priorités et de bornes inférieures pour les différentes instances considérées. En réalité, trois types de modélisation sont généralement utilisés pour modéliser un problème d'ordonnancement job shop. Ces trois catégories se distinguent particulièrement par le type de variables de décision utilisé (Unlu & Mason, 2010). Une étude comparative très intéressante a été faite entre ces trois

types de modélisation par Ku and al. (Ku & Beck, 2016). Les auteurs ont utilisé trois solveurs récents afin de comparer les performances de leurs modèles sur le problème d'ordonnancement job shop.

- Modélisation par positions : Cette modélisation consiste à utiliser des variables binaires pour représenter des positions sur les machines. En effet, à chaque machine, un ou plusieurs positions sont associées (selon le nombre des opérations à effectuer). A chaque position associée à une machine, une opération peut être affectée. Cependant, l'arrangement de ces positions détermine l'ordre par lequel les opérations vont être exécuter.

- Modélisation basée sur le temps : Dans ce type de modélisation, les variables de décision utilisées sont des variables qui affectent les jobs à des périodes de temps données. Ce type de variables a été introduit en ordonnancement pour la première fois par Sousa et Wolsey (Sousa & Wolsey, 1992).

- Modélisation disjonctive : La modélisation disjonctive ou la modélisation basée sur la contrainte de de précédence consiste à identifier l'ordre linéaire des jobs qui peut être considéré comme l'ensemble de toutes les permutations (solutions). Las variables de décisions dans ce cas sont des variables binaires qui sont généralement égale à un si un job donné succède à un autre emploi.

Pour construire notre modèle mathématique, nous avons choisi la programmation linière en nombre entier mixte avec une modélisation par positions, c'est-à-dire que les variables que nous avons choisi d'être utilisées sont des variables entières mixtes pour représenter le temps et les quantités de ressources consommables, et des variables binaires pour représenter les positions sur machines. L'objectif de ce modèle est premièrement, de modéliser le problème d'ordonnancement job shop avec la contrainte de ressources consommables et deuxièmement de trouver des solutions exactes qui seront utilisées ensuite pour évaluer les performances des méthodes heuristiques proposées.

III.4.1 Les données

Nous présentons dans cette section l'ensemble des paramètres et des indices que nous avons retenu pour modéliser notre problème.

III.4.1.1 Les indices

i :	Indice de l'opération
j :	Indice de job
k :	Indice de la machine
p :	Indice de la position
c :	Indice de la ressource consommable
V :	Un nombre très grand
J:	Ensemble de *n* job
M:	Ensemble de *m* machine
C :	Ensemble de *r* types of resources
$T=[1...T_{max}]$:	L'intervalle de temps représentant le temps d'arrivé de ressources consommable

III.4.1.2 Les paramètres (les données)

P_{jk} :	Processing time de job *j* sur la machine *k*
nb_j :	Nombres d'opérations de job *j*
np_m :	Nombre de positions de la machine *m*
$Qdem_j^c$:	Quantité de la ressource *c* demandée par le job *j*
$Qarr_t^c$:	Quantité de la ressource *c* arrivée dans le temps *t*
$Tarr_t^c$:	Le temps d'arrivé de la ressource *c*
Y_{jk}^i =	$\begin{cases} 1 & \text{Si l'opération } i \text{ de job } j \text{ doit être exécutée sur la machine } k \\ 0 & \text{Sinon} \end{cases}$

III.4.2 Les variables de décision

Comme il était définit, les variables de décision utilisées dans notre modèle sont de deux types :

III.3.2.1. Les variables entières mixtes

T_{jk} :	Le début de traitement de job *j* sur la machine *k*
F_k^p :	La fin de la position *p* de la machine *k*
Qst_k^{pc}:	La quantité totale demandée de la ressource *c* jusqu'à la fin de la position *p* de la machine *k*
C_{max} :	Le makespan

III.3.2.2. Les variables binaires de positions

$$X_{jk}^{p} = \begin{cases} 1 & \text{Si le job j est affecté à la position p de la machine k} \\ 0 & \text{Sinon} \end{cases}$$

$$Z_{k}^{pt} = \begin{cases} 1 & \text{Si } F_{k}^{p} \geq Tarr_{t}^{c} \\ 0 & \text{Sinon} \end{cases}$$

III.4.3 La fonction objectif et les contraintes

III.4.3.1 La fonction objectif

L'objectif visé dans cette partie du travail est la minimisation du temps total de traitement de toutes les pièces (jobs) dans le système c'est-à-dire le C_{max} (le Makespan). Alors la fonction objectif à satisfaire est présenté comme suit :

$$Z = \text{Min } (C_{max}) \tag{1}$$

III.3.3.2. Les contraintes

Les contraintes qui déterminent les limites de notre problème sont les suivantes

$$\sum_{j=1}^{n} X_{jk}^{p} = 1 \qquad \forall\, k \in M, \quad \forall\, p = 1 \dots np_k \tag{2}$$

$$\sum_{p=1}^{np_k} X_{jk}^{p} = 1 \qquad \forall\, k \in M, \quad \forall\, j \in J \tag{3}$$

$$F_{k}^{1} \geq \sum_{j=1}^{n} P_{jk} * X_{jk}^{1} \qquad \forall\, k \in M \tag{4}$$

$$F_{k}^{p} \geq F_{k}^{p-1} + \sum_{j=1}^{n} P_{jk} * X_{jk}^{p} \qquad \forall\, k \in M, \quad \forall\, p = 2 \dots np_k \tag{5}$$

$$\sum_{k=1}^{m} \left(Y_{jk}^{i} * F_{k}^{p}\right) + \sum_{k=1}^{m} \left(Y_{jk}^{i+1} * P_{jk}\right) \leq \sum_{k=1}^{m} \left(Y_{jk}^{i+1} * F_{k}^{p'}\right) + V * \left(1 - \sum_{k=1}^{m} \left(Y_{jk}^{i} * X_{jk}^{p}\right)\right) + V * \left(1 - \sum_{k=1}^{m} \left(Y_{jk}^{i+1} * X_{jk}^{p'}\right)\right) \tag{6}$$

$$\forall\, j \in J, \forall\, i = 1 \dots (nb_j - 1), \forall\, p, p' = 1 \dots np_k$$

$$Qst_k^{pc} = \sum_{j=1}^{n} \left(X_{jk}^{P} * Qdem_j^c \right) \qquad \forall\, k \in M, \quad \forall\, p = 1 \dots np_k, \quad \forall\, c \in C \qquad (7)$$

$$\sum_{s=1}^{p} Qst_k^{sc} \leq \sum_{t=1}^{tmax} Z_k^{pt} * Qarr_t^c \qquad \forall\, k \in M, \quad \forall\, p = 1 \dots np_k, \quad \forall\, c \in C \;\; \forall\, t \in T \qquad (8)$$

$$F_k^p - \left(Tarr_t^c + \sum_{j=1}^{n} \left(P_{jk} * X_{jk}^p \right) \right) \geq V * \left(Z_k^{pt} - 1 \right) \qquad \forall\, k \in M, \quad \forall\, p = 1 \dots np_k, \quad \forall\, c \in C \;\; \forall\, t \in T \qquad (9)$$

$$C_{max} \geq F_k^{np_k} \qquad \forall\, k \in M \qquad (10)$$

La contrainte **(2)** détermine qu'une position ne doit être occupée que par un seul job. Cela signifie qu'une machine ne peut exécuter qu'une seule opération à la fois. La contrainte **(3)** garantit qu'une opération du job j est affectée à une seule et unique position dans une seule machine à la fois. La contrainte **(4)** et **(5)** assurent qu'il n'y a pas de chevauchement entre les positions dans la même machine. La contrainte **(6)** permet de s'assurer qu'aucune opération ne commence le traitement avant que l'opération précédente du même job ne soit terminée. La contrainte **(7)** calcule le montant total des ressources demandées par un jobs affecté à chaque position. Cette quantité de ressources doit être inférieure ou égale à la quantité arrivée au début de ces positions ce qui est définie par la contrainte **(8)**. La contrainte **(9)** fait le lien entre le moment de l'arrivée des ressources et la fin des postes. Enfin, le Makespen est calculé par la contrainte **(10).**

III.5 Les méthodes de résolution utilisées

Afin de résoudre le problème étudié, nous avons adapté une méta-heuristique basée sur des algorithmes génétiques ainsi que quatre heuristiques basées sur des règles de priorité. Pour bien expliquer ces méthodes et notre choix concernant les paramètres et les critères fixées dans

chaque méthode, nous avons choisi un exemple illustratif défini par un problème de quatre jobs à exécuter sur les quatre machines.

Le tableau III.3 présente le temps nécessaire de traitement des jobs sur les machines ainsi que la quantité de ressources demandées par chacun de ces jobs. L'arrivée des ressources suit une courbe d'escalier qui est déterminé par l'arrivé d'une ressource ($Qarr_t^c$=1) toutes les cinq unités de temps ($Tarr_t^c$=5).

Tableau III.3. Temps et quantités de ressources nécessaires pour le traitement des jobs

Jobs	***Machine (temps de traitement)***				$\boldsymbol{Qdem_j^c}$
J_1	$M_3(2)$	$M_1(3)$	$M_2(5)$	$M_4(25)$	10
J_2	$M_1(15)$	$M_2(10)$	$M_3(20)$	$M_4(8)$	8
J_3	$M_3(7)$	$M_2(14)$	$M_1(5)$	$M_4(5)$	4
J_4	$M_2(9)$	$M_1(4)$	$M_3(3)$	-	-

III.5.1 Les heuristiques

III.5.1.1 L'heuristique Job Shop Short Processing Time (JSSPT)

Pour l'heuristique JSSPT, on suppose que le séquencement des jobs sur les machines se fait en fonction de l'ordre croissant de la somme des temps de traitement de chaque job. Dans notre exemple, nous constatons que les valeurs de la somme des temps de traitement des jobs J_1, J_2, J_3 et J_4 sont respectivement de 35, 53, 31 et 16. Ensuite, selon l'heuristique JSSPT, la solution du problème doit être définie par l'ordre mentionné dans le tableau suivant :

Tableau III.4. L'ordre des jobs selon l'heuristique JSSPT

Ordre sur machines	***Machines***			
	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$
1	J_4	J_4	J_4	J_3
2	J_3	J_3	J_3	J_1
3	J_1	J_1	J_1	J_2
4	J_2	J_2	J_2	-

Le tableau III.4 montre l'ordre d'exécution des travaux sur les machines. Cet ordre indique que les machines doivent d'abord traiter le job J_4, puis J_3, ensuite J_1 et enfin J_2. Ainsi, le C_{max} calculé à partir de ce séquencement est égale à **118** unités de temps.

III.5.1.2 L'heuristique Job Shop Longest Processing Time (JSLPT)

Contrairement à la précédente heuristique, l'heuristique JSLPT exige que le séquencement des jobs sur chaque machine se fasse dans l'ordre décroissant de la somme du temps de traitement de chaque job. C'est-à-dire que l'exécution du job sur chaque machine doit être lancée par le job ayant la plus petite valeur de la somme des durées de ses opérations. L'application de cette heuristique au même exemple nous donne l'ordre indiqué dans le tableau III.5. Cela donne une valeur de C_{max} qui est égale à **120** unités de temps.

Tableau III.5. L'ordre des jobs selon l'heuristique JSLPT

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_2	J_2	J_2	J_2
2	J_1	J_1	J_1	J_1
3	J_3	J_3	J_3	J_3
4	J_4	J_4	J_4	-

III.5.1.3 L'heuristique Job Shop Short Accumulation Processing Time (JSSPTcum)

JSSPTcum est une heuristique qui consiste à ordonnancer les jobs dans l'ordre croissant de leurs temps de traitement cumulé sur chaque machine. Pour cette heuristique, lorsqu'une machine est libérée, le job choisi pour être traité en premier est celui qui a la valeur cumulée la plus faible. Cette valeur cumulée est calculée par la somme des durées de toutes les opérations précédentes (précédentes de l'opération à effectuer sur cette machine). Cette technique vise à placer au début de l'ordonnancement les jobs qui consomment le moins de temps d'exécution pour quitter la machine.

Le tableau III.6 indique les durées cumulées sur chaque machine (en fonction de l'ordre des opérations). On peut voir, par exemple, pour le job J_3 qui sera exécuté avec sa deuxième opération sur la machine M_2, que la valeur cumulée sur cette machine est égale à **21**, qui est en effet la durée de la première opération **(7)** plus la durée de la deuxième opération **(14).** Selon l'heuristique JSSPTcum, le séquencement des jobs pour notre exemple est déterminé par l'ordre mentionnée dans le tableau III.7. Le C_{max} obtenu dans ce cas est de **118** unités de temps.

Tableau III.6. Cumul des temps de traitement sur machines

Jobs	***Cumul des temps de traitement sur machines***			
	M_1	M_2	M_3	M_4
J_1	5	10	2	35
J_2	15	25	45	53
J_3	26	21	7	31
J_4	13	9	16	-

Tableau III.7. L'ordre des Jobs selon l'heuristique JSSPTcum

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_1	J_4	J_1	J_3
2	J_4	J_1	J_3	J_1
3	J_2	J_3	J_4	J_2
4	J_3	J_2	J_2	-

III.5.1.4 L'heuristique Job Shop Short Resources Consumption (JSSRC)

Le principe de l'heuristique du JSSRC est de programmer les jobs en fonction de l'ordre croissant de la quantité de ressources requises pour chaque job. Cette heuristique vise à placer les jobs qui consomment le moins de composants au début de l'ordonnancement. L'application de cette heuristique sur le même exemple donne l'ordre indiqué dans le tableau III.8. Le C_{max} obtenu dans ce cas est de **135** unités de temps.

Tableau III.8. L'ordre des Jobs selon l'heuristique JSSRC

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_4	J_4	J_4	J_3
2	J_3	J_3	J_3	J_2
3	J_2	J_2	J_2	J_1
4	J_1	J_1	J_1	-

III.5.2 Méthode basée sur les algorithmes génétiques

Ce type de méta-heuristique est une technique de recherche proposée par la Hollande en 1975 (Holland, 1992). Les méta-heuristiques servent à résoudre un large éventail de problèmes d'optimisation difficiles dans un délai raisonnable, pour lesquels on ne connaît pas de méthodes traditionnelles plus efficaces. Dans un algorithme génétique, une représentation doit être choisie pour les individus d'une population. Par conséquent, chaque solution est stockée dans un chromosome artificiel représenté par un code.

En fait, lorsque l'on utilise des algorithmes génétiques, aucune information n'est nécessaire pour résoudre les problèmes. Il suffit de fournir une fonction de codage de chaque solution sous forme de gènes, et de fournir une fonction pour évaluer la pertinence de la solution obtenue pour ce problème. Pour notre problème, nous avons développé un algorithme génétique qui a la structure suivante :

III.5.2.1 Codage et population initiale

a) Le codage

La sélection d'un format de chaîne pour les individus est une étape très importante pour une mise en œuvre réussie d'un algorithme génétique. C'est pourquoi il est nécessaire de trouver un codage en même temps efficace et simple des individus, qui satisfasse toutes les contraintes du problème. En effet, pour notre problème nous avons proposé d'utiliser une représentation basée sur le séquencement des jobs sur les machines. Cela signifie que chaque individu de la population est illustré par une matrice $N \times M$ où les colonnes représentent les machines et les lignes représentent l'ordre d'exécution des jobs sur chaque machine. L'exemple du tableau III.9 présente un chromosome de quatre jobs et quatre machines (4x4).

Tableau III.9. Présentation du Chromosome

	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$
1	J_4	J_4	J_3	J_2
2	J_1	J_2	J_2	J_1
3	J_2	J_3	J_4	J_3
4	J_3	J_1	J_1	-

Dans cet exemple, il est remarqué que le premier job qui sera traité sur la machine $\boldsymbol{M_1}$ est le job J_4 suivi par le job J_1, puis le job J_2 et enfin le job J_3 de même manière pour les autres machines.

b) La population initiale

La population initiale est l'ensemble des chromosomes (des solutions admissibles du problème) à partir lesquels l'opération de reproduction va être commencé. Pour établir cette population de départ, nous avons choisi la méthode classique de génération, qui consiste à créer aléatoirement l'ensemble des chromosomes. En effet, cette tâche est assurée à travers l'affectation aléatoire des jobs qui vont être réalisés sur une machine, aux positions de ce dernière.

III.5.2.2 Sélection et Fonction d'aptitude (Fitness)

Le mécanisme de sélection consiste à choisir parmi les chromosomes de la population actuelle les chromosomes les plus forts pour générer la population suivante. L'évaluation des individus parents se réalise à la base de leurs valeurs d'aptitude « fitness ». Pour les problèmes d'ordonnancement, cette valeur est, généralement, représentée par la valeur de la fonction objectif. Citant par exemple (Hosseinabadi, Vahidi, Saemi, Sangaiah, & Elhoseny, 2019). Comme la fonction objective de notre problème est la minimisation du temps d'exécution total, les meilleures solutions sont donc celles qui aboutissent au C_{max} le plus bas.

III.5.2.3. Croisement

Dans cette étape, nous avons choisi la méthode d'un seul point pour faire le croisement entre de chromosomes sélectionnés aléatoirement. Le croisement dans ce cas s'effectuer par le changement de deux colonnes de la matrice (individu parents) pour produire deux nouveaux chromosomes enfants, comme le montre la figure III.4. Ce choix qui est basé sur deux colonnes a pour objectif de :

- Augmenter la probabilité de générer deux chromosomes totalement différents aux anciens, car si nous choisissons plus ou moins deux machines, il est fort possible de tomber dans les mêmes chromosomes pères.
- Augmenter la chance que les solutions générées soient admissibles, parce que si nous choisissons des lignes au lieu des colonnes, la probabilité que les chromosomes générés présentent des solutions inadmissibles est très élevée.

III.5.2.4 Mutation

La mutation dans la méthode de l'algorithme génétique est effectuée pour maintenir la diversité de la population par les perturbations causées dans une solution. Dans notre problème, la mutation n'a pas été prise en compte.

III.5.2.5 Critère d'arrêt

Le critère d'arrêt est l'indicateur selon lequel le processus de recherche doit être s'arrêté. Le critère choisi ici se réfère au nombre d'itérations effectuées.

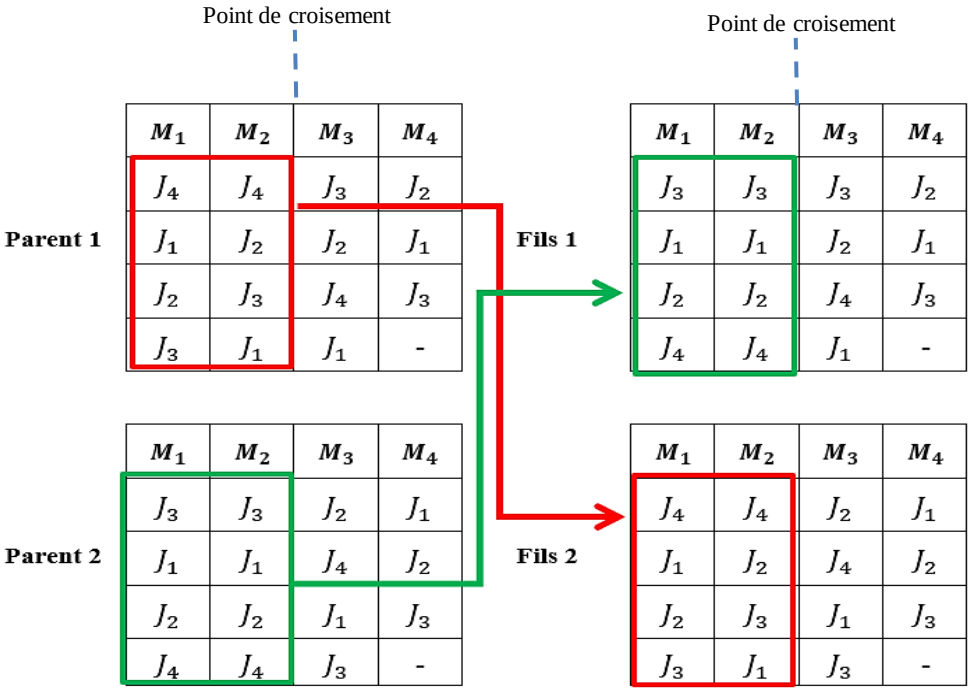


Figure III.4. Manière de croisement

III.6 Résultats expérimentaux

Pour évaluer les performances de l'AG adaptée et les quatre heuristiques proposées précédemment, nous les avons appliqué sur plusieurs exemples de petites, moyennes et grandes tailles. De ce fait, nous avons créé un programme sur Matlab permettant de calculer le makespan (C_{max}) et de trouver la meilleure solution. Le nombre des exemples retenus est fixé à dix exemples différents pour chaque taille du problème. Cependant, les données sont générées de manière aléatoire par un sous-programme qui définit aléatoirement les temps de traitement des jobs, leurs quantités de ressources consommables requises, la quantité et le temps d'arrivée des ressources consommables, etc.

En revanche, deux critères sont pris en compte afin d'examiner l'efficacité des approches proposées. Le premier est le rapport de déviation relative en pourcentage (RE) qui est défini par l'équation (11). Ce rapport détermine le pourcentage de rapprochement de la solution (valeur de makespan) trouvée par une approche donnée [$C_{max}(app)$] à la meilleure solution trouvée par toutes les approches [$C_{max}(best)$]. Le deuxième critère est le temps de calcul CPU, qui

indique le temps écoulé pendant l'exécution de certains algorithmes. Ces deux critères sont utilisés pour tester l'efficacité des approches proposées et pour trouver la meilleure qui donne des bonnes solutions dans un temps raisonnable.

$$\mathrm{RE} = \frac{C_{max}(app) - C_{max}(best)}{C_{max}(app)} \times 100 \qquad (11)$$

III.6.1 Résultats obtenus pour les petites instances

Cette première partie de test consiste d'un côté, à examiner et prouver l'efficacité de l'algorithme génétique et les heuristiques proposées et d'un autre côté, d'évaluer les limites du modèle mathématique proposé pour le critère du makespan. De ce fait, nous avons étudié trois problèmes de petites tailles (2x4, 3x4 et 4x4) à travers lesquels nous avons effectué une comparaissant entre les résultats obtenus par le programme Linéaire en Nombres Entiers qui représente la méthode exacte (EX), les quatre heuristiques de ***dispatching ruls*** et l'algorithme génétique. Notant ici que la méthode exacte (EX) a été programmée et simulée avec le solveur CPLEX, tandis que les autres approches ont été simulées en utilisant la programmation sous le logiciel Matlab. Les temps de traitement des jobs sont générés aléatoirement entre 1 et 30 unités de temps.

Les résultats trouvés par l'application de ces méthodes sur les trois instances, sont présentés dans les histogrammes de la figure III.5 et la figure III.6. Ces histogrammes montrent successivement, les moyennes de RE et les moyennes de CPU qui sont calculés à travers dix exemples différents de chaque taille de problème. A travers ces résultats, nous pouvons remarquer facilement que les solutions fournies par l'heuristique ***JSSPTcum*** sont les très proches de l'optimum, celui-ci est défini par les pourcentages inférieurs de RE calculé pour les trois instances. Ces valeurs proches de l'optimum démontrent très bien l'efficacité de cette heuristique par rapport aux autres méthodes. En outre, la deuxième méthode qui vient après l'heuristique ***JSSPTcum*** et qui est plus proche aussi de l'optimum est la méthode de l'algorithme génétique.

En termes de temps de calcul, nous remarquons que les heuristiques basées sur les règles de priorité ont montré leurs efficacités et leurs rapidités par rapport à l'AG. De plus la méthode exacte est incapable de résoudre le problème 4*4 dans un temps raisonnable, ce qui démontre la complexité du problème étudié

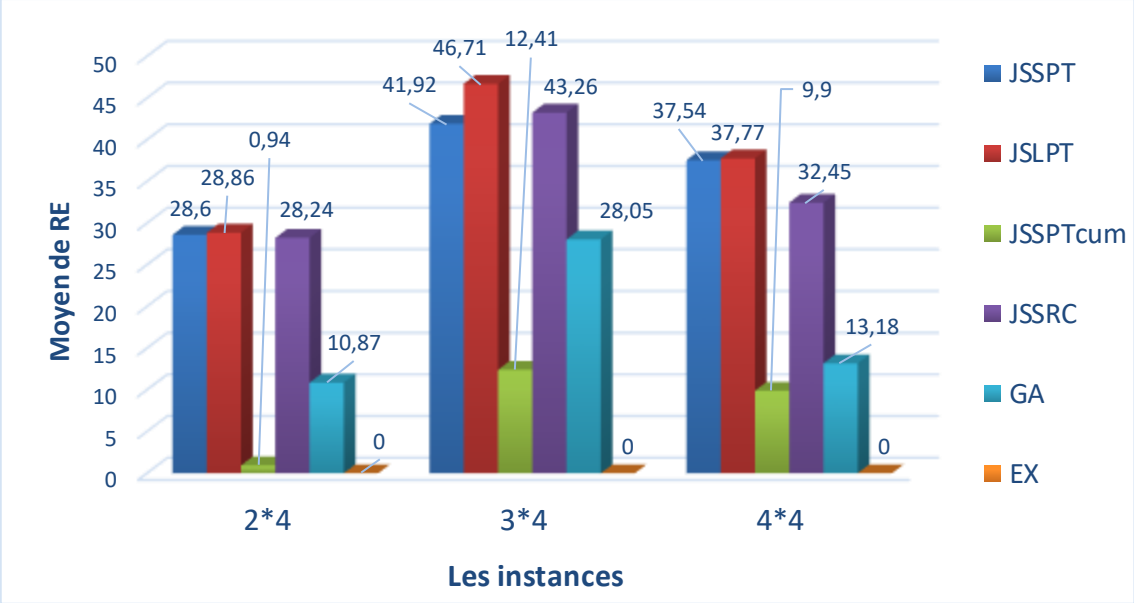


Figure III.5. Les moyennes du rapport RE calculé pour les petites instances

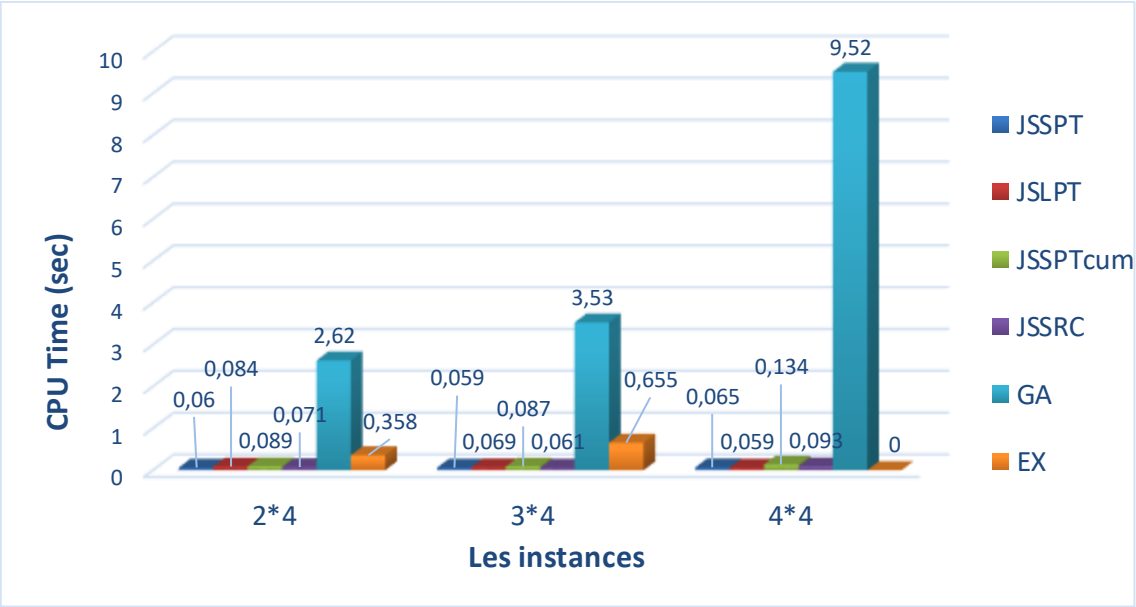


Figure III.6. Les moyennes de temps CPU obtenus pour les petites instances

III.6.2 Résultats obtenus pour les moyennes instances

Dans cette partie, l'expertise a été faite sur les problèmes de 8, 10 et 15 jobs. La variation du temps de traitement des opérations dans ce cas, suit une loi uniforme entre 1 et 50. Le test des méthodes proposées qui est effectué sur dix exemples donne les résultats mentionnés dans l'histogramme de la figure III.7. A travers ces résultats, on remarque que l'heuristique **JSSPTcum**, qui est basée sur le choix de la petite valeur du temps de traitement cumulé, donne toujours les meilleures solutions par rapport aux autres solutions trouvées. D'autre part, les valeurs du temps CPU présentées dans la figure III.8 démontrent également l'efficience de

JSSPTcum et des autres heuristiques par rapport à l'Algorithme Génétique. En outre, la méthode exacte dans cette partie de l'expertise reste incapable de trouver une solution dans un délai raisonnable, ce qui soutient également l'utilisation d'autres approches.

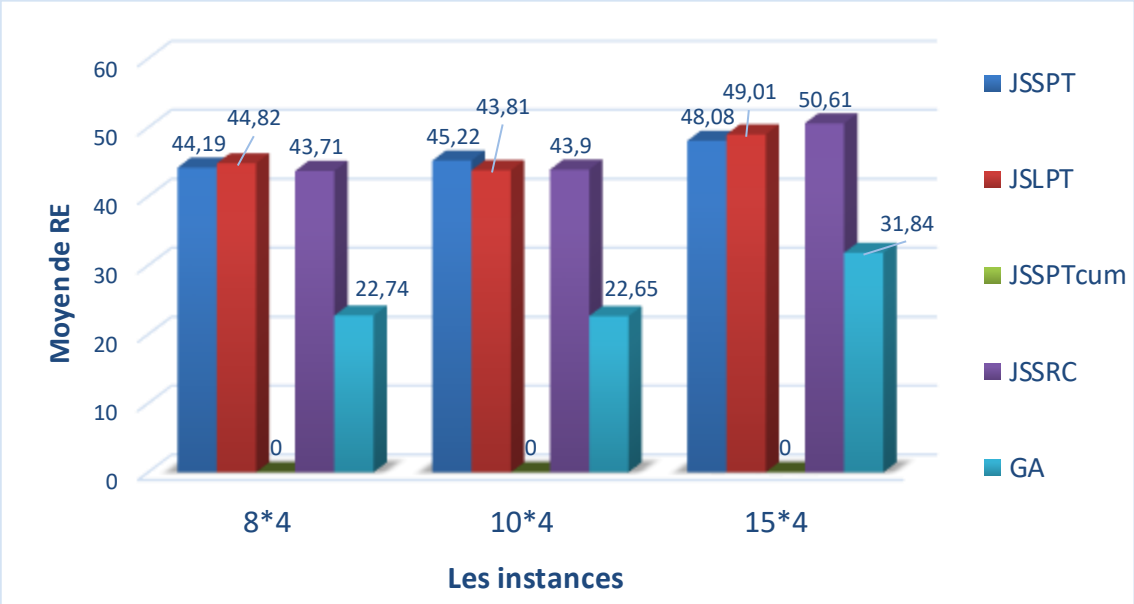


Figure III.7. Les moyennes du rapport RE calculé pour les moyennes instances

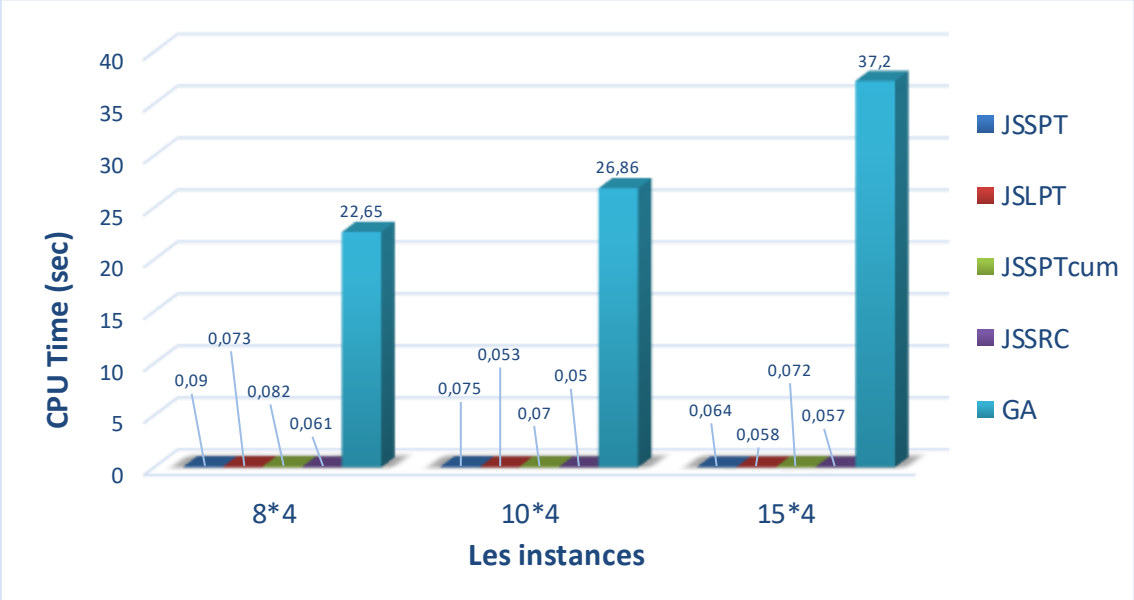


Figure III.8. Les moyennes de temps CPU obtenus pour les moyenne instances

III.6.3 Résultats obtenus pour les grandes instances

Les problèmes sélectionnés dans cette partie sont les problèmes de taille 20, 50 et 100 jobs qui doivent être réalisés sur les quatre machines. Comme dans la partie précédente, le temps de

traitement suit une loi uniforme entre 1 et 50. Selon la figure III.9, nous constatons que les meilleures solutions sont aussi obtenues par l'application de l'heuristique JSSPTcum, qui valide la performance de cette méthode.

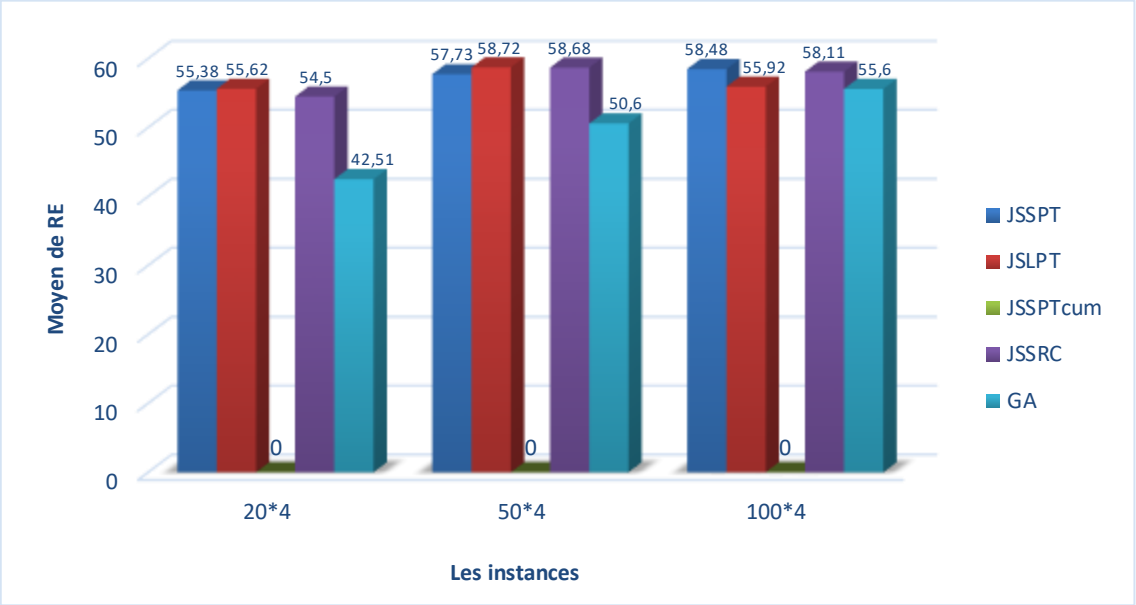


Figure III.9. Les moyennes du rapport RE calculé pour les grandes instances

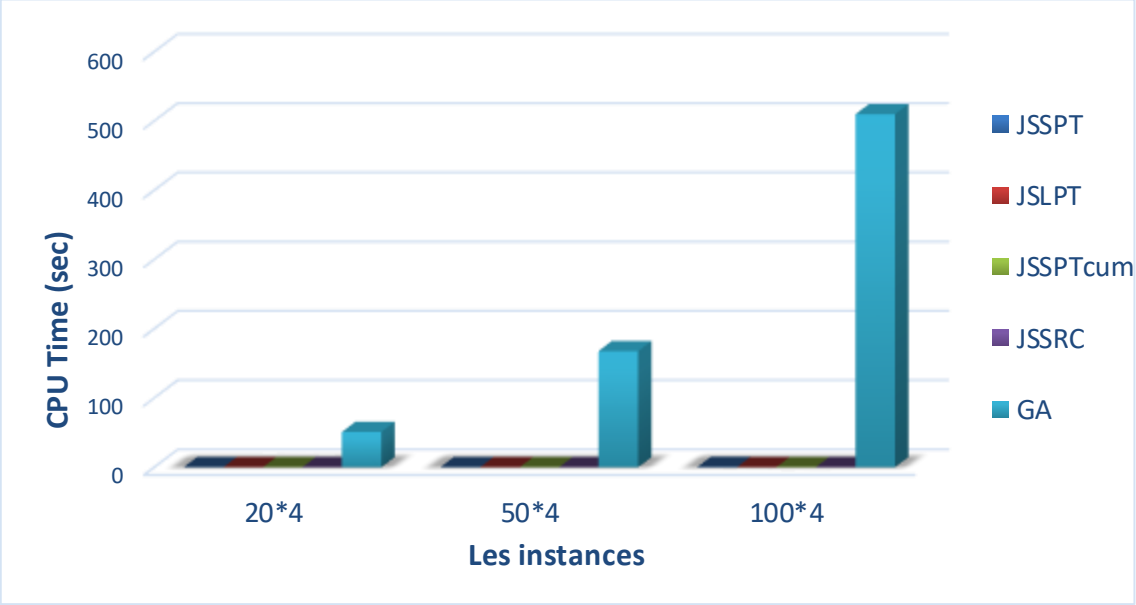


Figure III.10. Les moyennes de temps CPU obtenus pour les grandes instances

Là encore, on peut noter que la valeur du RE la plus proche de la meilleure valeur (valeur donnée par JSSPTcum) est donnée par l'AG dans les trois problèmes étudiés. Cela montre que l'AG est l'approche la plus efficace par rapport aux autres heuristiques en termes de fonction objectif.

III.7 Conclusion

Dans ce chapitre nous avons traité le problème d'ordonnancement job shop avec contrainte de ressources consommables. Ce problème se différencie au problème classique de job shop par le fait que le traitement des jobs sur les machines ne dépend pas uniquement de la disponibilité des machines mais aussi de l'existence des quantités suffisantes des ressources consommables requises. Pour résoudre ce problème nous avons développé tout d'abord un modèle mathématique par lequel nous avons pu examiner notre problème et trouver la solution exacte qui fait la base de l'évaluation de d'autre approches. Ensuite, nous avons proposé un algorithme génétique et un ensemble des heuristiques basées sur des règles de priorité, afin de trouver des solutions approchées au même problème.

L'étude effectuée dans ce chapitre et les résultats d'expertise sur plusieurs exemples de différentes instances nous a permis de montrer, d'un part, que les heuristiques et l'algorithme génétique proposées sont capables de résoudre des problèmes de grandes tailles dans un temps raisonnable toute en maintenant la qualité de la solution à un niveau acceptable. D'autre part, que l'heuristique JSSPTcum basée sur l'ordre croissant du cumule de temps de traitement est l'approche la plus performante concernant la minimisation du makespan dans le problème job shop avec des ressources non-renouvelables.

Le chapitre suivant est un prolongement des travaux conduits dans ce chapitre où nous allons rapprocher plus à la réalité par l'intégration de la contrainte de transport dans l'environnement de job shop avec des ressources consommables.

Chapitre IV : Ordonnancement job shop sous contrainte de transport unidirectionnel

IV.1. Introduction

Dans le chapitre précédent nous avons traité le problème job shop avec contrainte de ressources consommables dont nous avons considéré que chaque job termine son exécution sur une machine passe directement et sans retard à la machine suivante, c'est-à-dire qu'il n'y a aucune contrainte de transport et que le temps de déplacement des jobs est supposé intégré dans le temps de traitement des opérations. Mais en réalité, chaque job nécessite un certain temps de déplacement qui est indépendant du temps de traitement, pour arriver à la machine où il doit subi une opération de traitement.

Dans ce chapitre nous nous sommes intéressés au problème job shop avec deux contraintes ; contrainte de ressources consommables et contrainte de de transport. L'étude dans cette partie est également focalisée sur le système flexible iCIM 3000 qui est constitué de quatre stations (machines) de traitement de pièces, d'un système de stockage AS/RS et d'un système de transfert des pièces. En outre, le déplacement de ces jobs dans ce système, s'effectuer dans un sens unique c'est à dire que toutes les pièces suivent le même sens de déplacement (déplacement unidirectionnel). Cette contrainte est particulière car la plupart des travaux traitant le job shop avec contrainte de transport suppose souvent que les jobs peuvent être déplacés dans n'importe quel sens.

L'objectif visé dans ce chapitre, consiste à trouver les meilleurs séquencements des pièces (jobs) sur les quatre machines qui minimise le temps total d'achèvement des toutes les opérations (Makespen) en tenant compte en même temps le déplacement des pièces et la disponibilité des ressources consommables requise pour le traitement de chacune de ces pièces. Basant sur les résultats obtenus de l'étude du problème job shop avec contrainte de ressources consommables (traitée dans le chapitre précédent), qui dévoilent l'efficacité et la performance des heuristiques basés sur les règles de priorité, nous avons proposé six autres heuristiques de même caractère, afin de résoudre le problème JSP avec ces deux contraintes.

IV.2. Etat de l'art

L'introduction de la contrainte de transport dans l'étude de problèmes d'ordonnancement dans les systèmes FMS est l'une des préoccupations les plus importante qui permet aux gestionnaires des ateliers de rapprocher beaucoup plus à la réalité. En revanche cette considération rend le problème d'ordonnancement encore plus difficile et complexe (Caumond A. , 2006).

Dans ce cadre et afin de résoudre ce type de problème, plusieurs travaux ont été élaborés. Nouri et al (Nouri, Driss, & Ghédira, 2016) ont présenté un état de l'art bien détaillé basé sur sept critères tels que le nombre de ressources de transport, le type de ressources de transport, la complexité du travail, la flexibilité du routage, la contrainte de recirculation, les critères d'optimisation et les approches mises en œuvre.

Selon (Larabi, 2010), le problème d'ordonnancement dans les systèmes flexibles de production avec plusieurs transporteurs (AGVs) identiques est introduit pour la première fois en 1993 par Ulusoy et Bilge (Ulusoy & Bilge, 1992). Ce travail a été suivi par plusieurs d'autres travaux similaires. Nous pouvons citer par exemple (Bilge & Ulusoy, 1995) et (Lacomme, Larabi, & Tchernev, 2013).

Espinouse et al (Espinouse, Pawlak, & Sterna, 2017) traitent un problème d'ordonnancement dans un système flexible de production constitué d'une seule machine, un dépôt avec capacité illimitée et un véhicule auto guidé. Le véhicule qui ne peut transporter qu'une seule pièce à la fois, circule sans arrêt avec des cycles constants. L'objectif est de réduire au minimum le nombre de cycles de véhicules nécessaires pour transporter et exécuter tous les jobs dans le système. Ahmadi-Javid et al (Ahmadi-Javid & Hooshangi-Tabrizi, 2017) traitent un problème d'ordonnancement job shop qui intègre trois contraintes ; le calendrier des opérateurs, le temps de traitement des machines et le temps de transport avec un nombre fini de transporteurs où l'objectif est de minimiser le temps d'exécution total des travaux.

Les travaux de Li et al (Li, Shi, & Huang, 2018), sont focalisés sur un problème de job shop à trois machines dont les auteurs ont introduit le temps de transfère intermédiaire entre ces trois machines. L'objectif dans ce travail était de trouver une meilleure solution qui minimise le Makespan en proposant un nouveau modèle et en utilisant des heuristiques pour aboutir à cet objectif. Le travail de Gondran et al (Gondran, Huguet, Lacomme, Quilliot, & Tchernev, 2018) vise une nouvelle fonction objective du problème job shop avec transport, cet objectif réside dans la recherche d'une solution non semi-actives, mais qui minimise l'étendue du Makespan

et maximise ensuite la qualité de service grâce à une modélisation basée sur le décalage temporel et un processus itératif.

Un modèle basé sur la programmation linéaire à nombres entiers mixtes (MILP) a été proposé par Heger et al (Heger & Voss, 2018) afin de trouver une solution optimale du problème d'ordonnancement des AGVs dans un environnement job shop avec blocage. Les résultats obtenus par ce modèle ont été comparés par une approche de règle de priorité. Un autre modèle de programmation mathématique était proposé dans le papier (Homayouni & Fontes, 2019) dont le but est de programmer le problème d'ordonnancement conjointe de la production et du transport dans des systèmes de fabrication flexibles.

Vis-à-vis cette revue de la littérature, nous pouvons dire que le problème étudié ici est un problème de job shop avec contrainte de transport où le déplacement des pièces s'effectue par des palettes qui circulent dans un sens unidirectionnel. Notre contribution attribuée dans ce travail est la résolution du problème JSPT avec une contrainte additionnelle c'est la contrainte de ressources consommables.

IV.3. Description du problème et notations

IV.2.1. Description du problème

Notre étude est focalisée sur le système flexible de production iCIM 3000. Ce système a une configuration en boucle (la figure I.3). La spécification de cette configuration est que l'ensembles des machines sont reliées par un système de manutention de pièces sous forme circulaire (convoyeur FMF-Pallet pour l'iCIM300). Ce système est dédié à transférer les produits sur des palettes dans un sens unidirectionnel, c'est-à-dire que, quel que soit les gammes de fabrication des pièces, ces dernières doivent suivre un même sens de déplacement comme il est présenté dans la figure IV.1. En effet la contrainte du sens unidirectionnel de déplacement oblige certaines pièces de visiter plusieurs machines avant d'arriver à leurs destinations.

Dans le système iCIM3000, les stations ne peuvent traiter qu'une seule pièce à la fois et elles sont dotées, chacune, d'un stock de pièces considéré illimité. À côté de chaque station de traitement, il est installé un bras manipulateur qui sert à déplacer les pièces du tapie roulon vers le stock du station et vice versa. Les pièces déposées sur les stocks des stations doivent attendre la disponibilité de ces dernières afin d'être traités. L'ordre de traitement des pièces déposées dans le stock suit la règle de priorité FIFO (premier arrivé premier traité). De plus, nous

considérons que le temps de chargement, déchargement et d'attente d'une pièce sont tous inclus dans le temps de déplacement de cette pièce.

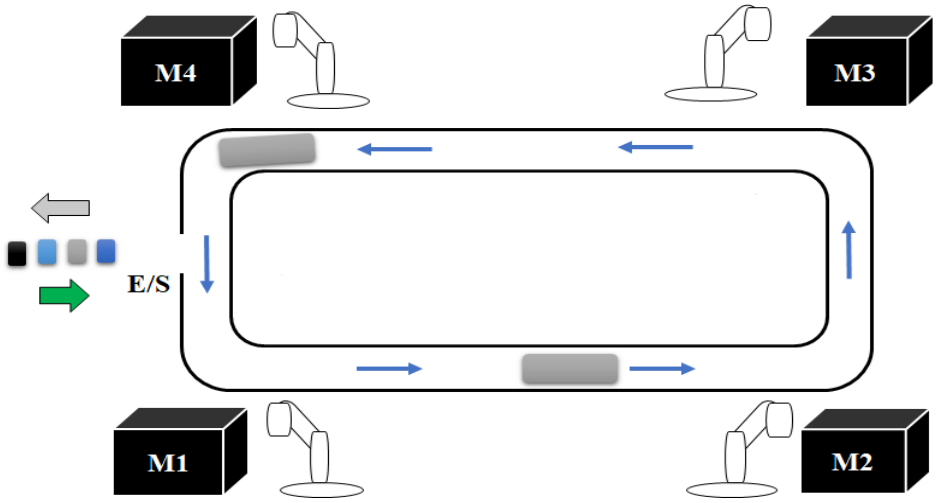


Figure IV.1. Sens de déplacement des palettes dans le système iCIM 3000

Le déplacement des pièces entre les stations s'effectuer sur des palettes standards à capacités unitaires, c'est à dire que ces palettes peuvent porter n'importe quel type de pièces mais avec une seule pièce à la fois. Les palettes sont toujours disponibles et se déplacent (à vide) d'une manière permanente sur le tapie roulon dans un sens unique avec une vitesse constante. Pour simplifier le problème nous considérons que le convoyeur peut supporter autan des palettes que des pièces à traiter. Cette supposition sert à éliminer le problème de blocage dû à la saturation du convoyeur.

IV.3.2. Notation

Le problème considéré ici est un problème d'ordonnancement job shop avec contrainte de ressources consommables et contrainte de transport avec plusieurs palettes. Ce problème consiste à ordonnancer n jobs $N=\{J_1, J_1, \ldots . J_n\}$ sur quatre machines $M=\{M_1, M_1, M_3, M_4\}$. Chaque job J_j est constitué de n_j opérations à réaliser dans un ordre bien déterminé. Chaque opération O_j^k du job J_j doit être réalisée sur la machine M_k sans préemption pendant un temps de traitement P_{jk}. Pour que l'opération O_j^k soit exécutée le job J_j doit être transporté vers la machine M_k avec un temps de déplacement noté T_j^k.

Le Tableau IV.1 présente le temps de déplacement (par unité de temps) des palettes entre les différentes stations. Dans ce tableau il est noté, par exemple, que le déplacement d'un job J_j , de l'entrée E vers la machine M_2, nécessite un temps $T_j^2 = 6.5$ unités de temps qui est, en effet,

égale au temps de déplacement de ce job de E vers M_1 avec 1.5 unité plus le temps de son déplacement de M_1 vers M_2 avec 5 unités de temps.

Tableau IV.1. Temps de transport des palettes entre stations

Destination		***Temps de displacement***
Départ	***Arrivée***	
E/S(entrée/sortie)	M1	1.5
M1	M2	5
M2	M3	3
M3	M4	5
M4	E/S(entrée/sortie)	1.5

IV.3.3 Fonction objectif

L'objectif visé dans cette partie est de trouver le meilleur ordonnancement qui minimise le temps total de traitement des pièces (jobs) tout en considérant les deux contraintes ; contrainte de transport et contrainte de ressources consommables. Alors, la fonction objectif (décrie par la formule 12) est la minimisation du temps total d'exécution C_{max} dont le C_j^k détermine la date de fin d'exécution du job J_j dans la machine M_k

$$\text{Min } (C_{max}) = \min \left[\max_j^k \left(C_j^k\right)\right] \qquad (12)$$

En effet, chaque job J_j à un instant donné et avant qu'il soit traité dans la machine M_k, peut être soit :

- (*i*) traité dans une machine $M_{k'}$ (machine précédente de M_k dans la gamme opératoire de J_j) ;
- (*ii*) transporté par une palette ;
- (*iii*) dans le buffer (stock) d'entrée de la machine M_k en attendant soit la disponibilité de la machine soit l'arrivée de la quantité requise des ressources consommables.

Alors, la date de fin d'exécution C_j^k est conditionnée par trois contraintes ; la disponibilité du job J_j (y compris la date de fin de son traitement sur la machine précédente et le temps de son déplacement), la disponibilité de la machine M_k et la disponibilité des ressources consommables nécessaires à la réalisation de ce job. De ce fait le C_j^k se calcule par l'équation suivante :

$$C_j^k = \max\left[D_j^k, D_k,\ DRC_j^k\right] + P_j^k \quad (13)$$

Avec :

$$D_j^k = (C_j^{k'} + T_j^k) \quad (14)$$

Dont :

$C_j^{k\prime}$: La fin de traitement du job J_j sur la machine précédente $M_{k\prime}$

D_k : La disponibilité de la machine M_k

D_j^k: La disponibilité du job J_j qui sera exécuté sur la machine M_k

DRC_j^k : La disponibilité des ressources consommables

Le but est la minimisation du temps de sortir du dernier job du système, il est donc, indispensable d'ajouter le temps de déplacement des pièces vers la sortie du système à la date de fin d'exécution sur la dernière machine, c'est-à-dire que pour le job J_j qui termine son exécution sur la machine M_g (la dernière machine selon sa gamme opératoire), la date de fin C_j^g de ce job devient $C_j^g + T_j^S$. La figure IV.2 présente un exemple d'une solution admissible pour le problème considéré avec 5 jobs et 4 machines. Cet exemple montre comment le temps de déplacement des jobs peut influencer sur la valeur de C_{max}.

En effet, pour ce séquencement, il est bien remarqué que la valeur de Makespan est définie par la fin de dernier déplacement du job J_5 qui sera le dernier job à quitter le système, malgré que le job J_4, en réalité, va terminer son traitement en dernier. la valeur de C_{max} dans ce cas égale à la valeur $C_5^1 + T_5^s$. En outre, il est bien clair que les machines seront disponibles directement dès la fin de traitement d'un job malgré que le job n'a pas terminer son déplacement, ce qui permet aux machines de recevoir un autre job pour le traiter.

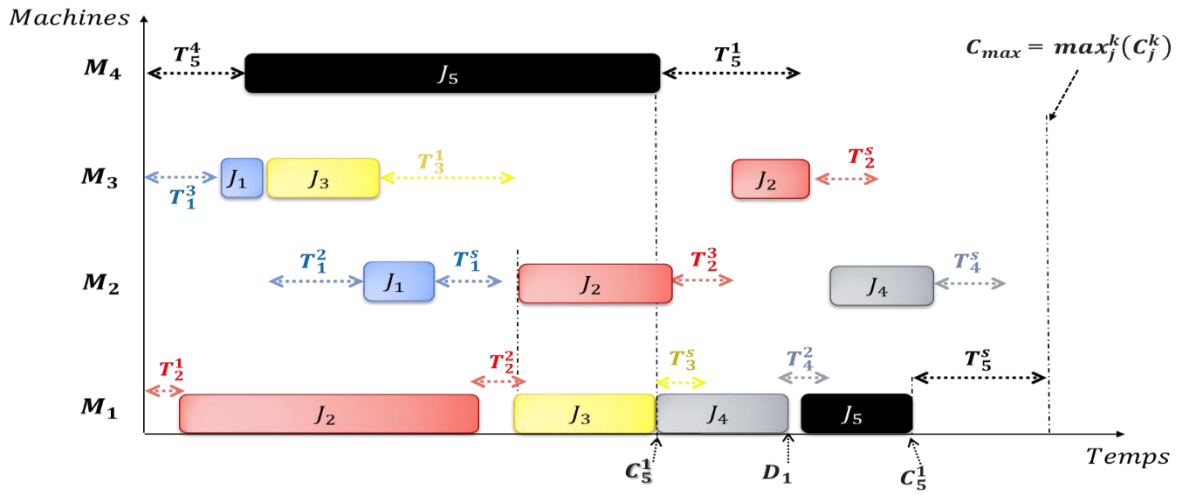


Figure IV.2. Exemple d'ordonnancement d'un problème 5x4

IV.3. Méthodes de résolution

La résolution du problème job shop avec contrainte de transport et de ressources consommables (JSPTRC) de manière optimale s'avère dans la plupart des cas très difficile à cause de son caractère fortement combinatoire. Les méthodes exactes dans ce cas requièrent un effort calculatoire qui croît exponentiellement avec la taille du problème. Alors, les méthodes approchées sont fortement préférées pour résoudre ce type de problèmes. Ces méthodes permettent de trouver des solutions acceptables dans des temps raisonnables.

Dans ce chapitre et afin de résoudre le problème JSPTRC, nous avons proposé d'étudier les performances de six autres heuristiques basées sur le même principe des méthodes développées dans le précédent. Ces heuristiques permettent de déterminer l'ordre de traitement des jobs sur les machines à la base des règles de priorité. L'objectif visé par cette étude est d'analysé l'influence de la contrainte de transport sur le choix de l'approche qui conduit à trouver les meilleures solutions. En outre, le critère d'évaluation que nous avons choisi pour déterminer le séquencement des jobs dans les six heuristiques proposées, repose essentiellement sur le rapport entre le temps de traitement et le temps de transport. Le choix de ce rapport permet de détermine, d'une part, la relation qui existe entre le temps écoulé pendant le traitement des pèces et le temps nécessaire à leurs déplacements, et d'autre part, quel est le degré d'influence du temps de transport de pièces sur l'ordre de traitement de ces dernières.

Pour bien présenter l'explication de l'ensemble des approches proposées, nous avons pris le même exemple illustratif étalés dans le chapitre précédent. Dans cet exemple il est défini un problème d'ordonnancement d'un ensemble de quatre jobs qui vont être exécutés sur les quatre machines. Chaque job suit sa propre gamme opératoire et nécessite (ou non) une quantité bien déterminée de ressources consommables. Les informations sur les gammes opératoires et les quantités de ressources requises sont exposées dans le tableau III.3.

En effet, la résolution du problème JSPTRC basant sur cet exemple, nécessite, dans une première étape, la détermination des temps de déplacement des jobs entre stations. Alors, d'après le tableau IV.1 qui définit le temps de transport entre les différentes stations et selon les gammes opératoires des jobs, nous pouvons déterminer la matrice des temps relatifs aux déplacements des jobs durant leurs trajectoires dans le système. Cette matrice de temps de déplacement est présentée dans le tableau IV.2.

Tableau IV.2. Temps nécessaire de déplacements des jobs entre stations

Jobs	*Destination*		*Temps de déplacement*
	Départ	*Arrivée*	
J_1	E/S	M_3	9.5
	M_3	M_1	8
	M_1	M_2	5
	M_2	M_4	8
	M_4	E/S	1.5
J_2	E/S	M_1	1.5
	M_1	M_2	5
	M_2	M_3	3
	M_3	M_4	5
	M_4	E/S	1.5
J_3	E/S	M_3	9.5
	M_3	M_2	13
	M_2	M_1	11
	M_1	M_4	13
	M_4	E/S	1.5
J_4	E/S	M_2	6.5
	M_2	M_1	11
	M_1	M_3	8
	M_3	E/S	6.5

IV.3.1. L'heuristique *JSSPTtran*

Pour cette heuristique, nous supposons que le séquencement des jobs sur les machines se fait en fonction de l'ordre croissant du rapport des temps Rtd_j. Ce rapport (décri par l'équation 14) définit une relation entre la quantité de temps allouée pour le de déplacement d'un job et le temps nécessaire au traitement de ce job.

$$Rtd_j = \frac{STT_j}{STD_j} = \frac{\sum P_j^k}{\sum T_j^k} \qquad (14)$$

Dans cette équation, STT_j détermine la somme des temps de traitement de toutes les opérations du job J_j et STD_j définit la somme de temps de déplacement de ce job entre les stations utilisées pour traiter ce dernier.

L'objectif de ce choix est de donner la priorité aux jobs qui ont la valeur du temps de déplacement important par rapport aux temps de traitement. En effet, cette décision a pour but de profiter du temps d'existence des jobs, possédant des petites valeurs de Rtd_j, sur le système de transport afin de traiter les jobs qui ont la valeur de Rtd_j plus élevée. Le tableau suivant montre les données relatives au rapport Rtd_j pour l'exemple choisi.

Tableau IV.3. Les rapports relatifs aux temps de traitement et de déplacements des jobs

Jobs	$\boldsymbol{STD_j}$	$\boldsymbol{STT_j}$	$\boldsymbol{Rtd_j}$
$\boldsymbol{J_1}$	32	35	1.5
$\boldsymbol{J_2}$	16	53	3.3
$\boldsymbol{J_3}$	48	31	0.6
$\boldsymbol{J_4}$	32	16	0.5

D'après le tableau IV.3 et selon l'heuristique *JSSPTtran*, il est bien clair que le job qui doit être traiter en premier dans le système est le job J_4 qui possède la valeur minimale de Rtd_j. Ce job est suivi par le job J_3 qui présente donc le deuxième job à traiter dans le système. Par la suite vient le job J_1 et en fin le job J_2. La solution sélectionnée pour notre problème est définie dans ce cas par l'ordre mentionné dans le tableau suivant. Le C_{max} calculé à partir de ce séquencement est égale à **168.5** unités de temps.

Tableau IV.4. L'ordre des jobs selon l'heuristique JSSPTtran

Ordre sur machines	***Machines***			
	$\boldsymbol{M_1}$	$\boldsymbol{M_2}$	$\boldsymbol{M_3}$	$\boldsymbol{M_4}$
1	J_4	J_4	J_4	J_3
2	J_3	J_3	J_3	J_1
3	J_1	J_1	J_1	J_2
4	J_2	J_2	J_2	-

IV.3.2. L'heuristique *JSLPTtran*

Contrairement à l'heuristique précédente, le séquencement des jobs selon l'heuristique JSLPTtran se fait en fonction de l'ordre décroissant du rapport Rtd_j. L'objectif visé ici, est de donner la priorité aux jobs qui ont des valeurs du temps de traitement plus important que seules du déplacement. L'application de cette heuristique sur l'exemple illustratif, nous permet d'obtenir l'ordre des jobs mentionné dans le tableau IV.5. Cette solution génère une valeur de C_{max} qui est égale à **133** unités de temps.

Tableau IV.5. L'ordre des jobs selon l'heuristique JSLPTtran

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_2	J_2	J_2	J_2
2	J_1	J_1	J_1	J_1
3	J_3	J_3	J_3	J_3
4	J_4	J_4	J_4	-

IV.3.3. L'heuristique *JSSRCtran*

Pour cette heuristique nous proposons d'introduire la quantité de ressources consommables dans le calcul du rapport de choix Rtd_j. Le principe est de programmer les jobs en fonction de l'ordre croissant de rapport $RtdRC_j$ (l'équation 15) qui égale au Rtd_j multiplié par la quantité de ressources à consommer $Qdem_j^c$. Cette heuristique vise à placer au début de l'ordonnancement, les jobs qui consomment le moins des composants et qui nécessite des temps de traitement plus faible par rapport aux temps de déplacement.

$$RtdRC_j = Rtd_j * Qdem_j^c \quad (15)$$

L'application de cette heuristique sur le même exemple donne l'ordre indiqué dans le tableau IV.6. Le C_{max} obtenu dans ce cas est de **168.5** unités de temps.

Tableau IV.6. L'ordre des Jobs selon l'heuristique JSSRCtran

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_4	J_4	J_4	J_3
2	J_3	J_3	J_3	J_1
3	J_1	J_1	J_1	J_2
4	J_2	J_2	J_2	-

IV.3.4. L'heuristique *JSLRCtran*

Le choix de l'ordre des jobs selon cette heuristique est basé sur l'ordre décroissant du rapport $RtdRC_j$, ce qui donne la priorité aux jobs qui consomment plus de ressources et qui nécessitent un temps de traitement plus important que le temps de leurs déplacements. Le tableau suivant présente l'ordre des jobs après l'application de l'heuristique JSLRCtran sur notre exemple. Le C_{max} obtenu ici est de **133** unités de temps.

Tableau IV.7. L'ordre des Jobs selon l'heuristique JSLRCtran

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_2	J_2	J_2	J_2
2	J_1	J_1	J_1	J_1
3	J_3	J_3	J_3	J_3
4	J_4	J_4	J_4	-

IV.3.5. L'heuristique *JSSPTcumtran*

JSSPTcumtran est une heuristique qui consiste à ordonnancer, sur chaque machine, les jobs dans l'ordre croissant de leurs rapports des temps cumulés $Rtdcum_j^k$ (décri par l'équation 16). Ce rapport est défini par la division du temps cumulé de traitement $STTcum_j^k$ sur le temps cumulé de déplacement $STDcum_j^k$. L'objectif voulu par cette heuristique est de placer, devant chaque machine et en premier, les jobs qui arrivent et terminent leurs exécutions le plus tôt possible.

$$Rtdcum_j^k = \frac{STTcum_j^k}{STDcum_j^k} \qquad (16)$$

Les valeurs de $STTcum_j^k$ et $STDcum_j^k$ calculées pour notre exemple sont montrées dans le tableau IV.8. Pour démontrer la manière comment trouver ces valeurs on prend par exemple les deux valeurs $STTcum_j^4$ et $SDTcum_j^4$ qui présentent successivement le cumul des temps de traitement et de déplacement du job J_1 sur la machine M_4. En effet, selon sa gamme opératoire le job J_1 qui doive être réalisé sur les machines M_3, M_1, M_2 et M_4 , sa valeur de $STTcum_j^4$ se calcule par la somme de temps de traitement sur la machine M_3 (P_{13}) plus le temps de traitement sur la machine M_1 (P_{11}) plus le temps de traitement sur la machine M_2 (P_{12}) plus le temps de traitement sur la machine M_4 (P_{14}). De la même manière le $STTcum_j^4$ se calcule par la somme $T_1^3+T_1^1+T_1^2+T_1^4+T_1^s$

Notons ici que la valeur du temps de déplacement d'un job J_j vers la sortie du système, est ajoutée au calcul de sa valeur $STDcum_j^g$ dont M_g représente la dernière machine dans la gamme opératoire de ce job (comme est le cas de $STTcum_j^4$).

Tableau IV.8. Valeurs Cumulées des temps de traitement et de déplacement des jobs

Jobs	M_1		M_2		M_3		M_4	
	$STTcum_j^1$	$STDcum_j^1$	$STTcum_j^2$	$STDcum_j^2$	$STTcum_j^3$	$STDcum_j^3$	$STTcum_j^4$	$STDcum_j^4$
J_1	5	17.5	10	22.5	2	9.5	35	32
J_2	15	1.5	25	6.5	45	9.5	53	16
J_3	26	33.5	21	22.5	7	9.5	31	48
J_4	13	17.5	9	6.5	16	25.5	-	-

A partir des données présentées dans tableau IV.8 et selon l'équation 16 nous pouvons calculer les valeurs de $Rtdcum_j^k$ qui sont montrées dans le tableau suivant :

Tableau IV.9. Les valeurs du rapport $Rtd_j cum$

Jobs	$Rtdcum_j^k$			
	M_1	M_2	M_3	M_4
J_1	0.28	0.44	0.21	1.09
J_2	10	3.85	4.74	3.31
J_3	0.78	0.93	0.74	0.65
J_4	0.74	1.38	0.50	-

Selon l'heuristique *JSSPTcumtran* la solution du problème de l'exemple est présenté dans le tableau IV.10

Tableau IV.10. L'ordre des Jobs selon l'heuristique JSSPTcumtran

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_1	J_1	J_1	J_3
2	J_4	J_3	J_4	J_1
3	J_3	J_4	J_3	J_2
4	J_2	J_2	J_2	-

IV.3.6. L'heuristique *JSLPTcumtran*

La décision du choix de solution dans *l'heuristique JSLPTcumtran* est basée sur l'ordre décroissant de rapport $Rtdcum_j^k$…Selon cette heuristique le séquencement des jobs s'effectue comme suit :

Tableau IV.11. L'ordre des Jobs selon l'heuristique JSLPTcumtran

Ordre sur machines	***Machines***			
	M_1	M_2	M_3	M_4
1	J_2	J_2	J_2	J_2
2	J_3	J_4	J_3	J_1
3	J_4	J_3	J_4	J_3
4	J_1	J_1	J_1	-

IV.4. Résultats expérimentaux

Afin d'étudier les performances des méthodes proposées nous avons effectué plusieurs simulations du problème sur plusieurs instances. En revanche, nous avons divisé notre analyse en deux parties. La première partie consiste à évaluer la qualité des solutions obtenues par les approches proposées vis-à-vis la solution optimale trouvée par la génération de toutes les solutions possibles. Les instances retenues dans cette partie sont des instances de petites tailles. Dans la deuxième partie l'étude était allongée pour des instances à moyennes et grandes tailles. L'objectif dans cette partie est de trouver quelle est l'approche la plus adaptée au notre problème et qui donne des meilleures solutions.

En effet, selon notre recherche, nous n'avons pas pu trouver des benchmarks utilisés dans la littérature où les auteurs traitent exactement le même problème étudié dans ce travail. De ce fait nous avons proposé de générer, d'une manière aléatoire, des benchmarks propres à cette étude (**annexe A**). Alors, des gammes opératoires, des temps de traitement, des temps de transport ainsi que des quantités de ressources consommables sont générés aléatoirement à l'aide d'un programme sous Matlab. La génération aléatoire de ces données ab été faite pour quinze exemples différents de chaque instance.

IV.4.1. Résultats obtenus pour les petites instances

Dans cette partie, nous avons focalisé notre étude sur trois problèmes de petites tailles 2x4, 3x4 et 4x4, dont nous avons effectué une comparaison entre l'heuristique JSSPTcum (présentée dans le chapitre III), les six heuristiques proposées, et la méthode exacte. L'objectif été d'évaluer les performances de ces heuristiques et perfectionner la qualité de leurs solutions vis à vie de la solution optimale. En effet, pour obtenir des solutions à notre problème, nous avons programmé l'ensemble des approches sous Matlab dont les temps de traitement sont générés entre 1 et 30 unités de temps. Le critère d'évaluation retenu pour effectuer la comparaison entre les approches proposées est la valeur de la fonction objectif C_{max}. Les résultats en termes de moyennes de Makespen (C_{max}) obtenues par l'application de ces approches sur 15 exemples différents, sont présentés dans les figures IV.3, IV.4 et IV.5.

Nous remarquons à partir des résultats montrés dans ces figures que l'heuristique JSSPTcum donne des résultats très proches de l'optimum pour les instances de 2x4 et 3x4 et des meilleures solutions pour l'instance de 4x4, de ce fait, elle domine toutes les autres heuristiques. Nous pouvons remarquer aussi que l'heuristique JSSPTcumtran se classe en deuxième position après JSSPTcum pour toutes les instances de petites tailles en dépassant donc les cinq autres heuristiques. Pour les cinq heuristiques qui restes, elles ont pratiquement le même classement, avec un petit dépassement des heuristiques JSLPTcum, JSSPTtran et JSLPTtran, qui n'intègrent pas la quantité de ressources consommables dans le rapport du choix, par rapport aux deux heuristiques JSSRCTtran et JSLRCTtran qui sont, dans la plupart du temps, en dernières positions. En outre, nous voyons que, entre deux heuristiques utilisant le même rapport du choix, les heuristiques qui se basent sur l'ordre croissant du rapport donnent des résultats plus performants que les autres heuristiques qui utilisent l'ordre décroissant dans le choix des solutions.

A travers ces résultats nous pouvons constater, premièrement, que l'utilisation de la valeur cumulée dans le rapport de choix des solutions démontre une grande efficacité concernant l'identification du séquencement des jobs sur les machines, ceci est bien prouvé par les résultats des deux heuristiques JSSPTcum et JSSPTcumtran. Deuxièmement, l'intégration de quantités de ressources consommables dans les rapports de sélection ne permet pas de trouver des bonnes solutions du problème étudié. Troisièmement, l'ordre croissant des rapports présente un critère très efficace pour le choix de séquencement des jobs et qui conduit souvent vers des meilleures solutions pour les petites instances de notre problème.

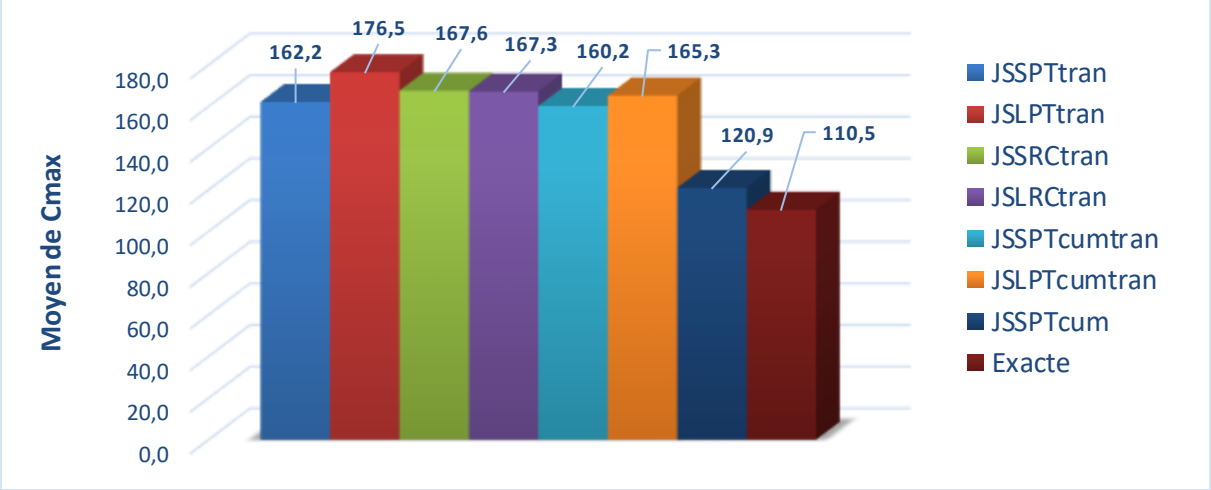


Figure IV.3. Résultats obtenus pour le problème 2*4

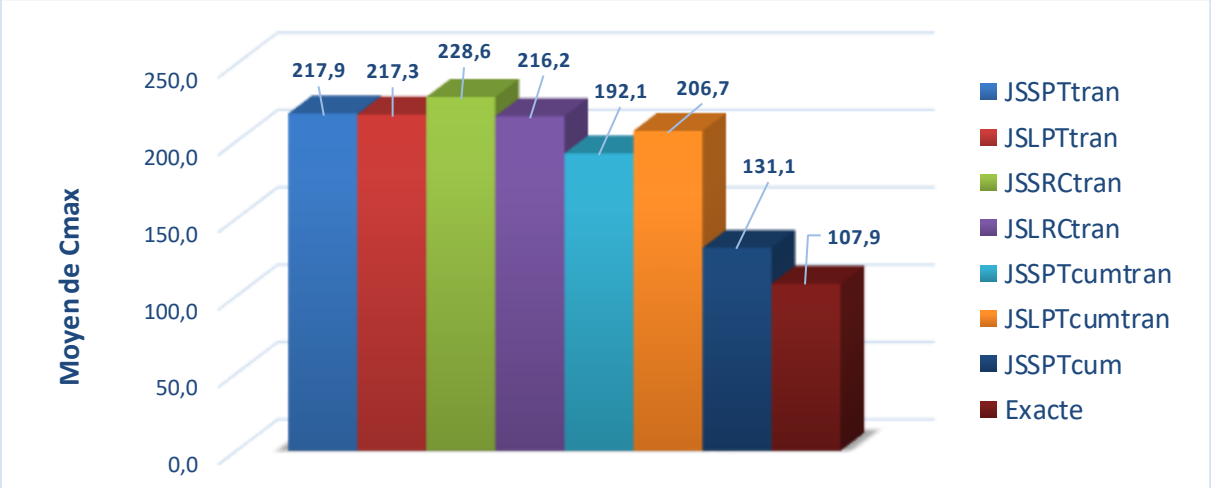


Figure IV.4. Résultats obtenus pour le problème 3*4

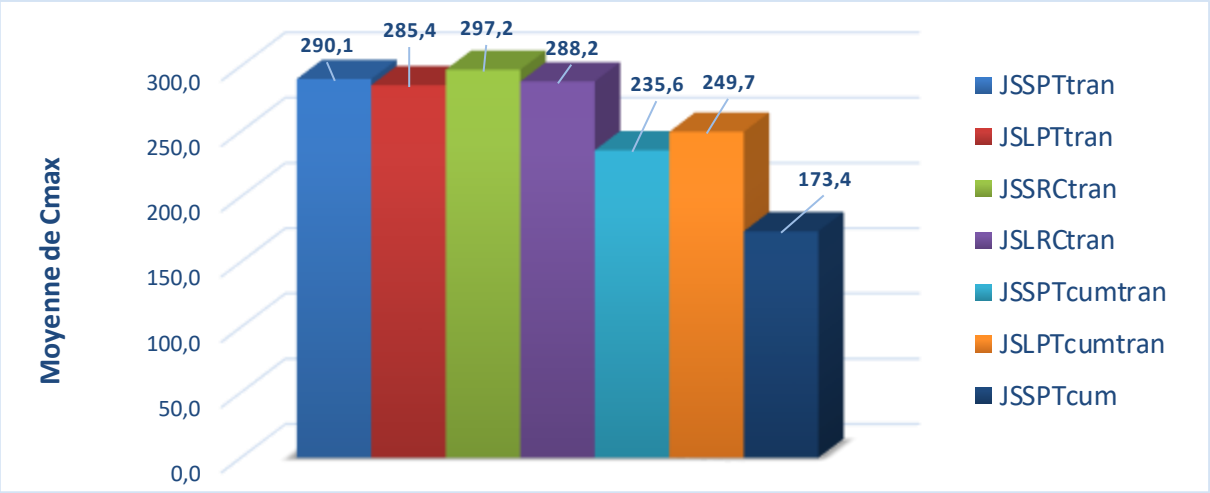


Figure IV.5. Résultats obtenus pour le problème 4*4

IV.4.1. Résultats obtenus pour les moyennes et les grandes instances

L'objectif visé dans cette partie est d'approuver l'efficience des heuristiques proposées dans la résolution du problème avec des complexités différentes. Le choix des instances a été fixé comme suit : 8x4, 10x4 et 15x4 pour les moyennes instances et 20x4, 50x4 et 100x4 pour les grandes instances. L'étude des approches dans cette partie est également basée sur l'évaluation des valeurs de Mekespan calculées à travers 15 exemples différents pour chaque instance. Les résultats obtenus de l'ensembles des tests effectués sont présentés dans les deux figures IV.6 et IV.7. Ces figures montrent successivement, la variation de la moyenne de C_{max} calculée pour les moyennes et grandes instances.

Notant ici que, dans cette partie d'expertise, le nombre des solutions admissibles trouvées par les deux heuristiques JSSPTcumtran et JSLPTcumtran ne dépasse pas deux solutions sur les quinze exemples étudiés, ce qui montre la faiblesse de ces deux heuristiques concernant les problèmes de moyennes et grandes instances. De ce fait, nous avons préféré d'écarter totalement leurs résultats de notre comparaison.

A travers les résultats obtenus après l'ensemble des tests effectués, nous remarquons très bien que l'utilisation de l'heuristique JSSPTcum a donné des très bons résultats par rapport aux autres heuristiques ce qui démontre encore une fois son efficacité de résolution du problème pour des instances plus grandes. En revanche, pour les autres heuristiques et contrairement à ce qui est constater dans la première partie d'expertise (les petites instances), les heuristiques qui utilisent l'ordre décroissant du rapport de choix des solutions (JSLPTtran et JSLRCtran), dépassent les autres heuristiques utilisant l'ordre croissant du rapport (JSSPTtran et

JSSRCtran), ce qui prouve et motivent leurs choix concernant la résolution du problème pour les moyennes et les grandes instances.

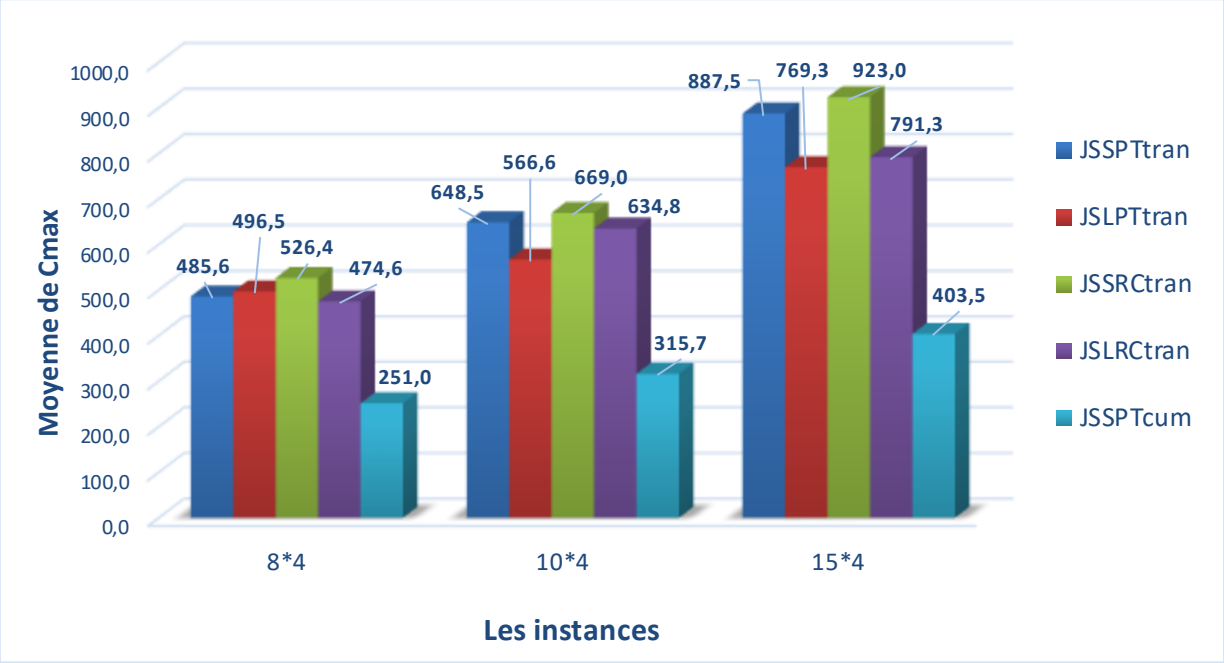


Figure IV.6. Résultats obtenus pour les problèmes de moyennes tailles

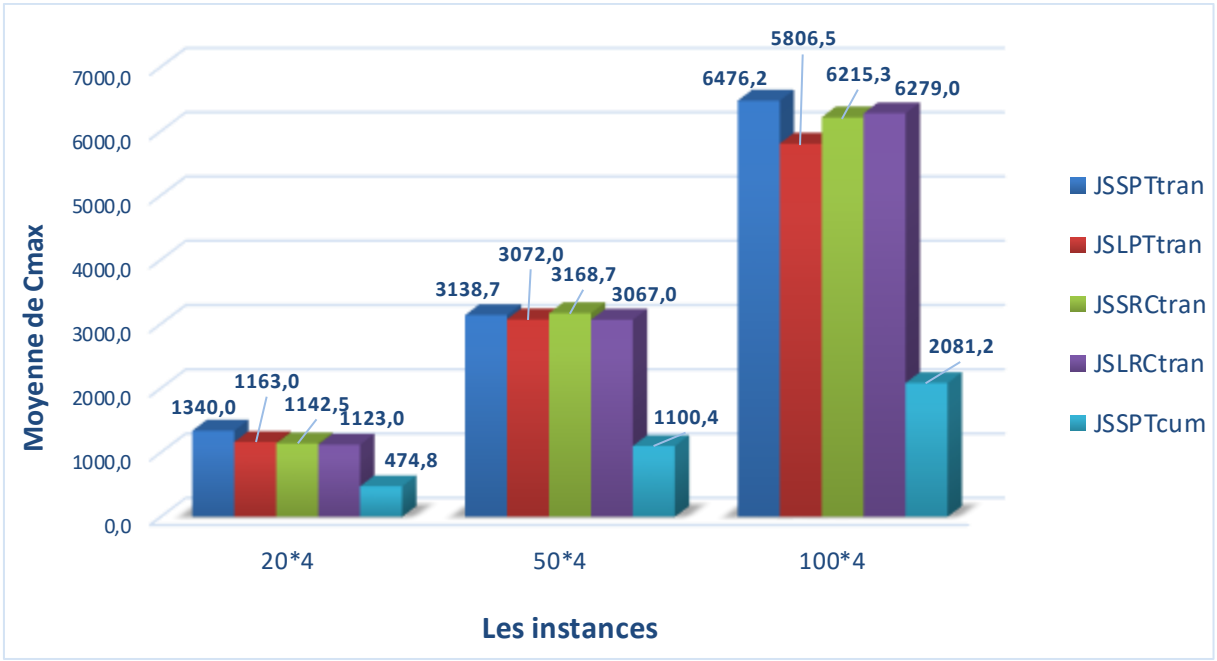


Figure IV.7. Résultats obtenus pour les problèmes de grandes tailles

IV.5. Conclusion

Dans ce chapitre, nous nous sommes intéressés à la résolution du problème de minimisation de makespan dans un environnement de job shop avec contrainte de transport unidirectionnel tout en prenant en compte les contraintes de ressources non-renouvelables qui compliquent le processus décisionnel d'ordonnancement des tâches qui peuvent nécessiter plusieurs composants en même temps de différents types. Nous avons montré l'efficacité des heuristiques basées sur les règles de priorité dans la résolution de ce type de problème sur un grand nombre de configurations différentes. La comparaison entre ces méthodes montre l'avantage de l'heuristique JSSPTcum sur les autres approches en terme de makespan.

L'étude menée dans ce chapitre nous a permis de dégager quelques résultats préliminaires qui donnent la possibilité d'évaluer le système décisionnel d'ordonnancement en utilisant les méthodes heuristiques ;

- L'utilisation de la valeur cumulée dans le rapport de choix des solutions, présente un outil très efficace concernant l'identification du séquencement des jobs pour la minimisation de C_{max} de notre problème.
- Concernant les heuristiques qui intègrent la quantité de ressources consommable dans le rapport de choix sont, dans la majorité des cas, moins performantes que les autres heuristiques.
- Nous avons constaté que, plus la taille du problème augmente, plus les heuristiques utilisant l'ordre décroissant du rapport de choix devienne efficaces.
- Finalement, à travers les différentes études qui ont été réalisées, nous avons constaté que, l'approche qui reste la plus performante, c'est l'heuristique JSSPTcum qui dépasse les autres approches concernant toutes les taille du problème.

Chapitre V : Approche temps réel pour l'ordonnancement job shop en présence de pannes de convoyeur

V.1. Introduction

Dans l'environnement réel des systèmes de production, l'exécution des travaux selon un planning prédéfini est presque impossible à cause de plusieurs types de perturbations (internes et/ou externes) qui empêchent le déroulement normal de ces travaux. De ce fait, les responsables de production sont souvent obligés de trouver de méthodes et des outils efficaces pour réagir rapidement et d'une manière optimale face à ces perturbations.

Dans le chapitre précédent, nous avons abordé le problème d'ordonnancement job shop avec deux contraintes (contrainte de transport et contrainte de ressources consommables) mais dans un contexte statique, c'est-à-dire, que toutes les données du problème, sont supposées connues à l'avance et aucune perturbation n'est présent pour perturber l'exécution de l'ordonnancement retenu. Dans ce chapitre, nous nous sommes intéressés au même problème mais avec la prise en compte de l'une de perturbations qui peuvent être apparues dans le système c'est la panne de système de transport. Pour résoudre ce problème nous avons proposé une démarche de réordonnancement en temps réel (en linge) qui consiste à établir un nouvel ordonnancement des jobs lors de l'apparition de cette perturbation.

Nous commençons ce présent chapitre par la présentation d'un ensemble des définitions concernant l'ordonnancement sous perturbations, puis nous présentons un état de l'art des travaux déjà réalisés dans ce contexte. La description du problème étudiée et la démarche proposée sont ensuite présentées et enfin nous exposons les applications effectuées de cette démarche sur plusieurs exemples ainsi que les résultats obtenus.

V.2. L'ordonnancement en présence de perturbations

L'ordonnancement de la production est une fonction qui permet d'organisation, sur un échèle de temps, l'exécution d'un ensemble des tâches sur un ensemble de ressources. Cette organisation nécessite obligatoirement la connaissance de plusieurs données comme le nombre des tâches à exécuter, la durée de chaque opération, la disponibilité des ressources… Dans la littérature et pour la plupart des travaux qui s'intéresse aux problèmes d'ordonnancement ces données sont généralement supposées constantes et connues avant le début du travail. Cependant, en réalité ces données peuvent avoir plusieurs changements à

cause de plusieurs perturbations ce qui rend la fonction ordonnancement moins efficace ou totalement inutile. Dans cette vison, Pinedo (Pinedo, 2016) a noté que : *" En pratique, l'environnement d'ordonnancement est généralement sujet à une quantité significative d'aléatoire ; en conséquence, il n'est pas intéressant de passer un temps énorme à déterminer une solution supposée optimale lorsque, en quelques heures à cause de quelque événement aléatoire, la structure du problème ou bien la liste des travaux aura changé."*

Dans ce qui suit nous allons donner un panorama rapide sur les notions les plus importantes relatives à l'ordonnancement sous perturbation.

V.2.1. Incertitudes et aléas

a. *Incertitudes* : la notion de l'incertitude dans l'ordonnancement réside dans le fait que les valeurs des données d'un problème d'ordonnancement ne sont pas connues à priori, ou si elles sont sujettes à des modifications, lors de l'établissement de l'ordonnancement ou pendant sa réalisation. Cette incertitude peut être modélisée par une variable aléatoire ou par un ensemble d'intervalles. C'est le cas, par exemple, d'une durée opératoire d'une tâche variant durant l'exécution de l'ordonnancement. (Ourari, 2011) (Yahouni, 2019).

b. *Aléas* : Ce terme désigne les événements provoquant la modification des données du problème comme la panne, retard de livraison, ordre urgent, etc. Dans un problème d'ordonnancement ces aléa sont considéré par leurs occurrences qui peuvent être représentées par des variables aléatoires

V.2.2. Type de perturbation

En réalité, pour un système de production, les perturbations ou les événements pouvant imposer des modifications sur les données prédéterminées, sont généralement divisés en quatre classes comme suit (Suresh & Chaudhuri, 1993):

a. ***Événements liés aux jobs*** : Ces évènements sont liés aux caractéristiques des jobs qui vont être traiter dans le système de production dans ce cas on peut distinguer les évènements suivants : Changement soudain du temps de traitement des jobs, l'arriver aléatoire des nouveaux jobs, changement de la date de livraison, modification de la gamme opératoire de certains jobs, changement de la priorité entre jobs...

b. ***Événements liés aux systèmes*** : Ces évènements sont liés à la disponibilité et la capacité des matériels utilisés dans la production (machine, moyen de transport…). Dans cette

classe on trouve : La panne de machines, la panne de moyen de transport, le conflit entre la capacité des machines et le volume de production…

c. ***Événements liés aux processus :*** Ces évènements sont liés au processus de production. L'ensemble des événements qui peuvent être distinguer ici sont : processus retardé, qualité rejetée, production instable…

d. ***Autres événements :*** Ces évènements ont des relations avec l'opération de production d'une manière générale. On peut citer ici : l'absence des opérateurs, l'arrivée tardive des matières premières, les défauts des matières premières, etc.

V.2.3. Méthodes utilisées pour l'ordonnancement sous perturbations

a. ***Méthodes purement réactives***

Les méthodes d'ordonnancement réactives sont des méthodes qui consistent à générer un ordonnancement en temps réel. Pour ce type d'ordonnancement, il n'y a pas un schéma prédéfini de l'ordonnancement, la stratégie de planification consiste à générer à chaque fois un nouvel ordonnancement en fonction des événements aléatoires qui se produisent dans l'atelier. L'inconvenant de cet ordonnancement est qu'il ne peut pas fournir un repère de planification pour les activités pertinentes de l'atelier. Il en résultera donc une modification fréquente du plan de planification, puis une augmentation des coûts de planification lorsque l'interruption se produit. (Mohan, Lanka, & Rao, 2019).

b. ***Méthodes prédictives-réactives***

La prédictif-réactif en l'ordonnancement est une stratégie courante qui comprend deux étapes principales ; la première étape consiste à générer une pré-planification sans tenir compte des événements perturbant de l'atelier. Quant à la deuxième étape consiste à mettre à jour le plan préprogrammé au fur et à mesure que les événements dynamiques apparaîtront. Ces mises à jour donnent des solutions qui peuvent améliorer les performances du programme initial (Mohan, Lanka, & Rao, 2019).

c. ***Méthodes proactives***

A la base d'une estimation préétablie des perturbations potentielles, le principe de ces approches est de construire durant la phase hors-ligne un ordonnancement de référence robuste. Ces approches prennent en compte les incertitudes uniquement durant la phase prédictive de l'ordonnancement. Pour pallier aux perturbations durant la phase réactive qui est supposée être triviale dans ce type d'approches (Yahouni, 2019).

d. ***Méthodes proactives-réactives***

Ce type d'approches consiste à établir, durant la phase hors-ligne, un ordonnancement flexible. Cet ordonnancement est figuré par plusieurs solutions permettant d'absorber les perturbations qui peuvent se produire durant la phase en ligne. Les méthodes proactives-réactives combinent certains avantages des approches précédentes en prenant en compte les deux phases de l'ordonnancement (Yahouni, 2019).

V.3. État de l'art

Depuis le début des années 80, période où les systèmes flexibles de production ont été adoptés, la notion temps réel dans le contrôle et l'ordonnancement des systèmes flexible de production ouvre des grandes pistes dans le domaine de recherche (Saygin & Kilick, 1996) (Peng & Chen, 1998). Actuellement et avec la révolution industrielle présentée par l'industrie 4.0, l'exploitation de l'information temps réel, est devenue plus en plus possible et précise, de plus, le temps de réaction vis à vie des alias est devenu plus en plus court grâce à la technologie avancée de ce type d'industrie comme l'internet phtisique, le big-data etc. (Turker, et al., 2019), (Ghaleb, Zolfagharinia, & Taghipour, 2020).

Dans l'environnement réel des systèmes de production, l'exécution des travaux selon un planning prés défini est presque impossible à cause de plusieurs types de perturbations qui empêchent le déroulement normal de ces travaux. Les planificateurs de production, dans ce cas, doivent non seulement générer des ordonnancements de haute qualité, mais également réagir rapidement afin de réviser leurs plannings de manière rentable. Ces perturbations peuvent être externes au système (comme l'arrivée des commandes urgentes, l'indisponibilité de la matière première…) ou internes (comme les pannes imprévues des machines, l'absences des opérateurs, l'indisponibilité des moyens de transport…) (Vieira, Herrmann, & Lin, 2003), (Subramaniam, Raheja, & Rama Bhupal Reddy, 2005) (Zhang & Wong, 2017).

Le problème d'arrivée des nouvelles commandes (jobs) est un problème très souvent rencontré dans l'industrie et plus particulièrement dans systèmes de fabrication travaillant à la commande. Dans ce contexte plusieurs travaux ont été proposés dont le but de proposer des techniques efficaces permettant de minimiser au maximum possible le coût généré par l'insertion de ces nouvelles commandes dans le planning initial. Cao et al. (Gao, Yang, Zhou, Pan, & Suganthan, 2018) étudient le problème d'insertion des nouveaux jobs dans un environnement job shop flexible. Ils ont proposé une méthode de réordonnancement basée sur une méta-heuristique appelé Jaya dont l'objectif est de minimiser à la fois l'instabilité,

le Makespan et le temps d'attente des machines. L'article (Wang, Zhang, & Yang, 2019) examine la question de savoir comment intégrer les nouveaux jobs arrivant d'une manière aléatoire afin de garantir la performance et la stabilité de l'ordonnancement dans un système job shop. Dans (Luo, El Baz, Xue, & Hu, 2020), les auteurs proposent une méthode basée sur les algorithmes génétiques afin de résoudre le problème de job shop en considérant l'arrivée des commandes urgentes. L'objectif est de minimiser le retard total et le coût totale d'énergie générés par cette perturbation. Plusieurs autres travaux s'intéressant au problème d'insertion de nouveaux jobs sont présentés dans (Rong, 2008) (Zhang, Gao, & Li, 2013) (Luo S. , 2020).

La panne de machines représente la perturbation la plus étudiée dans les problèmes d'ordonnancement. Ce là, à cause de son impact majeure sur le déroulement normal des travaux. Dans (Buddala & Mahapatra, 2019), une méthode prédictive réactive basée sur la technique d'apprentissage de deux étages a été développée afin de résoudre le problème de job shop flexible avec panne de machines. Pour introduire les données de cette évènement indésirable dans le problème job shop flexible, les auteurs ont proposé une technique d'insertion des temps morts dans le planning initial. L'objectif de la méthode est d'effectuer un réordonnancement efficace lors d'apparition d'une panne afin d'assurer la robustesse et la stabilité de l'ordonnancement prédictive. Songling Tian et al. (Tian, Wang, Zhang, & Wu, 2019) visent un problème de job shop flexible avec panne de machine. Pour résoudre ce type de problème les auteurs ont proposé un algorithme basé sur une approche intégrée de deux méthodes Réseau de Petri et Colonie de fourmi. Cet algorithme permet de trouver des solutions qui garantissent en même temps l'exécution de la production selon le planning initial et la fiabilité opérationnelle de l'atelier en cas de perturbation.

Le retard ou le blocage de moyen de transport à cause d'une panne de système de transport, est considéré parmi les perturbations le moins étudier dans la littérature. L'article (Poppenborg, Knust, & Hertzberg, 2012) traite le problème job shop flexible avec blocage de transport en supposant qu'une machine est bloquée l'hors de fin de traitement jusqu'à le job traité sera transporté ce qui défaire à notre travail dont nous considérant que la machine est libre dès qu'elle termine son traitement, de ce fait un autre job (le premier dans la liste d'attente) peut être traité immédiatement et sans retard dans cette machine.

Plusieurs méthodes ont été développées pour résoudre le problème de réordonnancement dans un environnement dynamique. Dans ce cadre, un résumé très important est décrit dans l'article (Mohan, Lanka, & Rao, 2019).

Les méthode réactives ou approches temps réel sont des méthodes qui permet de construire l'ordonnancement des tâches en fonction de l'état réel de l'atelier où aucun ordonnancement est prédéfini au préalable (Chan, Chan, & Lau, 2002) . Ce type d'approches est généralement utilisé lorsque le niveau de perturbations est trop important ou lorsque les variables du problème ne peuvent pas être définies préalablement (Yahouni, 2019). Dans ce contexte, Hassam (Hassam, 2012) traite un problème de sélection de routage alternative des tâches dans un système flexible de job shop. L'auteur propose plusieurs heuristiques basées sur les règles de priorité permettant de définir, à chaque instant et en temps réel, quel est le meilleur routage (des jobs) à implanter dans le système. Houari (Houari, 2018), Quant à elle propose l'utilisation des méta-heuristiques pour la sélection en temps réel des routages dans le même système mais avec présence d'incertitudes sur les ordres de fabrications, les incertitudes liées aux pannes de machines et les incertitudes sur les durées opératoires.

Des modèles d'ordonnancement temps réel sont développés par Ghaleb et al. (Ghaleb, Zolfagharinia, & Taghipour, 2020) afin de résoudre le problème de job shop flexible soumis à deux types de perturbation ; panne de machines et l'arrivée des nouveaux jobs. Ces modèles de réordonnancement ont pour objectif de générer nouvel ordonnancement basé sur la mise à jour temps réel des données tel que le temps d'arrivé de nouveaux jobs, l'indisponibilité de machines, le temps d'exécution des opérations, etc.

Dans le travail présenté dans l'article (Valledor, Gomez, Priore, & Puente, 2018), les auteurs s'intéressent au problème de flow shop avec trois types de perturbations ; la panne de machines, l'arrivé des nouveaux jobs et le changement de processing time. Ils ont proposé une architecture de ré-ordonnancement basée sur une approche prédictive-réactive dont ils ont utilisé les règles de priorité pour définir l'ordonnancement réactif à chaque instant. Une évaluation multi-objectifs basée sur trois objectifs a été effectuée afin de définir les meilleures solutions. Wang et al (Wang, Liu, & Jin, 2019) proposent une approche proactive permettant de trouver le meilleur séquencement des jobs en assurant la robustesse de l'ordonnancement prédéterminé. La perturbation considérée de ce travail et la panne de machine.

V.4 Description du problème

Notre étude est toujours focalisée sur le système flexible de production iCIM 3000 qui se compose de quatre stations (machines) reliées par un convoyeur à palettes sous forme d'une boucle (la figure V.1). Les pièces (jobs) déplacent à l'aide de ce convoyeur sur des palettes à capacité unitaires dans un sens unidirectionnel.

Dans le chapitre précédent, le problème traité était un problème d'ordonnancement job shop avec deux contraintes (JSPTRC) ; contrainte de transport et contrainte de ressources consommables. Ce problème consiste à ordonnancer *n* jobs (qui ont des gammes de fabrication différentes) sur les quatre stations avec prise en considération le temps de déplacement et la quantité de ressources à consommer lors du traitement de chaque job. En effet, ce problème a été abordé dans un contexte statique où toutes les données sont supposées connues à l'avance.

Figure V.1.a Image virtuelle du convoyeur (logiciel CIROS)

Figure V.1.b Image réelle du convoyeur à palettes iCIM3000

Figure V.1. Convoyeur à palettes du système iCIM3000

Dans cette partie de la thèse, nous intéressons au même problème (JSPTRC) mais à la partie dynamique de ce problème. Nous allons traiter ce problème en présence de l'une de perturbations qui peuvent être apparues dans le système, c'est la panne de système de transport.

Pour notre problème, si le convoyeur tombe en panne, aucun job ne peut être déplacé jusqu'à la fin de réparation (durée de panne). Cependant, une opération en cours sur une machine n'est pas interrompue c'est-à-dire que la machine doit terminer l'exécution de cette opération jusqu'à la fin. Si un job termine son exécution sur une machine avant la réparation du convoyeur, ce job quitte la machine vers un stock d'attente pour être déplacé par la suite. Après la réparation du convoyeur, le déplacement des jobs sera relance de nouveau.

V.5 Approche temps réel pour l'ordonnancement job shop avec pannes de convoyeur

Dans cette partie nous allons présenter la méthode de réordonnancement temps réel que nous avons proposé pour résoudre le problème job shop avec considération de panne de convoyeur. La démarche proposée consiste à établir un nouvel ordonnancement efficace lors de l'apparition de la panne du convoyeur. Ce réordonnancement consiste à modifier le plan préétabli en prenant en considération toutes les informations indiquant l'état réel du système. L'objectif est de trouver une nouvelle organisation des jobs de tel sort que le coût généré par cet événement perturbant ne soit pas élevé.

En effet, cette démarche en ligne est divisée, comme le montre l'organigramme de la figure V.1, en deux étapes principales ; la mise à jour des données et le réordonnancement

Etape 01 « mise à jour des données » : Cette étape consiste, premièrement, à collecter toutes les informations réelles sur l'état du système, c'est-à-dire déterminer toutes les données identifiant les positions des pièces dans le système à l'instant d'apparition de la panne (traitées par les machines, sur le convoyeur, à l'entrée du système, …) ainsi que les opérations déjà effectuées de chaque job programmé et les quantités des ressources existantes dans les stocks, etc. Deuxièmement, à mettre à jour les données du problème de l'ordonnancement initial à la base des nouvelles informations déjà collectées. Cette mise à jour vise à modifier les gammes opératoires des jobs en éliminant les opérations déjà effectuées, à modifier la disponibilité des jobs et la disponibilité des machines, à modifier les quantités de ressources consommables existantes, etc.

Etape 02 « le réordonnancement » : Cette étape consiste, en utilisant les nouvelles données du problème et à partir de la date de fin de la panne, à trouver le meilleur ordonnancement qui détermine l'exécution des jobs restants dans le système. Vue que le réordonnancement nécessite une méthode efficace et rapide et vue l'efficacité de l'heuristique JSSPTcum démontrée pour le problème job shop avec les deux contraintes transport et ressources consommables, nous avons choisi cette heuristique pour effectuer le réordonnancement.

En effet, l'opération de la mise à jour des données du problème est basée sur l'évolution du système pendant les deux phases, la phase avant l'apparition de la panne et la phase durant la période de la panne. Dans la première phase, tous les composants du système sont en état de

bon fonctionnement et le comportement du système est connu et déterminée par le planning prédéfini (plan prévisionnel).

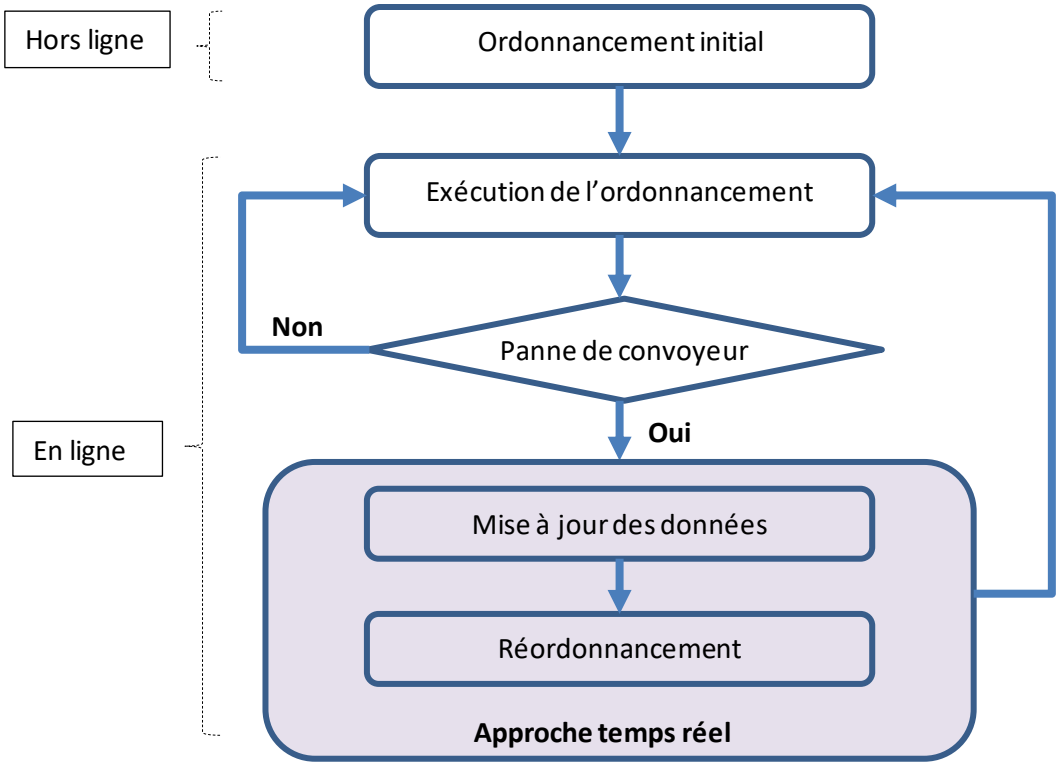


Figure V.2. Les deux phases de l'ordonnancement

Dans la deuxième phase, le convoyeur est en panne, par conséquent certains évènements vont être continues et certains évènements vont être s'arrêtés. Six cas sont possibles concernant la situation réelle des jobs au moment de l'apparition de la panne

Le premier cas : Le job a terminé son traitement dans le système, c'est-à-dire toutes ses opérations et ses déplacements ont été effectuées et le job est en d'hors du système (la figure V.3). Dans ce cas, toutes les données de ce job seront éliminées du problème.

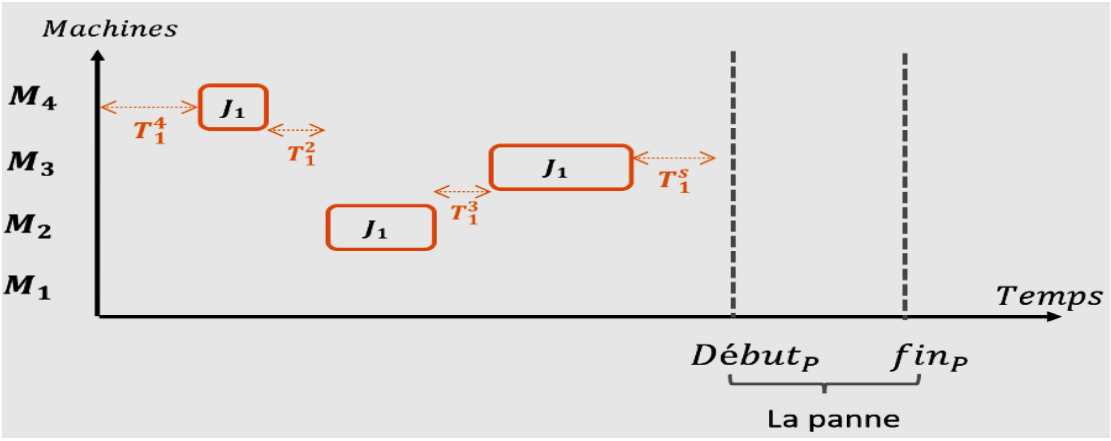


Figure V.3. Exemple d'un job son traitement va être terminer avant la panne

Le deuxième cas : Le job se trouve à l'entrée du système et son traitement n'a pas encore commencé. Dans ce cas, l'opération de mise à jour consiste à garder toutes les informations de ce job (le nombre d'opérations, gamme opératoire, les déplacements à effectuer…) et changer uniquement sa disponibilité. Pour l'exemple de la figure V.4, le traitement du job J_2 va commencer après la panne donc sa disponibilité dans le nouveau problème devient égale à la fin de la panne (fin_P) plus temps de déplacement de ce job (T_2^1) vers la première machine M_1.Selon l'équation (14) la disponibilité du job J_2 se calcule comme suit : $D_2^1 = (fin_P + T_2^1)$.

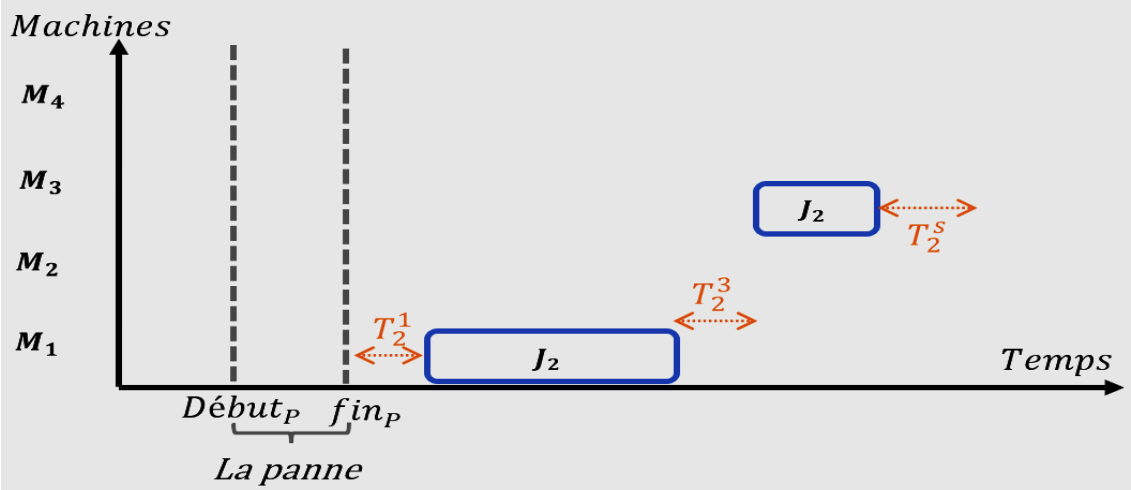


Figure V.4. Exemple d'un job son traitement va être commencer après la panne

Le troisième cas : Le job vient juste de terminer son exaction sur une machine Dans ce cas, la mise à jour consiste à éliminer du problème, toutes ses opérations et ses déplacements déjà effectués et modifier sa disponibilité qui sera égale à la fin de la panne. Comme le montre la figure V.5, le job J_3 vient de terminer son exaction sur une machine M_4 juste avec le début de la panne alors, sa nouvelle disponibilité devient $D_3^4 = fin_P$.

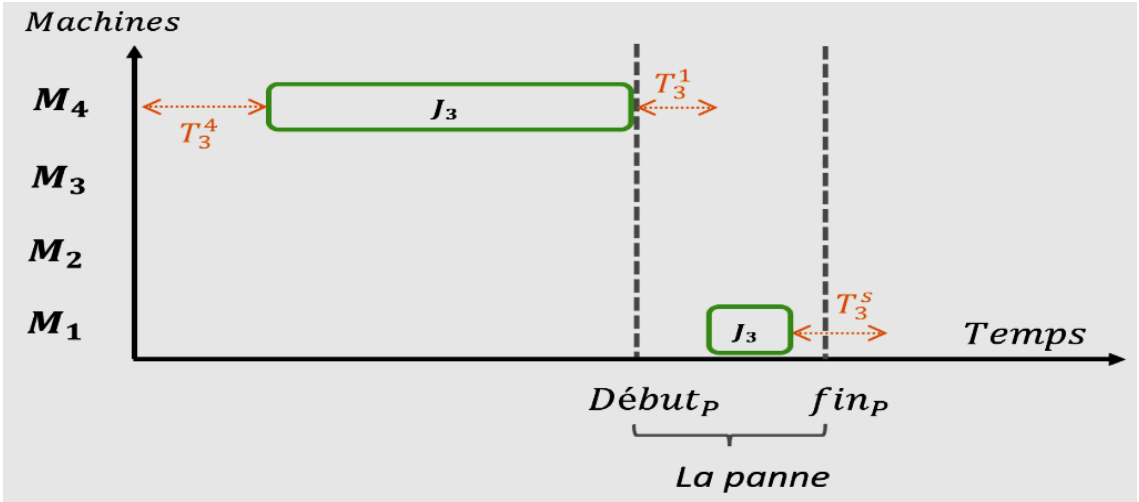


Figure V.5. Exemple d'un job vient juste de terminer son exaction sur une machine

Le quatrième cas : Le job est en déplacement, c'est dire que la pièce existe toujours sur le convoyeur et se déplace vers une machine pour qu'elle soit traitées. Dans ce cas, l'approche proposée consiste à éliminer toutes ses opérations et ses déplacements déjà effectués, garder la quantité du temps **restante** de son déplacement en cours et garder aussi toutes les opérations et les déplacements prochains. Dans la figure V.6 nous remarquons que, lors de l'apparition de la panne, le job J_4 a terminé son exécution sur la machine M_1 et se déplace vers la machine M_4. Dans ce cas, la nouvelle disponibilité de ce job sera égale à la fin de la panne plus le temps restant du déplacement $D_4^4 = (fin_P + T_4'^4)$.

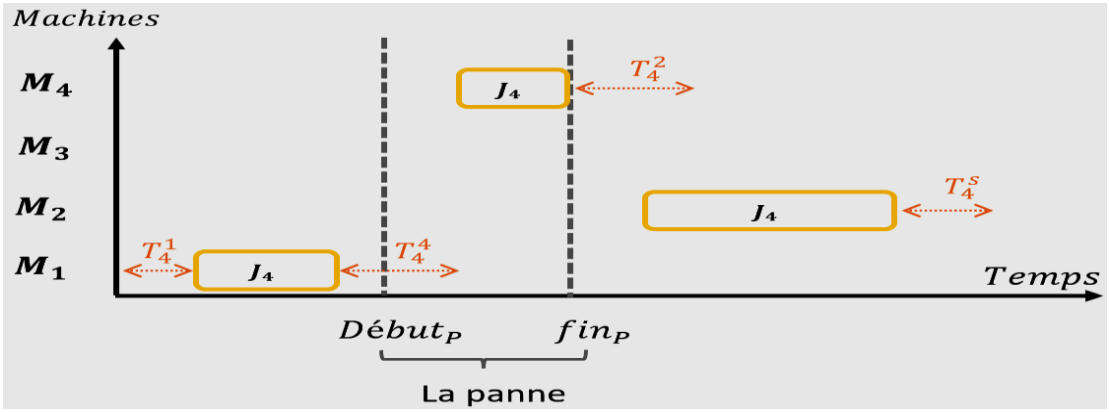


Figure V.6. Exemple d'un job en déplacement

Le cinquième Cas : Le job vient juste de commencer son exécution ou il est en cours d'exécution sur une machine et son traitement va être terminé après la fin de la panne (la figure V.7). Dans ce cas, il faut éliminer toutes les opérations et les déplacements déjà effectués et garder toutes ce qui est reste.

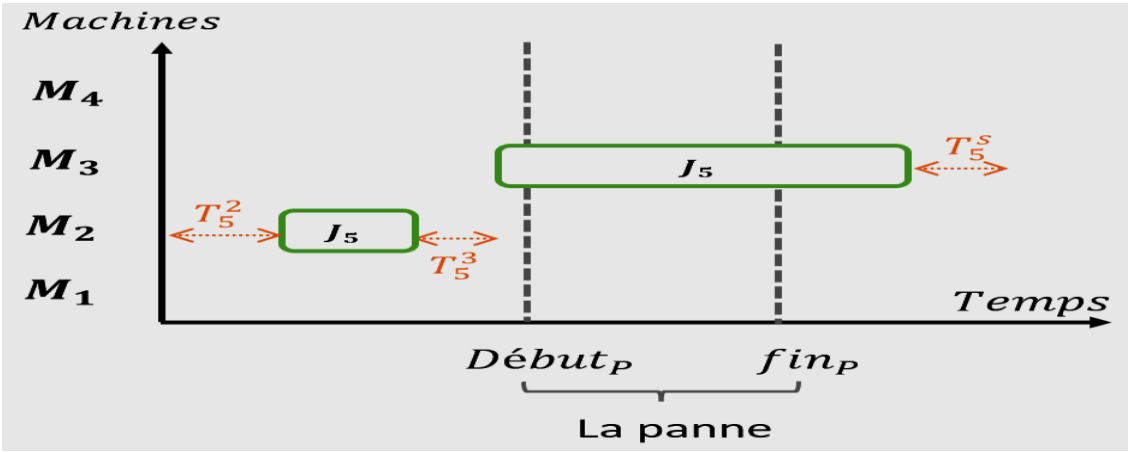


Figure V.7. Exemple d'un job son traitement sur une machine va être terminé après la panne

Le sixième cas : Le job vient juste de commencer ou il est en cours d'exécution sur une machine et son traitement va être terminé avant ou juste avec la fin de la panne. Dans ce cas, il faut éliminer toutes les opérations et les déplacements déjà effectués, sa disponibilité sera la fin de la panne plus le temps restant du déplacement. Pour exemple montré dans la figure V.7, la nouvelle disponibilité du job J_6 sera calculé comme suit : $D_6^3 = (fin_P + T_6'^3)$.

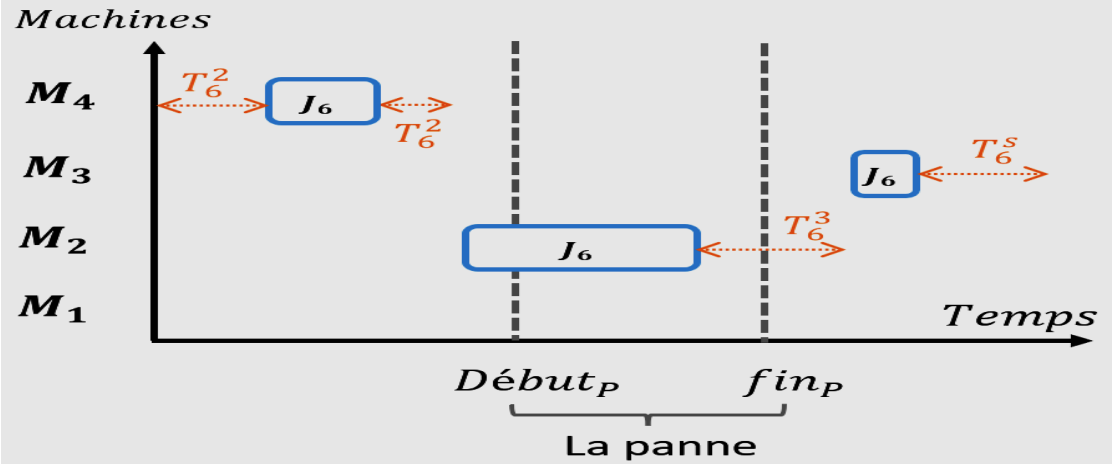


Figure V.8. Exemple d'un job en cours d'exécution sur une machine et son traitement va être terminé avant la panne

D'un autre côté, il est noté que, pendant la période de la panne du convoyeur, l'opération d'arrivé des ressources consommables ne s'arrête pas. De ce fait, la quantité de ressources consommables existant dans le stock à la fin de la panne se détermine par la somme de toutes les quantités arrivées jusqu'à l'instant de fin de panne moins la somme de quantité consommé par les opérations déjà effectuées.

V.6. Applications expérimentales et interprétation des résultats

Cette partie du chapitre est consacrée à la présentation de l'ensemble des applications que nous avons effectué sur la méthode de résolution proposée ainsi que les résultats obtenus.

V.6.1. Dimension des expériences

L'application de la méthode proposée a été effectuées sur plusieurs instances de tailles différentes. Le choix du temps de traitement des jobs (Processing time), les gammes opératoires, la quantité et la date d'arrivée des ressources consommables sont choisis aléatoirement pour chaque problème considéré.

Pour bien cerné notre étude, nous avons supposé que l'instant d'occurrence de la panne ($Début_P$) est choisi aléatoirement en fonction du makespen de l'ordonnancement prévisionnel L'équation.20 présente la formule que nous avons pris pour trouver la valeur de $Début_P$.

Dans cette équation, le $C_{max}(Init)$ représente le makespan initiale trouvé dans la phase hors-ligne et le facteur δ peut prendre des valeurs aléatoires appartenant à l'intervalle $[0,1]$. Le choix de cet intervalle réside dans le fait que dans notre travail nous ne considérons que les pannes apparues lors de l'exécution des travaux.

$$Début_P = \delta * C_{max}(Init) \quad (20)$$

De plus on suppose que la durée de la panne ($Durée_P$) prend toujours une valeur et égale au temps de réparation (TTR). Cette valeur dépend, en réalité, du taux de réparation du convoyer ($\boldsymbol{\mu}$) comme le montre l'équation suivante :

$$Durée_P = \frac{1}{\mu} \quad (21)$$

V.6.2. Mesure de performance

L'objectif visé dans cette partie est de minimiser le coût global généré par l'occurrence de la panne du convoyeur. Deux facteurs de performance sont souvent utilisés dans la littérature pour déterminer l'efficacité des approches de réordonnancement proposées ; le facteur robustesse et le facteur stabilité.

La robustesse de l'ordonnancement est généralement déterminée par la valeur du coût de changement de planning initial. Sevaux et Sörensen (Sevaux, 2004) ont défini la robustesse de solutions de l'ordonnancement par la propriété d'une solution dont sa qualité, mesurée par la valeur de la fonction objective, ne change pas trop par rapport à l'optimalité lorsqu'il y a de petits changements sur les données du problème.

Dans notre problème, le critère choisi pour évaluer l'efficacité des solutions, est basé sur la robustesse de l'ordonnancement trouvé vis à vie le temps total d'exécution. De ce fait, nous avons choisi le rapport (VE) qui détermine le pourcentage d'éloignement du makespan ($C_{max}(Réo)$) trouvée après le réordonnancement au makespen de l'ordonnancement initial ($C_{max}(Init)$). Ce rapport, montré par l'équation 22, définit le pourcentage du retard généré par la panne du convoyeur.

$$VE = \frac{C_{max}(Réo) - C_{max}(Init)}{C_{max}(Init)} \times 100 \quad (22)$$

De plus, pour bien étudier l'efficacité de l'approche proposée, nous avons choisi le facteur temps de calcul (CPU) qui détermine l'aspect temps réel.

V.6.3. Résultats et discutions

L'évaluation de la méthode de résolution temps réel, que nous avons proposé, a été effectuée sur trois types d'instances ; les petites, les moyennes et les grandes instances. Nous avons choisi trois problèmes différents pour chaque type d'instances.

Selon la variation du coefficient (**δ**), l'application a été faite sur cinq exemples différents pour chaque problème choisi. Ces applications sont implémentées sous forme de programmes en utilisant le logiciel Matlab.

a. Résultats obtenus pour les problèmes de petites tailles

Pour expérimenter notre approche sur des instances de petites tailles, nous avons choisi trois problèmes 2×4, 3×4 et 4×4. Les résultats trouvés par l'application sur les trois instances, sont présentés dans les histogrammes de la figure V.9 et la figure V.10. Ces histogrammes montrent successivement, les moyennes de VE et les moyennes de CPU qui sont calculés à travers cinq exemples différents de chaque taille de problème.

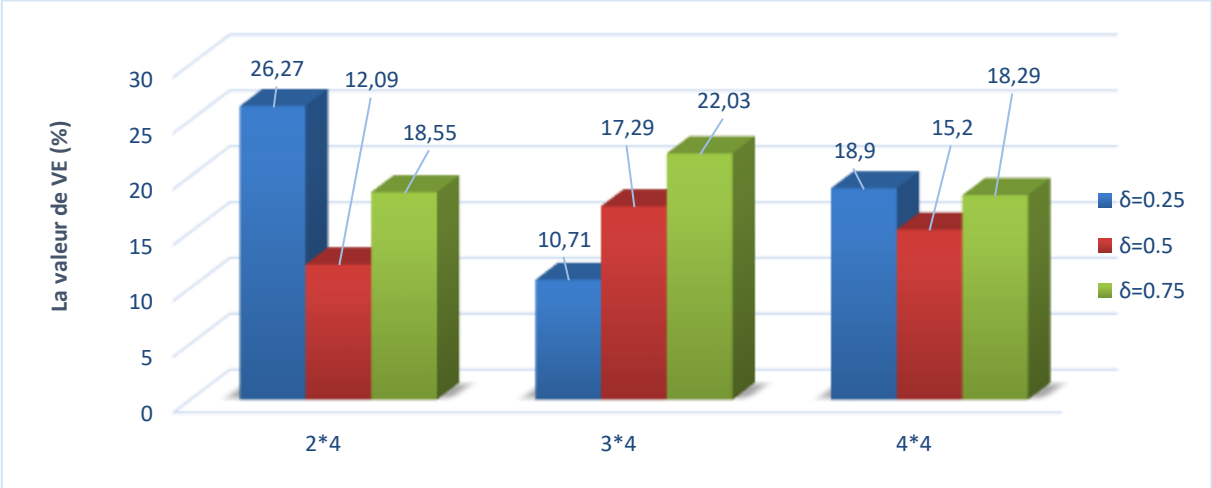


Figure V.9. La valeur du rapport (VE) trouvée pour les problèmes de petites tailles

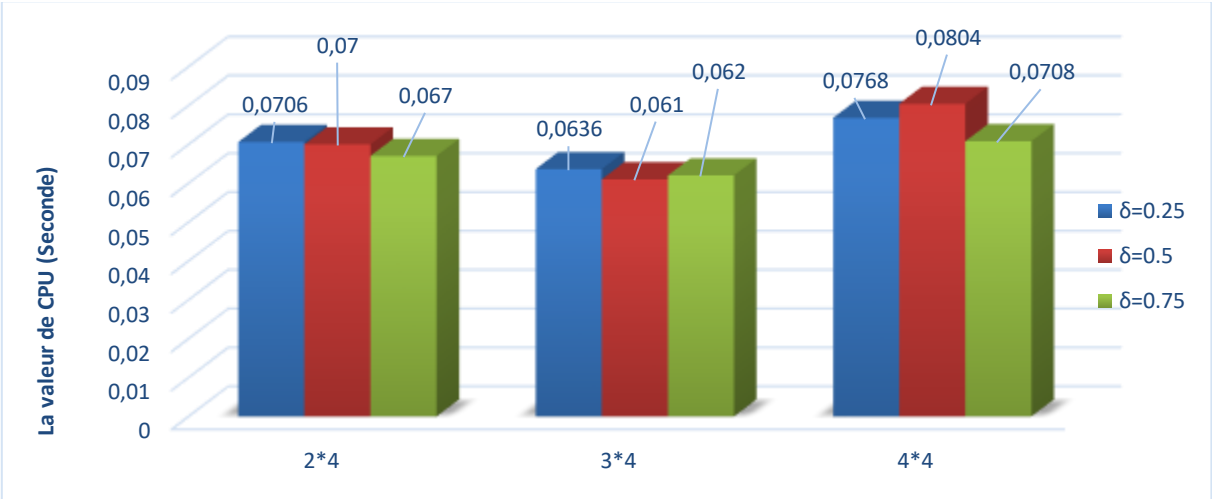


Figure V.10. La valeur du temps CPU calculée pour les problèmes de petites tailles

A travers ces résultats, nous pouvons remarquer que les valeurs calculées du rapport (VE) pour les trois problèmes, ne dépassent pas le 30%. Ces valeurs de robustesse démontrent très bien l'efficacité de l'approche proposée en termes de coût généré après la panne et qui ne représente que 1/3 du temps global de traitement des pièces défini dans la phase hors-ligne.

En termes de temps de calcul, nous remarquons, à travers les valeurs de CPU, que l'heuristiques JSSPTcum (basées sur les règles de priorité) a montré leur efficacité et leur rapidité vis à vie le temps de réaction.

b. Résultats obtenus pour les problèmes de moyennes tailles

Pour les instances de moyennes tailles, l'expertise a été faite sur les problèmes de 8×4, 10×4 et 15×4. La variation du temps de traitement des opérations dans ce cas, suit une loi uniforme entre 1 et 50. Le test de la méthode proposée sur cinq exemples donne les résultats mentionnés dans les histogrammes de la figure V.11 et la figure V.12.

Figure V.11. La valeur du rapport (VE) trouvée pour les problèmes de moyennes tailles

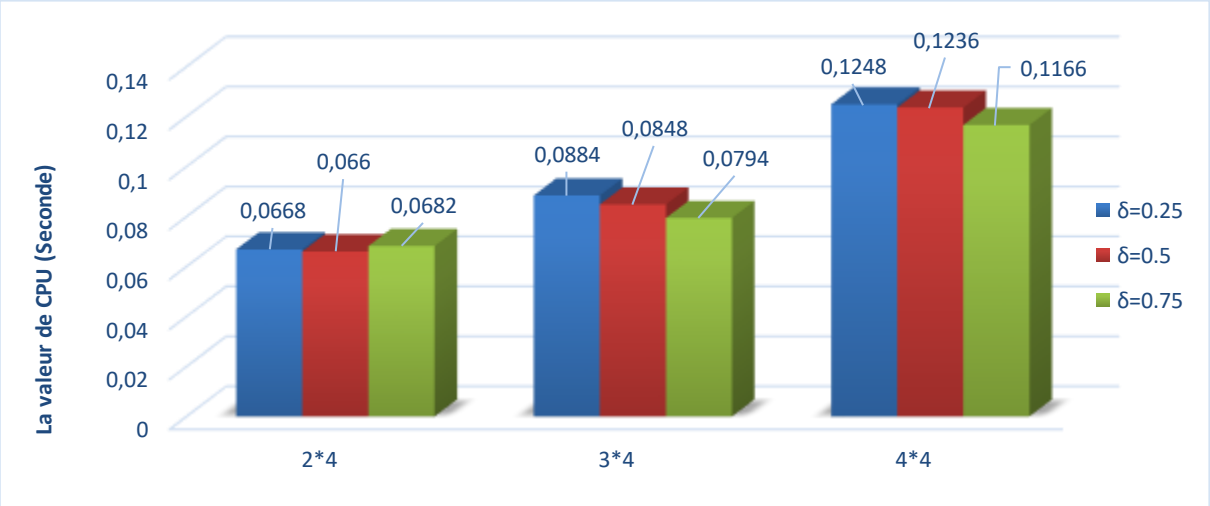


Figure V.12. La valeur du temps CPU calculée pour les problèmes de moyennes tailles

Les résultats mentionnés dans les deux figures montrent, encore une fois, l'efficacité de la méthode proposée. Nous remarquons que la moyenne de la valeur VE calculée pour les moyennes instances ne dépasse pas le 25% ce qui représente 1/4 du temps total initial.

c. *Résultats obtenus pour les problèmes de grandes tailles*

Les problèmes sélectionnés dans cette partie sont les problèmes de taille 20, 50 et 100 jobs qui doivent être réalisés sur les quatre machines. Comme dans la partie précédente, le temps de traitement suit une loi uniforme entre 1 et 50.

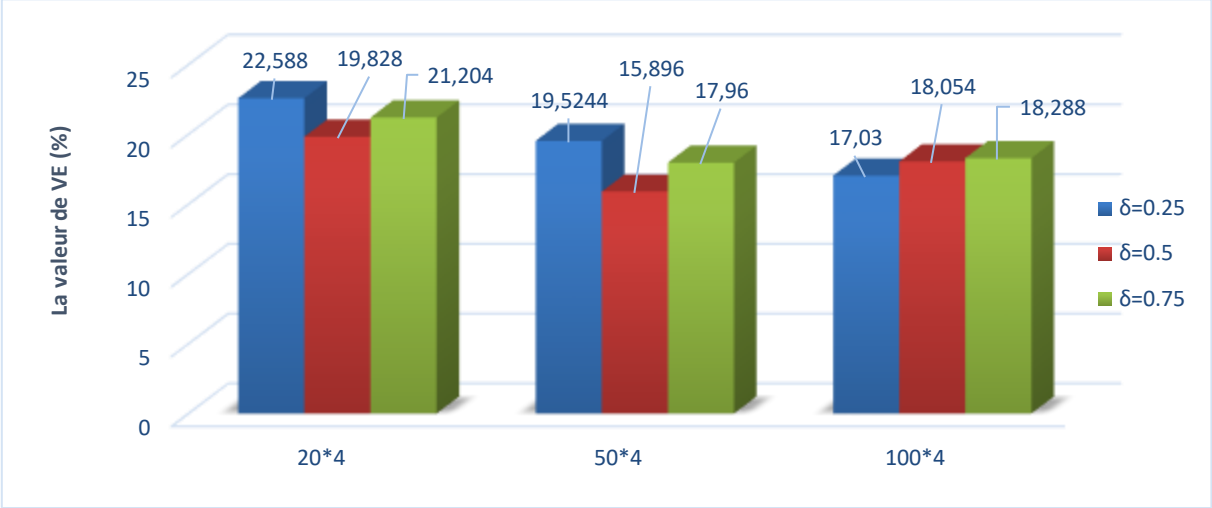


Figure V.13. La valeur du rapport (VE) trouvée pour les problèmes de grandes tailles

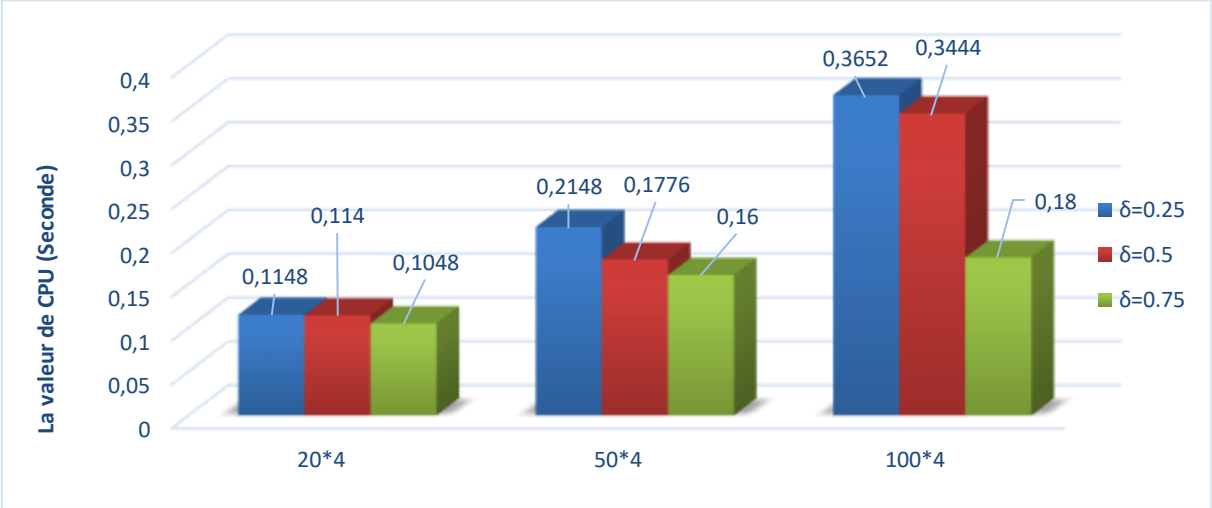


Figure V.14. La valeur du temps CPU calculée pour les problèmes de grandes tailles

A travers les résultats obtenus pour les instances de grandes tailles et pour les cinq tests effectués, nous remarquons très bien que l'utilisation de l'heuristique JSSPTcum a donné des très bons résultats.

V.7. Conclusion

Ce dernier chapitre a été consacré à la présentation de la méthode réordonnancement temps réel que nous avons proposé. Cette méthode consiste à trouver des solutions admissibles dans temps réduit au problème job shop en présence de panne de convoyeur. Plus l'aspect dynamique provoqué par la panne de système de transport, ce problème a été traité en considérant deux contraintes additionnelles.

L'évaluation de la méthode proposée, a été effectuée sur trois types d'instances ; les petites, les moyennes et les grandes instances. A travers les résultats obtenus nous avons remarqué que ces résultats sont très favorables vis à vie la robustesse de l'ordonnancement ainsi le temps de calcul, de ce fait on peut dire que la méthode proposée donne des solutions très satisfaisante dans un temps très court ce qui reflet l'aspect temps réel c'est-à-dire que dans des seconde le programme permet de trouver une solution qui peut être implémenter facilement dans le travail.

Conclusion générale

Dans ce travail, nous nous sommes intéressés à l'ordonnancement en temps réel dans un système de production flexible de type job shop.

Tous d'abord, nous avons donné, dans le premier chapitre, des généralités sur les systèmes de production flexibles, ses caractéristiques et ses différents types, ensuite nous avons présenté les différentes dimensions de flexibilité qui peuvent être existés dans les systèmes FMS. Après et dans le deuxième chapitre nous avons exposé une généralité sur l'ordonnancement et les problèmes d'ordonnancement, puis nous avons donné un aperçu sur les problème d'ordonnancement des ateliers en mettant l'actent sur les problèmes de type job shop. Enfin, nous avons illustré les approches de résolution utilisées pour résoudre ces problèmes. Dans le troisième chapitre nous avons entamé le problème d'ordonnancement job shop avec contrainte de ressources non renouvelables dont nous avons présenté au début le système de production iCIM 3000 (notre cas d'étude) et par la suite nous avons donné les dimensions du problème d'ordonnancement visé dans ce système.

En effet, un modèle mathématique a été développé et une méta-heuristique basée sur des algorithmes génétiques et plusieurs heuristiques ont été utilisées pour résoudre ce type de problème. L'objectif est de minimiser le temps d'exécution maximal (Makespan). Les heuristiques utilisées sont proposées à la base des règles de priorité comme la règle SPT, LPT, etc. Pour évaluer la performance de ces méthodes, plusieurs expériences avec des tailles différentes ont été faites. Les résultats obtenus sont ensuite comparés, sur des problèmes de taille réduite, avec des résultats obtenus en utilisant la méthode exacte basée sur la programmation mathématique. ses résultat ont fait l'objet d'une publication (Hadri, Bougloula, Belkaid, & Smadi). L'étude effectuée dans ce chapitre nous a permis de montrer, d'un part, que les heuristiques et l'algorithme génétique proposées sont capables de résoudre des problèmes de grandes tailles dans un temps raisonnable toute en maintenant la qualité de la solution à un niveau acceptable. D'autre part, que l'heuristique JSSPTcum basée sur l'ordre croissant du cumule de temps de traitement est l'approche la plus performante concernant la minimisation du makespan dans le problème job shop avec des ressources non-renouvelables.

Cependant, dans le quatrième chapitre nous avons étendu notre étude par la prise en compte de la contrainte de transport unidirectionnel (caractéristique du système iCIM 3000). Cette contrainte réside dans le faite que le déplacement des jobs s'effectuer dans un sens unique, c'est à dire que toutes les pièces suivent le même sens de déplacement. L'objectif consiste à trouver

les meilleurs séquencements des pièces (jobs) qui minimise makespen en tenant compte en même temps le déplacement des pièces et la disponibilité des ressources consommables. Pour résoudre ce problème nous avons proposé et utilisé d'autres heuristiques basée les règle de priorité en plus de l'heuristique JSSPTcum. La comparaison entre ces méthodes montre l'avantage de l'heuristique JSSPTcum sur les autres approches en terme de makespan sur un grand nombre de configurations différentes. L'étude menée dans ce chapitre nous a permis de dégager quelques résultats à savoir ;

- L'utilisation de la valeur cumulée dans le rapport de choix des solutions, présente un outil très efficace concernant l'identification du séquencement des jobs pour la minimisation de C_{max} de notre problème.
- Nous avons constaté que, plus la taille du problème augmente, plus les heuristiques utilisant l'ordre décroissant du rapport de choix devienne efficaces.
- Finalement, à travers les différentes études qui ont été réalisées, nous avons constaté que, l'approche qui reste la plus performante, c'est l'heuristique JSSPTcum qui dépasse les autres approches concernant toutes les taille du problème.

Dans l'étude mené dans le troisième et le quatrième chapitre, nous avons abordé le problème en supposant que les données du problème, sont toujours connues à l'avance et aucune perturbation n'est présent pour perturber l'exécution de l'ordonnancement retenu. Cependant, dans le chapitre cinq, nous nous sommes intéressés au problème de ré-ordonnancement en temps réel (en linge) qui consiste à établir un nouvel ordonnancement des jobs dans le cas d'apparition de panne de système de transport. En effet une démarche de deux étapes basé sur l'heuristique JSSPTcum a été présenté. Plusieurs expertises ont été effectuées pour évaluer la performance de la méthode proposée et les résultats de ces expériences ont montré que la méthode proposée donne des solutions très satisfaisante dans un temps très réduit, ce qui reflet l'aspect temps réel de l'ordonnancement.

Pour finir, nous pouvons dire que ces résultats ouvrent de nouvelles perspectives pour des travaux futures. Nous suggérons :

- D'ajouter la contrainte de recirculation des pièces au problème étudiée.
- De généraliser l'approche proposée pour les systèmes JSP flexible
- D'implanter cette approche sur le système réel.

Bibliographie

Ahmadi-Javid, A., & Hooshangi-Tabrizi, P. (2017). Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an Anarchic Society Optimization algorithm. *Computers & Operations Research, 84*, 73-91.

Ambrogio, G., Guido, R., Palaia, D., & Filice, L. (2020). Job shop scheduling model for a sustainable manufacturing. *Procedia Manufacturing, 42*, 538-541.

Artigues, C. (1997). *Ordonnancement en temps réel d'ateliers avec temps de préparation des ressources.* Université Paul Sabatier-Toulouse III.

Azem, S. (2010). *Ordonnancement des systèmes flexibles de production sous contraintes de disponibilité des ressources.* Thèse de doctorat, EMSE, Saint-Etienne.

Badiru, A. B., Ibidapo-Obe, O., & Ayeni, B. J. (2018). *Manufacturing and Enterprise: An Integrated Systems Approach.* New York: CRC Press.

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

Belkaid, F., Yalaoui, F., & Sari, Z. (2016). An efficient approach for the reentrant parallel machines scheduling problem under consumable resources constraints. *International Journal of Information Systems and Supply Chain Management, 9*(3), 1-25.

Belkaid, F., Yalaoui, F., & Sari, Z. (2016). Investigations on Performance Evaluation of Scheduling Heuristics and Metaheuristics in a Parallel Machine Environment. Dans C. Springer (Éd.), *Metaheuristics for Production Systems* (pp. 191-222).

Bellman, R. (1957). *Dynamic programming.* Princeton University Press.

Benkalai, I. (2018). *Ordonnancement d'ateliers en présence d'opérateurs.* Chicoutimi: Université du Québec.

Bilge, Ü., & Ulusoy, G. (1995). A time window approach to simultaneous scheduling of machines and material handling system in an FMS. *Operations Research, 43*(6), 1058-1070.

Billaut, J. C., & Roubellat, F. (1996). A new method for workshop real time scheduling. *International Journal of Production Research, 34*(6), 1555-1579.

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., Sterna, M., & Weglarz, J. (2019). Scheduling in Flexible Manufacturing Systems. Dans *Handbook on Schedulin* (pp. 671-711). Springer.

Blum, C., & Roli, A. (2003). Metaheuristics In Combinatorial Optimization: Overview And Conceptual Comparison. *ACM Computing Survey, 35*(3), 268-308.

Boukef, H. (2009). *Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques: optimisation par algorithmes génétiques et essaims particulaires.* Thèse de doctorat, Ecole centrale de Lille.

Bozorg-Haddad, O., Solgi, M., & Loáiciga, H. A. (2017). *Meta-heuristic and evolutionary algorithms for engineering optimization.* John Wiley & Sons.

Bronson, R., & Naadimuthu, G. (1997). *Schaum's Outline of Operation research.* McGraw Hill Professional.

Browne, J., Dubois, D., Rathmill, K., Sethi, S. P., & Stecke, K. E. (1984). Classification of flexible manufacturing systems. *The FMS magazine, 2*(2), 114-117.

Buddala, R., & Mahapatra, S. S. (2019). Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown. *The International Journal of Advanced Manufacturing Technology, 100*(5-8), 1419-1432.

Caumond, A. (2006). *Le problème de jobshop avec contraintes: modélisation et optimisation*. Thèse de doctorat.

Caumond, A. (2006). *Le problème job shop avec contrainte: modilisation et optimisation*. Thèse de doctorat, Université Blaise Pascal, Clermont-Ferrand II.

Chaari, T. (2011). *Un algorithme génétique pour l'ordonnancement robuste: application au problème du flow shop hybride.* Université de Valenciennes.

Chan, F. T., Chan, H. K., & Lau, H. C. (2002). The state of the art in simulation study on fms scheduling : a comprehensive survey. *International Journal of Advanced Manufacturing, 19*(11), 830–849.

Courtois, A., Martin-Bonnefous, C., & Pillet, M. (2003). *Gestion de production.* Les Ed. d'Organisation.

Dabaha, A., Bendjoudi, A., AitZaib, A., El-Bazc, D., & Taboudjemat, N. N. (2018). Hybrid Multi-core CPU and GPU-based B&B Approaches for the Blocking Job Shop Scheduling Problem. *Parallel and Distributed Computing, 117*, 73-86.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America, 2*(4), 393-410.

Delgado Sobrino, D. R., Košťál 2, P., & Vavruška, J. (2014). On the Analysis and Customization of an Icim 3000 System : a Take on the Material Flow, Its Complexity and a Few General Issues to Improve. *Applied Mechanics and Materials, 474*, 42-48.

Deng, G., Su, Q., Zhang, Z., Liu, H., Zhang, S., & Jiang, T. (2020). A population-based iterated greedy algorithm for no-wait job shop scheduling with total flow time criterion. *Engineering Applications of Artificial Intelligence, 88*, 103369.

Đurasević, M., & Jakobović, D. (2018). Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. *Genetic Programming and Evolvable Machines, 12*(1-2), 53-92.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. Dans *In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (pp. 39-43).

Eloundou, J. (2016). *Modélisation multi-contraintes d'un système de production flexible.* Thèse de doctorat, INSA de Rouen.

Eloundou, J. (2016). *Modélisation Multi-contraintes d'un Système de Production Flexible.* Thèse de doctorat, Rouen, INSA.

EMAG. (2020, Jun 19). *Machine Tools and Production Systems from EMAG*. Récupéré sur https://www.emag.com/

Erschler, J. F. (1986). Consistency of the disaggregation process in hierarchical planning. *Operations Research, 34*(2), 464-469.

Erschler, J., & De Terssac, G. (1988). *Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production.* Toulouse, France: Laboratoire d'Analyse et d'Architecture des Systèmes.

Espinouse, M. L., Pawlak, G., & Sterna, M. (2017). Complexity of scheduling problem in single-machine flexible manufacturing system with cyclic transportation and unlimited buffers. *Journal of Optimization Theory and Applications, 173*(3), 1042-1054.

Esquirol, P., Lopez, P., & Lopez, P. (1999). *l'ordonnancement.* Economica.

Eyers, D. R., Potter, A. T., Gosling, J., & Naim, M. M. (2018). The flexibility of industrial additive manufacturing systems. *International Journal of Operations & Production Management, 38*(12), 2313-2343.

Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. N. (2018). Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm. *IEEE transactions on cybernetics, 49*(5), 1944-1955.

Gendreau, M. &. (2019). *Handbook of metaheuristics* (Vol. 272). New York: Springer International Publishing AG.

Ghaleb, M., Zolfagharinia, H., & Taghipour, S. (2020). Real-time production scheduling in the Industry-4.0 context: Addressing uncertainties in job arrivals and machine breakdowns. *Computers & Operations Research, 123*, 105031.

Giard, V. (2003). *Gestion de la production et des flux.* Economica.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers operations research, 13*(5), 533-549.

Gondran, M., Huguet, M. J., Lacomme, P., Quilliot, A., & Tchernev, N. (2018). A Dial-a-Ride evaluation for solving the job-shop with routing considerations. *Engineering Applications of Artificial Intelligence, 74*, 70-89.

Gondran, M., Kemmoé-Tchomté, S., Lamy, D., & Tchernev, N. (Février, 2016). Une approche bi-objectif au problème du Job-shop sous contrainte de pics de consommation énergétique. *17ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision , 48.* France.

Gong, G., Deng, Q., Chiong, R., Gong, X., & Huang, H. (2019). An effective memetic algorithm for multi-objective job-shop scheduling. *Knowledge-Based Systems, 182*, 104840.

Grabowski, J., & Janiak, A. (1987). Job-shop scheduling with resource-time models of operations. *European Journal of Operational Rcsenrch, 28*(1), 58-73.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling : A Survey. *Annals of Discrete Mathematics, 5*, 287-326.

Graham, R., Lawler, E., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Math, 5*, 287-326.

Groover, M. P. (2016). *Automation, production systems, and computer-integrated manufacturing* (éd. 4th). Pearson Higher Education.

Gupta, Y. P., & Goyal, S. (1989). Flexibility of manufacturing systems: concepts and measurements. *European journal of operational research, 43*(2), 119-135.

Hadri, A., Bougloula, A., Belkaid, F., & Smadi, H. (s.d.). An efficient approach for solving a job shop scheduling problem with resources constraints: a case study iCIM 3000. *Int. J. Operational Research*.

Hassam, A. (2012). *Développement et analyse de méthodes d'ordonnancement temps réel pour les systèmes flexibles de production.* Université de Tlemcen.

Heger, J., & Voss, T. (2018). Optimal scheduling of AGVs in a reentrant blocking job-shop. *Procedia CIRP, 67*, 41- 45.

Heragu, S. S., & Kusiak, A. (1988). Machine layout problem in flexible manufacturing systems. *Operations research, 36*(2), 258-268.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press.

Homayouni, S. M., & Fontes, D. B. (2019). Joint scheduling of production and transport with alternative job routing in flexible manufacturing systems. *14th International Global Optimization Workshop* (p. 020045). AIP Conference Proceedings.

Hosseinabadi, A. A., Vahidi, J., Saemi, B., Sangaiah, A. K., & Elhoseny, M. (2019). Extended genetic algorithm for solving open-shop scheduling problem. *Soft computing, 23*(13), 5099-5116.

Houari, H. (2018). *L'intelligence artificielle pour résoudre les problèmes d'ordonnancement des systèmes flexibles de production.* Tlemcen: Université de Tlemcen.

Javel, G. (2010). *Organisation et gestion de la production-4e édition: Cours, exercices et etudes de cas.* Paris: Dunod.

Javel, G. (2010). *Organisation et gestion de la production-4e édition: Cours, exercices et etudes de cas.* Paris: Dunod.

Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly, 1*(1), 61-68.

Karimi, S., Ardalan, Z., Naderi, B., & Mohammadi, M. (2017). Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm. *Applied mathematical modellíng, 41*, 667-682.

Kaushal, A., Vardhan, A., & Rajput, R. S. (2016). Flexible manufacturing system a modern approach to manufacturing technology. *International Refereed Journal of Engineering and Science, 5*(4), 16-23.

Kebabla, M. (2008). *Utilisation des stratégies méta heuristiques pour l'ordonnancement des ateliers de type Job Shop.* Magister, Génie industriel, Batna.

Kemmoé, S., Lamy, D., & Tchernev, N. (2015). A Job-shop with an Energy Threshold Issue Considering Operations with Consumption Peaks. *International Federation of Automatic Control, 48*(3), 788-793.

Kouloughli, S. (2013). *Optimisation de systèmes de stockage/déstockage multiallées et à rack glissant.* Thèse de doctorat, Université de Tlemcen.

Ku, W. Y., & Beck, J. C. (2016). Mixed integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research, 73*, 165-173.

Kuhpfahl, J. (2015). *Job Shop Scheduling with Consideration of Due Dates: Potentials of Local Search Based Solution Techniques.* Springer.

Kumar, H., Kumar, P., & Sharma, M. (2019). A genetic algorithm for a flow shop scheduling problem with breakdown interval, transportation time and weights of jobs. *International Journal of Operational Research, 35*(4), 470-483.

Lacomme, P., Larabi, M., & Tchernev, N. (2013). Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics, 143*(1), 24-34.

Larabi, M. (2010). *Le problème de job-shop avec transport : modélisation et optimisation*. Clermont Ferrand II: Université Blaise Pascal.

Larabi, M. (2010). *Le problème de job-shop avec transport: modélisation et optimisation*. Clermont-Ferrand: Université Blaise Pascal.

Laribi, I. (2018). *Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables.* Université de Tlemcen.

Li, Y., Shi, S. Y., & Huang, Q. D. (2018). Three-machine job shop scheduling with intermediate transfer. *International Journal of Simulation Modelling, 17*(2), 359-368.

Lopez, P., & Roubellat, F. (2001). *Ordonnancement de la production.* Hermès science publications.

Luenberger, D. G. (2016). *Linear and Nonlinear Programming.* Springer.

Lunardi, W. T., Birgin, E. G., Laborie, P., Ronconi, D. P., & Voos, H. (2020). MixedInteger Linear Programming and Constraint Programming Models for the Online Printing Shop Scheduling Problem. *Computers & Operations Research*, 105020.

Luo, J., El Baz, D., Xue, R., & Hu, J. (2020). Solving the dynamic energy aware job shop scheduling problem with the heterogeneous parallel genetic algorithm. *Future Generation Computer Systems, 108*, 119-134.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing*, 106208.

Maache, Y. (2011). *Aide à la décision d'ordonnancement inter-entreprises dans le cadre des travaux de la maintenance Cas : Société de Maintenance de l'Est (SME).* Universite Hadj Lakhdar Batna.

Marmier, F. (2007). *Contribution à l'ordonnancement des activités de maintenance sous contrainte de compétence: une approche dynamique, proactive et multi-critère*. L'UNIVERSITÉ DE FRANCHE-COMTÉ.

Meng, L., Zhang, C., Ren, Y., Zhang, B., & Lv, C. (2020). Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering, 142*, 106347.

Meng, L., Zhang, C., Shao, X., & Ren, Y. (2019). MILP models for energy-aware flexible job shop scheduling problem. *Journal of cleaner production, 210*, 710-723.

Meng, L., Zhang, C., Shao, X., & Ren, Y. (2019). MILP models for energy-aware flexible job shop scheduling problem. *Journal of Cleaner Production, 210*, 710-723.

Merchichi, S. (2016). *approches de résolution exacte du problème de composition des cellules dans le système cellulaire de production.* Université de Boumerdes.

Meziane, M. (2018). *proposition d'une approche d'ordonnancement pour les ateliers de type job shop flexible.* Thèse de doctorat, Université d'oran.

Mohan, J., Lanka, K., & Rao, A. N. (2019). A review of dynamic job shop scheduling techniques. *Procedia Manufacturing, 30*, 34-39.

Muritiba, A. E., Rodrigues, C. D., & da Costa, F. A. (2018). A Path-Relinking algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research, 92*, 145-154.

Nagata, Y., & Ono, I. (2018). A Guided Local Search with Iterative Ejections of Bottleneck Operations for the Job Shop Scheduling Problem. *Computers and Operations Research, 90*, 60-71.

Nearchou, A. C. (2004). The effect of various operators on the genetic search for large scheduling problems. *International Journal of Production Economics, 88*(2), 191-203.

Nouri, H. E., Driss, O. B., & Ghédira, K. (2016). A classification schema for the job shop scheduling problem with transportation resources: state-of-the-art review. Dans *Artificial Intelligence Perspectives in Intelligent Systems* (pp. 1-11). Cham: Springer.

Ourari, S. (2011). *De l'ordonnancement déterministe à l'ordannancement distribué sous incertitudes.* Toulouse: Université Paul Sabatier-Toulouse III.

Ozturk, G., Bahadir, O., & Teymourifar, A. (2019). Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming. *International Journal of Production Research, 57*(10), 3121-3137.

Pei, Z., Zhang, X., Zheng, L., & Wan, M. (2020). column generation-based approach for proportionate flexible two-stage no-wait job shop scheduling. *nternational Journal of Production Research, 58*(2), 487-508.

Peng, C., & Chen, F. F. (1998). Real-time control and scheduling of flexible manufacturing systems: an ordinal optimisation based approach. *The International Journal of Advanced Manufacturing Technology, 14*(10), 775-786.

Pinedo, M. (2012). *Scheduling: Theory, Algorithms and Systems* (éd. Fifth). New York: Springer.

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems.* New York: Springer.

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems.* New York: Springer.

Poppenborg, J., Knust, S., & Hertzberg, J. (2012). Online scheduling of flexible job-shops with blocking and transportation. *European Journal of Industrial Engineering, 6*(4), 497-518.

Portmann, M. C. (1988). Méthodes de décomposition spatiales et temporelles en ordonnancement. *RAIRO-APII, 22*(5), 439-431.

Québec. (1996). *Lexique des convoyeurs et des transporteurs : terminologie technique et industrielle.* Bibliothèque nationale du Québec.

Rajendran, S., Rajendran, C., & Leisten, R. (2017). Heuristic rules for tie-breaking in the implementation of the NEH heuristic for permutation flow-shop scheduling. *International Journal of Operational Research, 28*(1), 87-97.

Rashid, R., Arani, S. D., Hoseini, S. F., & Omran, M. M. (2018). A new supply chain network design approach, regarding retailer's inventory level and supplier's response time. *International Journal of Operational Research, 31*(4), 421-241.

Rifai, A. P., Dawal, S. Z., Zuhdi, A., Aoyama, H., & Case, K. (2016). Reentrant FMS scheduling in loop layout with consideration of multi loading-unloading stations and shortcuts. *The International Journal of Advanced Manufacturing Technology, 82*(9-12), 1527-1545.

Rong, Z. (2008). *Dynamic Job Shop Scheduling Using Ant Colony Optimization Algorithm Based On A Multi-Agent System*. Université de Technologie Chine du Sud.

Roy, B., & Sussmann, B. (1964). *Les problèmes d'ordonnancement avec contraintes disjonctives.* SEMA.

Salido, M. A., Escamilla, J., Barber, F., Giret, A., & Tang, D. (2016). Energy efficiency, robustness, and makespan optimality in job-shop scheduling problems. *AI EDAM, 30*(3), 300-312.

Sari, Z. (2003). *Modélisation, analyse et évaluation des performances d'un AS/RS à convoyeur gravitationnel.* Thèse de doctorat, Université de Tlemcen.

Satheesh Kumar, R. M., Asokan, P., Kumanan, S., & Varma, B. (2008). Scatter search algorithm for single row layout problem in fms. *Advances in Production Engineering & Management, 3*(4), 193-204.

Saygin, C., & Kilick, S. E. (1996). Effect of flexible process plans on performance of flexible manufacturing systems. Dans *proceedings of 7th International DAAM symposium* (pp. 393-394). Vienna, Austria .

Seidgar, H., Rad, S. T., & Shafaei, R. (2017). Scheduling of assembly flow shop problem and machines with random breakdowns. *International Journal of Operational Research, 29*(2), 273-293.

Sevaux, M. &. (2004). A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 2*(2), 129-147.

Shivanand, H. K., Benal, M., & Koti, V. (2006). *Flexible manufacturing system*. New Age International.

SOUIER, M. (2012). *Investigations sur la sélection de routages alternatifs en temps réel basées sur les métaheuristiques-les essaims particulaires.* Thèse de doctorat , Université de Tlemcen.

Sousa, J. P., & Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical programming, 45*(1-3), 353-36.

Subramaniam, V., Raheja, A. S., & Rama Bhupal Reddy, K. (2005). Reactive repair tool for job shop. *International Journal of Production Research, 43*(1), 1-23.

Suniya, N. K. (2013). *Analysis and Modeling of Flexible Manufacturing System*. Rourkela: National Institute of Technology.

Suresh, V., & Chaudhuri, D. (1993). Dynamic scheduling—a survey of research. *International journal of production economics, 32*(1), 53-63.

Tabrizi, B. H., Ghaderi, S. F., & Haji-Yakhchali, S. (2019). Net present value maximisation of integrated project scheduling and material procurement planning. *International Journal of Operational Research, 34*(2), 285-300.

Taghezout, N. (2011). *conception et développement d'un système multi-agent d'aide à la décision pour la gestion de production dynamique.* Université Toulouse III.

Talavage, J. (1987). *Flexible manufacturing systems in practice: design: analysis and simulation.* CRC Press.

Tempelmeier, H., & Kuhn, H. (1993). *Flexible manufacturing systems: decision support for design and operation.* John Wiley & Sons.

Tetzlaff, U. A. (1990). *Optimal Design of Flexible manufacturing systems.* Heidelberg: Physica.

Tian, S., Wang, T., Zhang, L., & Wu, X. (2019). Real-time shop floor scheduling method based on virtual queue adaptive control: Algorithm and experimental results. *Measurement, 147*, 106689.

Toker, A., Kondakci, S., & Erkip, N. (1991). Scheduling under a non-renewable resource constraint. *Journal of the Operational Research Society, 42*(9), 811–814.

Toker, A., Kondakci, S., & Erkip, N. (1994). Job Shop Scheduling under a Non-Renewable Resource Constraint. *Journal of the Operational Research Society, 45*(8), 942–947.

Turker, A. K., Aktepe, A., Inal, A. F., Ersoz, O. O., Das, G. S., & Birgoren, B. (2019). A decision support system for dynamic job-shop scheduling using real-time data with simulation. *Mathematics, 7*(3), 278.

Ulusoy, G., & Bilge, Ü. (1992). Simultaneous scheduling of machines and material handling system in an FMS. *IFAC Proceedings Volumes, 25*(8), 15-25.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering, 58*(4), 785-800.

Valledor, P., Gomez, A., Priore, P., & Puente, J. (2018). Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions. *International Journal of Production Research, 56*(19), 6363-6377.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of scheduling, 6*(1), 39-62.

Wang, D. J., Liu, F., & Jin, Y. (2019). A proactive scheduling approach to steel rolling process with stochastic machine breakdown. *Natural Computing, 18*(4), 679-694.

Wang, Y., Yang, O., & Wang, S. N. (2019). A solution to single-machine inverse job-shop scheduling problem. *International Journal of Simulation Modelling, 18*(2), 335-343.

Wang, Z., Zhang, J., & Yang, S. (2019). An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. *Swarm and Evolutionary Computation, 51*, 100594.

Xiong, H., Fan, H., Jiang, G., & Li, G. (2017). A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *European Journal of Operational Research, 257*(1), 13-24.

Yahouni, Z. (2019). *Le meilleur des cas pour l'ordonnancement de groupes: Un nouvel indicateur proactif-réactif pour l'ordonnancement sous incertitudes.* Thèse de doctorat, Unioversité de Tlemcen.

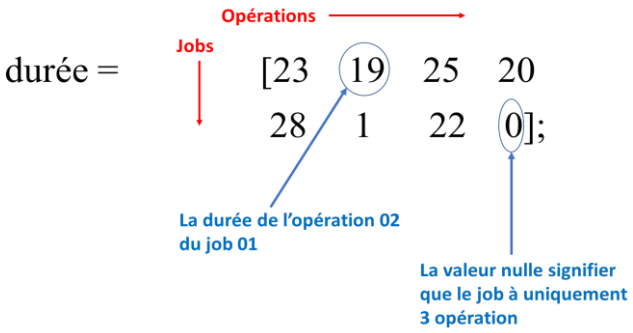
Zhang, L., Gao, L., & Li, X. (2013). A hybrid intelligent algorithm and rescheduling technique for job shop scheduling problems with disruptions. *The International Journal of Advanced Manufacturing Technology, 65*(5-8), 1141-1156.

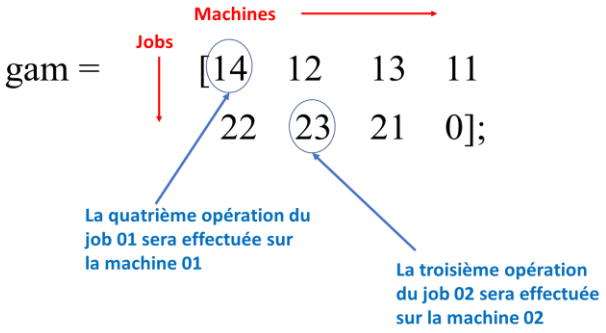
Zhang, S., & Wang, S. (2018). Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules. *IEEE Transactions on Engineering Management, 65*(3), 487-504.

Zhang, S., & Wong, T. N. (2017). Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. *International Journal of Production Research, 55*(11), 3173-3196.

Zhou, D., Cherkassky, V., Baldwin, T., & Olson, D. (1991). A neural network approach to job-shop scheduling. *IEEE Transactions on Neural Networks, 2*(1), 175 - 179.

Annexe A





Problème 2*4

Ex	Durée	Gamme	Quantité demandé
1	duree = [23 19 25 20 28 1 28 22];	gam = [14 12 13 11 22 24 23 21];	QC = $\begin{bmatrix}13\\6\end{bmatrix}$
2	duree = [11 23 14 19 22 5 13 21];	gam = [12 14 13 11 22 21 23 24];	QC = $\begin{bmatrix}4\\15\end{bmatrix}$
3	duree = [20 28 4 7 26 16 4 25];	gam = [13 11 12 14 22 23 21 24];	QC = $\begin{bmatrix}16\\11\end{bmatrix}$
4	duree = [8 22 17 1 22 11 2 15];	gam = [13 14 11 12 22 24 23 21];	QC = $\begin{bmatrix}10\\03\end{bmatrix}$
5	duree = [19 22 2 27 20 13 6 4];	gam = [14 12 11 13 24 22 21 23];	QC = $\begin{bmatrix}17\\1\end{bmatrix}$;

Problème 3*4

Ex	Duree	Gamme	Quantité demandé
1	duree = [24 27 8 28 27 18 16 4 3 2 28 29];	gam = [14 12 13 11 21 23 22 24 32 31 33 34];	QC = $\begin{bmatrix}7\\13\\3\end{bmatrix}$
2	duree = [1 20 1 22 2 9 13 23 24 28 11 5];	gam = [12 11 13 14 22 24 23 21 32 31 33 34];	QC = $\begin{bmatrix}15\\5\\10\end{bmatrix}$
3	duree = [28 4 7 27 16 7 24 10 4 25 7 5];	gam = [11 14 13 12 23 22 21 24 31 34 33 32];	QC = $\begin{bmatrix}10\\0\\0\end{bmatrix}$
4	duree = [14 4 15 7 0 23 4 19 10 9 18 20];	gam = [13 14 12 11 0 22 23 21 33 34 31 32];	QC = $\begin{bmatrix}15\\16\\0\end{bmatrix}$

Résumé

Dans l'environnement réel des systèmes de production, l'exécution des travaux selon un planning prédéfini est presque impossible à cause de plusieurs types de perturbations (internes et/ou externes) qui empêchent le déroulement normal de ces travaux. Dans ce cadre notre thèse s'intéresse à l'étude du problème d'ordonnancement en temps réel d'un atelier job shop avec la présence de panne du système de transport. Nous abordons, tout d'abord, le problème job shop dans leur contexte statique et avec deux contraintes additionnelles. Afin de résoudre ce problème nous développons un programme linéaire en nombres entiers et nous proposons une méta-heuristique basé sur les algorithmes génétiques et plusieurs heuristiques basées sur les règles de priorité. Nous nous intéressons par la suite au problème de ré-ordonnancement en temps réel qui consiste à établir un nouvel ordonnancement des jobs dans le cas d'apparition de la panne de système de transport. Une méthode d'ordonnancement/ré-ordonnancement est proposé afin de trouver le meilleur ordonnancement qui minimise le coût générer par cet évènement perturbant.

Mots clé : Job shop ; Ordonnancement en temps réel ; Heuristiques ; Méta-heuristiques.

Abstract

In the real environment of production systems, executing jobs according to a predefined schedule is almost impossible due to several types of disturbances (internal and / or external) that prevent the normal processing of these jobs. In this context, our thesis focuses on the study of the real-time scheduling problem of a job shop workshop with the presence of a transport system failure. We first address the job shop problem in their static context and with two additional constraints. In order to solve this problem, we develop a linear integer program and we propose a meta-heuristic based on genetic algorithms and several heuristics based on priority rules. We are then interested in the real-time rescheduling problem which consists in creating a new job scheduling in the moment of the occurrence of the transport system failure. A scheduling / re-scheduling method is proposed in order to find the best scheduling that minimizes the cost generated by this disturbing event.

Keywords: Job shop; Real-time scheduling; Heuristics; Meta-heuristics.

ملخص

في البيئة الحقيقية لأنظمة الإنتاج، يكاد يكون من المستحيل تنفيذ العمل وفقًا لجدول زمني محدد مسبقًا بسبب عدة أنواع من الاضطرابات (الداخلية و / أو الخارجية) التي تمنع التقدم الطبيعي لهذه الأعمال. في هذا الإطار، تهتم أطروحتنا بدراسة مشكلة الجدولة في الوقت الفعلي لورشة عمل من نوع (متعدد المسارات) مع وجود عطل في نظام النقل. نتعامل أولاً مع مشكلة الجدولة في سياقها الثابت ومع قيدين إضافيين ومن أجل حل هذه المشكلة، قمنا بتطوير برنامج عدد صحيح خطي واقترحنا طريقة تتأسس على الخوارزميات الجينية والعديد من الطرق التقريبية مرتكزة على أساس قواعد الأولوية. وبعد ذلك درسنا مشكلة إعادة الجدولة في الوقت الفعلي والتي تتمثل في إنشاء جدولة جديدة في لحظة حدوث فشل نظام النقل. تم اقتراح طريقة الجدولة / إعادة الجدولة من أجل العثور على أفضل الحلول التي تقلل التكلفة الناتجة عن هذا الحدث المعرقل.

الكلمات المفتاحية: متعدد المسارات؛ جدولة الوقت الحقيقي؛ الاستدلال؛ الاستدلال الفوقي.