Table des matières

Table des figures

Rapport-Gratuit.co

Liste des tableaux

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Liste des Acronymes

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AGV	Automatic Guided Vehicle
AI	Artificial Intelligence
$AIP - PRIMECA$	Atelier Inter-établissement de Productique (Pôle de Ressources Informatique pour la Mécanique)
AL	Approche par localisation
CSM	Conic Scalarization Method
DML	Dedicated Manufacturing Lines
EDD	Earliest Due Date
ELS	Evolutionary Local Search
ERP	Entreprise Resource Planning
FJS	Flexible Job Shop
$FJSP$	Flexible Job Shop Problem
FMC	Flexible Manufacturing Cell(ou Cellule de Production Flexible)
FMS	Flexible Manufacturing System
FMU	Flexible Manufacturing Unit
GH	Greedy Heuristtic (ou Heuristique gloutonne)
$GRASP$	Greedy Randomized Adaptive Search Procedure
$IGIRT$	Iterated Greedy Insertion Randomized Technique
$IGIT$	Iterated Greedy Insertion Technique

ILP	Integer Linear Programming
IMS	Intelligent Manufacturing System
$INLP$	Integer Non-Linear Programming
ISA	Integrated Simulated Annealing
ITS	Integrated Tabu Search
$KBACO$	Knowledge-Based Ant Colony Optimization
LPT	Longest Processing Time
$MCFMS$	Multi-cell Flexible Manufacturing System (ou Système de Production flexible multi-cellule)
MES	Manufacturing Execution System
$MILP$	Mixed Integer Linear Programming
$MMFMS$	Multi-machine Flexible Manufacturing System (ou Système de Production flexible multi-machine)
$MOCN$	Machines Outil à Commande Numérique
$MOGA$	Multiple Objective Genetic Algorithm
NGA	Neighborhood-based Genetic Algorithm
$NPGA$	Niched-Pareto Genetic Algorithm
$NSGA$	Non-dominated Sorting Genetic Algorithm
$ORCA - FMS$	Architecture for an Optimized and Reactive Control for a Flexible Manufacturing Systems
$PESA$	Pareto Envelope-based Selection Algorithm
PF	Potential Fields (ou Champs Potentiels)
PSO	Paricle Swarm Optimization
RCL	Restricted Candidate List
RMS	Reconfigurable Manufacturing System
SFM	Single Flexible Machine (ou Machine Flexible Unique)
$SPEA$	Strength Pareto Evolutionary Algorithm
SPT	Shortest Processing Time
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search

INTRODUCTION GÉNÉRALE

L'environnement actuel des entreprises est caractérisé par des marchés confrontés à une forte concurrence et une clientèle de plus en plus exigeante en matière de qualité, de coût et de délais de livraison. Cette évolution est davantage renforcée par le développement rapide de nouvelles technologies de l'information et de la communication qui assurent une connexion directe entre les entreprises (entreprise à entreprise B2B) et entre les entreprises et leurs clients (entreprise à client B2C). Dans ce type de contexte, la performance de l'entreprise repose sur deux dimensions :

- **Dimension technologique**, où l'objectif est de développer les performances intrinsèques des produits commercialisés afin de répondre aux exigences de qualité et de réduction du coût de possession de ces produits. L'innovation technologique joue un rôle important et peut être un élément crucial pour le développement et l'obtention de marchés. Par ailleurs, la croissance technologique rapide des produits et les exigences de personnalisation de ces produits attendues par les marchés amènent souvent les entreprises à abandonner la production de masse pour s'orienter vers la production de petites et moyennes séries, voire sur la demande. Cela nécessite des systèmes de production flexibles, capables de s'adapter rapidement et efficacement aux demandes et aux besoins du marché ;

- **Dimension organisationnelle** : destinée au développement des performances en termes de temps de production, de respect des délais de livraison, d'adaptation et de réactivité aux variations des commandes commerciales, etc. Le rôle de cette dimension croît dans la mesure où les marchés sont de plus en plus volatils et progressifs et nécessitent des délais de réponse plus courts de la part des entreprises.

Par conséquent, les entreprises doivent disposer de méthodes et d'outils puissants pour l'organisation et le contrôle de la production pour satisfaire les besoins de ses clients dans les meilleures conditions.

Pour atteindre ces objectifs, une organisation d'entreprise repose sur la mise en œuvre d'un certain nombre de fonctions, dont l'ordonnancement qui est au cœur des systèmes de production.
En effet, la fonction d'ordonnancement a pour but d'organiser l'utilisation des ressources (humaines/technologiques) dans les ateliers d'entreprise pour répondre directement aux exigences des clients. Cette fonction doit organiser l'exécution simultanée de plusieurs tâches en utilisant des ressources disponibles en quantité limitée tout en tenant compte des tendances et des exigences du marché, ce qui en fait un problème complexe à résoudre. Bien que cette fonction ait toujours existé au sein des entreprises, elle doit faire face aujourd'hui à des contraintes de plus en plus complexes en raison des objectifs économiques, sociaux et environnementaux à atteindre. Malgré les recherches effectuées dans ce domaine, il n'existe pas à ce jour de méthode d'ordonnancement «unique» applicable à toutes sortes de scénarios possibles. Il existe un certain nombre de problèmes génériques différenciés en fonction des caractéristiques des tâches à exécuter ou des ressources disponibles dans l'atelier. Des méthodes spécifiques peuvent alors être associées à la résolution de chacun de ces problèmes génériques. Ces méthodes sont soit une interprétation du problème d'une manière générale, soit une méthode spécifique dédiée au problème posé.

La flexibilité du marché est importante pour la prospérité d'une entreprise dans des environnements en constante évolution. Ces changements sont généralement le résultat d'innovations technologiques rapides, de changements dans les goûts des clients, des courts cycles de vie des produits, de l'incertitude dans les sources d'approvisionnement, etc. Pour ces raisons, le fait de basculer vers des systèmes de production flexibles est le meilleur choix s'offrant aux entreprises afin de répondre aux changements et à l'évolution du marché sans compromettre l'activité ni ralentir le rendement.

Un système de production flexible (Flexible Manufacturing System, FMS) consiste en l'agencement de plusieurs machines à commande numérique (Computer Numerical Control, CNC) reliées entre elles via un système de transport et de chargement automatisés des pièces et des outils, l'ensemble étant contrôlé et coordonné par un ordinateur central. Un tel système combine donc la flexibilité individuelle (les machines CNC) à une flexibilité accrue du système de manutention. Le FMS est caractérisé par la disponibilité de ressources alternatives permettant d'accroitre les performances du système, gérer la maintenance préventive et faire face aux pannes et aux évènements imprévus. Ainsi, une opération peut être réalisée par plusieurs ressources et chaque ressource peut proposer divers services. Ce type de problème est considéré par

la communauté de Recherche Opérationnelle comme un Flexible Job-Shop Problem (FJSP) avec des contraintes supplémentaires associées par exemple aux files d'attente des ressources et aux temps de transport.

Le problème d'ordonnancement dans le job shop flexible (FJSP) est une extension du problème d'ordonnancement du job shop (JSP) classique. La flexibilité peut être considérée de différentes manières. Selon, Tsubone [2], les types de flexibilité les plus étudiés sont *(i)* la flexibilité de la machine : la capacité d'une machine à exécuter différentes opérations sur un ensemble donné de types de pièces *(ii)* la flexibilité du routage : différents itinéraires existent pour la fabrication du même type de pièces (routage alternatif).
Ainsi, compte tenu de la nature NP-difficile du JSP [3] combinée à la flexibilité du FJSP, il a été prouvé que ce problème est fortement NP-difficile et combinatoire [4] [5] [6]. Ainsi, compte de cette complexité, l'emploi d'approches heuristiques est favorisé par rapport aux techniques mathématiques traditionnelles.
En effet, les problèmes d'optimisation difficile sont définis comme étant des problèmes qui ne peuvent être résolu d'une manière optimale ou garantir une limite par une méthode exacte (déterministe) dans un délai raisonnable. Pour trouver des solutions satisfaisantes à ces problèmes, des méthodes approchées sont employées. Ces méthodes nommées heuristiques/métaheuristiques représentent des algorithmes conçus pour résoudre approximativement un large éventail de problèmes d'optimisation difficile (ordonnancement, sac à dos, voyageur de commerce, routage, ...).
L'objectif principal de notre thèse est de proposer des méthodes de résolution pour le problème d'ordonnancement dans un système de production flexible de type job shop flexible avec contrainte de temps de transport.

1. Objectifs et Contributions de la thèse

- L'incorporation des temps de transports dans les job shops flexibles
- La proposition de deux approches combinant une méthode d'insertion et heuristiques gloutonnes itérées.
- Les deux approches proposées ont été testées sur un Benchmark spécialisée prenant en compte les temps de transport.
- Les deux approches proposées ont été évaluées et comparées à d'autres méthodes et les résultats obtenus démontrent leurs hautes performances.

Ces travaux ont fait l'objet d'une publication et d'une communication dans une conférence :
A. Bekkar, O. Guemri, A. Bekrar, N. Aissani, B. Beldjilali, D. Trentesaux (2016).

An Iterative Greedy Insertion Technique for Flexible Job Shop Scheduling Problem. IFAC-PapersOnLine, volume 49, issue 12, pp. 1956-1961.

A. Bekkar, G. Belalem, B. Beldjilali (2019). *Iterated Greedy Insertion Approaches for the Flexible Job Shop Scheduling Problem with transportation times constraint.* International Journal of Manufacturing Research, volume 14, issue 1, pp. 43-66.

2. Structure de la thèse

Cette thèse est composée de six chapitres organisés de la manière suivante :

Chapitre 1 :
Ce chapitre a pour but d'introduire les systèmes de production flexibles. Pour cela, nous commencerons par présenter les caractéristiques générales des systèmes de production avant d'aborder les notions fondamentales et nécessaires à cette thèse. Ensuite, nous citerons les différents type de système de production rencontrés en industrie, avant d'enchainer avec une étude détaillée des systèmes de production flexibles ainsi que toutes les notions industrielles touchées par la flexibilité. Puis, nous définissons le pilotage de production avant de conclure avec une étude comparative entre les différentes architectures de pilotage.

Chapitre 2 :
Ce chapitre sera divisé en deux grandes parties :
La première partie du chapitre sera consacrée à la fonction d'ordonnancement dans les systèmes de production. Nous commencerons par définir la fonction d'ordonnancement de manière générale et introduire ses concepts de base. Ensuite, nous évoquons les différentes classes d'ordonnancement. Puis, nous énumérons les types d'ordonnancement d'atelier les plus récurrents et rencontrés dans la littérature.
Dans la seconde partie du chapitre, nous nous concentrons sur la fonction d'ordonnancement dans un type de système de production spécifique : le *Job Shop Flexible.* En plus d'avoir la particularité d'être flexible (les opérations peuvent être traitées par plus d'une ressource), le cheminement des opérations n'est pas linéaire ce qui rend la fonction d'ordonnancement dans ce type d'atelier plus complexe et en fait un des problèmes les plus intéressants à étudier et l'un des plus rencontrés dans la littérature concernant l'optimisation.
Par conséquent, cette partie présentera les différentes approches de résolution présentes dans la littérature ainsi qu'un état de l'art sur la résolution du FJSP.

Chapitre 3 :
Nous abordons une variante du problème d'ordonnancement dans le job shop flexible qui est le problème du job shop flexible avec temps de transport. Nous présenterons pour ce problème, un état de l'art élargi englobons divers types d'ateliers avant de nous focaliser sur les job shops flexibles et les principaux travaux menés dans cet axe.

Chapitre 4 :
Ce chapitre définit les instances du Benchmark de la cellule flexible AIP-PRIMECA de l'Université de Valenciennes et du Hainaut Cambrésis, qui a pour particularité de se baser sur un système de production flexible réel. Nous commencerons par schématiser et définir les spécificités de cet atelier flexible (ressource, jobs, opérations, système de transport, temps de transport entre les machines, etc). Puis, nous définirons les paramètres, variables, hypothèses et contraintes prises en considération lors de la modélisation mathématique du problème d'ordonnancement.

Chapitre 5 :
Nous présentons dans ce chapitre la première des deux méthodes proposées à savoir la *Iterated Greedy Insertion Technique (IGIT)* [7]. Méthode résultante de la combinaison d'une technique d'insertion et d'une heuristique gloutonne itérée. Nous dresserons un bref état de l'art sur les approches présentées dans la littérature ayant utilisées ces deux heuristiques pour la résolution de problème d'ordonnancement. Ensuite, nous présenterons l'algorithme en détail ainsi que son fonctionnement. Puis, les résultats obtenus sur les instances du Benchmark présenté dans le chapitre précédent seront présentés, comparés, illustrés et discutés.

Chapitre 6 :
Dans ce chapitre nous exposons la seconde approche proposée à savoir la *Iterated Greedy Insertion Randomized Technique (IGIRT)* [8]. Nous commencerons par définir et présenter. Ensuite, nous schématiserons son fonctionnement. Puis, tout comme le chapitre précédent, les résultats seront présentés, illustrés, analysés et comparés afin de souligner les avantages de l'approche IGIRT. Enfin, nous expliquerons comment nous sommes parvenus, grâce à cette méthode, à améliorer l'approche IGIT.
Enfin, une discussion sera proposée sur les éléments requis pour améliorer nos méthodes afin de nous permettre d'orienter les principales perspectives de cette thèse.
Nous concluons cette thèse, en rappelant les contributions apportées avec les deux approches proposées dans le domaine du FJSP et proposons des perspectives de recherches envisageables à court, moyen et long termes.

Chapitre 1

Les systèmes de production flexibles

Sommaire

1.1 Introduction

L'objectif de ce chapitre est de contextualiser notre sujet de thèse et de poser les bases de la problématique.
Une étude récente indique que la communauté des systèmes de production intelligents orientent les recherches actuelles sur trois axes fondamentaux [9] :

- **Production rapide et adaptative centrée sur l'utilisateur :**
  Les systèmes de production doivent avoir la capacité de répondre aux besoins du client, en termes de personnalisation du produit/service dans de courts délais. Ce qui implique une forte capacité d'adaptation et de réactivité du système de production aux besoins du marché.
- **Système de production hautement flexible et reconfigurable :**
  Les systèmes de production doivent être capables de s'adapter et de s'organiser aux niveaux architectural (pilotage de production) et physique (positionnement des machines, système de transport des produits) afin de faciliter la production de nouvelles gammes de produits.
- **Production durable tenant compte des changements culturels de la société et des entreprises :**
  Cela implique les consommations et les économies d'énergie ainsi que la relation entre l'industrie et l'environnement.

Ces contextes ont poussé les chercheurs et industriels à développer des systèmes de pilotage et de contrôle de production capables de réagir efficacement mais aussi en évolution permanente pour améliorer leurs performances et la qualité des produits proposés. Ainsi, ces systèmes doivent pouvoir exploiter au mieux les ressources pour produire le maximum dans les plus brefs délais et à moindre coût.
Dans cette optique, il nous semble important de présenter les systèmes de production flexibles et la manière dont ils sont pilotés. Pour cela, nous commençons ce chapitre par certaines caractéristiques du système de production d'une manière générale, avant d'introduire la notion de flexibilité et les différents niveaux auxquels elle peut agir lors du processus de fabrication. Puis, nous définissons le pilotage du système et les différentes fonctions inhérents à ce système, avant de clôturer le chapitre avec les architectures de pilotages les plus importantes et fréquemment rencontrées en industrie ainsi qu'une étude comparative entre elles.

1.2 Définition d'un système de production

La production concerne le processus de transformation de la matière première en un produit fini. Une gestion efficace du système de production est nécessaire à l'entreprise pour atteindre ses objectifs tout en optimisant le rendement et en réduisant les couts.
Le système de production peut être divisé en trois sous-systèmes : physique, informationnel et décisionnel [10].

1. **Le sous-système physique :**

   Ce sous-système est composé de l'ensemble des dispositifs physiques du système de production : machines, postes de travail, moyens de transport, main-d'œuvre et matières premières.

   Le rôle de cette partie du système de production est de transformer la matière première en produits finis en procédant à des transformations successives de composants simples ou multiples en articles grâce à des opérations complétées par les dispositifs physiques présents dans le système. Ces dispositifs peuvent effectuer trois types de transformation : spatiale (transport), temporelle (stock) ou physique. Les transformations physiques sont effectuées par des équipements qui consistent à transformer physiquement des objets ou du matériel.

   Ces transformations peuvent être de trois types différents : assemblage (fusion de composants), désassemblage (découpage d'un élément) et transformation (simple modification d'un seul élément, comme pour les processus de trempe ou de flexion).

2. **Le sous-système informationnel :**

   Il s'agit du système d'information rassemblant toutes les données existantes dans le système de production, soit statiques tels que les nomenclatures, le routage de fabrication, la durée d'opération standard ou dynamique telle que l'état actuel de la ressource ou l'emplacement d'un produit sur telle ou telle ressource. Dans les entreprises le système d'information est le plus souvent géré par les progiciels de gestion intégrée (ERP).

3. **Le sous-système décisionnel :**

   Responsable de la gestion du fonctionnement de l'ensemble du système de production. Ce sous-système établit et reçoit des ordres pour programmer l'exécution des opérations, obéissant à des restrictions telles que la disponibilité des composants ou des dispositifs physiques, minimisant le retard d'une bonne production et maximisant l'utilisation des ressources. Par ailleurs, le sous-système décisionnel englobe les politiques de gestion des stocks et d'ordonnancement et

s'appuie le plus souvent sur le sous- système informationnel pour fonctionner. Le sous-système décisionnel est lui-même composé de trois niveaux selon l'horizon dans laquel la décision est prise : *stratégique*, *tactique* ou *opérationnel* correspondant respectivement au long, moyen ou court terme [11].

(a) **Stratégique :** ce niveau correspond aux problèmes de conception d'un système de production et concerne les décisions à long terme relatives aux composantes du système

(b) **Tactique :** ce niveau comprend le choix des produits à usiner simultanément et conformément aux demandes de production. D'une manière générale, ce niveau inclut les décisions qui surviennent à la suite du niveau stratégique avant d'entamer l'exploitation (niveau décisionnel à court et moyen terme).

(c) **Opérationnel :** appelé aussi ***contrôle de production*** ce niveau concerne les décisions liées au pilotage d'un système de production et par ailleurs représente un niveau décisionnel à court terme.
Ce niveau est essentiellement responsable de trois activités : l'ordonnancement, la répartition des produits et le suivi des processus on-line [1] [12].

1.3 Typologie des systèmes de production

D'après Ghédira [13], les systèmes de production peuvent être classés selon leur mode de production. Nous retrouvons dans la littérature les classes suivantes :

1.3.1 Production en série unitaire

Ce type de production mobilise sur une période assez longue l'essentiel des ressources d'une entreprise pour réaliser un nombre très limité de projets. Comme la construction de navires de grande taille, les grands travaux publics, ou la construction d'avions. Étant donné le caractère non répétitif des tâches un personnel hautement qualifié est nécessaire. En ce qui concerne le problème d'ordonnancement, le problème majeur est l'arbitrage entre la recherche d'un coût compétitif et le respect des délais. En effet, d'une part, les commandes seront rapidement honorées si beaucoup de ressources sont mises en œuvre. Mais, d'autre part, le coût des ressources est généralement croissant avec leur niveau d'utilisation. L'ordonnancement des tâches est donc essentiel. En effet, non seulement l'ordre d'exécution des tâches détermine la date de

1. rapport des performances de l'ordonnancement au module responsable pour assurer le fonctionnement continu du système

livraison, et exerce une grande influence sur les coûts dans la mesure où une mauvaise coordination engendre un non-respect des délais.

1.3.2 Production à flux continu

Ce type de production est caractérisé par un flux régulier et important de matières premières destinées à être transformées en matières plus élaborées, comme l'industrie de la pétrochimie ou le secteur agroalimentaire. Étant donné, l'importance et la régularité de la demande, le problème d'organisation des ressources au coût minimum est généralement assez simple. Il peut être résolu par la programmation linéaire.

1.3.3 Production de petite et moyenne série

Ce type de production implique la réunion de tous les équipements assurant une fonction spécialisée en un même lieu, tel un atelier de peinture dans une usine d'assemblage automobile. La ressource humaine est plutôt qualifiée et les équipements sont polyvalents. Deux principaux problèmes sont à considérer concernant les ressources :

1. Le principal problème concerne l'emplacement des ressources dans l'atelier. En effet, lors de la conception de l'atelier, la gestion des coûts de manutention entre les différents postes de travail est très couteuse. Donc pour diminuer ces coûts des méthodes d'agencement dans l'espace sont utilisées pour déterminer la meilleure localisation des machines les unes par rapport aux autres.
2. Le second problème concerne la gestion quotidienne de l'atelier. Elle consiste en la détermination de l'ordre d'exécutions des différentes tâches sur une ou plusieurs machines.

1.3.4 Production en ligne (de masse)

Ce type de production est employé lorsqu'un flux régulier de produits passe d'un poste à l'autre (l'ordre de passage étant fixé), telles que les lignes d'assemblage d'automobiles. En ce qui concerne les ressources utilisées, les équipements sont généralement très spécialisés. L'un des problèmes majeurs consiste en l'équilibrage de la chaîne : c'est-à-dire à définir les tâches à réaliser à chaque poste de manière à avoir le même temps de réalisation à chaque poste. En effet, un mauvais équilibrage de la chaîne entraîne une mauvaise exploitation des ressources. Parmi les autres problèmes fréquents nous pouvons citer : la fiabilité de la chaîne (un maillon défectueux affecte l'ensemble de la chaîne) et la fiabilité du système d'informations.

TABLE 1.1 – Typologie des systèmes de production

	Faible Quantité	**Forte Quantité**
Produit Unique	Production par projet (en série unitaire) (1)	Production à flux continue (en industries de process) (2)
Produits Multiples	Production de petite série (en atelier spécialisée) (3)	Production de masse (en grande série) (4)

1.4 Les types des systèmes de production

D'après les travaux de Essafi [14], Chalfoun [15] et Eloundou [16] les systèmes de production peuvent être répartis en trois grandes familles selon leur configuration, leur objectif de production, leur capacité d'évolution et d'adaptation ainsi que l'horizon temporel sur lequel ils opèrent.

1.4.1 Lignes de production dédiées (DML)

Lignes de production dédiées (Dedicated Manufacturing Lines (DML)) appelées aussi lignes de transfert au vu de la configuration linéaire du système. Ce type de système de production représente un système d'usinage conçu pour la production de masse à long terme d'un seul type de produit. Le rapport coût-efficacité est le moteur de la pré-planification et de l'optimisation. Cependant, le principal inconvénient des DML est que ses lignes ne sont pas conçues pour être modifiables. Par conséquent, elles ne peuvent pas être converties facilement pour produire de nouveaux produits, ce qui en fait un système de plus en plus rare dans la production moderne [15].

1.4.2 Systèmes de production reconfigurables (RMS)

Les systèmes de production reconfigurables (Reconfigurable Manufacturing systems(RMS)) sont définis par Koren et al. [17] comme étant des systèmes de production conçus pour changer rapidement de structure, de composants matériels et logiciels afin d'ajuster rapidement la capacité de production en réponse aux changements des exigences du marché.
Ainsi, la propriété majeure d'un RMS est de jumeler la haute productivité des DMS à l'agilité face aux changements comme des FMS.
Les systèmes de production reconfigurables sont conçus pour une production de moyenne et grande sérise d'un seul produit à moyen et à long terme [14].
Selon, Koren and Shpitalni [18], les RMSs sont caractérisés par :

- **La modularité :** dans un RMS, la majeure partie des composantes sont modulables (architecture, machines, logiciel, etc.). Ainsi, dans le cas d'une reconfiguration, le concepteur peut ajuster, remplacer, modifier ou même supprimer des modules rapidement et à moindre coût afin de mieux répondre aux nouvelles exigences et/ou applications.

- **L'intégrabilité :** c'est la capacité à intégrer des composants rapidement et avec précision, grâce à des ensembles mécaniques, informationnels et des interfaces de contrôle.
- **La personnalisation :** cette caractéristique a pour but de réduire le coût total du système et possède deux aspects distincts à savoir : la personnalisation de la flexibilité (ou du système physique) et du système de contrôle (ou de la commande). La personnalisation du système physique consiste en la conception de machines prenant en compte le produit (ou la famille de produits) à fabriquer pour assurer une personnalisation physique possible et la flexibilité nécessaire pour réduire le coût de production. La personnalisation du système de contrôle est établie en intégrant les modules de contrôle nécessaires à l'aide d'une technologie à architecture ouverte permettant ainsi d'apporter les modifications adéquates en cas de reconfiguration (ajout/modification/suppression des modules de contrôle).
- **La convertibilité :** c'est la capacité à changer rapidement et facilement les fonctionnalités d'un système existant afin de répondre aux nouvelles exigences de la production.
- **La diagnosticabilité :** c'est la capacité à évaluer de manière automatique l'état actuel du système et cela à travers la détection et la correction rapide de pannes et des sources de défaillance au niveau de la qualité afin de réduire les délais de démarrage. En cas de reconfiguration, cette caractéristique est essentielle pour réduire le temps de reconversion du système de production.

1.4.3 Système de production flexibles (FMS)

Les systèmes de production flexibles (Flexible Manufacturing Systems (FMS)) représentent un type de système conçu pour une production de plusieurs types de produits de la même famille en petite et moyenne séries à de faibles coûts.
Le terme « flexible » décrit l'adaptabilité ou la souplesse du système de production. Ainsi, un système de production flexible a pour but d'aboutir à une productivité importante et possède la capacité de gérer la variabilité de la production, de s'adapter à son environnement et ainsi suivre les variations du marché.
L'important intérêt accordé par la communauté de la productique et de l'optimisation aux FMS ainsi que le nombre considérable d'avantages que propose ce type de système nous a poussé à focaliser nos recherches sur ce sujet.
D'ailleurs, nous pouvons citer parmi ces avantages :

- La grandes variétés de produits pouvant être réalisée grâce à la flexibilité des machines.
- La réactivité face aux marchés.
- L'adaptabilité aux variations de volumes de production.
- La flexibilité d'expansion du système permettant ainsi l'évolutivité de ce dernier contrairement aux DMSs.
- La structure permettant l'introduction de nouvelles gammes de produit ainsi que la

production d'une très large gamme de produits tout en étant moins sujette à un crash économique.

Avec l'émergence des nouvelles technologies, de nouvelles perspectives de recherche intéressantes se présentent et nous encourage à approfondir nos recherches sur cette thématique pour identifier les principales problématiques relatives à ce type de système de production et apporter notre contribution.
Cependant, la flexibilité peut se situer à divers niveaux du processus de production.
Nous détaillons dans la section suivante ces aspects de la production concernés par la notion de flexibilité.

1.4.4 Comparaison entre les types de systèmes de production

Le tableau 1.2 résume les caractéristiques des différents types de systèmes de production énumérés précédemment afin de synthétiser les avantages et inconvénients de chacun d'entre eux et justifier les raisons pour lesquelles nous avons basé nos recherches sur les FMSs.

TABLE 1.2 – Comparaison entre les différents types de systèmes de production

Caractéristiques	**DML**	**RMS**	**FMS**
Structure du système	Fixe	Modifiable	Modifiable
Structure de la machine	Fixe	Modifiable	Modifiable
Flexibilité	Absente	Personnalisable	Générale ou Partielle
Évolutivité	Absente	Présente	Présente
Coût	Faible	Moyen	Onéreux
Réactivité	Absente	Réactif	Réactif
Capacité de production	Capacité de production élevée/Variété de produit unique	Capacité de production élevée/Variété de produit moyenne	Capacité de production moyenne/Variété de produit élevée

1.5 La flexibilité dans un système de production

La notion de flexibilité dans un contexte manufacturier peut être catégorisée de différentes manières. Selon [19] la flexibilité peut être répartie en cinq grandes catégories selon les notions suivantes : *la ressource*, *la manutention*, *l'atelier*, *la production* et *le système de production*.
Une autre classification a été proposée par Sethi & Sethi [20], et est considérée comme étant la plus complète du fait qu'elle regroupe tous les aspects de l'industrie flexible et aborde toutes les étapes du processus de fabrication.
Cette classification est divisée en trois grandes classes étant elles-mêmes divisées en sous-classes comme suit :

1. **Flexibilité basique :** ce type de flexibilité est divisé en trois sous-catégories :

   - **Flexibilité de la machine (ou ressource) :** fait référence aux différents types d'opérations qu'une ressource peut effectuer.
     Concerne la capacité d'une ressource à effectuer différentes opérations requises par un ensemble donné de pièces en un faible temps de transition. Ce type de flexibilité peut être obtenu par l'utilisation de dispositifs automatiques de haute technologie de changement d'outils en conjonction avec des chargeurs d'outils et des machines reconfigurables qui permettent de remplacer des modules entiers de machine pour effectuer d'autres opérations.
     Par conséquent, la flexibilité de la ressource permet de réduire la taille des lots, ce qui entraîne des délais de livraison plus courts et une exploitation accrue des machines.
   - **Flexibilité de la manutention :** concerne la manœuvre du matériel, lorsqu'il est possible de fournir différents chemins de transfert aux produits.
     Ce type de flexibilité permet d'acheminer n'importe quel produit (ou ensemble de produits) en fonction d'un critère précis, tel le volume ou le poids du produit.
   - **Flexibilité de l'opération :** ce type de flexibilité concerne le produit, lorsque différentes séquences d'opérations de production aboutissent au même type de produit final (i.e, un produit peut être réalisé selon différents process plan).
     C'est la capacité d'intervertir ou de remplacer les opérations qui permettent la fabrication d'un produit. Cela signifie qu'il n'existe pas de contraintes de précédence entre toutes les opérations de fabrication d'un produit. 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. La flexibilité des opérations dépend essentiellement de la conception et de la nature du produit.

2. **Flexibilité du Système :** ce type de flexibilité est divisé en cinq sous-catégories :

   - **Flexibilité du volume :** concerne la capacité de modifier le volume de production d'un processus manufacturier. Étant donnée, l'évolution perpétuelle du marché ainsi que le comportement du consommateur, la rentabilité de l'entreprise réside dans la mesure dans laquelle elle peut adapter le volume de production en rapport avec la demande [21].
   - **Flexibilité de l'expansion :** concerne la facilité avec laquelle de nouvelles machines peuvent être ajoutées ou remplacées dans la configuration originale du système sans interruption du processus de production.
     En d'autres termes, il s'agit de la capacité d'expansion d'un système de production selon la volonté du concepteur. Pour atteindre ce niveau de flexibilité il est

primordial de posséder une flexibilité au niveau des machines et des moyens de manutentions ainsi qu'un haut niveau d'automatisation. Ce type de flexibilité touche les entreprises qui connaissent une croissance d'activité.

- **Flexibilité du routage :** trouver des chemins alternatifs à suivre à travers le système pour un process plan donné. Ce type de flexibilité a pour but de faire face aux aléas de la production : panne d'une machine, surexploitation d'une ressource.
  La flexibilité de routage est l'une des plus considérées dans la littérature du fait de son avantage majeur qui est celui de gérer des événements non planifiés tels que des pannes de machine, des commandes urgentes ou imprévues [22].
- **Flexibilité du processus de fabrication :** Ensemble de produits pouvant être effectués sans modification majeure sur le système.
- **Flexibilité du produit :** concerne la capacité à ajouter ou remplacer sur un produit de nouvelles pièces tout en minimisant le temps de changement ainsi que les coûts. Cependant, le changement de pièces dans un produit implique automatiquement certaines configurations. C'est pour cela que la flexibilité du produit ne peut avoir lieu avec la flexibilité du processus.

3. **Flexibilité globale :** ce type de flexibilité est divisé en trois sous-catégories :

- **Flexibilité du programme :** concerne la capacité du système à s'exécuter sans surveillance pendant une importante période.
  La mise en place d'une telle flexibilité nécessite une flexibilité d'acheminement ainsi que des processus, la présence de capteurs pour le contrôle et la détection. La flexibilité des programmes nécessite une compréhension accrue des processus de fabrication afin de mettre en places des procédures pour toutes les éventualités qui surviendraient sur le système. La flexibilité des programmes de contrôle exige aussi une expérimentation contrôlée pour l'apprentissage, la modification des connaissances accumulées sous forme de programmes informatiques et la capacité de transfert de ces connaissances pour des produits, des processus ou des procédures similaires
- **Flexibilité de la production :** dans ce type de flexibilité la production est automatisée. Elle est donc contrôlée par ordinateur, permettant ainsi la fabrication d'une grande variété de pièces. La nature de ces pièces ainsi que la taille du flux de production varient sans qu'il y ait nécessairement besoin d'engager des investissements importants en équipements.
- **Flexibilité face au marché :** concerne la capacité du système de production à s'adapter à l'environnement changeant du marché et faire face aux concurrents. Ce type de flexibilité est très important pour la survie d'entreprises et souligne l'importance du lien entre le côté production et marketing d'une entreprise.
  La flexibilité du marché prend en considération différents paramètres pour faire

face à des changements tels que : les innovations technologiques, le comportement des consommateurs, la courte durée de vie des produits ou encore l'incertitude des sources d'approvisionnement.
Pour assurer une flexibilité face au variation du marché, il est nécessaire que le système soit flexible au niveau du produit, du volume et de l'expansion [23].

1.6 Les types de systèmes de production flexibles

Les Systèmes de Production Flexibles sont devenus plus populaires et répandus en raison des faibles coûts de production et de leur haut niveau de productivité.
La littérature propose divèrses classifications des FMS. Selon MacCarthy & Liu [24] les FMS définissent n'importe quelle structure automatisée et sont classés selon leurs caractéristiques de fonctionnement et de contrôle en quatre catégories :

- **Machine Flexible Unique "Single Flexible Machine" (SFM) :** constituée d'une machine à commande numérique avec un chargeur d'outils, un système de transport et une aire de stockage.
- **Cellule de Production Flexible "Flexible Manufacturing Cell" (FMC) :** constituée d'un groupe de SFM reliés par un système de transport unique.
- **Système de Production flexible multi-machine "Multi-machine Flexible Manufacturing System" (MMFMS) :** constitué de plusieurs SFM reliées par un système de transport desservant plusieurs machines à la fois (transport multiple).
- **Système de Production flexible multi-cellule "Multi-cell Flexible Manufacturing System" (MCFMS) :** constitué de plusieurs FMC et éventuellement de SFM connectées par un système de transport automatique.

Une autre classification des systèmes flexibles est proposée par Widmer [25] et se base sur le nombre de machines à commandes numérique (CN) et leur agencement :

- **Le module flexible (MF) :** est une machine à commande numérique avec une aire de stockage, un changeur de pièces et un changeur automatique d'outils.
- **La cellule flexible (CF) :** représente plusieurs modules reliés par un véhicule filoguidé permettant la fourniture des machines en pièces.
- **Le groupe flexible (GF) :** est un ensemble de CF et de MF formant la même zone de production (fabrication, usinage, ou assemblage) joints par des véhicules filoguidés, le tout géré par un ordinateur central.
- **Le système flexible (SF) :** représente plusieurs CF reliées entre elles par des véhicules filoguidés composant les divers zones de production.
- **La ligne flexible (LF) :** est un ensemble d'instruments attribués aux diverses machines comme une ligne de véhicules filoguidés, de robots, de convoyeurs, ...

Quant à la classification de Askin & Standridge [26], elle est principalement basée sur le nombre de machines à commande numérique et est divisée en cinq catégories :

- **L'équipement (equipement) :** représente une machine à commande numérique ou un robot manipulateur.
- **La station de travail (workstation) :** représente une machine à commande numérique avec son aire de stockage, son système de chargement/déchargement, son changeur d'outils.
- **La cellule flexible (cell) :** représente une ou deux stations de travail.
- **L'atelier flexible (shop) :** représente un ensemble de cellules reliées par un système de transport automatique.
- **L'unité industrielle (facility) :** représente l'ensemble de plusieurs ateliers.

1.7 Le pilotage des systèmes de production flexibles

Le pilotage consiste à décider dynamiquement des commandes pertinentes à donner à un système soumis à l'incertitude pour atteindre un objectif donné décrit en termes de maîtrise de performances [27].
La Figure 1.1, se base sur la norme ISA95 [28]. Elle synthétise et illustre les différents niveaux de contrôle et les interactions entre ceux-ci. Notons que, dans cette figure, le niveau pilotage englobe les niveaux tactique et opérationnel. Nos travaux se focalisent sur les niveaux pilotage (ordonnancement) et contrôle (routage de produit). Par conséquent, nous allons, dans un premier temps, introduire les fonctions de pilotage ainsi que le rôle de chacune d'entre elles. Puis nous étudierons les architectures de pilotage existantes.

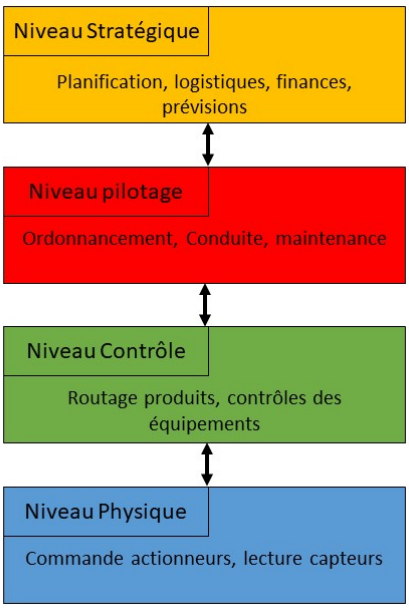

FIGURE 1.1 – Niveaux de contrôle d'un système de production

1.7.1 Les fonctions de pilotage

Le pilotage consiste à décider dynamiquement des consignes pertinentes à donner à un système soumis à perturbation pour atteindre un objectif donné décrit en termes de maîtrise de performances [27].

Le pilotage des systèmes de production peut être considéré comme la superposition de cinq niveaux décisionnels opérant à des horizons temporels distincts, et agissant du niveau le plus global au plus particulier. Ces cinq fonctions sont réparties en deux phases (gestion prévisionnelle et pilotage temps réel).
La gestion prévisionnelle concerne les décisions prises off-line (soit avant lancement du processus de fabrication). Le but principal est de concevoir un planning détaillé de la production en fonction des produits, des marchés, des moyens, etc.
La gestion prévisionnelle est composée des trois fonctions suivantes :

1. **La planification :** se situe au niveau stratégique du système de pilotage et agit sur le long terme. La planification à pour but d'établir des plans de production nécessaires appelés PDP (Plan Directeur de Production) générés selon les objectifs commerciaux, financiers et de production en utilisant généralement des systèmes d'informations spécialisés comme les ERP.
2. **La programmation :** établit le programme prévisionnel de production à partir du PDP et définit les besoins nets en fonction des quantités et des délais de production (besoins bruts), des approvisionnements ainsi que des quantités déjà en stock.
3. **L'ordonnancement :** représente le sujet principal de notre thèse. Par conséquent, il sera abordé plus en détail dans les chapitres suivants.

Quant au pilotage en temps réel, il concerne les décisions prise on-line (soit lors du lancement et pendant l'exécution du processus de fabrication).
Le pilotage en temps réel est composé des deux fonctions suivantes :

1. **La conduite :** cette fonction correspond au niveau décisionnel responsable de la mise en œuvre des tâches ordonnancées pour être lancées par la partie commande. Elle assure la flexibilité quotidienne pour faire face aux fluctuations du système. Si par exemple un état du système ne permet pas de prendre une décision une résolution locale peut être établie.
   La fonction de la conduite est en lien directe avec l'ordonnancement et nous pouvons affirmer qu'elles sont complémentaires. En effet, en cas d'imprévu dans le système dû à des perturbations ou des incertitudes, la conduite adapte localement l'ordonnancement afin de correspondre à la nouvelle situation, nous pouvons ainsi parler de dynamicité du système.
2. **La commande :** cette fonction est en relation directe avec le système physique, car elle a un rôle d'interface et d'interpréteur. Elle décode les tâches à exécuter, lance les commandes au système physique et concerne aussi la commande des équipements de production.

La Figure 1.2 inspirée des travaux de Pujo [29] montre les fonctions de pilotage, leurs répartitions et l'horizon sur laquelle elles opèrent.

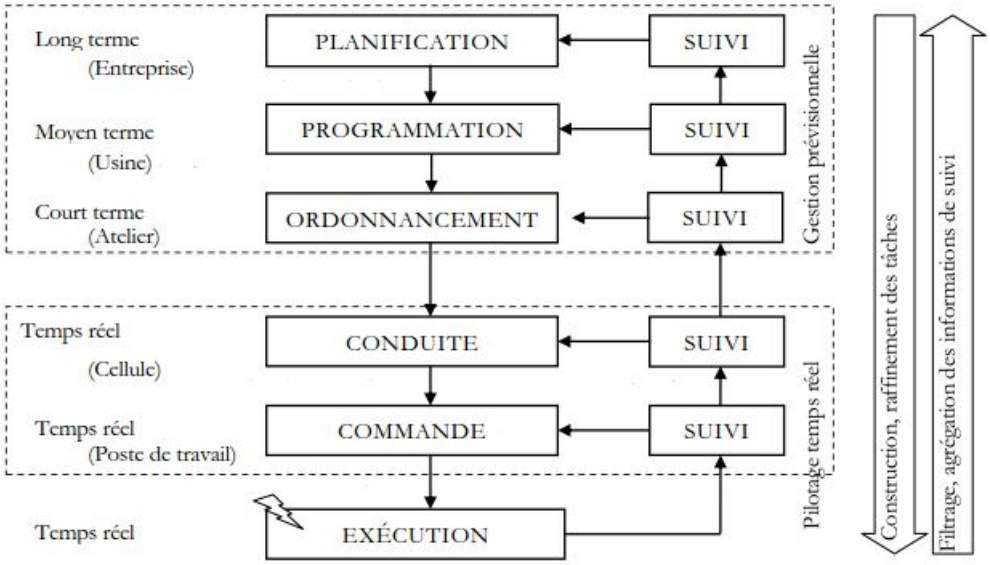

FIGURE 1.2 – Les fonctions de pilotage d'un système de production

Comme indiqué dans la Figure 1.2 la fonction d'ordonnancement se trouve au centre du système de pilotage, et représente le lien directe entre la gestion prévisionnelle et le pilotage en temps réel, ce qui en fait le point névralgique du système de production et est étroitement liée au bon pilotage du système.

1.8 Les architectures de pilotage dans les FMSs

Une architecture de pilotage est une description de la composition et de la structure de pilotage [30]. Elle se définit par rapport à l'ensemble des relations multilatérales, durables et structurées existant entre ses différents sous-systèmes et reflétant son organisation [29]. L'implantation physique d'un système de pilotage résulte souvent de son architecture fonctionnelle. Nous pouvons distinguer deux types d'architectures : centralisée et décentralisée (ou distribuée). Les systèmes centralisées représentent les premières architectures de pilotage ayant vu le jour et étaient organisés autour d'architectures exclusivement centralisées et considéraient la production d'un point de vue global. L'apparition de nouvelles technologies, l'évolution de la complexité des systèmes manufacturiers et la difficulté à piloter tous ses éléments ainsi que le manque de réactivité face aux aléas de la production et la très faible tolérance aux pannes, a poussé les industriels et les chercheurs à envisager d'autres organisations plus efficaces et plus en adéquation avec la nécessaire répartition des fonctions de pilotage. C'est ainsi que les architectures décentralisées virent le jour où les capacités

décisionnelles sont réparties sur différents éléments du système appelés "*entités*".
La définition générique d'une entité a été proposée par Trentesaux [1] et décrite comme : "*un terme générique qui fait référence à une unité autonome capable de communiquer, prendre des décisions et agir.*"

1.8.1 Architecture centralisée

Appelé aussi architecture de classe 0, elle est caractérisée par une seule entité décisionnelle, qui contrôle la totalité du système. Cette entité centrale gère en temps réel les évènements, synchronise et coordonne toutes les tâches. Ce type d'architecture est devenu de plus en plus rare voir obsolète vu le perfectionnement des technologies et la sensibilité de ce type d'architecture où la panne de l'entité décisionnelle signifie l'arrêt total du système. Les facteurs comme : la lenteur des temps de réponse (lors de la présence d'un nombre important d'entrées/sorties), la faible modularité (difficulté de modification dans la structure physique) et le manque de robustesse (très faible tolérance aux pannes) fragilisent grandement ce type d'architecture et ont donné, par conséquent, naissance aux architectures distribuées. Bien que leurs organisation set structures soit plus complexes, ces architectures sont plus prometteuses et permettent de palier aux inconvénients des systèmes centralisés à travers la distribution de la décision via différentes entités décisionnelles.

1.8.2 Architectures décentralisées (distribuées)

Les architectures décentralisées (distribuées) ont la particularité de posséder plusieurs entités décisionnelles. Ces architectures, bien qu'étant plus complexes que les architectures centralisées sur les plans structurel et organisationnel sont plus prometteuses, réactives, flexibles et moins sujettes aux pannes.
Ces architectures ont été divisées par Trentesaux [1] en trois classes selon le type de relation existant entre les entités décisionnelles (Figure 1.3).

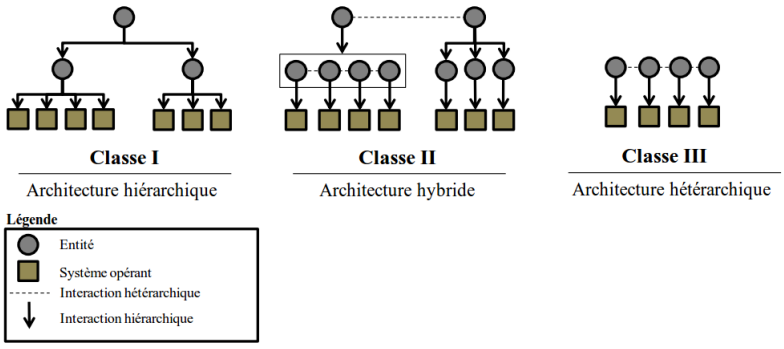

FIGURE 1.3 – Les architectures de pilotages décentralisées selon Trentesaux [1]

Classe I : Architecture hiérarchique (au sens strict)

Ce type d'architecture est considéré comme une extension de l'architecture centralisée [15]. Les entités décisionnelles sont réparties sur plusieurs niveaux de pilotage (forme pyramidale), donnant lieu à une relation de type maître-esclave entre ces différentes entités. Dans ce type d'architecture, les décisions sont prises par les entités se trouvant au plus haut niveau où une connaissance globale relative au système est primordiale. Bien que cela remédie aux problèmes rencontrés dans les architectures centralisées, un problème majeur subsiste, celui de la "réactivité". En effet, le processus décisionnel en cascade entraine un laps de temps considérable entre le moment où un évènement inattendu est détecté et le moment où la décision est prise par l'entité de plus haut niveau. En conséquence, ce type d'architecture devient rapidement inefficace lors de l'apparition d'un nombre important d'aléas.

Classe II : Architecture hybride

Aussi connu dans la littérature sous l'appellation "architecture semi-hétérarchique", ce type d'architecture bénéficie des avantages de l'architecture hiérarchique en terme d'optimisation globale, et de l'architecture hétérarchique en matière de réactivité, d'adaptabilité et de tolérance aux pannes. D'après la thèse de Pach [31] les architectures hybrides peuvent être divisées en quatre sous-classes selon deux critères :

- Le dynamisme du pilotage : concerne l'évolution des interactions entre entités (le basculement d'une configuration à une autre)
- L'homogénéité : concerne la manière dont le pilotage est appliqué sur les entités.

Classe III : Architecture hétérarchique (au sens strict)

Ce type d'architecture possède un seul niveau hiérarchique qui regroupe les entités décisionnelles, ainsi le fonctionnement de cette structure se base sur un contrôle décentralisé.
Les temps de réponse critiques sont gérés localement sans l'intervention d'autres entités. L'indépendance entre les entités décisionnelles de l'architecture permet une exploitation équitable des ressources.
L'inconvénient majeur rencontré dans ce type d'architecture et qui freine sa mise en œuvre dans le monde industriel, est l'absence de vision globale, de perspectives à long terme et de référence de l'ensemble du système pour garantir une prise de décision optimale, ce phénomène est appelé *"Myopie"*.
En effet, dans les architectures hétérarchiques, la décision est prise en fonction des informations et de l'environnement propre à chaque entité. Ces informations peuvent être donc incomplètes, incorrectes ou insuffisantes. Par conséquent, une décision prise peut s'avérer optimale "aux yeux" de l'entité, sans pour autant correspondre aux objectifs globaux du système [32]. Ce phénomène peut être divisé en deux catégories :

1. **Myopie spatiale :** chaque entité du système possède des capacités décisionnelles locales et manque d'informations sur les autres entités ce qui engendre une vue globale

réduite du système et interfère par conséquent avec les performances globales du système.

2. **Myopie temporelle :** une entité peut avoir besoin de plus d'informations sur les états futurs afin d'envisager une séquence de décisions plus optimale pour éviter une dégradation des performances dans le FMS.

1.8.3 Étude comparative entre les architectures de pilotage

Le Tableau 1.3 présente une étude comparative entre les principaux types d'architectures de pilotage : centralisée, hiérarchique et hétérarchique. L'architecture de pilotage hybride, étant le résultat de l'hybridation des architectures hiérarchique et hétérarchique, reprend donc les avantages de chacune de ces dernières. Pour cela, nous ne jugeons pas nécessaire d'intégrer l'architecture de pilotage hybride au tableau comparatif et préférons donc, nous concentrer sur les trois architectures standards. Nous résumons dans ce tableau les point forts et les points faibles de chacune des architectures de pilotage, tout en mettant l'accent sur l'intérêt de mener nos recherches sur les architectures hétérarchiques.

TABLE 1.3 – Étude comparative entre les architectures de pilotage

Architecture / Caractéristiques	**Centralisée**	**Hiérarchique**	**Hétérarchique**
Prise de décision	Unique. L'entité centrale a le contrôle total sur tous les nœuds et est la seule qui influence le système.	Selon un classement de puissance décisionnelle. Les entités de niveau supérieur ont un degré d'influence plus important que celles du niveau inférieur.	Égalitaire. La plupart des entités ont les mêmes propriétés décisionnelles. Cependant, il peut y avoir des entités mieux inter-connectées conduisant à des propriétés améliorées.
Relation entre entité	L'ensemble des nœuds est directement lié à l'entité centrale.	Chaîne linéaire de commande.	En réseaux (avec dans certains cas des relations complexes).
Structure	En forme d'étoile. L'entité centrale (centre de décision) est reliée à toutes les autres entités du système.	Forme pyramidale. Le nombre d'entité décroit considérablement au fur et à mesure que le niveau monte.	Horizontale. Toutes les entités se trouvent au même niveau (hormis, certains cas où il existe une entité de niveau supérieur).
Interaction & Contrôle	Uni-directionnelle. De l'entité décisionnelle vers les autres nœuds.	Contrôle descendant. Les entités de niveaux supérieurs peuvent contrôler les décisions des entités de niveaux inférieurs.	Bi-directionnelle. Les entités se trouvent au même niveau et possèdent les mêmes capacités décisionnelles.

TABLE 1.3 – Étude comparative entre les architectures de pilotage

Architecture / Caractéristiques	**Centralisée**	**Hiérarchique**	**Hétérarchique**
Optimisation globale	Forte. Dépend des capacités de calcul et du fonctionnement de l'entité centrale. (++)	Relativement forte. Dépend des entités de niveau supérieur et leur capacité de calcul. (+)	Faible. L'existence de plusieurs entités décisionnelles autonomes et leurs manques d'information globale peut entrainer une concurrence/conflits d'objectifs et conduire à la détérioration de l'objectif global. (- -)
Réactivité	Absence de réactivité (- -)	Limitée. Dépend du niveau de l'entité décisionnelle touchée. (-)	Forte. Une entité est facilement remplaçable et l'intégration en temps réel des perturbations est meilleure. (++)
Robustesse	Très fragile. Étant donnée la totale responsabilité de l'entité décisionnelle (- -)	Peu robuste. Dû au partage de responsabilité (-)	Forte. L'ajout/suppression d'une entité (pour cause de panne) est facile à mettre en œuvre (++)

TABLE 1.3 – Étude comparative entre les architectures de pilotage

Architecture / Caractéristiques	**Centralisée**	**Hiérarchique**	**Hétérarchique**
Tolérance aux pannes	Très faible. La panne de l'entité décisionnelle signifie l'arrêt total du système (- -)	Moyenne. - Entité décisionnelle affectée = Dommages importants (totalité du système affectée). - Entité de niveau inférieur = Pas d'impact sur le système. (-)	Forte. La panne d'une entité n'affecte pas le fonctionnement du système. (++)
Autonomie	Inexistante. Toutes les entités du système dépendent de l'entité centrale. (- -)	Peu autonome. Une dépendance décisionnelle verticale subsistent dû à la relation maitre/esclave entre les entités. (-)	Autonomie des entités décisionnelles. Chaque entité dispose de ses propres objectifs et la prise de décision est locale. (++)
Coopération par niveau	Inexistante (existence d'une seule entité décisionnelle) (- -)	Les entités de niveau inférieur obéissent à celle de niveau supérieur. (- -)	Les entités se trouvent toutes au même niveau décisionnel. (+)
Myopie	Inexistante. Étant donnée, l'existence d'une seule entité décisionnelle.	Inexistante. La structure par niveau permet une meilleure vision globale du système.	Forte. Représente l'inconvénient principal de cette architecture.

1.9 Conclusion

Bien qu'un pilotage performant représente le nerf du système de production ce dernier a dû faire face à de nombreuses contraintes internes dues à la nature des processus de fabrication et leurs complexités, aux événements incertains qui peuvent se produire tels que les pannes, le manque d'effectif ou d'expérience, le manque de matière première etc. mais aussi, des contraintes externes qui découlent de l'environnement du système de production, des marchés en perpétuelle évolution, de la nature de ses produits et des exigences de la clientèle.

Pour cela le système de pilotage exige toujours d'être réactif, adaptable et surtout flexible afin de répondre à toutes ces exigences et ces contraintes.

À travers, ce chapitre nous avons présenté des généralités sur les systèmes de production, la notion de flexibilité dans le domaine industriel ainsi que les différentes architectures de contrôle d'un système de production flexible afin de poser les bases du chapitre suivant et d'introduire la fonction d'ordonnancement qui représente le point essentiel du pilotage de production permettant à l'entreprise d'atteindre les objectifs fixés.

Chapitre 2

L'ordonnancement dans les systèmes de production flexible (job shop flexible)

Sommaire

2.1 Introduction

Le pilotage de production permet de piloter le système par rapport aux stratégies de l'entreprise. Comme nous l'avons vu dans le chapitre précédent, il comporte plusieurs fonctions. La fonction d'ordonnancement est directement liée aux performances globales et à la réactivité du système de production. L'étude de cette fonction est une branche de la recherche opérationnelle et de la gestion de production qui vise à améliorer l'efficacité et le rendement d'une entreprise d'un point de vue économique en réduisant les coûts de production.
D'ailleurs, d'après les travaux de Zhang et al. [33], les deux problèmes fondamentaux étudiés dans les ateliers flexibles sont l'ordonnancement prédictif (off-line) et l'ordonnancement réactif (temps réel).

- **L'ordonnancement prédictif :** consiste à prévoir "à priori" un certain nombre de décisions en fonction de données prévisionnelles et d'un modèle de l'atelier. Ce type d'ordonnancement se fait off-line (avant le début de la production) et fait suite à la planification.
- **L'ordonnancement réactif (ou dynamique) :** consiste à adapter les décisions prévues en fonction de l'état courant du système et des déviations entre la réalité et le modèle (ce qui est prévu en théorie). En effet, dans un environnement de production réel, la probabilité qu'un ordonnancement prévisionnel soit exécuté exactement comme prévu est très faible en raison des aléas et imprévus (panne de machine, retard dans les livraisons de matières premières, etc). Dans un soucis de réalisme, un ordonnancement doit tenir compte des incertitudes de l'environnement et s'adapter aux données qu'il perçoit en temps-réel.

Nous nous intéressons, donc, dans notre thèse à cette fonction d'ordonnancement du système de pilotage, compte tenu de sa plus grande importance dans la production par rapport aux autres fonctions [34].
Dans un premier temps, il est primordiale de placer la fonction d'ordonnancement dans le cadre général des systèmes de production. Il est donc essentiel de définir la fonction d'ordonnancement et de souligner son rôle majeur dans un système de production.
Pour cela, nous diviserons ce chapitre en deux grandes parties. Dans la première partie du chapitre, nous présenterons de manière synthétique, les concepts de base nécessaires à la formulation d'un problème d'ordonnancement, ses différentes classes, les méthodes utilisées pour la modélisation du problème ainsi que les différentes organisations d'ateliers qui déterminent des problèmes d'ordonnancement. Cette section servira de prémisse à la seconde partie du chapitre consacrée au type d'atelier qui fait l'objet de notre thèse, à savoir le job shop flexible. Ainsi, nous définirions dans cette partie le problème d'ordonnancement dans le job shop flexible plus en détail. Puis nous présenterons les différentes méthodes de résolution rencontrées dans la littérature, pour conclure ce chapitre avec un état de l'art sous forme de tableau qui énumère les travaux pertinents effectués dans le domaine du FJSP.

2.2 La fonction d'ordonnancement

La fonction d'ordonnancement est un pilier du système de production. Elle permet d'optimiser la production et joue un rôle fondamentale pour déterminer l'efficacité du système tout comme elle représente une activité importante pour le maintien de l'entreprise dans la concurrence.
Étant un des thèmes majeurs traités dans la littérature des problèmes d'optimisation, plusieurs définitions ont été proposées. Celles qui reviennent le plus souvent dans la littérature et que nous jugeons les plus exhaustives et les plus en adéquation avec notre sujet de thèse sont les suivantes.

Définition 1 : *L'ordonnancement est l'allocation des ressources, humaines ou techniques, aux tâches sur une durée déterminée avec le but d'optimiser un ou plusieurs objectifs [35].*

Définition 2 : *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement, etc.) et de contraintes portant sur l'utilisation et la disponibilité des ressources requises par les tâches [36].*

Définition 3 : *L'allocation des ressources aux tâches sur une durée prédéfinie en ayant pour but d'optimiser un ou plusieurs objectifs [37].*

2.3 Concepts de base de l'ordonnancement

Le fait que la fonction d'ordonnancement se trouve au niveau pilotage, la positionne au cœur du système de production. Par conséquent, elle est en lien directe avec les autres fonctions et a pour rôle d'organiser l'exécution simultanée de plusieurs activités en tenant compte des contraintes du système.
Pour formuler un problème d'ordonnancement les notions suivantes sont essentielles :

2.3.1 Les tâches

Une définition standard de la tâche a été proposée par la norme (AFNOR NF X50-310) [38] : « Une tâche est le processus le plus élémentaire. Elle est constituée d'un ensemble d'actions à accomplir, dans des conditions fixées, pour obtenir un résultat attendu et identifié, en termes de performances, de coûts et de délais ».
Une tâche est considérée comme l'exécution d'une opération sur une machine m et peut être divisée en deux catégories :

- **Les tâches morcelables (préemptibles) :** le traitement de ces dernières peut être interrompu. Par conséquent, ce type de tâche peut être exécuté en plusieurs fois, facilitant ainsi la résolution de certains problèmes.

- **Les tâches non morcelables (indivisibles) :** sont exécutées en une seule fois et leur traitement ne peut être interrompu qu'une fois la tâche terminée.

Un ensemble de tâches dans un ordre précis forme un « job » pour lequel on donne la gamme de production. Une tâche est définie par un ensemble de variables où chacune est une information particulière se rapportant à la réalisation d'une opération [39].

- r_i pour représenter la date de disponibilité de la tâche i (release date) ;
- t_i pour représenter la date de début de la tâche i (start date) ;
- c_i pour représenter la date de fin d'exécution de la tâche i (completion time) ;
- d_i pour représenter la date d'échéance de la tâche i (due date) ;
- p_i pour représenter la durée opératoire de la tâche i (processing time date) ;
- $F_i = c_i$ - r_i pour représenter la durée de séjour de l'opération i sur la machine avant qu'elle redevienne disponible (flow time) ;
- $L_i = c_i$ - d_i exprime le retard algébrique (lateness) entre la fin d'exécution de la tâche i par rapport à sa date d'échéance ;
- $T_i = \max(L_i,0)$ qui exprime le retard absolu de la tâche i (tardiness) ;
- $E_i = \max(-L_i,0)$ qui exprime l'avancement (earliness) de la tâche i ;

La Figure 2.1 indique les relations entre les différentes variables représentant une tâche.

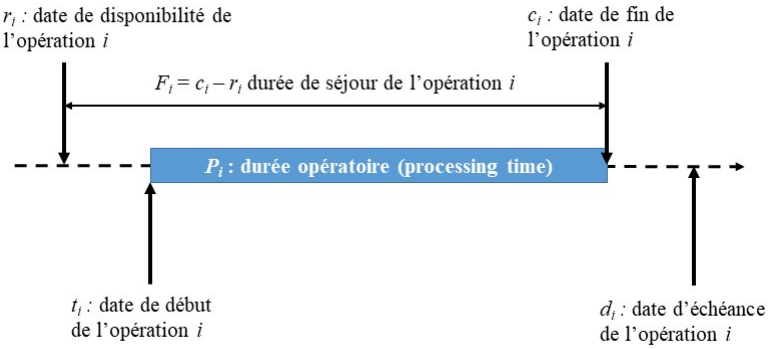

FIGURE 2.1 – Caractéristiques d'une tâche faisant référence à l'exécution d'une opération

2.3.2 Les ressources

Selon la situation les ressources peuvent prendre différentes formes. En effet une ressource peut être représentée par un moyen technique ou humain utilisé pour la réalisation d'une tâche.

Les ressources peuvent être divisées en deux grandes catégories, qui elles-même peuvent être divisées en sous catégories :

— **Ressources renouvelables :**
Ce type de ressource est réutilisable une fois l'exécution de l'opération achevée. Les ressources renouvelables peuvent concerner les machines, l'homme, les outils de production, les robots ou les moyens de transport.
Ces ressources peuvent être divisées en deux sous-catégories :
- Ressources disjonctives : se dit des ressources ne pouvant exécuter qu'une seule tâche à la fois, tels que : les machines-outils et les robots manipulateurs.
- Ressources cumulatives : se dit des ressources pouvant exécuter plus d'une tâche au même moment, comme les machines parallèles.

— **Ressources consommables :**
Ce type de ressources devient indisponible après utilisation, tels que l'énergie, la matière première ou le budget.

La Figure 2.2 indique la classification des ateliers à partir des différents types de ressources.

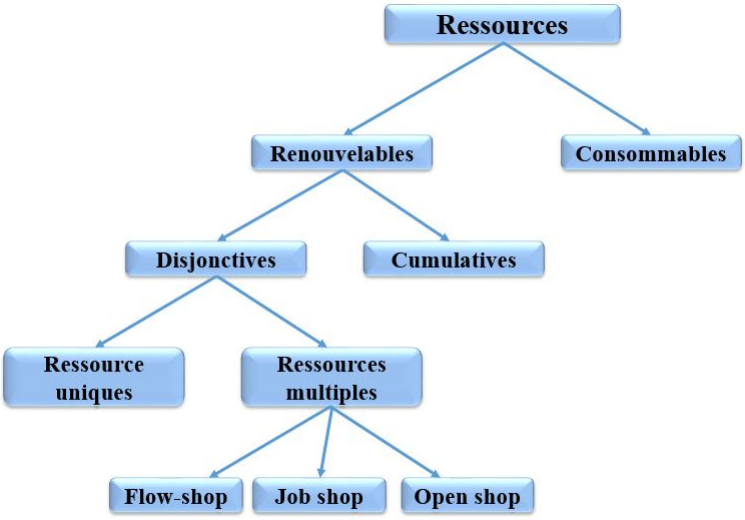

FIGURE 2.2 – Classification des ateliers en fonction des ressources

2.3.3 Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision.

1. **Contraintes temporelles :**
Les contraintes temporelles intègrent en général :
- Les contraintes de temps alloué, issues généralement d'impératifs de gestion et relatives aux dates limites des tâches (tels que les délais de livraison) ou à la durée totale d'un projet.

- Les contraintes d'antériorité et plus particulièrement les contraintes de cohérence technologique décrivant le positionnement relatif de certaines tâches par rapport à d'autres (e.g contraintes de gammes dans le cas des problèmes d'ateliers).
- Les contraintes de calendrier liées au respect d'horaires de travail, etc.

2. **Contraintes de ressources :**
   Ces contraintes sont liées directement à la ressource. Elles spécifient la capacité et la disponibilité des ressources, tout comme elles distinguent les différents types de ressources (disjonctives ou cumulatives). Les ressources ne disposent ni de la même disponibilité ni de la même capacité, celles-ci pouvant être modulée par la modification des calendriers d'utilisation ou l'emploi de ressources externes. Une ressource peut aussi être consommable, si à sa libération, elle n'est pas disponible en même quantité.

Une autre classification divisée en deux catégories est présentée par Toumi [40] :

1. **Les contraintes endogènes :** elles représentent les contraintes en relation directe avec le système de production, à savoir :
   - La capacité des machines et des moyens de transports,
   - Les dates de disponibilités des machines et des moyens de transport,
   - Les séquences des opérations (gammes des produits).
2. **Les contraintes exogènes :** elles regroupent les contraintes imposées par l'environnement extérieur du système de production :
   - Les dates d'échéance des produits imposées par le client ou par la nature du produit,
   - Les priorités des produits/commandes,
   - Les retards possibles permis pour certains produits/commandes.

2.3.4 Les objectifs

La résolution d'un problème d'ordonnancement d'atelier aboutit à l'optimisation d'un objectif qui peut s'exprimer par la minimisation ou la maximisation d'un ou de plusieurs critères. Ces critères ont pour but de déterminer l'efficacité d'une méthode de résolution en évaluant la qualité des résultats obtenus.
Dans les problèmes d'ordonnancement d'atelier, divers critères existent pour classifier les objectifs. La classification la plus récurrente est celle proposée par Esquirol & Lopez [41] où les objectifs sont classés selon les critères suivants :

1. **Objectifs liés au temps**
   — Le temps total d'exécution de l'ensemble des tâches qui correspond aussi à l'achèvement de la dernière opération dans l'atelier. Cette objectif, nommé *Makespan* et dénoté C_{max} représente l'objectif le plus étudié dans la littérature dédiée à l'ordonnancement d'atelier
   — Les temps de retards : ce type d'objectif peut avoir divers variantes d'optimisation (maximum, moyen, somme,...)

2. **Objectifs liés aux ressources**
   - La charge des ressources : cet objectif générique peut donner lieu à deux types d'objectifs : la charge totale ou moyenne de chaque ressource.
   - Le nombre de ressources : tel que la maximisation de la charge d'une ressource ou la minimisation, le nombre total ou moyen de ressources nécessaires à la réalisation de l'ensemble des jobs de l'atelier.
   - Le nombre (total ou pondéré) de ressources nécessaires à l'exécution d'un ensemble d'opérations.
3. **Objectifs liés aux coûts**
   Il peut s'agir des coûts de productions, de transport, de stockage, etc.

Le tableau 2.1 définit les objectifs les plus étudiés dans la littérature concernant l'ordonnancement en milieu manufacturier.

TABLE 2.1 – Critères d'optimisation

Fonction Objective	**Symbole**	**Définition**
Makespan (Maximal Total Completion time)	C_{max}	Le temps total du traitement de toutes les jobs (la fin de traitement de la dernière opération de l'atelier)
Total workload of machines	W_T	Charge de travail total des machines représente le temps d'opération total sur toutes les machines
Critical machine Workload	W_C/W_M	La machine avec la charge de travail critique est celle avec la plus grande charge d'opérations à traiter
Maximal machine workload	W_{max}	Charge maximale de la machine représente le temps maximal passé sur chacun des machines.
Total weighted tardiness	$\bar{T_W}$	Somme des retards pondérés
Maximum tardiness	T_{max}	Le retard total représente la différence entre le temps d'achèvement et la date d'échéance de tous les jobs
Mean tardiness/ Average tardiness	$\bar{T}$	Le retard moyen représente la différence moyenne entre le temps d'achèvement et la date d'échéance d'un seul job.
Maximum lateness	L_{max}	Permet d'évaluer le respect des dates d'échéances.
Number of tardy jobs	N_t	Le nombre de Jobs en retard.
Maximum Flowtime	F_{max}	Le plus long temps qu'un job passe dans l'atelier (le coût de l'ordonnancement est directement lié au plus long job).
Mean Flowtime	$\bar{F}$	Le temps moyen passé par un job dans l'atelier (cet objectif, aussi, a un impact direct sur le coût de l'ordonnancement).
Maximum Earliness	E_{max}	Le temps d'avance total représente la somme des temps passés entre le moment où les jobs sont affectés aux ressources et le début de leurs traitements.
Mean Earliness	$\bar{E}$	Le temps moyen passé par un job sur une ressource avant son exécution

Récemment, la consommation d'énergie est de plus en plus étudié par la communauté. Bien que de nombreux problèmes d'ordonnancement aient été étudiés dans la littérature, le domaine de l'ordonnancement éco-énergétique est relativement nouveau [42].
En effet, la consommation mondiale d'énergie ayant doublée au cours des 40 dernières années et étant susceptible de doubler à nouveau avant 2030 [43]. Les secteurs industriels sont confrontés à d'énormes défis en termes de problèmes environnementaux et économiques, et par conséquent, la réduction de la consommation d'énergie est devenue une de leur principale préoccupation.
Différentes études ont porté sur plusieurs approches pour concevoir des opérations efficaces sur le plan énergétique à divers niveaux d'industrie [44] [45].
Cependant, d'un point de vue opérationnel, l'optimisation de l'ordonnancement de production est un mécanisme approprié et efficace pour les installations industrielles. De plus, dans les processus de production réels, l'ordonnancement est l'un des facteurs clés qui influent sur l'efficacité, la qualité et le coût de la production ajouté à cela l'effet de l'ordonnancement sur l'utilisation de ressources et par conséquent les émissions de dioxyde de carbone résultantes. Ainsi, avec un ordonnancement éco-énergétique, il est possible de réduire la consommation d'énergie sans aucune refonte dans les paramètres structurels et d'obtenir des améliorations tangibles de la consommation d'énergie et des coûts.

Notons aussi l'intérêt grandissant des chercheurs pour l'optimisation multi-objectifs. Comme son nom l'indique, le but est d'optimiser au minimum deux objectifs. Cependant, la présence de plusieurs objectifs à optimiser peut entrainer la dégradation de l'un au profit d'un autre. Afin de palier à ce problème, différentes méthodes ont été proposées pour trouver un compromis permettant de satisfaire tous les objectifs.
Les méthodes d'optimisation multi-objectifs peuvent généralement être classées en trois types différents [46] :

1. La transformation en problème mono-objectif qui consiste à combiner les différents objectifs en une somme pondérée [47] [48] [49]. Les méthodes de cette classe peuvent être basées sur les fonctions d'utilité, de ε-Constraint [50], ou la programmation d'objectif (goal programming) [51].
2. L'approche non-Pareto qui utilise des opérateurs pour traiter les différents objectifs de manière séparée [52] [53] [54].
3. Les approches de Pareto qui sont directement basées sur le concept d'optimalité de Pareto. Elles visent à atteindre deux objectifs en convergeant vers le front de Pareto et obtenir des solutions diversifiées réparties sur tout le front. Les méthodes d'optimisation basées sur le front de Pareto sont favorisées par les chercheurs car elles permettent d'obtenir un ensemble de solutions optimales, au lieu d'une solution unique, ce qui est parfaitement cohérent avec un problème d'ordonnancement réel.

Il existe différents algorithmes se basant sur ce principe. Les plus utilisés et fréquemment rencontrés dans la littérature sont :

- **Multi-Objective Genetic Algorithm (MOGA) :** mise en œuvre par Foncesa & Fleming [55]. Cet algorithme a été le premier a utilisé la notion de dominance directement pour l'évaluation des performances des individus. Dans cet algorithme, pour chaque solution i, le nombre n_i de solutions la dominant est calculé et le rang $r_i = (1 + n_i)$ lui est associé. Le rang de chaque solution non-dominée peut être donc compris entre 1 et N (N correspondant à la taille de la population). Le fitness F_i attribué à chaque individu est basé sur son rang. La fonction qui transforme le rang en fitness est telle que tous les individus du même rang ont le même fitness et doit être maximiser.
  L'inconvénient majeur de cette méthode réside dans le manque de diversité dans la représentation des solutions.

- **Niched Pareto Genetic Algorithm (NPGA) :** proposé par Horn et al. [56]. Cette méthode diffère des précédentes lors de la phase de sélection. Deux individus sont choisis au hasard et comparés à un sous-ensemble Q de la population initiale. Si l'une d'elles domine toutes les solutions de Q et que l'autre est dominée par au moins un individu, alors la solution non dominée est sélectionnée. Si les deux individus sont dominés ou non dominés, la sélection est effectuée via la méthode de la fonction de partage appelé partage de classe équivalent.

- **Non-dominated Sorting Genetic Algorithm (NSGA) :** proposée par Srinivas & Deb 1994 [57], dans cette méthode, avant que d'entamer le processus de sélection, les solutions sont classifiées à base de non-dominance. Tous les individus non-dominés sont classés dans une catégorie avec une valeur de fitness factice proportionnelle à la taille de la population afin de fournir une possibilité de reproduction égale pour tous les individus.

- **Non-dominated Sorting Genetic Algorithm (NSGA-II) :** proposé par Deb et al. [58], cet algorithme emploie une population d'individus et obtient une approximation du front optimal de Pareto. Les deux principaux opérateurs du NSGA-II qui ont été largement appliqués par les chercheurs sont les procédures de tri non dominé et les distances de crowding. L'algorithme de tri non dominé classe les solutions dans différents fronts de Pareto. La procédure de distance de crowding calcule la dispersion des solutions sur chaque front et préserve la diversification de l'algorithme. À chaque génération de cet algorithme, ces deux fonctions forment les fronts de Pareto.
  Avec NSGA-II, Deb et al. tente de répondre aux problèmes rencontrés avec la méthode NSGA concernant : la complexité élevée de l'algorithme, le non élitisme et l'utilisation du partage (diversité de la population).

- **Non-dominated Sorting Genetic Algorithm-III (NSGA-III) :** proposé par Deb & Jain [59], NSGA-III est une méthode d'optimisation de Pareto pour la résolution de problèmes à plusieurs objectifs (de trois jusqu'à quinze). C'est une extension de l'algorithme NSGA-II. La principale différence entre les deux algorithmes réside dans le fait que le NSGA-III utilise un ensemble de points de référence pour maintenir la diversité des points de Pareto pendant la recherche. Cela se traduit par une distribution très uniforme des points de Pareto dans l'espace des objectifs, même lorsque le nombre d'objectifs est élevé.

- **Pareto Archived Evolution Strategy (PAES) :** méthode crée par Knowles et Corne [60], cette méthode se base sur une stratégie évolutionniste simple de la recherche locale où une archive externe des solutions non dominées est maintenu dans le but de diversifier les solutions.

- **Pareto Envelope-based Selection Algorithm (PESA) :** proposé par Cornes et al. [61], cette méthode utilise deux paramètres relatifs à la population : P_I qui représente la taille de la population interne et P_E qui représente la taille de l'archive (population externe), si une solution dans P_I est non-dominée, elle est systématiquement transférée à l'archive, en même temps, toutes les solutions dominées par cette nouvelle solution sont supprimées de la population.

- **Pareto Envelope-based Selection Algorithm (PESA-II) :** proposé par Corne et al. [62], cet algorithme utilise une technique de sélection basée sur l'utilisation d'hypercubes dans l'espace objectif. Au lieu d'effectuer une sélection en fonction du fitness des individus comme dans PESA, cette méthode effectue une sélection par rapport aux hypercubes occupés par au moins un individu. Après avoir sélectionné l'hypercube, un individu de l'hypercube est choisit aléatoirement. Cette méthode se montre plus efficace à repartir les solutions sur le front de Pareto que la sélection par tournoi classique.

- **Strength Pareto Evolutionary Algorithm (SPEA) :** proposé par Zitzler & Thiele [63], cet algorithme utilise le concept de force (Strength) pour localiser et maintenir un front de solutions non dominées. La "force" indique le degré de domination d'une solution. L'attribution du fitness de chaque individu est réalisée en fonction des "forces" de toutes les solutions non dominées qui le dominent. La diversité de l'algorithme est garantie par l'utilisation de la méthode de liaison moyenne.

- **Strength Pareto Evolutionary Algorithm-II (SPEA-II) :** est un algorithme génétique multi-objectif proposé par Zitzler et al. [64] et basé sur le front de Pareto. SPEA-II adopte un schéma appelé force qui prend en compte, pour

chaque solution de la population, non seulement le nombre de solutions qu'ils dominent, mais également le nombre de solutions par lequel il est dominé. Cela fournit une estimation plus précise du classement réel d'une solution.
L'algorithme multi-objectif SPEA-II diffère de son prédécesseur, SPEA, dans les points suivants [65] :

- Un schéma amélioré de l'affectation du fitness, prenant en compte pour chaque individu le nombre d'individus qu'il domine et par lequel il est dominé.
- Une technique d'estimation de la densité basé sur l'algorithme du plus proche voisin (k-ppv) permettant un guidage plus précis du processus de recherche.
- Des nouvelles méthodes de troncature des archives garantissant la préservation des solutions de bornes (supérieure et inférieure).

Le NSGA-II est l'algorithme évolutionnaire multi-objectif le plus utilisé [66] [67] en raison de sa meilleure distribution ainsi que sa vitesse de convergence.

2.4 Les classes d'ordonnancement

Les travaux de Caumond [68], Shahzad [69] et Zahmani [70] permettent de conclure que lors de la recherche d'un ordonnancement, plusieurs solutions répondant aux critères sélectionnées peuvent être trouvées. Par conséquent, il est primordial de diviser l'ordonnancement en différentes classes afin de restreindre la recherche des solutions à des sous-ensembles limités pour éviter une exploration intégrale de l'espace de recherche et faire un gain considérable en temps de résolution.
La Figure 2.3 illustre la manière dont laquelle les différentes classes d'ordonnancement sont intégrées l'une dans l'autre.

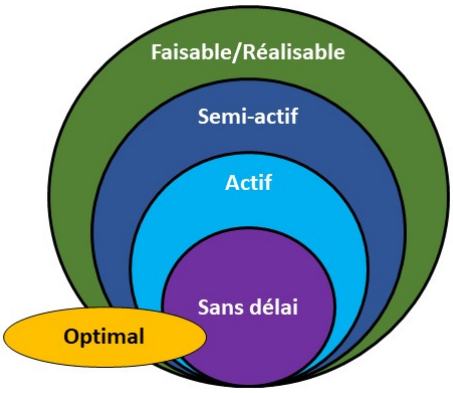

FIGURE 2.3 – Les classes d'ordonnancement

- **Ordonnancement faisable (ou réalisable) :**
  Un ordonnancement est dit faisable (ou réalisable) si et seulement si tous les jobs respectent l'ensemble des contraintes spécifiées du problème d'ordonnancement.
- **Ordonnancement semi-actif :**
  Un ordonnancement semi-actif est un ordonnancement réalisable aligné à gauche où aucune tâche ne peut être déplacée vers sa gauche sans changer l'ordre des opérations. Autrement dit, il est impossible de commencer une opération plus tôt sans pour autant violer une contrainte ou sans changer l'ordre des opérations sur les ressources. Dans un tel ordonnancement (l'ordre de passage des tâches), l'unique façon d'améliorer le rendu est de réordonner la séquence des opérations sur les machines. Par conséquent, si un ordonnancement semi-actif est représenté par un diagramme de Gantt, il est impossible de décaler une tâche à gauche.
- **Ordonnancement actif :**
  Un ordonnancement faisable est dit « actif » aucune opération ne peut être traitée plus tôt sans retarder la date de début d'une autre opération. Par ailleurs, il est aussi impossible d'insérer une nouvelle opération entre deux tâches sans remettre en cause l'ordre d'exécution ou enfreindre une des contraintes. Un ordonnancement actif est nécessairement semi-actif. Sur un diagramme de Gantt, un ordonnancement est considéré comme actif s'il est impossible de décaler une tâche à gauche.
- **Ordonnancement sans délai :**
  Un ordonnancement est *« sans délai »* si à chaque instant toute opération qui dispose des ressources nécessaires est commencée. Ainsi, un ordonnancement est sans délai si aucune machine n'est maintenue inactive pendant qu'une opération attend d'être traitée sur cette machine. Par conséquent, un ordonnancement sans délai est nécessairement un ordonnancement actif et donc semi-actif.
  Les ordonnancements sans délais sont le plus souvent utilisés dans les heuristiques étant donnée leur nature incertaine à trouver la solution optimale [68].
- **Ordonnancement optimal :**
  Un ordonnancement optimal est un ordonnancement qui donne la meilleure valeur possible pour une mesure de performance donnée. Dans un problème d'ordonnancement il est possible de trouver plus d'une solution qui donnent le même résultat (même valeur de la fonction objectif). Pour plus d'un objectif, le sens de l'optimalité est assez différent (cas du multi-objectif).

2.5 Modélisation d'un problème d'ordonnancement

La modélisation est une étape importante de la résolution d'un problème d'ordonnancement. Elle peut être définie comme étant une écriture simplifiée de toutes les données du problème, en utilisant un formalisme bien adapté pour représenter un problème choisi [71]. Nous pouvons distinguer dans la littérature deux types de méthodes pour modéliser un problème d'ordonnancement : les méthodes mathématiques et les méthodes graphiques.

1. **Modélisation mathématique :** dans ce type de modélisation, les données, les contraintes et la fonction d'évaluation des critères sont écrites sous forme d'équations et d'inéquations mathématiques. Ces méthodes, couramment utilisées, sont relativement simples et directement exploitables par les algorithmes de résolution [41].

2. **Modélisation graphique :** il existe différents outils de modélisation graphique, parmi eux nous citons :

   - **Méthode PERT (Program Evaluation and Research Task) :** : cette méthode utilise un graphe potentiels-évènements qui permet de représenter une tâche par un arc auquel lui est affecté un chiffre indiquant la durée de cette tâche. Les sommets quant à eux représentent les débuts et fins d'opérations (évènements).

   - **Réseau de Petri (RdP) :** cet outil de modélisation des systèmes dynamiques à événements discrets a été proposé par Carl Adam Petri [72]. Le RdP décrit les relations existantes entre les conditions et les événements d'un système. Sa représentation graphique permet d'illustrer et de visualiser de manière naturelle, les intéractions, le parallélisme, le choix et le partage des ressources au sein du système. Il existe différents types de réseaux de Petri (RdP) [16] temporisé, coloré, flou, déterministe, stochastique, cyclique, synchronisé et à capacité.
     Les RdPs permettent de traduire un nombre considérable de phénomène ayant un lien avec les problèmes d'ordonnancement :
     - Les conflits sur les ressources
     - Les durées opératoires certaines (RdP temporisé) et incertaines (RdP stochastique)
     - La disponibilité, la multiplicité et la capacité des ressources (RdP synchronisés et à capacités).

   - **Diagramme de Gantt :** ce type de modélisation graphique est la méthode la plus utilisée pour la représentation des solutions aux problèmes d'ordonnancement compte tenu de sa facilité de lecture et d'interprétation [73] [70] [74].
     Élaboré par Henry Laurence Gantt en 1917, le diagramme de Gantt est un graphique à barre horizontale où chaque ligne est composée de différents rectangles qui sont pour chacun la représentation du temps nécessaire à la réalisation d'une opération : le temps de réalisation est proportionnel à la longueur du rectangle. Dans le cadre d'un problème d'ordonnancement, le diagramme de Gantt prend en ordonnée les différentes ressources utilisées et en abscisses les temps opératoires correspondant à une échelle temporelle. Ce type de modélisation graphique sera utilisé dans les chapitres suivants pour la représentation des affectation et séquencement de tâches sur les machines.

En dehors des modélisations graphiques et mathématiques, il existe un système de notation : *la notation* $(\alpha \mid \beta \mid \gamma)$. Ce système de notation introduit par Graham et al. [75] est couramment utilisé pour représenter un problème d'ordonnancement. Le champ α renseigne les informations relatives à l'environnement machine, i.e. le type d'atelier et le nombre de machines disponibles. Le champ β indique les caractéristiques du problème d'ordonnancement, ce qui implique les contraintes prises en considération lors de l'étape d'ordonnancement. Ce champ peut contenir diverses entrées standards ou spécifiques au type d'atelier. Enfin, le champ γ correspond au critère à optimiser.

2.6 Type de problèmes d'ordonnancement

Selon la manière dont les tâches sont ordonnancées et dont les machines sont configurées, nous distinguons différents problèmes d'ordonnancement. Connu sous le nom d'atelier nous citons ci-dessous les types d'atelier les plus rencontrés dans la littérature :

2.6.1 Atelier à machine unique *(Single machine shop)*

Ce type d'atelier est composé d'une seule machine qui traite toutes les gammes de production. Ce type d'atelier est le plus problématique et est de moins en moins répandu dans l'industrie en raison des différents handicaps rencontrés (goulot d'étranglement, surexploitation de la ressource, file d'attente interminable, temps de production exponentiel,...) [76].
La Figure 2.4 illustre un atelier à cheminement unique avec cinq jobs à produire.

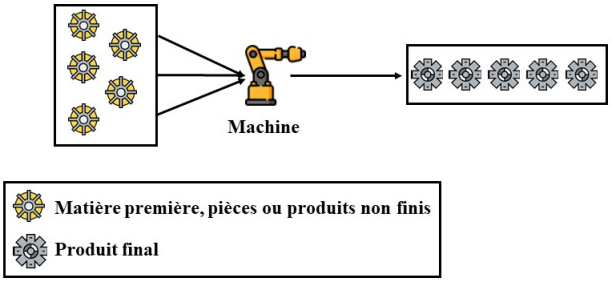

FIGURE 2.4 – Atelier à machine unique

2.6.2 Atelier à machines parallèles *(Parallel machine shop)*

Dans ce type d'atelier, chaque job à la même séquence d'opération. L'atelier à machines parallèles peut être considéré comme étant un atelier à machine unique avec redondance de machine étant donné que toutes les machines sont identiques. Cependant, contrairement au type précédent, la redondance de machine permet d'accélérer la production et d'éviter l'arrêt du système dans le cas d'une panne de machine [77].
La Figure 2.5 illustre un atelier à machines parallèles avec n machines.

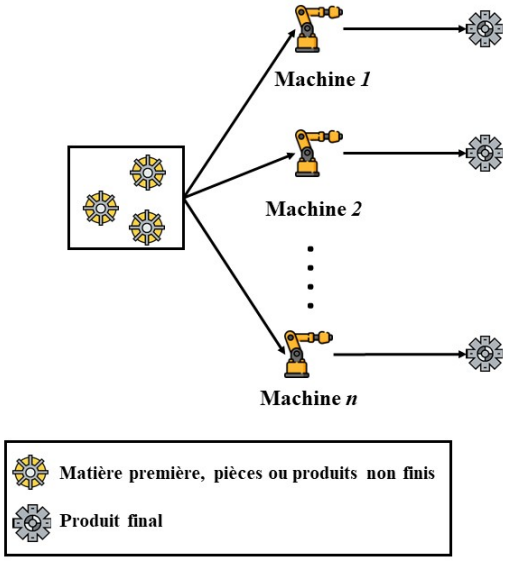

FIGURE 2.5 – Atelier à machines parallèles

2.6.3 Atelier à cheminement unique *(Flow shop)*

Les ateliers de type Flowshop ou atelier à cheminement unique ont pour particularité d'avoir un processus de fabrication linéaire. Ce type d'atelier est constitué d'un ensemble de ressources où le traitement de chaque produit se fait de manière chaînée. Le flux de chaque produit est unidirectionnel et chaque opération de chaque job est exécutée dans le même ordre. Ce type d'atelier est rencontré généralement dans les productions en série, où les gammes opératoires sont identiques. Bien que la production du même type de produit est répétée, les temps de traitement peuvent différer, ce qui donne lieu à un problème de délai à optimiser. Ce type de problème à pour but de trouver un ordre optimal de passage des pièces afin de minimiser la date fin au plus tard de la dernière opération de l'atelier.
La Figure 2.6 illustre un flow shop composé de n machines (n $\in \mathbb{N}$) et quatre jobs.

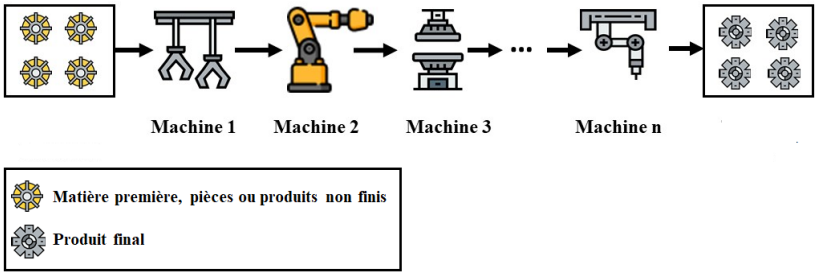

FIGURE 2.6 – Atelier à cheminement unique (Flow shop)

2.6.4 Atelier à cheminements multiples *(Job shop)*

L'une des caractéristiques des ateliers à cheminement multiple est que la demande pour un produit particulier est généralement d'un volume petit ou moyen. L'objectif le plus considéré dans le job shop est le même que pour le flow shop, à savoir trouver une séquence de tâches sur les machines qui minimise le temps total de production. Le Job Shop est considéré comme un des problèmes les plus généraux. En effet, certaines recherches [78] arrivent à la conclusion que les ateliers Flow Shop et open shop peuvent être considérés comme des cas spéciaux du problème Job Shop, justifiant ainsi l'intérêt constant de la communauté scientifique envers cette problématique.
La Figure 2.7 illustre un job shop composé de cinq machines et trois job.

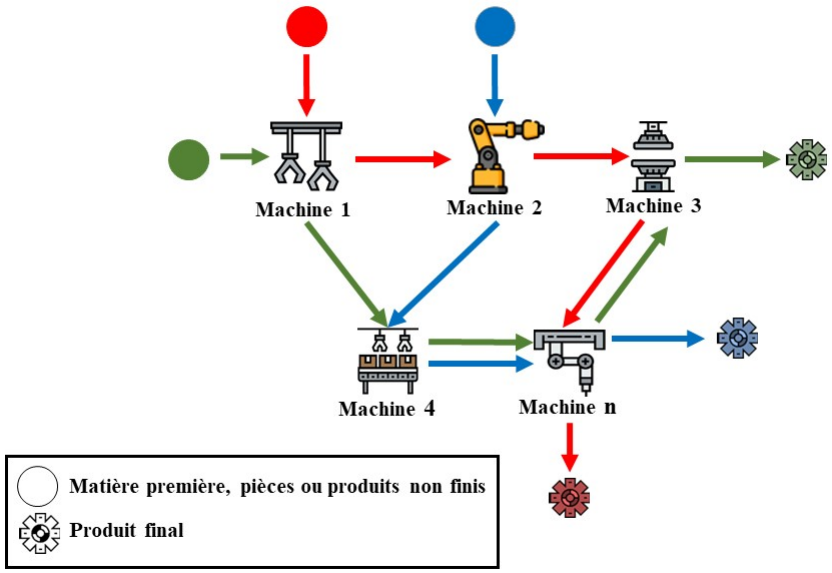

FIGURE 2.7 – Atelier à cheminements multiples (Job shop)

2.6.5 Atelier à cheminement libre *(Open shop)*

Dans ce type d'atelier, l'ordre des opérations n'est pas fixé à priori. Ainsi, le problème d'ordonnancement se décompose en deux sous-problèmes, celui de déterminer le cheminement de chaque job (routage) et celui d'ordonnancer les jobs en prenant en considération les gammes trouvées étant donné, son niveau de complexité élevé et son côté très stochastique. Ce type d'atelier n'est pas très reproduit en industrie ni très étudié dans la littérature des problèmes d'ordonnancement.

2.6.6 Atelier à production "lineless"

Dans les systèmes de production *lineless*, tous les éléments de production peuvent se déplacer librement sur le site de production en utilisant l'auto-organisation afin de s'adapter aux fluctuations telles que la diversité des demandes de production et le dysfonctionnement des machines [79]. Ces ateliers sont considérés comme une généralisation des *Open shop* où la flexibilité des opérations est améliorée [80]. Bien que très peu utilisé pour leur coût élevé et l'espace considérable nécessaire à l'installation et au déploiement de ce type d'atelier, de nombreux avantages existent [79] :

1. La production d'une haute variété de produits : étant donné que les installations ne sont pas connectées entre elles, différents types de produits peuvent être réalisés dans un seul atelier.
2. Robustesse face aux dysfonctionnements : même en cas de défaillance d'une installation, les autres installations ne sont que relativement affectées, car seule l'installation défaillante doit être réparée individuellement.
3. Facilité de reconfiguration : lorsqu'un nouveau produit est traité, de nouvelles installations peuvent être ajoutées sans interrompre les processus de production en cours ou de modifier la configuration de l'ensemble du système.

2.6.7 Atelier à cheminement unique hybride *(Flow shop hybride)*

Connu aussi sous le nom de flow shop flexible [81], il est défini pour la première fois par [82], ce problème est une extension du problème du flow-shop. C'est une combinaison du problème d'ordonnancement du flow-shop et du problème d'ordonnancement de l'atelier à machines parallèles [83]. Ce type d'atelier est composé de m étages où chaque étage est composé de plusieurs machines parallèles identiques, proportionnelles ou non identiques. Les jobs doivent être traités sur une seule machine de chaque étage. étant donné le processus de fabrication linéaire, l'ordre des étages est le même pour tous les produits. En plus du problème d'ordonnancement des jobs, le fait d'être constitué de plusieurs étages donne lieu à un problème d'affectation des jobs aux machines sur les différents étages.
Ce type d'atelier est courant dans divers industries, notamment celles des textiles, des armes à feu, des horloges et montres, des locomotives de chemin de fer et des bicyclettes [81].
La Figure 2.8 montre un Flow-Shop Hybride composé de n étages.

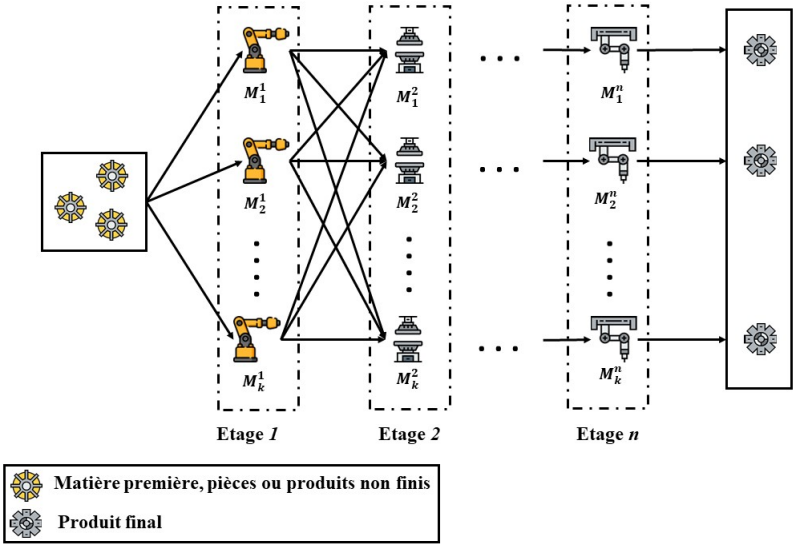

FIGURE 2.8 – Atelier à cheminement unique hybride (Hybrid Flowshop)

2.6.8 Atelier à cheminement mixte

Initié par Masuda et al. [84], ce problème a été défini comme étant un mélange d'atelier à cheminement multiple (Jobshop) et libre (openshop) de manière à ce que certains jobs ont des ordres de machines fixes (job shop) tandis que les opérations des autres jobs peuvent être traitées dans un ordre arbitraire (comme dans l'open shop). Bien que très récente comme organisation, les chercheurs éprouvent un intérêt constant envers cette problématique compte tenu des potentiels sujets de recherches ainsi que des possibilités à offrir [85] [86] [87].

2.6.9 Atelier à cheminement multiple flexible *(Job Shop Flexible)*

Le Job Shop Flexible ou Flexible Job Shop très souvent abrégé dans la littérature par FJSP est une extension du Job Shop Classique. La particularité du job shop flexible est qu'une opération peut être traitée par différentes ressources alternatives, et qu'une ressource peut fournir plusieurs services (exécuter différentes tâches).
Parmi les différents type de problèmes d'ordonnancement existant dans la littérature nous nous sommes focalisés sur la résolution du problème d'ordonnancement dans les *Job Shop Flexible* ou *Flexible Job Shop* (JSF ou FJS).

2.7 Le problème d'ordonnancement dans le job shop flexible

Le problème d'ordonnancement dans le job shop flexible ou le *Flexible Job Shop Scheduling Problem* est une extension du problème de Job Shop classique. Ce type d'atelier a pour particularité de fournir le même type de service/opération sur un ensemble ou sur la totalité des machines/ressources d'un système de production (flexibilité partielle ou totale).
Les performances des machines peuvent être équivalentes ou pas (i.e la durée d'exécution de la même opération peut différer d'une machine à l'autre).
L'aspect non déterministe de l'ordonnancement dans les job shops flexibles en fait un problème NP-difficile. En effet, l'éventail de choix de ressources alternatives qui s'offre à une tâche peut entraîner une explosion combinatoire.
Étant donnée la nature et la complexité du problème le temps de résolution s'avère généralement exponentiel. Cela est appuyé par Garey et al. [88], qui selon lui même si la taille du problème est réduite (atelier composé de *deux machines* et *trois jobs* au plus) il reste néanmoins NP-difficile.
Bien que dans certains travaux, des techniques transversales ont été incorporées aux méthodes exactes, telle que l'augmentation de la capacité de calcul de l'ordinateur/simulateur ont été tentées, les résultats obtenus n'ont pas abouti à une réduction significative du temps de résolution du problème [31].
Au vu de cette contrainte temporelle, les méthodes de résolution exactes s'avèrent donc inefficaces. Afin de pallier à cet handicap, des méthodes de résolutions approchées ont été mises en œuvre. Ces méthodes ont pour but de réduire considérablement le temps de résolution au détriment de l'optimalité de la solution.

Comme indiqué dans le premier chapitre, la flexibilité est un concept pouvant intervenir à différents niveaux du processus de fabrication et concerner différents concepts. Cependant, les aspects de la flexibilité ayant un impact direct sur la fonction d'ordonnancement sont ceux de *la ressource* et du *routage*.
D'ailleurs la flexibilité du routage conduit directement au problème des FJSP. Ce problème considéré comme de la planification à très court terme, cherche à résoudre deux sous-problèmes :

- **Le séquencement :** ce sous-problème consiste à établir une séquence d'opérations sur chaque ressource, et à déterminer les dates de début et de fin de chacune.
- **L'affectation :** aussi appelée routage ou allocation, consiste en l'affectation des opérations aux ressources. L'attribution des ressources nécessaires aux opérations prend plus d'ampleur dans les ateliers flexibles, où plusieurs machines alternatives peuvent réaliser la même opération avec des temps d'exécution identiques ou différents.

Le problème du job shop flexible a été abordé pour la première fois par Brucker & Schlie [89] il y a une trentaine d'années en traitant la notion de flexibilité du système par

l'intermédiaire de machines "polyvalentes". Depuis, plusieurs méthodes ont été proposées pour résoudre ce problème. Ces méthodes peuvent être divisées en deux grandes classes, qui seront présentées plus en détail dans la section suivante.

2.8 Les méthodes de résolution du FJSP

La communauté de l'optimisation combinatoire a longtemps privilégié les méthodes exactes au détriment des méthodes approchées pour la résolution du problème d'ordonnancement du job shop flexible (FJSP). Le recours à des méthodes exactes permet de bénéficier de résultats théoriques forts accompagnant les notions de convergence et d'optimalité globale. De ce fait, plusieurs problèmes d'ordonnancement de taille raisonnable peuvent être résolus avec des méthodes exactes, en un laps de temps raisonnable. Ces méthodes consistent généralement à énumérer l'ensemble des solutions de l'espace de recherche et disposent de techniques pour détecter le plus tôt possible les échecs (calcul de bornes). Cependant le temps de calcul nécessaire à ces méthodes est exponentiel, raison pour laquelle, elles ne sont utilisées que sur des problèmes de petite taille.
Depuis la fin des années 80, les méthodes approchées ont suscité un intérêt croissant pour leur capacité à produire des solutions d'excellente qualité en un temps de calcul réduit. La perte en optimalité est compensée par la réduction considérable des temps de calcul. Ces méthodes bénéficient également d'autres avantages tels que la rapidité d'adaptation à des modifications structurelles du problème.
Nous présentons, dans cette section les deux classes de méthodes de résolution du FJSP.

2.8.1 Méthodes exactes

Ces méthodes sont issues de la recherche opérationnelle ; elles sont dites exactes car leur convergence vers une solution optimale à un problème donné a été démontrée.
Les méthodes exactes sont multiples et variées, elles sont proposées pour résoudre les problèmes combinatoires de manière optimale et leurs efficacités dépend en grande partie de la structure du problème. Les méthodes de résolution exactes les plus rencontrées dans la littérature sont : le Branch & Bound et les techniques de programmation mathématique comme la programmation linéaire mixte (MILP) et la programmation dynamique [90].
Les méthodes de résolution exactes ont pour principe d'effectuer une recherche complète de l'espace de recherche pour trouver une solution optimale au problème, pour cela, leur efficacité dépend principalement de la taille du problème et l'optimalité de la solution n'est garantie que pour des problèmes de petites tailles.
Ainsi, l'inconvénient majeur avec ce type de méthodes réside dans le fait qu'elles nécessitent des calculs mathématiques complexes et d'importantes ressources calculatoires pour des problèmes de grandes taille, qui par conséquent les rendent très lentes, difficiles à mettre en œuvre et restreint radicalement leur utilisation (utilisable dans des cas très spécifiques).

2.8.2 Méthodes approchées

Malgré l'évolution permanente de l'informatique et l'apparition de calculateurs ultra-performants, il existe toujours, pour un problème polynomial, une taille critique au-dessus de laquelle une énumération, même partielle, des solutions admissibles devient prohibitive. Compte tenu de ces difficultés, les chercheurs s'orientent de plus en plus vers le développement de méthodes approchées. Le FJSP faisant parti de la classe des problèmes d'optimisation combinatoire peut être résolu avec des méthodes approchées. Nous distinguons dans la littérature deux types de méthodes approchées étroitement liées : les heuristiques et les métaheuristiques.

Les heuristiques

Une heuristique peut être définie comme étant un algorithme stochastique itératif composé de règles empiriques simples basées sur l'expérience et qui agit de manière à progresser vers un optimum global.
Une heuristique diffère d'une métaheuristique par rapport à son domaine d'application. En effet, la particularité d'une métaheuristique est que contrairement à une heuristique, elle peut résoudre des problèmes de différentes nature sans pour autant s'y adapter profondément, d'où le préfixe grecque "Méta", pour indiquer le niveau supérieur.
Ce type d'approche est un recours efficace face à des problèmes complexes dans des domaines comme l'industrie, les services, la finance, la gestion de production et l'ingénierie.

Les métaheuristiques

Les métaheuristiques sont des algorithmes capables de guider et d'orienter le processus de recherche dans un espace de solution de taille importante.
Selon Blum & Roli [91] les métaheuristiques doivent remplir les propriétés suivantes :

- Offrir une stratégie générale qui guident le processus de recherche.
- Exécuter une exploration efficace de l'espace de recherche pour trouver des solutions optimales ou proches de l'optimalité.
- Leur comportement est non-déterministe (stochastique) et ne donne donc aucune garantie d'optimalité
- Posséder une structure de base prédéfinie
- Posséder des concepts de bases s'appuyant sur des concepts abstraits (Méta) (contrairement aux heuristiques qui ne peuvent être appliqués qu'à un problème spécifique).

Les métaheuristiques peuvent être divisées en deux catégories :

1. **Métaheuristiques à base de solution unique :**
   Elles permettent de définir aisément des méthodes avec une forte capacité d'intensification dans le but d'obtenir des solutions de bonnes qualités de manière rapide. L'évolution des métaheuristiques à base de solution unique et les recherches effectuées

dans ce domaine ont permis de combiner des phases de diversification et d'intensification et d'utiliser des voisinages multiples [92].

Parmi les métaheuristiques à base de solution unique nous pouvons citer : les algorithmes de recherche locale (LS), de recherche tabou (TS) et le recuit simulé (SA).

2. **Métaheuristiques à base de population de solutions :**
   Elles proposent un ensemble de solutions potentielles à un problème donné. Leur principe général consiste à combiner des solutions entre elles pour en former de nouvelles en essayant d'hériter des "bonnes" caractéristiques des solutions parents. Ce processus est répété jusqu'à ce qu'un critère d'arrêt soit satisfait (nombre de générations maximum (ou sans améliorations), temps maximum, borne atteinte, etc.).
   Parmi les métaheuristiques à base de population de solutions, nous pouvons citer : les méthodes évolutionnistes tels que les algorithmes génétiques (AG) ou les algorithmes de recherche globale tels que l'optimisation par essaim particulaire (PSO) ou l'optimisation par colonies de fourmis (ACO).

Approches hybrides

Bien que les méthodes approchées soient puissantes et efficaces, la combinaison entre elles peut augmenter leurs puissances et améliorer considérablement les résultats.
En outre, les approches hybrides sont le résultat de la combinaison de deux ou plusieurs algorithmes en utilisant les forces de chacune d'entre elles.
L'hybridation des méthodes de résolution peut donner lieu à deux types d'approches :

- **L'hybridation entre méthodes approchées :** l'hybridation entre les métaheuristiques basées population et les méthodes de recherche locale.
- **Math-heuristiques :** concerne l'hybridation de méthodes exactes avec des (méta) heuristiques en exploitant des techniques de programmation mathématique dans des cadres (méta)heuristiques ou en accordant à des approches de programmation mathématique la robustesse croisée à l'efficacité qui caractérisent les métaheuristiques.
  Ce type d'algorithme se présente sous la structure "maître-esclave" d'un processus de guidage et d'un processus d'application [93]. Soit (i) la métaheuristique agit à un niveau supérieur et contrôle les appels à l'approche exacte, soit (ii) la technique exacte agit en tant que maître et contrôle l'utilisation du schéma métaheuristique.

2.9 État de l'art sur la résolution du FJSP

Nous retrouvons dans la littérature deux grandes familles de méthodes pour la résolution des deux sous-problèmes d'ordonnancement (séquencement et affectation) :

1. **Méthodes séquentielles (ou hiérarchiques) :** résolution d'un sous-problème à la fois.
2. **Méthodes intégrées (ou concurrentes) :** résolution des deux sous-problèmes en parallèle (de manière simultanée).

Généralement les sous-problèmes d'affectation et de séquencement sont traités séparément à différents niveaux du système de production.
Selon Yazdani et al. [94], bien qu'il soit plus difficile de résoudre les deux sous-problèmes parallèlement (approches intégrées), les résultats obtenus sont généralement meilleurs que lorsque les deux sous-problèmes sont résolus séparément (approches séquentielles).

Le tableau 2.2 représente un état de l'art que nous avons réalisé et qui résume les articles les plus cités et référencés dans le domaine du FJSP. Pour cela, nous nous sommes basées sur les deux métriques LCS [1] et LCR [2] pour établir un état de l'art aussi complet qu'exhaustif. Classer de manière chronologique, ce tableau comporte cinq colonne : la première colonne indique le nom du ou des auteurs ainsi que la référence de leurs travaux. La deuxième colonne, indique le type d'approche, cette colonne peut contenir deux entrées : intégrée ou séquentielle selon la manière dont les deux sous-problèmes d'ordonnancement (affectation et séquencement) sont résolus. La troisième colonne indique le type de méthodes utilisées (exactes, heuristiques, métaheuristiques, hybrides). La quatrième colonne explicite la méthodologie de résolution du problème ainsi que la particularité et les apports de chacun des travaux mentionnés. La dernière colonne présente le ou les critères de performance (objectifs) optimisés.

1. Local Citation Score : nombre de fois que l'article x a été cité par les autres articles de l'ensemble de références analysées
2. Local Citation References : nombre d'articles, de l'ensemble de références, mentionné dans la liste de références de cet article

TABLE 2.2 – État de l'art sur la résolution du FJSP

Référence	**Type d'approche**	**Approches utilisées**	**Méthodologie de résolution**	**Objectif(s) optimisé(s)**
Brucker & Shlie [89]	Intégrée & Séquentielle	Algorithme polynomial	Résolution du FJSP avec deux Jobs. Ces travaux sont les précurseurs dans la résolution du FJSP	C_{max}
Brandimarte [95]	Séquentielle	Règles de priorité + TS	**Règles de priorité :** pour le sous-problème de routage. **TS :** pour le sous-problème de séquencement. Introduction des instances pour le FJSP générale avec divers degrés de polyvalence de production pour les machines.	C_{max} T_{max}
Hurink et al. [96]	Intégrée	Méthode de TS	**Recherche locale & deux voisinages :** basés sur le concept de bloc pour résoudre le problème avec machines à capacités multiples. Génération d'instances avec indépendance des temps de traitement des opérations par rapport aux machines.	C_{max}
Dauzère-Pérès & Paulli [97]	Intégrée	Structure de voisinage + TS	**Structure de voisinage** traitement parallèle de l'affectation et séquencement ainsi que de la réaffectation et du reséquencement. **TS :** classification des solutions voisines en utilisant un graphe disjonctif pour la connexion des voisins.	C_{max}
Mastrolilli & Gambardella [98]	Intégrée	TS améliorée + Deux fonctions de voisinage	Amélioration des techniques de Recherche Tabou de [97]	C_{max}
Mati et al. [99]	Intégrée	Heuristique gloutonne	Traitement simultanément des deux sous-problèmes d'affectation et séquencement pour le FJSP avec plus de deux jobs.	C_{max}
Mati et al. [100]	Intégrée	Heuristique gloutonne (GH) + Algorithme Génétique (AG)	**GH :** résolution du cas général avec plus de deux jobs. **AG :** guidage de l'heuristique gloutonne afin d'explorer des séquences de jobs intéressantes.	C_{max}

TABLE 2.2 – État de l'art sur la résolution du FJSP

Référence	**Type d'approche**	**Approches utilisées**	**Méthodologie de résolution**	**Objectif(s) optimisé(s)**
Kacem et al. [101]	Intégrée	Approche par localisation (AL) + Algorithme évolutionniste (AE)	Résolution de la flexibilité partielle et totale. **AL :** pour le problème d'allocation de ressources en localisant la ressource idéale et construire un modèle d'affectation adéquat. **AE :** contrôlé par le modèle d'affectation et utilisation de manipulations génétiques avancées pour accroitre la qualité de la solution.	C_{max} W_T
Kacem et al. [102]	Intégrée	Approche de Pareto hybride = Algorithme évolutif + Logique floue	Application du concept biologique des *organismes génétiquement modifiés (GMO)* pour améliorer la qualité de la solution.	C_{max} W_T W_M
Kim et al. [103]	Intégrée	AG symbiotique	Résolution des deux sous problèmes de l'ordonnancement simultanément	C_{max}
Xia et Wu [5]	Séquentielle	PSO + SA	**PSO :** affectation des opérations aux machines **SA :** séquencement des opérations sur chacune des machines.	C_{max} W_{max} W_T
Chan et al. [104]	Intégrée	AG à gènes dominants	La notion de gènes dominants est utilisée pour identifier et sauvegarder les meilleurs gènes du chromosome au cours de l'évolution. L'approche est utilisé de manière itérative pour évaluer les différents niveaux de flexibilité de l'affectation des opérations aux machines. Traitement de la flexibilité totale.	C_{max}

TABLE 2.2 – État de l'art sur la résolution du FJSP

Référence	**Type d'approche**	**Approches utilisées**	**Méthodologie de résolution**	**Objectif(s) optimisé(s)**
Liu et al. [105]	Séquentielle	VNPSO (Variable Neighborhood Particle Swarm Optimization) = PSO + SA	VNPSO pallie au ralentissement du PSO lorsque le nombre de générations augmente et évite une terminaison au minimum local. L'idée de base est de diriger les particules par une stratégie de shaking et de les amener à explorer des espaces de voisinage variables pour de meilleures solutions d'ordonnancement.	C_{max} C_{sum} (flow-time)
Fattahi et al. [6]	Intégrée	Deux approches intégrées (ISA et ITS)	• **ISA** : approche intégrée combinée avec une heuristique de recuit simulé • **ITS** : approche intégrée combinée avec une heuristique de Recherche Tabou	C_{max}
	Séquentielle	Quatre heuristiques séquentielles	Combinaison de quatre différentes heuristiques : le recuit simulé, le recuit simulé hybride, la recherche tabou et la recherche tabou hybride.	
Gao et al. [106]	Intégrée	AG hybride + Variable Neighborhood Descent (VND)	**AG :** utilise deux vecteurs (pour l'affectation et le séquencement). **VND :** appliquée dans chaque nouvelle génération pour améliorer la qualité de la progéniture avant de l'injecter dans la population.	C_{max} W_{max} W_T
Pezzella et al. [107]	Intégrée	Framework d'un AG	Intégration de différentes stratégies pour générer la population initiale, sélectionner les individus pour la reproduction.	C_{max}
Tay & Ho [66]	Intégrée	Programmation génétique + Règles de priorités composites (CDR)	**Framework de programmation génétique :** Génération de *cinq* CDRs en combinant : le temps de traitement, la date d'échéance, le nombre d'opérations et le temps total moyen d'exécution d'un job.	C_{max} $\bar{T}$ $\bar{F}$

TABLE 2.2 – État de l'art sur la résolution du FJSP

Référence	**Type d'approche**	**Approches utilisées**	**Méthodologie de résolution**	**Objectif(s) optimisé(s)**
Gen at al. [108]	Intégrée	AG multi-étapes avec changement du goulot d'étranglement	Résolution du FJSP avec flexibilités partielle et totale. AG multi-étapes est composé de : • K étapes (nombre total d'opérations par job) • m état (nombre total de machines).	C_{max} W_T W_C
Zhang et al. [109]	Intégrée	PSO + TS	**PSO :** Affectation des opérations aux machines et le séquencement des opérations sur chaque machine. **TS :** Amélioration du sous-problème de séquencement provenant de chaque solution obtenue.	C_{max} W_T W_C
Li et al. [110]	Séquentielle	TS hybride (HTSA)	**Recherche tabou :** production des solutions voisines dans le module d'affectation de machine **Algorithme VNS :** Recherche locale dans le module du séquencement d'opérations.	C_{max} W_{max} W_T
Bagheri et al. [111]	Intégrée	Algorithme immunitaire artificiel (AIA)	Combinaison de stratégies pour générer une population initiale ainsi que plusieurs mutations différentes pour la réaffectation le re-séquencement.	C_{max}
De Giovanni & Pezzella [112]	Intégrée	AG	Résolution du FJSP dans un environnement de production décentralisé. Les jobs sont traités par un FMU [3]	C_{max} de tout les FMU

3. Flexible Manufacturing Unit : système de plusieurs unités de fabrication flexibles

TABLE 2.2 – État de l'art sur la résolution du FJSP

Référence	**Type d'approche**	**Approches utilisées**	**Méthodologie de résolution**	**Objectif(s) optimisé(s)**
Xing et al. [113]	Séquentielle	KBACO = ACO + Modèle de Connaissance (Knowledge Model)	Le modèle intègre certaines connaissances via l'ACO et applique l'information existante pour guider la recherche.	C_{max}
Moslehi & Mahnam [114]	Intégrée	Méthode hybride = PSO + Méthode de recherche locale	**PSO :** Recherche approfondie de l'espace de solution **Recherche locale :** réaffectation des opérations aux machines et réordonanncement des résultats obtenus afin d'améliorer la vitesse de convergence.	C_{max} W_T W_C
Azab & Naderi [115]	Séquentielle	MILP + Heuristiques Gloutonnes	**MILP :** pour la résolution des problèmes de petites tailles. Trois heuristiques ont d'abord été adaptées au problème. Trois heuristiques gloutonnes ont été développées. L'idée générale étant d'insérer les opérations de manière itérative dans une séquence pour construire une permutation d'opération complète.	C_{max}
Karthikeyan et al. [116]	Séquentielle	Algorithme discret hybride des lucioles (HDFA) = Algorithme discret des lucioles (DFA) + LS	**DFA :** recherche approfondie de l'espace de solution. **LS :** réaffectation des machines aux opérations et reprogrammation des résultats obtenus à partir de DFA pour améliorer la vitesse de convergence.	C_{max} W_{max} W_T
Zhang & Rose [117]	Intégrée	Simulation de processus de production + AG	Des variables de décision sont utilisées pour représenter l'affection des opérations de chaque job sur les machines (comme un individu dans l'AG).	C_{max}

2.10 Conclusion

Nous avons abordé dans ce chapitre l'aspect commun des problèmes d'ordonnancement dans les systèmes de production manufacturier. Ce chapitre peut être divisé en deux grandes parties. La première partie du chapitre sert à replacer la fonction d'ordonnancement dans le cadre général des systèmes de production. Ainsi, nous avons tout d'abord définit l'ordonnancement dans un contexte industriel tout en exposant ses concepts de base et ses différentes classes avant de citer les types de problèmes d'ordonnancement les plus souvent étudiés dans la littérature et qui résultent de la manière dont sont acheminées les tâches ainsi que l'organisation des machines dans l'atelier.
Quant à la seconde partie, elle est en lien étroit avec la section précédente et sert de postulat de départ au troisième chapitre. En effet, dans cette partie, nous abordons le problème d'ordonnancement dans l'atelier qui fait office de sujet à notre thèse, à savoir le job shop flexible. Nous traitons donc plus en détail dans cette section le problème de l'ordonnancement dans le job shop flexible en dressant un état de l'art complet sur les différentes méthodes exactes et approchées rencontrées dans la littérature servant à résoudre ce type de problème d'optimisation NP-difficile. Nous avons présenté dans ce chapitre une revue de l'état de l'art sur les techniques/méthodes de résolution publiées dans la littérature pour résoudre le FJSP. Le problème d'ordonnancement du job shop flexible (FJSP) est une extension du problème du job shop classique et est difficile à résoudre en raison de sa nature NP-difficile. Les chercheurs ont essayé de proposer des approches de résolution efficaces au cours des 30 dernières années. Avec les progrès de la puissance de calcul, les approches pour résoudre les FJSP sont devenues de plus en plus efficaces et puissantes. Ce chapitre nous a permis de constater une certaine tendance de la part de la communauté à utiliser des méthodes approchées et plus particulièrement des métaheuristiques pour la résolution de ce problème.
Les métaheuristiques les plus courantes sont des méthodes hybrides car ces dernières tirent parti des forces de chacune des méthodes combinées pour trouver de meilleures solutions. Cependant, malgré la diversité des méthodes utilisées, la perpétuelle proposition et l'utilisation de nouvelles approches pour la résolution du FJSP [118] [119] [120], aucune des méthodes n'a été jugée comme étant la meilleure.

Chapitre 3

L'ordonnancement dans le job shop flexible avec contrainte de transport

Sommaire

3.1 Introduction

La résolution du FJSP en tenant compte de certaines contraintes fournit un cadre formel pour la prise de décision et le choix de la solution dans un problème d'ordonnancement. D'ailleurs, un problème est défini essentiellement par ses contraintes, en vue de caractériser l'ensemble des solutions admissibles.
Pour cela, il est impératif de proposer des algorithmes de complexité raisonnable pour résoudre le problème et satisfaire les contraintes prises en considération.
L'extension que nous étudions dans notre thèse, en plus des contraintes habituelles d'un FJSP (précédence, allocation, disjonctive, etc.), concerne l'inclusion du transport des jobs entre les différentes machines.
L'étude de la contrainte de transport dans l'ordonnancement peut être abordée sous différents aspects : plus court chemin, optimisation de l'utilisation des ressources de transport, planification de trajectoire, temps de transport entre ressources, etc. L'aspect du transport étudié dans notre thèse concerne les temps de transport d'un job entre deux ressources.
Du fait de la difficulté générée par l'ajout de cette contrainte, les travaux réalisés dans ce domaine sont relativement rares [39].
Ce chapitre servira d'état de l'art sur le problème d'ordonnancement avec transport. Nous commencerons par une revue générale de la littérature sur le problème d'ordonnancement avec transport dans les différents types d'atelier. Avant de présenter un état de l'art sur l'ordonnancement du FJSP avec les différents aspects de la contrainte de transport.

3.2 État de l'art sur l'ordonnancement avec transport

Afin d'illustrer de manière plus détaillée l'état de l'art sur le problème d'ordonnancement avec contrainte de transport, nous divisons l'état de l'art en deux parties bien distinctes.
La première partie de l'état de l'art est consacrée au problème d'ordonnancement avec transport dans les différents types d'atelier pouvant être rencontrés en milieu industriel (job shop, flow shop, flow shop hybride, etc.).
Cet état de l'art est présenté sous forme d'un tableau récapitulatif (Tableau 3.1) composé de cinq colonnes : la première colonne référence les travaux de recherches, la deuxième colonne identifie les approches utilisées pour la résolution du problème, la troisième colonne la définit la méthodologie de résolution ainsi que les particularités des méthodes employées, la quatrième colonne énumère le ou les objectifs optimisés et la dernière colonne présente le type d'atelier dans lequel les tâches ont été ordonnancées. Ce tableau sert de prélude afin de mieux présenter, ensuite, l'état de l'art concernant le type d'atelier qui fait l'objet de notre thèse, à savoir le job shop flexible.

3.2.1 L'ordonnancement avec transport dans les différents ateliers

Un certain nombre d'études incluent le transport, lors de la modélisation et de la résolution du problème d'ordonnancement, en se ramenant à des problèmes de flow-shop ou de job-shop. Ainsi, le problème d'ordonnancement du job-shop avec transport a été formalisé, la première fois en 1999 par Knust [121]. Hurink & Knust et Lacomme font partis des rares chercheurs à s'être intéressé aux problèmes d'ordonnancement avec transport. Parmi les travaux de Hurink & Knust, nous pouvons citer entre autre ceux sur le job shop généralisée avec un seul robot transporteur [122] ou du flow shop avec transporteur unique [123]. Quant à Lacomme, ses recherches se sont grandement concentrées sur le job shop avec transport et plusieurs robots différents [124] et [125] [126]. Parmi, les autres chercheurs à avoir étudiés le problème d'ordonnancement avec transport dans différents ateliers, nous pouvons citer, Afsar et al. [127] qui ont proposé un framework maître/esclave utilisant les algorithmes GRASP et ELS pour la résolution du JSP où le job est transporté entre les machines par une flotte de véhicules homogènes avec une capacité unitaire. Zeng et al. [128] ont traité une extension du Blocking Job Shop (BJS) [1] prenant en compte le transfert des tâches entre les machines en utilisant un nombre limité d'AGV. López et al. [129] ont étudié le makespan et les capacités inutilisées des ressources dans le job shop en tenant compte des activités de transport.

Le tableau 3.1 dresse un état de l'art sur les principaux travaux réalisés dans le domaine de l'ordonnancement avec transport dans différents types d'ateliers. Afin de bien résumer les différents travaux et de clairement illustrer les apports de chacun d'eux, nous avons divisé ce tableau en cinq colonnes : la première colonne présente la référence ainsi que les noms des auteurs, la deuxième colonne définit les types d'approches utilisées (exacte, heuristique, métaheuristiques, hybride, etc...), la troisième colonne explicite le fonctionnement de l'approche mentionné dans la précédente colonne ainsi que ses particularités, la quatrième énumère le ou les objectifs optimisées et la dernière colonne indique le type d'atelier d'ordonnancement étudié.

1. Le Blocking Job shop (BJS) est une extension du job shop sans contrainte de buffer. Cela signifie qu'après qu'un Job est terminé sur la machine courante, il reste sur cette machine jusqu'à ce que la machine suivante soit disponible.

TABLE 3.1 – État de l'art sur le problème d'ordonnancement avec transport dans les différents ateliers

Référence	**Approches Utilisées**	**Méthodologie de résolution/Particularité de l'approche**	**Objectif(s) optimisé(s)**	**Type d'atelier**
Bilge & Ulusoy [130]	Modèle de programmation non linéaire en nombres entiers mixtes (MIP) + Procédure itérative	**MIP :** Formulation des contraintes des deux sous-problèmes d'ordonnancements de tâches et de véhicules et interagissant à travers des fenêtres temporelles. **Procédure itérative :** génération d'un nouvel ordonnancement et construction de fenêtres temporelles à partir des temps d'exécution obtenus pour la recherche de nouvelles solutions. Proposition de la première bibliothèque d'instances pour le job shop avec transport (deux véhicules identiques).	C_{max}	Flexible Manufacturing System
Strusevich [131]	Heuristique	Ordonnancement des *Open shop* à deux machines en considérant les temps de transport job entre les machines.	C_{max}	Open shop
Abdelmaguid et al. [132]	Schéma de codage hybride AG/heuristique gloutonne	Combinaison du schéma proposée avec un ensemble d'opérateurs AG sélectionnés à partir de la littérature sur les applications des AG aux problèmes d'ordonnancement.	C_{max}	Job shop
Allaoui & Artiba [133]	Approche hybride	L'approche proposée résulte de : Trois règles de priorité (LPT, SPT et EDD) + le recuit simulé + un modèle de simulation flexible. Prise en compte des : temps de transport et de configuration, maintenance préventive et pannes stochastiques.	C_{max} T_{max} $\bar{T}$ $\bar{F}$ N_t	Flowshop hybride
Hurink & Knust [122]	Algorithme de recherche locale	Les mouvements du robot sont considérés comme une généralisation du problème de voyageur de commerce. Proposition d'une bibliothèque d'instances.	C_{max}	Job shop (un seul transporteur)

TABLE 3.1 – État de l'art sur le problème d'ordonnancement avec transport dans les différents ateliers

Référence	**Approches Utilisées**	**Méthodologie de résolution/Particularité de l'approche**	**Objectif(s) optimisé(s)**	**Type d'atelier**
Lacomme et al. [124]	Graphe Disjonctif + Algorithme de recherche locale	Ce travail étend la proposition de Hurink & Knust [122] en incluant plusieurs robots transporteurs.	C_{max}	Job shop (plusieurs RT)
Deroussi et al. [134]	Métaheuristique hybride (SA + Recherche locale itérative)	Classement des différents mouvements pour générer un sous-ensemble de voisins de la solution courante afin de les utiliser dans l'amélioration des solutions.	C_{max}	FMS (plusieurs AGV)
Caumond et al. [135]	MILP	Prise en compte des contraintes de : temps de transport, capacité limitée des ressources, gestion des tampons d'entrée/sortie, nombre maximum de tâches simultanément autorisées dans le système et déplacements vides du véhicule.	C_{max}	FMS (un AGV)
Boudhar & Haned [136]	Plusieurs heuristiques	Étude de l'ordonnancement des jobs préemptifs en prenant en compte les temps de transport. Obtention d'une borne inférieure de haute qualité pour le C_{max}.	C_{max}	Atelier à machines parallèles identiques
Naderi et al. [137]	Métaheuristique basée sur le SA	Établissement d'un compromis entre les mécanismes d'intensification et de diversification pour augmenter les performances. Intégration des temps de transport et de configuration .	C_{max} T_{max}	Flow shop hybride

TABLE 3.1 – État de l'art sur le problème d'ordonnancement avec transport dans les différents ateliers

Référence	**Approches Utilisées**	**Méthodologie de résolution/Particularité de l'approche**	**Objectif(s) optimisé(s)**	**Type d'atelier**
Naderi et al. [138]	Trois MILP + Six heuristiques + Trois métaheuristiques basées sur des systèmes immunitaires artificielles	MILPs : Formulation des deux systèmes (un et plusieurs transporteurs). Application de six heuristiques. Application de trois méta-heuristiques : 1. Algorithme immunitaire purement artificiel. 2. Algorithme immunitaire artificielle + Recherche locale. 3. Algorithme immunitaire artificielle + Recuit simulé.	C_{max}	• Flow shop (transporteurs illimités) • Flow shop (un seul transporteur)
Khoukhi et al. [139]	ILP + ACO	L'approche permet la gestion des critères suivantes : la capacité des mémoires tampons, les priorités et capacités de transfert des véhicules transporteurs, le coût de retard des opérations. L'objectif principal est de minimiser les pénalités de précocité/retard (earliness/tardiness) concernant les délais de livraison des produits finis.	C_{max} pénalités des E_{max}/T_{max}	Job shop (plusieurs RT)
Erol et al. [140]	Modèle multi-agents	Résolution du problème d'ordonnancement dynamique. Ordonnancement simultané des AGV et des machines. Utilisation des mécanismes de négociation/enchères entre les agents.	C_{max}	FMS (plusieurs AGV)
Lacomme et al. [126]	Algorithme mémétique + Code génétique	Ordonnancement simultané des machines et des transporteurs. Le code est divisé en deux parties : • *Sélection Ressource* pour le fonctionnement de la machine • *Séquencement* pour l'opération de transport.	C_{max}	• FMS (un RT) • FMS (plusieurs RT)

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

TABLE 3.1 – État de l'art sur le problème d'ordonnancement avec transport dans les différents ateliers

Référence	**Approches Utilisées**	**Méthodologie de résolution/Particularité de l'approche**	**Objectif(s) optimisé(s)**	**Type d'atelier**
Zhang et al. [141]	Heuristique de goulot d'étranglement modifié	Prise en considération des contraintes de transport et temps de traitement limités. Atelier représenté par un graphe disjonctif. Procédure de goulot d'étranglement + heuristique : affectation et séquencement des tâches de transport de manière itérative.	C_{max}	Job shop
Zeng et al. [128]	Deux modèles INLP + heuristique en deux étapes	L'heuristique combine une méthode d'amélioration de l'ordonnancement et une recherche locale avec une structure de voisinage proposée à partir d'un modèle de graphe disjonctif.	C_{max}	Blocking Job shop
Nageswararao et al. [142]	BPSVHA (Binary Particle Swarm Vehicle Heuristic Algorithm) = PSO + Vehicle heuristic	**PSO :** ordonnancement des machines. **Vehicle Heuristic :** heuristique intégrée responsable de l'affectation de véhicule.	C_{max} $\bar{T}$	FMS (plusieurs AGV)
Afsar et al. [127]	GRASP + ELS (Evolutionary Local Search)	Transport des jobs à travers les machines grâce à une flotte de véhicules homogènes.	C_{max}	Job shop

3.2.2 L'ordonnancement avec transport dans le FJSP

Depuis que le FJSP a été divisé en deux sous-problèmes par Brandimarte [95], différentes variantes du problème ont été étudiées. Malgré les différentes variantes du problème, l'objectif principal reste inchangé : affecter toutes les opérations aux machines en respectant la séquence d'opérations de chaque job, pour optimiser un ou plusieurs objectifs.
Le FJSP avec transport représente une de ces variantes, et bien que cette contrainte soit l'extension la plus naturelle du job-shop flexible, elle est rarement prise en compte, et les jobs sont généralement considérés comme naissant sur la machine. Cette extension, en plus de la flexibilité des machines, génère de nouvelles contraintes disjonctives entre les ressources de transport, souvent représentées par des robots de transport appelés Automated Guided Vehicles (AGV) ou navettes.
En terme de transport de produits (jobs), deux problèmes sont principalement considérés dans la littérature. D'une part, les problèmes de routage qui consistent à déterminer un chemin prédéfini sans collision pour atteindre la destination souhaitée (la ressource) et d'autre part une planification de mouvement qui consiste à générer une trajectoire sans collision de la configuration initiale à la destination souhaitée.
L'augmentation considérable de la complexité par rapport au *jsf* standard, suite à l'ajout du problème du mouvement des matériaux et des opérations de transport en font un sujet très peu abordé par la communauté.
Parmi les recherches menées dans ce domaine, Zhang et al. [33] ont proposé une approche hybride pour le FJSP avec des contraintes de transport. Bien que très efficace, l'approche n'a pas été testée sur des instances de grande taille ni sur un cas réel. Liu et al. [143] ont abordé le FJSP multi-objectifs avec des contraintes de ressources AGV en proposant un algorithme de colonie d'abeilles micro-artificielles pour minimiser le temps total de production et la charge totale des machines. Karimi et al. [144] ont proposé deux modèles mathématiques, l'un basé sur la séquence et l'autre sur la position et ont développé un nouvel algorithme compétitif impérialiste avec un algorithme de recuit simulé pour résoudre le FJSP avec temps de transport. Rossi [145] a proposé un algorithme ACO basé sur un graphe disjonctif où un modèle relation-apprentissage renforcé est implémenté en tenant compte de différentes contraintes comme la ressource alternative et le problème de transport.
Ainsi, après avoir cité certains des principaux travaux sur l'ordonnancement avec transport dans les différents types d'ateliers existants (tableau 3.1), nous nous concentrons cette fois sur le type d'atelier qui fait l'objet de notre thèse, à savoir le jsf avec temps de transport.
Le tableau 3.2 dresse un état de l'art sur le FJSP avec transport, certains des travaux majeurs de ce domaine y sont résumés. Tout comme pour les tableaux 2.2 et 3.1, les travaux sont classés de manière chronologique. Ce tableau comprend quatre colonne : dans la première colonne nous citons le ou les auteurs ainsi que la référence de leurs travaux. Dans la deuxième colonne, nous indiquons les approches utilisées. Les entrées de cette colonne peuvent être de type exacte (MILP, ILP,...), heuristique, métaheuristiques (ACO, PSO, SA,...) ou hybride. Dans la troisième colonne nous explicitons le fonctionnement des méthodes utilisées ainsi que leurs spécificités. Quant à la dernière colonne, nous y indiquons le ou les objectifs optimisés.

TABLE 3.2 – État de l'art sur le FJSP avec transport

Référence	**Approches Utilisées**	**Méthodologie de la résolution**	**Objectif(s) optimisé(s)**
Rossi & Dini [146]	ACO	Proposition d'un système logiciel basé sur l'ACO pour la résolution du FJSP en temps réel et prenant en compte les temps de transport et de configurations. Le problème est représenté par un graphe disjonctif et la génération du chemin est réalisée par un algorithme de recherche locale (basé sur une structure de voisinage) permettant aux fourmis de visiter le graphe.	C_{max}
Pezzella et al. [107]	Algorithme génétique	**AG :** intégrer différentes stratégies pour générer la population initiale et reproduire de nouveaux individus. **Approche AL de [101] :** trouver des affectations initiales et les améliorer en réorganisant les tâches et les machines.	C_{max}
Deroussi & Norre [147]	Recherche Locale Itérative (ILS) + Recuit Simulé (SA)	**ILS :** basée sur les déplacements classiques d'échanges, d'insertions et de perturbations. **SA :** utilisé pour le critère d'acceptation. Nouvelle bibliothèque d'instances proposée (extension de [130]).	C_{max}
Pandian et al. [148]	Algorithme génétique	Ordonnancement simultané des jobs et AGV. **AG :** basé sur la technique de saut de gènes entre chromosomes.	C_{max} Flux de matériel.
Zhang et al. [33]	Métaheuristique hybride (AG + TS)	**AG :** résolution du problème d'affectation des opérations aux machines **TS :** trouver de nouvelles solutions améliorant les précédentes.	C_{max} Stockage [2].
Liu et al. [143]	Multi-objective Micro Artificial Bee Colony (MMABC)	Chaque solution correspond à une source de nourriture codée pour représenter l'affectation des tâches AGV, des opérations de la machine et de la séquence d'opération.	C_{max} W_T

2. somme des temps d'attente des tampons entrée/sortie des ressources

TABLE 3.2 – État de l'art sur le FJSP avec transport

Référence	**Approches Utilisées**	**Méthodologie de la résolution**	**Objectif(s) optimisé(s)**
Rossi [145]	ACO basé sur un graphe disjonctif	Implémentation d'un modèle d'apprentissage relationnel renforcé prenant en compte le problème de transport, les ressources alternatives et les temps de configuration.	C_{max}
Karimi et al. [144]	Deux modèles MILP + Algorithme hybride compétitif impérialiste	**MILPs :** pour les instances de petite taille (l'un basé sur la séquence et l'autre sur la position). **Algorithme compétitif impérialiste & Recuit simulé** : pour les instances de grande taille.	C_{max}
Zhang & Rose [117]	Approche intégrée hybride (AG + simulation des processus de production)	**AG :** représentation des individus via des variables décisionnelles. **Simulation des processus de production :** évaluation de la faisabilité et la valeur du fitness des individus.	C_{max}
Nouri et al. [149]	Algorithme génétique basé sur le voisinage (NGA) + Recherche Tabou	**NGA :** appliqué par un agent ordonnanceur pour une exploration globale de l'espace de recherche. **Recherche Tabou :** utilisée par un ensemble d'agents cluster pour guider la recherche dans les régions prometteuses.	C_{max}
Deliktas et al. [150]	Mixed nonlinear integer programming + Méthodes de scalarisation	Optimisation mono et bi-objectifs dans un environnement de production intercellulaire. **Mono-objectif :** modèles mathématiques prenant en compte les pièces exceptionnelles, les mouvements intercellulaires, les temps de transport intercellulaire et de configuration ainsi que la re-circulation. **Bi-objectifs :** méthodes de scalarisation [3] pour convertir les objectifs du modèle mathématique en une fonction mono-objectif.	C_{max} T_{max}

3. sommes pondérées, ε-constraints et CSM

3.3 Conclusion

Nous avons présenté dans ce chapitre la variante du problème d'ordonnancement dans le job shop flexible faisant l'objet de notre thèse, le FJSP avec temps de transport.
Le FJSP étant de nature NP-difficile, la prise en compte d'une nouvelle contrainte concernant les temps de transport accroit la complexité du problème.
Le traitement de la contrainte de transport dans l'ordonnancement sous ses diverses formes (temps de transport, planification et ordonnancement des AGVs,...) est une contrainte rarement prise en compte malgré son fort impact sur la fonction d'ordonnancement et son lien direct avec le rendement de l'atelier.
Ce chapitre nous a permis de définir le problème étudié et de nous situer par rapport à l'état de l'art. Nous avons commencé par présenter les différentes approches présentes dans la littérature pour la résolution du problème d'ordonnancement avec transport dans divers types d'ateliers avant de nous concentrer sur le job shop flexible.
Le chapitre suivant sera consacré à la modélisation du problème étudié. Nous y présenterons le cas d'étude, le jeu d'instances utilisé lors de l'évaluation des performances des approches proposées ainsi que le modèle mathématique utilisé pour la formulation du problème du FJSP avec temps de transport.

Rapport-Gratuit.co

Chapitre 4

Cas d'étude

Sommaire

4.1 Introduction

Afin de valider une quelconque approche il est nécessaire d'effectuer des tests de performance en utilisant un cas d'étude. Pour les besoins d'évaluation des approches proposées dans notre thèse nous nous sommes basés sur une cellule flexible existante, "la cellule de production AIP-PRIMECA de l'Université de Valenciennes".
Cette cellule de production flexible a été présentée, modélisée et formalisée dans le Benchmark proposé par Trentesaux et al., [151].
Ce chapitre a pour but de présenter cette cellule ainsi que le modèle mathématique qui nous ont permis de modéliser, d'évaluer et de valider les approches proposées à la résolution du FJSP avec temps de transport.
Pour cela, ce chapitre sera divisé en deux grandes sections. En premier lieu, nous nous intéressons à l'aspect physique du problème en présentant les données relatives au cas d'étude (la cellule AIP-PRIMECA) en incluant les ressources/machines disponibles au sein de l'atelier, les produits, les ordres de fabrication prêt au montage ainsi que le système de transport utilisé pour transporter les jobs d'une ressource à l'autre. Puis, la second partie du chapitre sera concentrée à l'aspect logique du problème qui sera modélisé à travers une formulation mathématique. Nous décrirons dans cette partie, les différents paramètres, variables, hypothèses et contraintes prises en considération lors de la résolution du problème.

4.2 Présentation de la cellule AIP-PRIMECA

D'après Chaudhry & Khan [152] la majorité des articles de recherche traitent de simples FJSP et comme indiqué dans les chapitres précédents, les travaux prennant en considération les temps de transport sont très rares. Toujours, selon Chaudhry & Khan [152] plus de 90% des travaux de recherche sur le FJSP sont axés uniquement sur l'aspect théorique du problème en omettant les contraintes réelles rencontrées dans un système de production manufacturier tels que les temps de transport, la maintenance, la panne des machines, les temps de traitement flous/incertains, etc.
En raison, du faible taux de recherche sur les réelles contraintes rencontrées dans le FJSP. Les jeux de données respectant toutes les contraintes du domaine ne sont que très rares sauf pour certains benchmarks standards, tel que celui de Brandimarte [95] ou Taillard [153] qui malgré cela ne se base pas sur un cas d'usine concret.
Dans une optique de respect des réelles contraintes rencontrées dans un FMS, nous avons utilisé les instances du Benchmark proposé par Trentesaux et al. [151] et qui s'inspire d'une véritable cellule de montage : la cellule AIP-PRIMECA de l'Université de Valenciennes. Cette cellule est considérée comme un job shop flexible, menant ainsi à la formulation d'un problème d'ordonnancement dans un job shop flexible (FJSP). Ce Benchmark propose aussi différents scénarios permettant de tester l'efficacité et la robustesse d'une approche de résolution. Nous présenterons dans cette section les données clés du Benchmark, relatives aux ressources, aux produits et au système de transport.

4.2.1 Données relatives aux ressources/machines

La cellule de l'AIP-PRIMECA est composée d'un ensemble de composants industriels, comprenant des robots, des capteurs, des actionneurs et un système de manutention reliés par un système de transport afin de traiter un ensemble de tâches définies.
Cette cellule offre les types de flexibilité suivants : flexibilité de la machine (diverses opérations peuvent être effectuées sans changement de configuration) ; flexibilité de routage (existence de différentes routes de transfert entre au moins deux machines différentes) ; flexibilité du processus (un ensemble de produits peut être fabriqué sans modification majeure de la configuration) [154] ; et flexibilité de la séquence machine (des séquences machines alternatives peuvent être appliquées pour produire la même séquence) [155].
La cellule est composée de sept machines (Figure 4.1). Chaque machine peut effectuer différentes opérations, mais seulement une à la fois et sans interruption (non-préemptives).
Machines :

- **M1 :** unité de chargement/déchargement, composée d'un manipulateur Festo.
- **M2, M3 et M4 :** postes de travail d'assemblage, composés de robots Kuka.
- **M5 :** unité d'inspection automatique, avec une caméra Cognex.
- **M6 :** unité de récupération (l'unique poste de travail manuel dans la cellule).
- **M7 :** un poste de travail optionnel qui n'est utilisé que pour le scénario dynamique.

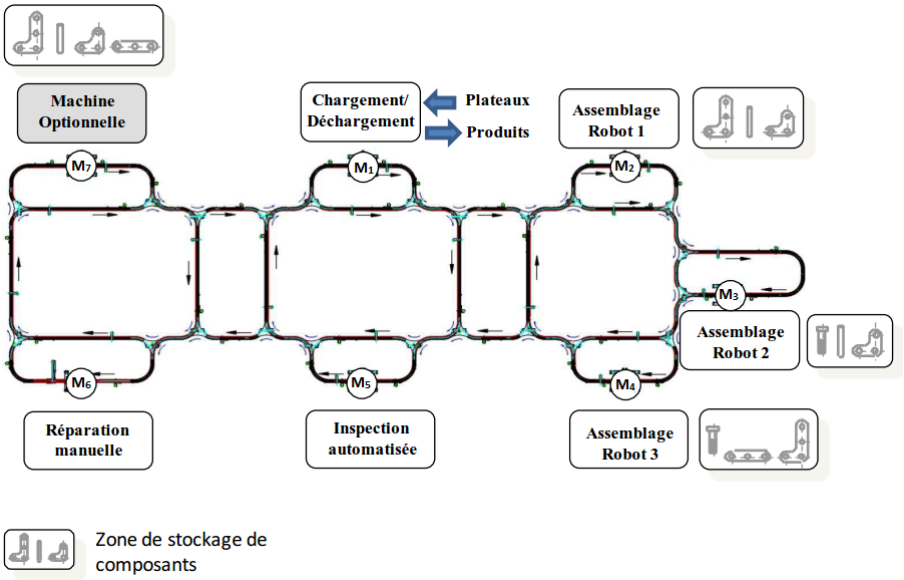

FIGURE 4.1 – Disposition des machines de la cellule flexible AIP-PRIMECA

La flexibilité du FJS peut être divisée en deux catégories selon Kacem et al. [101] :

- **Flexibilité Totale :** Chaque opération peut être effectuée par n'importe quelle machine m de l'atelier.

- **Flexibilité Partielle :** Certaines opérations ne peuvent être traitées que par une ou un nombre limité de machines m de l'atelier.

En se basant sur cette classification, la cellule peut être vue comme un job shop flexible partiel, où les opérations "Montage axe", "Montage composant r", "Montage composant l", "Montage composant L" et "Montage composant vis" peuvent être effectuées par plusieurs machines, tandis que les opérations "Inspection" et "déchargement plaques" sont effectuées par une seule machine (respectivement M5 et M1). Les postes de travail optionnels M6 et M7 n'ont pas été utilisés dans notre travail.

4.2.2 Données relatives aux jobs (produits) et opérations

Le tableau 4.2 indique les affectation des opérations aux ressources ainsi que les temps opératoires de chaque opérations. Les valeurs indiquées représentent le temps nécessaire (en seconde) à la machine/ressource pour effectuer l'opération. L'absence de valeur signifie que la ressource ne peut effectuer l'opération (notion de flexibilité partielle).

TABLE 4.1 – Temps de traitement des opérations (sec)

Id	**Opérations**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**
1	Montage_Axe		20	20				
2	Montage_r		20	20				
3	Montage_l				20			
4	Montrage_L		20		20			
5	Montage_Vis			20	20			
6	Inspection (Inspec)					5		
7	Chargement (Chrgt)	10						
8	Déchargement (Déchrgt)	10						

Dans la cellule flexible AIP-PRIMECA chaque job consiste en une séquence d'opérations qui commence par un chargement de plaque sur une navette, suivie d'autres opérations en fonction du produit, et se termine par une opération d'inspection et de déchargement.
Sept jobs différents peuvent être produits en réalisation le montage des composants bruts pour former les lettres suivantes : "B", "E", "L", "T", "A", "I" et "P" (Figure 4.2).
Le tableau 4.2 indique les opérations nécessaires à la production de chaque job.

TABLE 4.2 – Séquence de production des produits (jobs)

B	**E**	**L**	**T**	**A**	**I**	**P**
Chrgt	Chrgt	Chrgt	Chrgt	Chrgt	Chrgt	Chrgt
Axe	Axe	Axe	Axe	Axe	Axe	Axe
Axe	Axe	Axe	Axe	Axe	Axe	Axe
Axe	Axe	Axe	r_comp	Axe	l_comp	r_comp
r_comp	r_comp	l_comp	L_comp	r_comp	Vis_comp	L_comp
r_comp	r_comp	l_comp	Inspec	L_comp	Inspec	Inspec
l_comp	L_comp	Vis_comp	Déchrgt	l_comp	Déchrgt	Déchrgt
Vis_comp	Inspec	Vis_comp		Vis_comp		
Inspec	Déchrgt	Inspec		Inspec		
Déchrgt		Déchrgt		Déchrgt		

Les ordres de fabrication des clients peuvent prendre trois formes : "BELT", "AIP" et "LATE" (Figure 4.2). Ces ordres sont composés de trois ou quatre produits et les plus souvent commandés sont le "BELT" et le "AIP". L'ordre de fabrication est considéré comme achevé une fois tous ses produits réalisés.

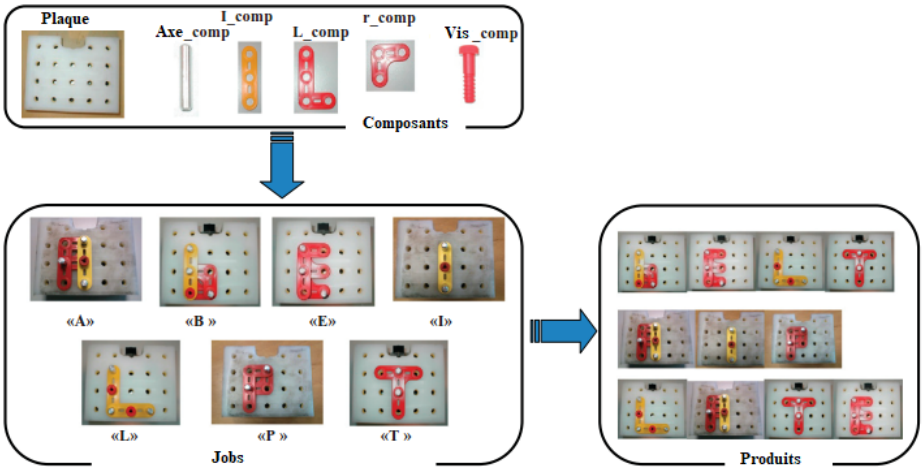

FIGURE 4.2 – Composants de l'AIP-PRIMECA

4.2.3 Données relatives au système de transport

Les ressources sont reliées entre elles via un système de transport monorail unidirectionnel (Figure 4.3). Ce système de transport est muni d'un système d'aiguillage rotatif se trouvant avant chaque divergence/convergence et peut être représenté par un graphe fortement connexe composé de deux type de nœuds :

- **Les nœuds machines :** M_1, M_2, M_3, M_4, M_5, M_6, M_7
- **Les nœuds de routage :** n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}, n_{11} (les décisions de routage sont prises au niveau de ces nœuds).

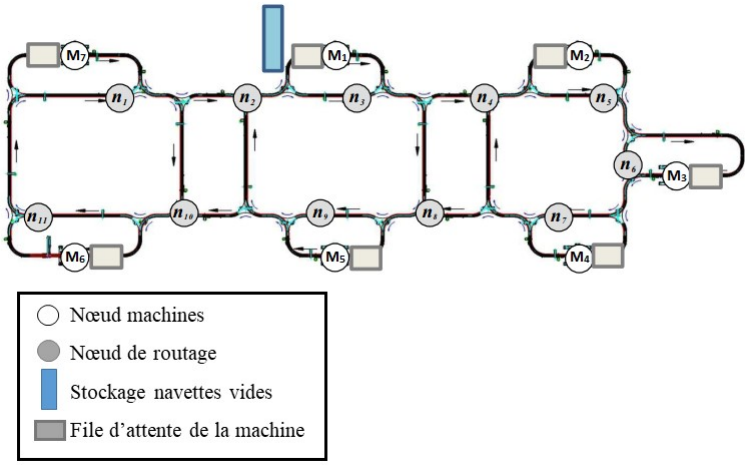

FIGURE 4.3 – Système de transport de l'AIP-PRIMECA

Si deux opérations successives du même job ne peuvent être effectuées par la même machine, une opération de transport via des navettes est nécessaire et le produit sera transporté vers une machine alternative fournissant le même type de service.
Le temps de transport dépend de la disposition des machines et une opération de transport consiste en la récupération, le transport et le dépôt du job (produit) au niveau de la machine adéquate. Pour notre cas d'étude une matrice de distance est fournie représentant les temps théoriques nécessaires aux opérations de transport.
Le tableau 4.3 représente les temps théoriques nécessaires aux transports d'un job d'un nœud à l'autre.

TABLE 4.3 – Temps de transport entre les nœuds

	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	M1	M2	M3	M4	M5	M6	M7
n1		4	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-
n2	-		4	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-
n3	-	-		4	-	-	-	5	-	-	-	-	-	-	-	-	-	-
n4	-	-	-		4	-	-	-	-	-	-	-	5	-	-	-	-	-
n5	-	-	-	-		3	-	-	-		-	-	-	11	-	-	-	-
n6	-	-	-	-	-		4	-	-	-	-	-	-	-		5	-	-
n7	-	-	-	5	-	-		4	-	-	-	-	-	-	-	-	-	-
n8	-	-	-	-	-	-	-		4	-	-	-	-	-	-	5	-	-
n9	-	5	-	-	-	-	-	-		4	-	-	-	-	-	-	-	-
n10	-	-	-	-	-	-	-	-	-		4	-	-	-	-	-	7	-
n11	9	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	10
M1	-	-	-	6	-	-	-	7	-	-	-		-	-	-	-	-	-
M2	-	-	-	-	-	5	-	-	-	-	-	-		13	-	-	-	-
M3	-	-	-	-	-	-	6	-	-	-	-	-	-		7	-	-	-
M4	-	-	-	7	-	-	-	6	-	-	-	-	-	-		-	-	-
M5	-	7	-	-	-	-	-	-	-	6	-	-	-	-	-		-	-
M6	12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		13
M7	-	6	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	

4.3 Définition mathématique du problème

En plus des contraintes de base (précédence, disjonctives, etc), nous prenons en compte dans notre étude des contraintes réalistes qui sont généralement omises. Cela concerne le transport entre deux machines dans le système de production, la capacité de la file d'attente des machines et la limitation des tâches dans l'atelier.

4.3.1 Hypothèses

- Toutes les machines sont disponibles au moment *t=0*
- Tous les jobs sont prêts à être traités au moment *t=0*
- Chaque job possède une séquence unique d'opérations. Cependant différentes routes peuvent être empruntées selon la combinaison des machines employées (flexibilité du routage)
- Certains services (opérations) sont fournis par une seule machine.
- Une opération ne peut être interrompue.
- Le temps de traitement d'une opération est le même sur les machines alternatives.
- Le nombre de machines et de jobs est connu au préalable.
- Une machine peut traiter une seule opération à la fois.

— Les machines sont indépendantes les unes des autres.
— Les pannes des machines et des navettes ne sont pas considérées.
— Les tampons d'entrées et de sorties sont considérés illimités
— Les temps de configuration (set up) sont négligés.

4.3.2 Notations pour les Paramètres

$\boldsymbol{J}$	ensemble de Jobs, J = { 1,2,...,j,..., $\|J\|$ }
$\boldsymbol{M}$	ensemble de machines, M = { 1,2,...,m,..., $\|M\|$ }
I_j	ensemble des opérations du job j, I_j = { 1,2,...,j,..., I_j }, j ∈ J
O_{ij}	opération i du job j
$TT_{mm'}$	le temps de transport entre la machine m et m'
Pt_{ij}	temps d'exécution de l'opération i du job j.
p_{ij}	début de l'opération i du job j
M_{ij}	ensemble de machines qui exécutent l'opération i du job j
NM_{ij}	le nombre de machine pouvant exécuté l'opération O_{ij}

4.3.3 Notations pour les Variables

$stp_{m,l}$	Temps de début de l'opération réalisée sur la machine m avec la priorité l
$Tr_{ijmm'}$	variable binaire : 1 si il y a une opération de transport entre les machines m et m' après l'exécution de l'opération i du job j ; 0 sinon
a_{ijm}	variable binaire : 1 si l'opération i du job j peut être réalisée par la machine m ; 0 sinon
b_{ijm}	variable binaire : 1 si l'opération i du job j est réalisée par la machine m ; 0 sinon
R_{ijml}	variable binaire : 1 si l'opération i du job j est effectuée sur la machine m avec la priorité l ; 0 sinon

4.3.4 Fonction Objectif

Comme indiqué dans le tableau 2.1 il existe différentes mesures de performance étudiées dans la littérature concernant l'ordonnancement.
Cependant, l'objectif le plus répandu quel que soit le type d'atelier est celui de la minimisation du Makespan aussi appelé C_{max}. En effet, d'après Chaudhry & Khan [152] environ 45% des travaux menés dans le domaine du FJSP ont pour but de minimiser le temps total de production.
Compte tenu du contexte industriel et de l'objectif principal de toute fonction d'ordonnancement, la fonction objectif que nous cherchons à optimiser dans nos travaux est le C_{max}.

Ce critère d'optimisation correspond au temps d'achèvement de la dernière opération dans l'atelier (la cellule) et est formulé de la manière suivante :

$$C_{max} = max_{\forall i \in I_j \forall j \in J}\ Pt_{ij}$$

4.3.5 Contraintes

— ***Contraintes de précédence*** : la première contrainte (1) assure que le traitement d'une opération ne peut commencer qu'après que l'opération précédente du même job soit terminée (cas de deux opérations du même job j). Cependant, si la même machine ne peut pas exécuter l'opération suivante, une opération de transport est effectuée vers la machine adéquate (temps de transport pris en considération). La seconde contrainte (2) assure qu'une opération ne peut commencer qu'une fois la machine libre (deux opérations de deux jobs différents).

$$p_{ij} + Pt_{(i+1)j} + \sum_{m,m' \in M_{ij}} TT_{mm'}\ Tr_{ijmm'} \leq p_{(i+1)j} \quad (1)$$

$$stp_{m,l+1} \geq stp_{m,l} + \sum_{ij} Pt_{ij}\ R_{ijml} \quad (2)$$

— ***Contraintes d'allocation*** : la contrainte (3) assure qu'une opération ne peut être effectuée que par une machine qui offre le service demandé. Les contraintes (4) et (5) garantissent qu'une opération n'est allouée qu'à une seule machine.

$$b_{ijm} \leq a_{ijm} \quad (3)$$

$$\sum_{m \in M} b_{ijm} = 1 \quad (4)$$

$$\sum_{m \in M} R_{ijml} = 1 \quad (5)$$

— ***Contrainte disjonctive*** : Une machine (ressource) peut effectuer une seule opération à la fois.

$$p_{ij} + Pt_{kl}\ a_{klm} \leq p_{kl} \qquad \forall \text{ i,k} \in \text{I}, \forall \text{ j,l} \in \text{J}, \forall \text{ m} \in M_{ij} \quad (6)$$

— ***Contrainte de transport*** : si deux opérations successives du même job ne sont pas effectuées par la même machine, alors une opération de transport est nécessaire.

$$a_{ijm} + a_{(i+1)jm'} \text{ - } 1 \leq Tr_{ijmm'} \qquad \forall \text{ i} \in I_j, \forall \text{ j} \in \text{J}, \forall \text{ m,m'} \in M_{ij}, \text{ m} \neq \text{m'} \quad (7)$$

4.4 Conclusion

Nous avons présenté dans ce chapitre le Benchmark sur lequel nous avons testé nos approches. La particularité de ce jeu de données proposé par Trentesaux et al., [151] est de s'inspirer d'un système de production réel, celui de l'AIP-PRIMECA de l'Université de Valenciennes.
Le fait que ce Benchmark prend en considération des contraintes réelles rencontrées dans les systèmes de production, tels que les temps de transport nous a encouragé à nous basé dessus pour développer et développer nos approches.
Après avoir présenté en détail les données relatives à ce cas d'étude, nous avons donné la formulation mathématique utilisée lors de la mise en œuvre des approches proposées en présentant les différents paramètres, variables et contraintes prises en considération lors de la résolution du FJSP avec temps de transport.
Le chapitre suivant présentera la première des deux approches proposées : l'approche IGIT (Iterated Greedy Insertion Technique).

Chapitre 5

Approche IGIT pour la résolution du FJSP avec temps de transport

Sommaire

5.1 Introduction

Notre thèse porte sur la résolution du problème d'ordonnancement dans le job shop flexible en prenant en considération les temps de transport. Bien que cette contrainte est d'une importance majeure, elle n'est que très rarement considérée dans la littérature (chapitre 3). Pour faire face à cette problématique nous proposons deux approches heuristiques : *Iterated Greedy Insertion Technique (IGIT)* et *Iterated Greedy Insertion Randomized Technique (IGIRT)*.
Nous présentons dans ce chapitre, la première de ces deux approches, l'heuristique (*IGIT*) [7] qui combine une heuristique gloutonne itérée itérée et une heuristique d'insertion.
Nous commençons ce chapitre par une définition et un bref état de l'art sur les heuristiques utilisées pour la mise en œuvre de l'IGIT, à savoir, les heuristiques gloutonnes itérées, et les heuristiques d'insertion. Ensuite, nous présentons l'approche proposée et exposons son algorithme complet ainsi que son fonctionnement. Puis, nous effectuons des expérimentations sur différentes instances de petites tailles et illustrons la manière dans l'approche résoud les deux sous-problèmes d'affectation et de séquencement. Enfin, nous clôturons ce chapitre par une présentation et une analyse du comportement de l'approche face aux instances de grandes tailles.

5.2 Approche proposée : Iterated Greedy Insertion Technique (IGIT)

5.2.1 Heuristiques de construction gloutonne

Les algorithmes constructifs élaborent des solutions potentielles aux problèmes d'optimisation ou de décision, étape par étape, à partir d'une solution initiale vide. À chaque étape, un composant de solution est ajouté à la solution partielle actuelle et ce procédé est répété jusqu'à obtention d'une solution candidate complète. Les algorithmes constructifs utilisent généralement une fonction heuristique qui estime pour chaque composant de la solution l'intérêt de l'inclure dans une solution candidate partielle. Le meilleur composant est choisit, généralement, en fonction de son potentiel en termes de contribution à la minimisation locale de la fonction objectif. L'heuristique calculera donc le profit de chaque élément. L'optimalité locale n'implique pas une optimalité globale.
À la base, un algorithme de construction est formé d'algorithmes dits "gloutons" (constructifs) qui ajoutent à chaque étape un composant de solution pour lequel la valeur de la fonction heuristique est la meilleure. Si plusieurs composants de solution ont la même meilleure valeur heuristique, un critère de départage est utilisé pour décider quel composant de solution est réellement ajouté ; dans le cas le plus simple, cette égalisation se fait de manière uniforme et aléatoire, mais elle peut également être effectuée par une fonction heuristique secondaire.
La fonction heuristique gloutonne représente généralement l'augmentation incrémentielle de la fonction de coût due à l'incorporation de l'élément choisi, dans la solution partielle en

construction. Le critère d'intégration établit que l'élément possédant la plus petite augmentation incrémentale est sélectionné.
L'algorithme 1 représente le schéma basique d'une heuristique de construction gloutonne.

Algorithm 1 Schéma d'une heuristique de construction gloutonne

```
1: s = ∅
2: while s n'est pas une solution complète do
3:     Choisir le meilleur composant évalué c ;
4:     s = s+c ;
5: end while
```

Malgré leur rapidité à générer des solutions, ces dernières ne sont pas souvent optimales. Pour cela, les algorithmes gloutons sont généralement utilisées pour construire des solutions initiales qui sont ensuite améliorées à l'aide d'autres heuristiques ou de méthodes de recherche locale.
Un algorithme glouton construit une solution pas à pas et est caractérisé par les propriétés suivantes :

- Aucun retour arrière (backtracking) sur une décision n'est possible ;
- Le choix est intuitif et par conséquent le potentiel meilleur choix est effectué à chaque étape ;
- Obtenir un résultat optimum global ;
- Peu couteux (comparé à une énumération exhaustive).

D'après Talbi [156] les algorithmes constructifs gloutons sont utilisés dans la résolution de plusieurs problèmes d'optimisation, au vue du nombre considérable d'avantages qu'ils offrent. En effet, ils sont faciles à concevoir et sont couramment utilisés pour accélérer la recherche, ils ont un temps polynomial réduit et conduisent souvent à des optimums locaux d'excellente qualité.

5.2.2 Heuristique gloutonne itérée

L'heuristique gloutonne itérée utilisée dans l'approche IGIT est la combinaison, elle aussi, de deux méthodes : algorithme glouton constructif et d'un algorithme glouton itéré.
Les algorithmes gloutons itérés (Iterated Greedy Algorithms) sont étroitement liés à d'autres méthodes de recherche locale stochastique en général et particulièrement à la recherche locale itérée (Iterated Local Search) [157].
Les principaux avantages de l'algorithme glouton itéré sont la facilité de mise en œuvre et l'indépendance du framework par rapport aux propriétés spécifiques du problème adressé.
L'heuristique gloutonne itérée (Iterated Greedy) est une approche conceptuellement très simple pouvant être utilisée pour améliorer la performance d'une heuristique gloutonne, qui est utilisée comme une boîte noire dans l'algorithme. L'idée de base est d'itérer un cycle de Destruction-Construction. De nouvelles solutions sont construites et évaluées à chaque

itération jusqu'à ce que la condition d'arrêt soit atteinte [158]. Ce processus nécessite trois phases : *la destruction*, *la construction* et *le critère d'acceptation* pour construire une solution réalisable [159] :

- La phase de destruction est répétée dans le but d'éliminer une partie de la solution.
- La phase de construction applique un algorithme glouton à partir de la solution partielle obtenue lors de la phase précédente afin d'obtenir une nouvelle solution en insérant des parties de solution jusqu'à obtenir une solution complète.
- Le critère d'acceptation est appliqué à la solution candidate pour déterminer si elle améliore la solution précédente afin d'être remplacée ou pas.

Le framework de l'algorithme glouton itéré est présenté d'une manière générale dans 'algorithme 2.

Algorithm 2 The Iterated Greedy Framework

```
1: Input : Une solution initiale P
2: while termination_criterion do
3:     P_R = Destruction (P) ;
4:     P' = Construction (P_R) ;
5:     P = AcceptanceCriterion (P, P') ;
6: end while
```

5.3 Un état de l'art

Les algorithmes gloutons itérés sont utilisés dans divers problèmes d'optimisation et différents types d'ateliers. Nous présentons dans cette section, un bref état de l'art sur certains travaux pertinents ayant utilisés l'algorithme glouton itéré pour la résolution d'un problème d'ordonnancement.

Mati et al. [99] furent des pionniers dans l'utilisation des algorithmes gloutons itérés pour la résolution du FJSP, et cela en proposant une approche intégrée (résolution simultané du séquencement et affectation) basé sur une procédure gloutonne itérative pour le traitement du FJSP avec temps d'exécution différents entre machines. Plus tard, Mati et al. [100] proposèrent un autre algorithme glouton guidé par un algorithme génétique afin d'explorer des séquences de job intéressantes dans le job shop multi-ressources avec flexibilité de la ressource (variante du job shop flexible). Framinan & Leisten [160] proposent une méthode de recherche gloutonne itérée multi-objectifs pour l'ordonnancement du flowshop avec minimisation du *makespan* et *temps d'écoulement (flowtime)*. Cet algorithme utilise une heuristique constructive itérative intégrée dans une approche de recherche locale gloutonne où seules les solutions partielles non dominées sont conservées lors de la phase de construction des solutions. Ying & Cheng [161] ont proposé une heuristique gloutonne itérée pour l'ordonnancement dynamique dans les ateliers à machines parallèles avec temps de configuration dépendant des séquences. Naderi et al. [162] présentent un algorithme glouton itéré pour résoudre le problème d'ordonnancement dans les lignes de flux flexible (*flexible flow*

line (FFL)[1]) [163] avec contrainte des temps d'installation dépendants des séquence afin de minimiser le temps d'exécution total pondéré. Minella et al. [164] proposent un nouvel algorithme multi-objectifs basé sur les idées de la méthodologie gloutonne itérée pour l'ordonnancement dans le flowshop. Cet algorithme se caractérise par une initialisation efficace de la population, la gestion du front de Pareto et une recherche locale spécialement adaptée pour optimiser les objectifs du *Makespan*, du *retard (tardiness)* et du *temps d'écoulement (flowtime)*. Naderi et al. [165] considèrent un problème bi-objectifs d'ordonnancement de camions dans les systèmes de cross-docking[2] avec stockage temporaire et proposent pour cela un algorithme glouton itérée multi-objectifs où ils considèrent les objectifs du makespan et du retard total. Azab & Naderi [115] ont proposé trois heuristiques gloutonnes pour l'ordonnancement dans le job shop distribué (mutli-installation) où l'idée de base est d'insérer de manière itérative des opérations (une à chaque itération) dans une séquence pour constituer une permutation complète des opérations. Pranzo & Pacciarelli [166] proposent une métaheuristique gloutonne itérée pour la résolution des deux variantes du blocking job shop (avec et sans permutations d'opérations entre les machines). Karabulut [167] propose un algorithme glouton itéré hybridé avec un algorithme de recherche aléatoire. Cette approche utilise une nouvelle formule de calcul du critère d'acceptation pour résoudre le problème d'ordonnancement du flowhsop de permutation dans le but de minimiser le retard total (*total tardiness*). Dubois-Lacoste et al. [168] ont proposé un algorithme glouton itéré pour l'ordonnancement du flowshop de permutation et utilisent une méthode de recherche locale pour améliorer les solutions partielles après la phase de destruction.

5.4 Méthodologie de la solution

5.4.1 Heuristique d'insertion

Les algorithmes d'insertion sont des heuristiques constructives simples et très efficaces. Dans notre cas, l'heuristique d'insertion commence par un sous-ensemble d'opérations de fabrication à insérer sur la machine appropriée, ensuite une séquence d'opérations est établie progressivement en ajoutant une opération à la fois en utilisant une fonction fitness gloutonne afin de construire une solution satisfaisante. Cette technique est avantageuse en termes de temps d'exécution et de qualité de solution, en particulier dans les cas d'instances à grande taille.

La vitesse à laquelle une solution initiale est générée peut conduire à une importante déviation en terme de qualité de solutions obtenues au fur et à mesure des itérations de l'algorithme. Dans ce cas, les algorithmes gloutons peuvent être utilisés pour garantir la solution finale. Cependant, l'utilisation de meilleures solutions comme solutions initiales ne conduit pas tou-

1. Dans un FFL, un ensemble de n tâches doit être traité à un ensemble de g étages de production, chacune ayant plusieurs machines identiques en parallèle

2. Technique logistique de gestion des approvisionnements utilisée notamment dans les industries de transport et de produits périssables où le but est de faire passer des marchandises des quais d'arrivée aux quais de départs, sans passer par le stock.

jours aux meilleurs optimums locaux [169]. Par conséquent, une stratégie hybride combinant les techniques d'insertion et une heuristique gloutonne itérée peuvent être utilisée.
Ainsi, les solutions initialement construites résultent de méthodes gloutonnes, auxquelles s'appliquent ensuite des techniques d'insertions dans le but de réduire la convergence prématurée [156]. Cependant, pour la validation de cette approche un compromis ainsi qu'un bon équilibre doivent être trouvés entre l'utilisation de solutions aléatoires et gloutonnes pour garantir la bonne qualité des solutions en un temps de calcul réduit notamment pour les instances de grande taille et la réduction de la probabilité de convergence prématurée vers un optimum local.

5.4.2 Algorithme complet (IGIT)

L'algorithme de l'approche IGIT a été initialement proposé dans [7] puis amélioré dans [8] et se déroule de la manière suivante :

Algorithm 3 Iterated Greedy Insertion Technique

1: $SO \leftarrow \{(O_ij)\}$; i.e. toutes les opérations (SO est un ensemble d'opérations) ;
2: $Solution \leftarrow \varnothing$; initialement un ensemble vide, puis a ensemble de triplé <opération, machine, position>
3: **while** SO n'est pas vide **do**
4: o ← next operation in SO ;
5: $Cmax\_min \leftarrow BN$; BN : big number
6: **for** chaque machine m de M_{ij} (ensemble de machines pouvant traiter O_{ij}) **do**
7: **for** chaque position p de P_m (ensemble de positions dans la séquence de la machine m) **do**
8: Insérer l'opération o à la position p de la machine m ;
9: **if** $Cmax\_min > Cmax_x$ **then**
10: $Cmax\_min \leftarrow Cmax_x$;
11: $Best\_Machine \leftarrow m$;
12: $Best\_Position \leftarrow p$;
13: **end if**
14: **end for**
15: **end for**
16: $Solution \leftarrow SolutionU\{(o, Best\_machine, Best\_position)\}$;
17: $SO \leftarrow SO - \{o\}$;
18: **end while**

Critère d'acceptation

Une nouvelle solution réalisable (*Solution U*) est toujours acceptée si elle est meilleure que celle existante (*Solution*), c'est-à-dire si $Cmax\_min > Cmax_x$. Si cette condition est remplie, alors la solution candidate (*Solution U*) devient la nouvelle meilleure solution. Par conséquent, l'opération o est retirée de l'ensemble des opérations SO et affectée à la *Best_machine* de *Solution U* et insérée à la séquence *Best_position* de cette machine.

Critère d'arrêt

Différents critères d'arrêt peuvent être définis, tels qu'une limite de temps, un nombre maximum d'itérations, etc. Dans notre cas, après avoir effectuer plusieurs tests, le critère d'arrêt que nous avons retenu correspond au nombre maximal d'itération.
La meilleure solution a été retenu après avoir exécuter l'algorithme de l'IGIT 10 fois et lors de chacune de ces exécutions, un certain nombre d'itérations du cycle construction-deconstruction de solutions a été effectué.

5.5 Expérimentations

Afin de tester la performance de l'approche IGIT proposée, l'ensemble de données de référence C0 du Benchmark [151] a été utilisé.
Tout comme mentionné dans le chapitre 4, ce Benchmark est une instanciation des jeux de données extraits de la cellule de l'AIP-PRIMECA (Pôle de Ressources Informatique pour la Mécanique) de l'Université de Valenciennes. Les données essentielles à la résolution du problème du FJSP avec temps de transport ont été définies dans le chapitre précédent et sont disponibles en intégralité sur le lien : `http://www.univ-valenciennes.fr/bench4star/content/download`.
L'approche IGIT a été codée en Java, et toutes les expérimentations ont été menées sur un processeur Intel Core i7-4510 avec 2.6 GHz et 8 Go de RAM.
Les résultats de l'approche proposée IGIT sont comparés avec ceux de la méthode MILP (Mixed Integer Linear Program) et l'approche Potential Fields (PF).
Le MILP utilisé représente le modèle formel du Benchmark [151] et est implémenté à l'aide d'IBM ILOG CPLEX Optimizer. La solution générée est de type offline, elle prend en compte toutes les contraintes, ce qui rend la résolution du problème très difficile.
Le MILP étant une méthode de résolution exacte, les résultats obtenus avec cette méthode pour des instances de taille réduite sont optimaux. Cependant, cette méthode réagit très mal aux perturbations et changements, ce qui a un impact négatif et direct sur les résultats et ne correspond donc pas à la nature complexe et flexible du FJSP (environnement en perpétuels changement et évolution). Ajouter à cela le fait que le temps de résolution croit de manière exponentiel à mesure que la taille des instances évolue.
Pour palier à ce genre de problème certaines mesures et configurations sont effectuées, telle que la limitation du temps de résolution maximal à "1 heure" pour éviter toute explosion combinatoire, bug de la machine, ou temps de calcul de solution interminable. S'ajoute à cela, le re-paramétrage du modèle dans le cas de l'apparition d'une perturbation, entraînant par conséquent une perte de temps importante et rend cette méthode obsolète pour la résolution des grandes instances.
La négligence de ces contraintes malgré leur importance cruciale et leur incidence directe sur le temps total de production sont les raisons pour lesquels nous avons intégré une technique gloutonne aux deux approches proposées.
Cette technique une fois intégrée à nos approches nous permet de gérer des instances de

grandes tailles jusque là jamais aborder. En effet, bien que l'approche MILP et les champs potentiels se soient révélés efficaces, ces techniques gèrent très mal les instances de grandes tailles ce qui représente un inconvénient majeur. D'après, Pinedo [170], la méthode du MILP ne peut résoudre en générale pas plus de 10 lots et peut difficilement trouver une solution pour les instances composées de plus de 20 jobs, ce qui représente un nombre relativement insignifiant dans un contexte industriel (et encore plus dans une production en série).
Par conséquent, résoudre le problème avec une méthode exacte peut s'avérer interminable, compte tenu de la taille des instances (nombre élevé de tâches et donc nombre élevé d'opérations).

TABLE 5.1 – Comparaison du Makespan

Product	**IGIT**	**MILP**	**PF**
A-I-P	216	216	-
2 x A-I-P	346	326	-
3 x A-I-P	522	482	-
B-E-L-T-A-I-P	**419**	570	448

Le tableau 5.1 compare les résultats du *makespan* [8] obtenus avec l'approche proposée "IGIT", la méthode exacte MILP et la méthode approchée des champs potentiels (PF) sur les ordres de fabrication standards présentés dans le Benchmark de l'AIP-PRIMECA à savoir : A-I-P, 2 x A-I-P, 3 x A-I-P et B-E-L-T-A-I-P [151]. Notons que la méthode des champs potentiels n'a été testée que sur l'ordre de fabrication B-E-L-T-A-I-P d'où l'absence de résultat pour les autres type instances [151].
Nous pouvons remarquer que l'approche "IGIT" donne de meilleurs résultats que le MILP et les champs potentiels (PF) sur l'ordre de fabrication standard B-E-L-T-A-I-P.
Avec un C_{max} de **419**, l'approche IGIT enregistre des gains importants en matière de temps de production comparé aux méthodes avec lesquels le scénario B-E-L-T-A-I-P a été testé. En effet, avec un gap de 6,48% comparé au PF et de 26,5% comparé à la méthode exacte MILP, l'approche IGIT se montre très efficace avec ce type d'instances.

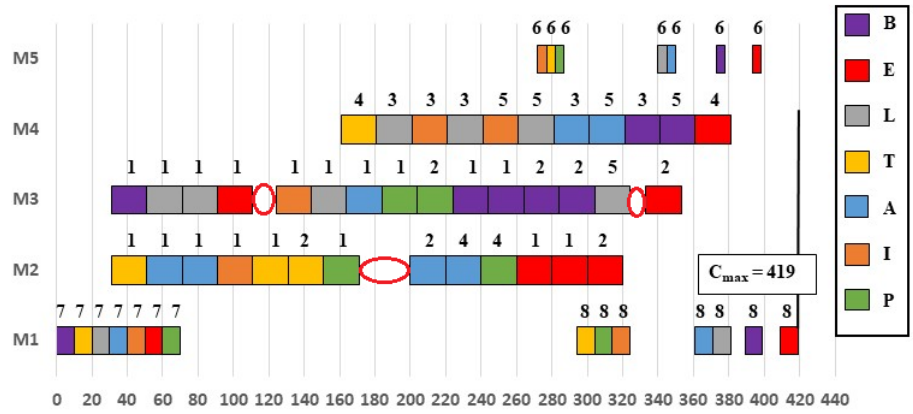

FIGURE 5.1 – Diagramme de Gantt pour l'ordre de fabrication B-E-L-T-A-I-P

La figure 5.1 représente l'ordonnancement de l'ordre de fabrication B-E-L-T-A-I-P avec l'approche IGIT [8]. Nous pouvons observer dans cette figure, une totale exploitation de la machine M4 sans aucun temps d'inactivité. En effet, cette ressource fonctionne sans arrêt pendant 220 secondes (temps nécessaire pour 11 opérations d'assemblage).
Avec 15 opérations d'assemblage à son actif, la machine M3 est la plus sollicitée des trois ressources d'assemblage.

5.6 Analyse de l'IGIT sur les grandes instances

Compte tenu des résultats satisfaisants de l'approche IGIT sur des instances de taille réduite (A-I-P, 2 x A-I-P, 3 x A-I-P et B-E-L-T-A-I-P) et le fait que les approches de [7], [171] et [172] n'ont été testé que sur des problèmes de petite taille, nous avons jugé intéressant d'évaluer notre approche sur des instances de plus grandes tailles.
Pour cela, nous avons simulé notre approche sur trois types d'instances de grandes tailles : 10 x A-I-P, 5 x B-E-L-T-A-I-P et 10 x B-E-L-T. Ces instances prennent respectivement en charge 30, 35 et 45 jobs et peuvent traiter jusqu'à 600 opérations (5 x B-E-L-T-A-I-P).

L'approche IGIT étant une méthode gloutonne itérée hybride, un certain nombre de cycle de déconstruction et reconstruction partielle des solutions est effectué.
Comme nous l'avons expliqué précédemment, les résultats obtenus sont comparés avec ceux du cycle précédent jusqu'à ce que le C_{max} ne peut être amélioré. Par conséquent, le nombre d'itérations effectué lors du déroulement de l'approche IGIT représente un paramètre essentiel pour la garantie d'une solution optimale et de bonne qualité.

Le tableau 5.2 représente l'évaluation des performances de l'approche IGIT sur les instances de grandes tailles précédement mentionnées. En plus des résultats obtenus, nous présentons dans ce tableau les paramètres nécessaires (nombre d'itérations et temps CPU)

pour traiter des instances de grandes tailles [8].

TABLE 5.2 – Performances de l'approche IGIT sur les grandes instances : 10 x A-I-P, 10 x B-E-L-T et 5 x B-E-L-T-A-I-P.

Nombre d'itérations	**Produit**	**Nombre de jobs**	**Nombre d'opérations**	**Cmax (IGIT)**	**CPU time (sec)**
10.000	10 x A-I-P	30	240	1773	9.878
	10 x B-E-L-T	40	360	2822	16.838
	5 x B-E-L-T-A-I-P	35	600	2258	12.194
50.000	10 x A-I-P	30	240	1713	39.176
	10 x B-E-L-T	40	360	2813	81.994
	5 x B-E-L-T-A-I-P	35	600	2206	57.542
100.000	10 x A-I-P	30	240	1692	125.149
	10 x B-E-L-T	40	360	2735	214.406
	5 x B-E-L-T-A-I-P	35	600	2201	107.817
500.000	10 x A-I-P	30	240	1653	508.731
	10 x B-E-L-T	40	360	2670	1127.886
	5 x B-E-L-T-A-I-P	35	600	2171	595.277

Dans un problème d'ordonnancement, l'objectif premier de la méthode de résolution est de réduire les coûts tout en optimisant la production. Par conséquent, le temps consommé est un facteur essentiel pour déterminer l'efficacité d'une approche.
Après avoir présenté le premier paramètre, à savoir, le nombre d'itérations et montré son importance sur le bon fonctionnement de l'approche IGIT, nous nous concentrons sur le second paramètre, à savoir, le temps CPU consommé pour obtenir une solution optimale.
La Figure 5.2 illustre et synthétise le temps CPU nécessaire à l'approche IGIT pour atteindre un C_{max} optimal pour les scénarios : 10 x A-I-P, 10 x B-E-L-T-A-I-P et 5 x B-E-L-T-A-I-P. Dans notre cas, le temps CPU, représente la durée nécessaire à l'approche IGIT pour trouver la combinaison optimale d'affectation et de séquencement pour obtenir le meilleur temps de production totale (C_{max}) selon l'ordre de fabrication (instance) exécuté.

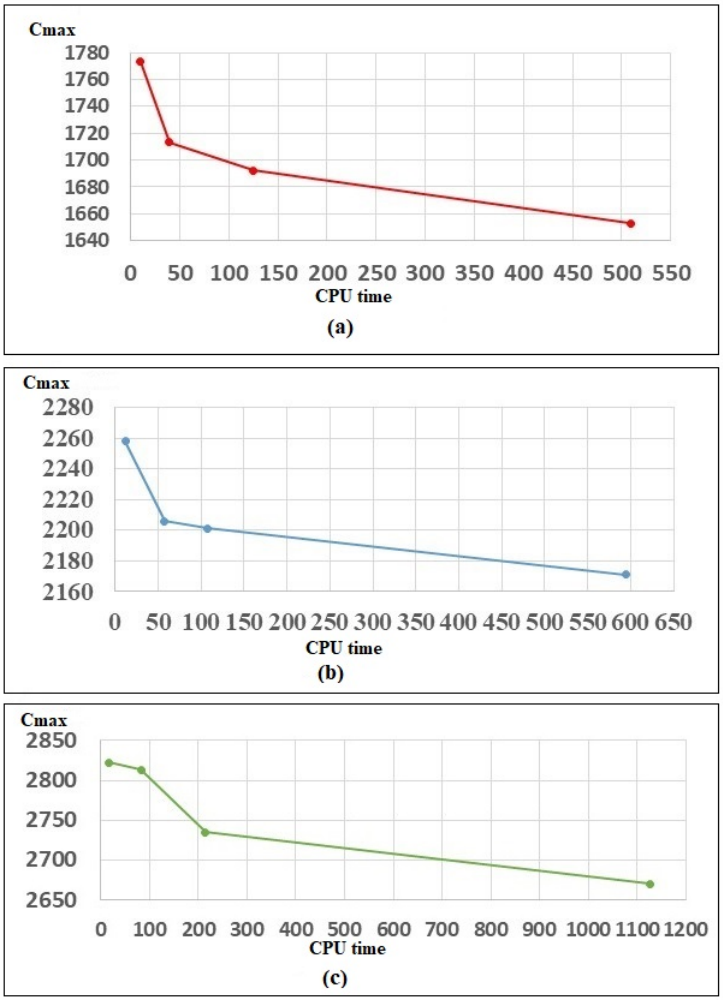

FIGURE 5.2 – Simulations de l'heuristique IGIT pour : (a) 10 x A-I-P, (b) 5 x B-E-L-T-A-I-P et (c) 10 x B-E-L-T

Nous remarquons que pour les scénarios : 10 x A-I-P et 5 x B-E-L-T-A-I-P (figure 5.2 sections "a" et "b"), l'approche IGIT a le même comportement, autrement dit, elle décroit d'une manière considérable durant les 50 premières secondes après le lancement de la simulation, puis progressivement les 50 secondes suivantes commence à diminuer graduellement jusqu'à se stabiliser après environ dix minutes. Le scénario inverse se produit avec le jeu de données 10 x B-E-L-T (figure 5.2 (section "c")). En effet, le temps de convergence observé est deux fois plus élevé comparé aux deux scénarios précédents et que cet ordre de fabrication nécessite un nombre élevé d'itérations et par conséquent de temps CPU pour obtenir un C_{max} de bonne qualité.

5.7 Conclusion

Dans ce chapitre, nous avons présenté la première des deux approches proposées dans notre thèse pour la résolution du problème d'ordonnancement du job shop flexible avec temps de transport, l'approche Iterated Greedy Insertion Tehcnique (IGIT).
Cette nouvelle approche est le résultat de la combinaison d'une heuristique gloutonne itérée et d'une technique d'insertion.
L'approche a été testée sur le jeu d'instances du Benchmark proposé par Trentesaux et al., [151] représentant une cellule flexible de production réelle.
Les résultats obtenus sont intéressants, et l'approche nous permet d'obtenir de meilleurs résultats que les méthodes anciennement testées sur certains types d'instances. En plus de cela, l'approche IGIT nous permet de traiter des instances de grande tailles du Benchmark [151], chose qui n'a jamais été faite auparavant. Á notre connaissance, bien que très prometteurs, en vue d'améliorer les résultats et l'efficacité de notre approche, nous proposons, dans le chapitre suivant, une variante de l'approche IGIT, en introduisant la notion de liste d'opérations restreinte et de procédure d'insertion aléatoire.

Chapitre 6

Approche IGIRT pour la résolution du FJSP avec temps de transport

Sommaire

6.1 Introduction

Le but est de montrer l'intérêt d'intégrer le concept de liste restreinte comportant les opérations candidates à l'insertion. Nous présentons dans ce chapitre notre seconde approche pour la résolution du FJSP avec temps de transport. Cette approche nommée Iterated Greedy Insertion Randomized Technique est une variante de l'approche proposée "IGIT" décrite dans le chapitre précédent. Nous commençons par présenter l'algorithme IGIRT en détail et sa méthodologie de fonctionnement. Ensuite, nous enchainons avec des expérimentations effectuées sur le Benchmark [151] et comparons les résultats obtenus du C_{max} avec l'approche IGIT, le MILP et les champs potentiels. Puis, nous continuons avec une analyse et une schématisation de nos résultats à l'aide du diagramme de Gantt, afin de mieux illustrer la manière dont les deux sous-problèmes d'affectation et de séquencement ont été résolus. Enfin, nous clôturons ce chapitre, avec des tests de performance sur des instances de grandes tailles que nous comparons avec l'approche IGIT afin de montrer son efficacité et les gains obtenus.

6.2 Approche proposée : Iterated Greedy Insertion Randomized Technique (IGIRT)

L'approche proposée Iterated Greedy Insertion Randomized Technique (IGIRT) [8] est une variante de l'approche précédemment présentée IGIT. L'approche IGIRT est considérée comme une hybridation de l'approche IGIT où une procédure gloutonne randomisée a été intégrée.
L'heuristique gloutonne randomisée peut être vu comme étant une étape de la métaheurisque GRASP. Cette même méthode, étant elle aussi composée de deux phases : *une procédure gloutonne randomisée (Greedy Randomized Procedure (GRP))* et *une recherche locale (Local Search (LS))*.
Dans la technique gloutonne randomisée utilisée, une solution réalisable de bonne qualité est construite en ajoutant progressivement un élément à la fois. En commençant à partir d'une solution partielle vide, à laquelle nous insérons des éléments. Les éléments non insérés sont évalués à chaque itération à l'aide d'une fonction fitness, puis une liste restreinte de candidats (RCL) composée des opérations potentielles à l'affectation aux machines est créée. Ensuite, une opération choisie aléatoirement à partir de la RCL est intégrée à la solution.

L'évaluation des éléments à l'aide de la fonction fitness conduit à la création de la liste restreinte de candidats (RCL) formée à partir des meilleurs éléments, c'est-à-dire dont l'incorporation à la solution partielle actuelle entraîne les plus petits coûts incrémentaux (cela représente l'aspect glouton de l'algorithme).
L'élément à intégrer dans la solution partielle est choisi aléatoirement parmi ceux de la RCL. Une fois l'élément sélectionné intégré à la solution partielle, la liste RCL est mise à jour et les coûts différentiels sont réévalués.
Ce type de méthode de construction aléatoire gloutonne utilisant le principe de la RCL peut

également être rencontré dans la littérature sous le nom d'heuristique semi-gloutonne [173]. D'ailleurs, d'après Glover [174], l'utilisation de stratégies, pour créer des listes restreintes de candidats, est essentielle pour limiter le nombre de solutions examinées dans une itération donnée dans des situations où les options sont très importantes ou si les solutions sont coûteuses en évaluation. L'efficacité de la stratégie de la RCL devrait être évaluée en fonction de la qualité de la meilleure solution trouvée dans un certain temps de calcul.
De plus, cette stratégie peut être améliorée par l'utilisation de structures de mémoire pour des mises à jour efficaces des évaluations des parties de solution insérée à chaque itération.

6.3 Algorithme complet (IGIRT)

La stratégie aléatoire peut générer un écart important en termes de solutions obtenues. Pour améliorer la robustesse, l'approche IGIRT utilise une stratégie hybride consistant à combiner deux stratégies principales : une approche aléatoire (à travers la notion du RCL) et une approche gloutonne (hybridation avec l'IGIT). Un compromis est établi entre l'utilisation de solutions initiales aléatoires et gloutonnes en termes de qualité des solutions et de temps de calcul. La meilleure réponse à ce compromis dépend principalement de l'efficacité des algorithmes aléatoires et gloutons. Générer une solution initiale aléatoire est une opération rapide, mais peut nécessiter un nombre important d'itérations pour converger. Pour accélérer la recherche, une heuristique gloutonne est utilisée. En effet, comme mentionné dans le chapitre précédent, dans la plupart des cas, les algorithmes gloutons ont une complexité polynomiale réduite. Ainsi, l'utilisation d'heuristiques gloutonnes conduit souvent à des optima locaux de meilleure qualité. Par conséquent, l'approche IGIRT nécessitera, moins d'itérations pour converger vers un optimum local.
L'algorithme 4 présente l'approche IGIRT et se déroule de la manière suivante :

Algorithm 4 Iterated Greedy Insertion Randomized Technique

1: i=0 ; current iteration
2: $Cmax\_min \leftarrow BN$; BN est un grand nombre
3: $Best\_Solution \leftarrow null$;
4: **while** (i < NI) **do**
5: $SO \leftarrow O_{ij}$; ensemble de toutes les opérations
6: $Solution \leftarrow \varnothing$; un ensemble vide
7: Calculer le fitness des opérations de SO en utilisant la règle "R" ;
8: Construire la RCL à partir des N premières opérations avec le meilleur fitness ;
9: Supprimer les N opérations du SO ;
10: Sélectionner aléatoirement une opération o à partir de RCL ;
11: Appliquer Algorithme 2 pour obtenir la meilleure machine et la meilleur position pour l'opération o ;
12: $Solution \leftarrow SolutionU(o, best\_machine, best\_position)$;
13: Supprimer l'opération du RCL ;
14: **if** SO is not empty **then**
15: **if** R est dynamique **then**
16: Recalculer le fitness pour les opérations restantes de SO en utilisant R ;
17: **end if**
18: Ajouter l'opération avec le meilleur fitness à la RCL ;
19: Supprimer l'opération o du SO ;
20: **end if**
21: **if** RCL n'est pas vide **then**
22: Aller à ligne 10
23: **end if**
24: Calculer le C_{max} de la Solution ;
25: **if** $Cmax < Cmax\_min$ **then**
26: $Cmax\_min \leftarrow Cmax$;
27: $Best\_Solution \leftarrow Solution$;
28: **end if**
29: i = i+1 ;
30: **end while**
31: **Return** *Best_Solution*

L'algorithme de l'IGIRT (algorithme 4) passe par différentes étapes définies comme suit : Tout d'abord, nous disposons d'un ensemble d'opérations SO qui comporte toutes les opérations O_{ij} nécessaire à l'élaboration d'un ordre de fabrication.
Rappelons que O_{ij} signifie l'opération i du job j et $i,j \in \mathbb{N}$.
Ensuite, le fitness de chaque opération est calculé en fonction de la règle "R". Cette règle "R" se base sur l'analyse sous contraintes basées sur la gestion des conflits entre jobs pour l'utilisation d'une ressource ainsi que de règles de propagation temporelle.
Une fois le fitness des opérations calculé nous sélections les N opérations avec le meilleur fitness.
Dans l'heuristique constructive, à chaque itération, les éléments pouvant être inclus dans la solution partielle sont classés à l'aide de l'heuristique gloutonne. À partir de cette liste, un

sous-ensemble représentant la liste des candidats restreints (RCL) est généré. la RCL représente l'aspect probabiliste de l'IGIRT. Le critère de restriction de cette liste est basé sur la cardinalité. La RCL est constituée des n meilleurs éléments en termes de coût incrémentiel, le paramètre n représentant le nombre maximal d'éléments de la liste.

Après plusieurs tests, nous avons déduit que le nombre adéquat d'opérations formant le RCL est de 3.
En effet, pour une bonne implémentation et une résolution optimale des sous-problèmes d'affectation et de séquencement le nombre maximal d'opérations autorisés dans la RCL est de 3, dépassé ce chiffre, nous observons une détérioration des performances de l'approche IGIRT. Une fois la RCL construite une opérations est choisie aléatoirement parmi celles formant la liste restreinte.
Puis, l'algorithme IGIT est appliqué sur cette même opération afin d'être affecter à la machine m offrant le service demandé. Dans le cas de l'indisponibilité de cette machine, l'algorithme IGIRT se réfère aux tableaux 4.1 et 4.3 pour affecter l'opération à la machine nécessitant le moins de temps de transport et offrant le même type de service.
Après l'application de l'IGIT sur l'opération sélectionnée o un triplet d'informations nommé *SolutionU* (o, best_machine, best_position) est construit. La construction de la variable *SolutionU* résout les deux sous-problèmes du FJSP simultanément, à savoir, l'affectation et le séquencement. En effet, la variable best machine m /m $\in \mathbb{N}$ et m=[1,5] répond au sous-problème de l'affectation. Et la variable p /p $\in \mathbb{N}$ répond au sous-problème de séquencement en insérant l'opération sur la meilleure position de la machine.
Cette portion de solution *SolutionU* est ajoutée à la solution en construction *Solution*. La variable *Solution* est représentée par un vecteur : *Solution*=$SolutionU_1$,...$SolutionU_n$.
Les $SolutionU_n$ sont ajoutées jusqu'à ce que *SO* soit vide et formant ainsi une solution complète qui représente le C_{max}.
A chaque fin de cycle de construction de solution complète (C_{max}), le résultat obtenu lors de l'itération courante i est comparé au résultat final obtenu lors de l'itération précédente *i-1* .
Cette dernière étape détermine si le critère d'acceptation a été rempli ou pas. En effet, si la solution obtenue améliore l'ancienne elle sera maintenue et stockée dans la variable $Cmax_{min}$.
La condition d'arrêt de l'algorithme correspond au nombre maximal d'itérations autorisé. En effet, l'algorithme IGIRT se répète jusqu'à ce que le nombre maximal d'itérations défini au préalable est atteint et retourne la meilleure solution obtenue.

L'organigramme présenté dans la figure 6.1 représente le principe général de fonctionnement de l'approche IGIRT.

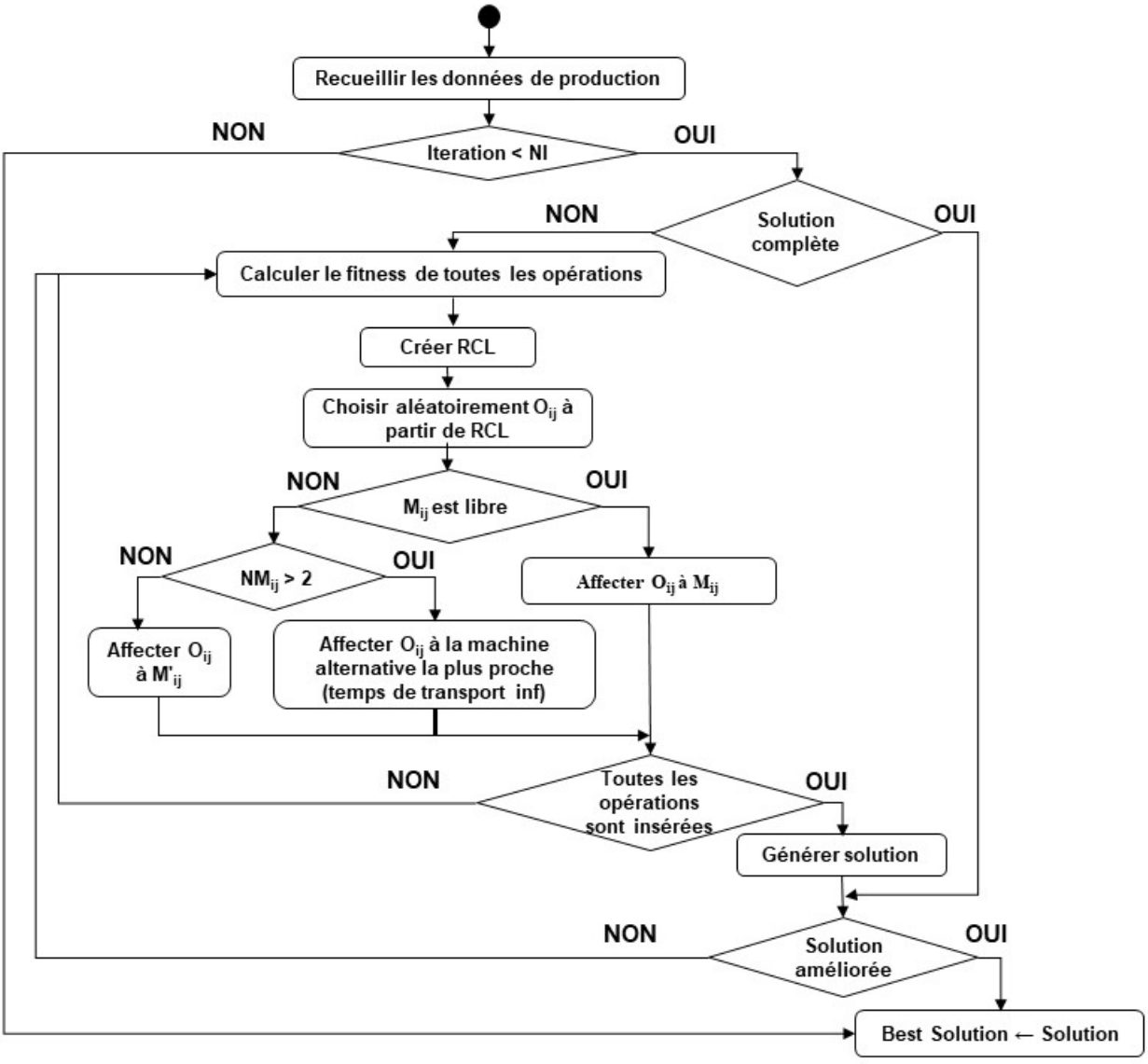

FIGURE 6.1 – Organigramme IGIRT

6.4 Expérimentations

Tout comme pour l'approche IGIT, l'approche IGIRT a été codée en Java et les expérimentations faites sur un processeur Intel Core i7-4510 avec 2.6 GHz et 8 Go de RAM.
Le tableau 6.1 compare les résultats du C_{max} obtenus avec l'approche IGIRT et ceux obtenus grâce aux méthodes du MILP, des champs potentiels (PF) et de l'IGIT.
Nous reprenons dans ce tableau les résultats précédemment indiqués dans le tableau 5.1 pour les comparer avec ceux de l'IGIRT afin de montrer l'efficacité de cette approche par rapport aux autres approches testées sur les instances du Benchmark de l'AIP-PRIMECA.
Nous indiquons dans le tableau, le nom de l'instance, la méthode utilisée, la solution obtenue, et le gap de l'IGIRT par rapport aux autres méthodes.

TABLE 6.1 – Comparaison du Makespan

Product	**IGIRT**	**IGIT**	**MILP**	**PF**
A-I-P	216	216	216	-
2 x A-I-P	**306**	346	326	-
3 x A-I-P	**406**	522	482	-
B-E-L-T-A-I-P	**369**	419	570	448

Comme indiqué dans le tableau 4.2, les opérations de chargement et de déchargement ne peuvent être effectuées que par M1 et l'opération d'inspection uniquement par la ressource M5. Étant donné que chaque job commence par une opération de chargement et que la ressource M1 est la seule dans l'atelier à fournir ce type de service, les opérations incluses dans la liste RCL correspondent à celles traitées en premier sur la ressource M1.

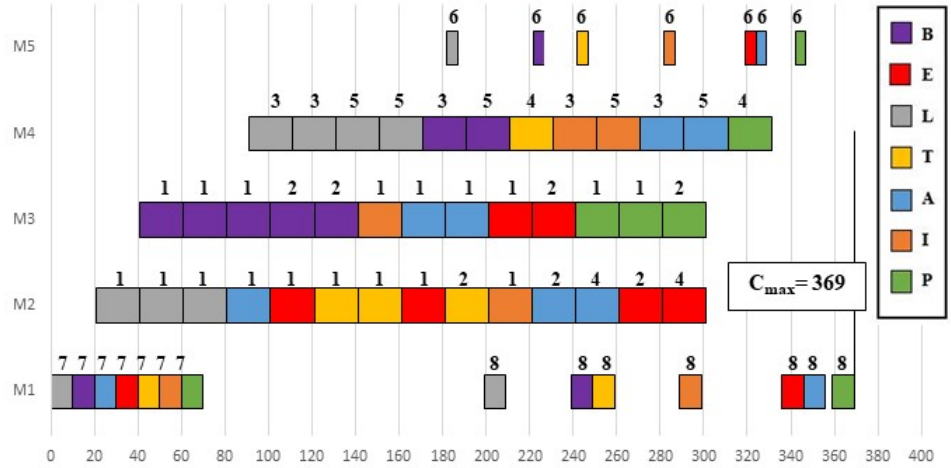

FIGURE 6.2 – Diagramme de Gantt pour l'ordre de fabrication B-E-L-T-A-I-P avec l'approche IGIRT

La figure 6.2 est un diagramme de Gantt représentant le meilleur ordonnancement obtenu pour l'ordre de fabrication B-E-L-T-A-I-P en utilisant l'approche IGIRT.
Cette figure indique une totale exploitation des ressources d'assemblage M2, M3 et M4. En effet, aucun temps d'arrêt n'est observé pour ces trois machines pendant leurs périodes d'activité. Cette exploitation complète permet un gain important dans le temps de production total comparée aux deux autres approches (MILP et PF) testées sur ce type d'instance.

Bien qu'une légère surcharge peut être observée sur les machines M2 et M3 avec respectivement 14 et 13 opérations effectuées, par rapport à la machine M4 qui traite 12 opérations, la répartition des opérations reste relativement bien équilibrée sur les trois *machines d'assemblage*. Cette légère différence d'opérations traitées est dûe au fait que l'opération "r_mounting" ne peut être réalisée que par les machines M2 et M3 (notion de flexibilité partielle). Sachant, que l'opération ("r_mounting") représente plus de **11%** des opérations effectuées, c'est ainsi que la notion de RCL intervient afin de distribuer équitablement ce type d'opération sur les ressources alternatives les plus proches ce qui permet de gérer efficacement cette opération, équilibrer la charge de travail, éviter les interruptions et les files d'attente interminable au niveau d'une seule ressource. Cette bonne gestion de répartition des tâches effectuée par la RCL est entre autre la raison principale du gain important de temps de production pour ce type de scénario.

Nous pouvons également remarquer que l'insertion des opérations est beaucoup plus équilibrée avec les tâches "L" (gris) et "B" (violet).

En effet, une fois l'insertion des opérations de ces jobs effectuée, les machines exécutent une grande partie des opérations de ces deux tâches avant de commencer l'exécution d'une autre, ceci est dû aux premières opérations insérées. La figure 1 (a) indique que les ressources d'assemblage (M2, M3 et M4) sont exploitées de manière optimale, sans temps d'attente entre les opérations. Ainsi, une fois les trois machines démarrées, elles ne s'arrêtent qu'une fois la totalité des opérations effectuées. Cette utilisation optimale des machines s'explique par la notion de RCL, qui regroupe trois opérations candidates correspondant au nombre de machines d'assemblage. Le C_{max} atteint en utilisant l'IGIRT sur l'ordre de produit B-E-L-T-A-I-P est égal à 369 sec. Ainsi, l'IGIRT améliore significativement les résultats obtenus avec les autres approches (Tableau 6.1) et prouve son efficacité.
Bien que l'IGIT ait démontré son efficacité par rapport à la méthode MILP sur ce type d'instance avec un gap de **3,8%**, cette amélioration est insignifiante par rapport au gain considérable obtenu avec la méthode IGIRT avec un gap de **11,94%** par rapport à l'IGIT, **17,64%** par rapport à l'approche PF et **35,26%** par rapport au MILP. En effet, la notion de RCL permet d'assigner efficacement les opérations aux machines.

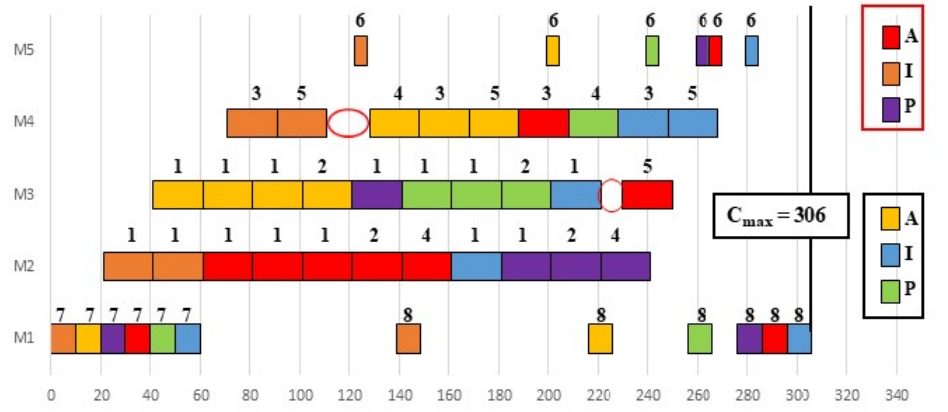

FIGURE 6.3 – Diagramme de Gantt pour l'ordre de fabrication 2 x A-I-P avec l'approche IGIRT

La figure 6.3 est un diagramme de Gantt représentant le meilleur ordonnancement obtenu pour l'ordre de fabrication 2 x A-I-P en utilisant l'approche IGIRT.
Le diagramme de Gantt pour 2 x A-I-P tout comme celui de l'ordre de fabrication B-E-L-T-A-I-P (Figure6.2) indique une utilisation légèrement supérieure de la machine M2 par rapport aux autres machines.
Cependant, contrairement au scénario B-E-L-T-A-I-P où l'exploitation des machines est totale (aucun temps d'attente entre le traitement de deux opérations sur une machine), en utilisant l'approche IGIRT sur le scénario 2 x A-I-P certains temps intermédiaires subsistent. En effet, comme indiqué dans la figure 6.3 nous pouvons observer un léger temps mort sur la machine M4 avant qu'elle ne commence à prendre en charge les opérations du second job "A" (ligne jaune). Le même phénomène peut être observé sur la machine M3, étant donné le léger temps d'inactivité existant avant de traiter la dernière opération de montage du 1er job "A" (ligne rouge).
De plus, le transport de ces deux emplois (1er et 2e A) d'une machine à une autre a conduit à une exploitation non-complète de M3 et M4. Ce phénomène est dû au fait que la machine M4 soit la seule à fournir le service "l_montage" et qu'elle représente l'une des opération formant le job "A". Notons aussi que la machine M2 soit la seule machine à être utilisée de manière optimale sans aucun temps d'inactivité.
Bien que l'approche IGIRT utilisant la technique d'insertion aléatoire cherche à réduire considérablement les opérations de transport en les évitant ou en cherchant la ressource alternative la plus proche, nous notons que 2 x A-I-P est l'ordre de production avec le moins bon gain. dispersion de l'emploi (bleu). En effet, c'est le seul job exploitant toutes les ressources, ce qui nécessite donc plusieurs opérations de transport lors du processus de fabrication.
Par rapport à l'ordre de production B-E-L-T-A-I-P où il n'y a pas de temps d'attente entre les trois machines d'assemblage (M2, M3 et M4), l'approche IGIRT reste efficace par rapport aux autres approches utilisées sur ce type d'instance car avec un C_{max} de **306** sec, les gains

obtenus sont relativement bons, avec un gap de **(11,57%)** comparés à ceux obtenus avec l'approche l'IGIT et **(6,14%)** à la méthode MILP.

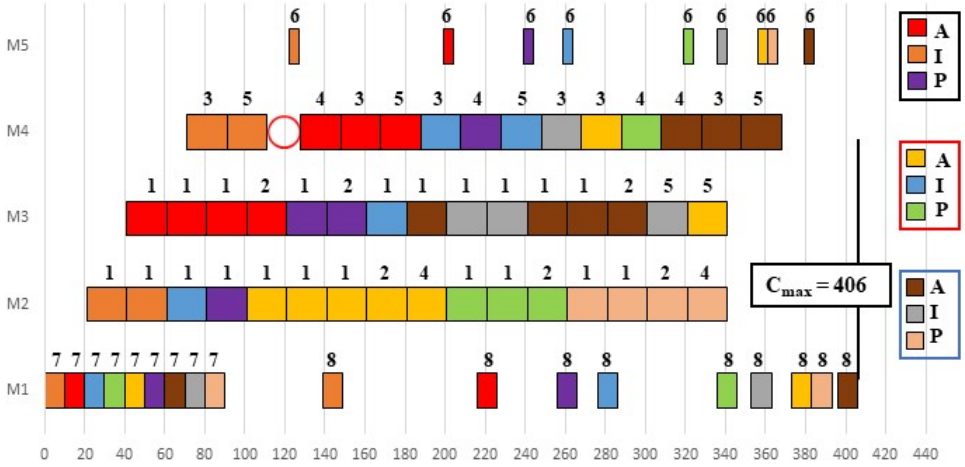

FIGURE 6.4 – Diagramme de Gantt pour l'ordre de fabrication 3 x A-I-P

La Figure 6.4 est un diagramme de Gantt représentant le meilleur ordonnancement obtenu pour l'ordre de fabrication 3 x A-I-P en utilisant l'approche IGIRT.
Nous pouvons remarquer que le même événement avec le même job "A" (ligne rouge) et les mêmes machines M3 et M4 se produit avec le scénario 2 x A-I-P (Figure 6.3).
Néanmoins, comparé au scénario 2 x A-I-P, les temps d'inactivités dûs aux opérations de transport d'un job ont eu lieu qu'une seule fois, ajouté à cela l'activité non interrompue des machines M2 et M3, a conduit à de meilleures performances comparé au scénario 2 x A-I-P avec des gains obtenus importants. En effet le gap observé pour l'instance 3 x A-I-P est de **22,23%** par rapport à l'IGIT et **15,77%** par rapport au MILP.

6.5 Analyse de l'IGIRT sur les grandes instances

Le tableau 6.2 représente l'évaluation de l'approche IGIRT sur les instances de grandes tailles : 10 x A-I-P, 10 x B-E-L-T et 5 x B-E-L-T-A-I-P.
Ces expérimentations nous permettent, comme pour l'approche IGIT, d'indiquer la capacité de l'approche IGIRT à gérer les instances de grandes tailles contrairement aux méthodes MILP et PF.

TABLE 6.2 – Performances des approches pour les grandes instances : 10 x A-I-P, 10 x B-E-L-T et 5 x B-E-L-T-A-I-P.

Nombre d'itérations	**Produit**	**Nombre de jobs**	**Nombre d'opérations**	**Cmax IGIRT**	**CPU time (sec)**
10.000	10 x A-I-P	30	240	1136	18.948
	10 x B-E-L-T	40	360	1799	38.042
	5 x B-E-L-T-A-I-P	35	600	1467	27.424
50.000	10 x A-I-P	30	240	1125	78.76
	10 x B-E-L-T	40	360	1791	187.263
	5 x B-E-L-T-A-I-P	35	600	1466	225.376
100.000	10 x A-I-P	30	240	1119	165.21
	10 x B-E-L-T	40	360	1786	451.294
	5 x B-E-L-T-A-I-P	35	600	1456	200.105
500.000	10 x A-I-P	30	240	1116	949.689
	10 x B-E-L-T	40	360	1778	1988.558
	5 x B-E-L-T-A-I-P	35	600	1449	1264.696

La résolution du problème d'ordonnancement du job shop flexible avec contrainte de temps de transport est un problème difficile et sa résolution avec une méthode exacte peut selon la taille du problème s'avérer impossible. Compte tenu aussi de la grande taille des instances (nombre élevé de job implique un nombre élevé d'opérations), il faut environ dix minutes à l'IGIT pour trouver une solution optimale à l'instance la plus grande du jeu de test, à savoir 5 x B-E-L-T-A-I-P.
Certains paramètres restent fixes, tels que le nombre de machines utilisées (fixé à cinq), les opérations effectuées par chacune d'entre elles (Tableau 4.2) ainsi que la distance entre les machines (Tableau 4.3). Par conséquent, pour trouver une solution optimale, plusieurs expérimentations avec un nombre différent d'itérations sont effectuées.
Tout comme pour l'approche IGIT, l'approche IGIRT a été testée avec quatre nombres d'itérations différents du cycle de destruction-construction, pour déterminer le nombre de cycles nécessaires à la convergence de l'approche IGIRT vers une solution optimale.
L'approche IGIRT a été testée sur les mêmes jeux d'itérations : (i = 10.000, i = 50.000, i = 100.000 et i = 500.000) afin de pouvoir comparer les résultats et déterminer laquelle des deux méthodes proposées est la plus optimale.

La Figure 6.5 illustre le temps CPU nécessaire à la méthode IGIRT pour obtenir une solution optimale (C_{max}) pour les différents scénarios : 10 x A-I-P, 10 x B-E-L-T-A-I-P et 5 x B-E-L-T-A-I-P.
L'ordonnancement se produit de manière off-line, ces expérimentations nous permettent donc de déterminer deux des paramètres les plus important au bon déroulement des méthodes proposées et permettant d'atteindre les résultats optimaux :

1. le nombre d'itérations (phase de construction et déconstruction),
2. le temps CPU requis pour la convergence des solutions.

En effet, plus le nombre d'itérations est élevé, plus le temps de simulation augmente.

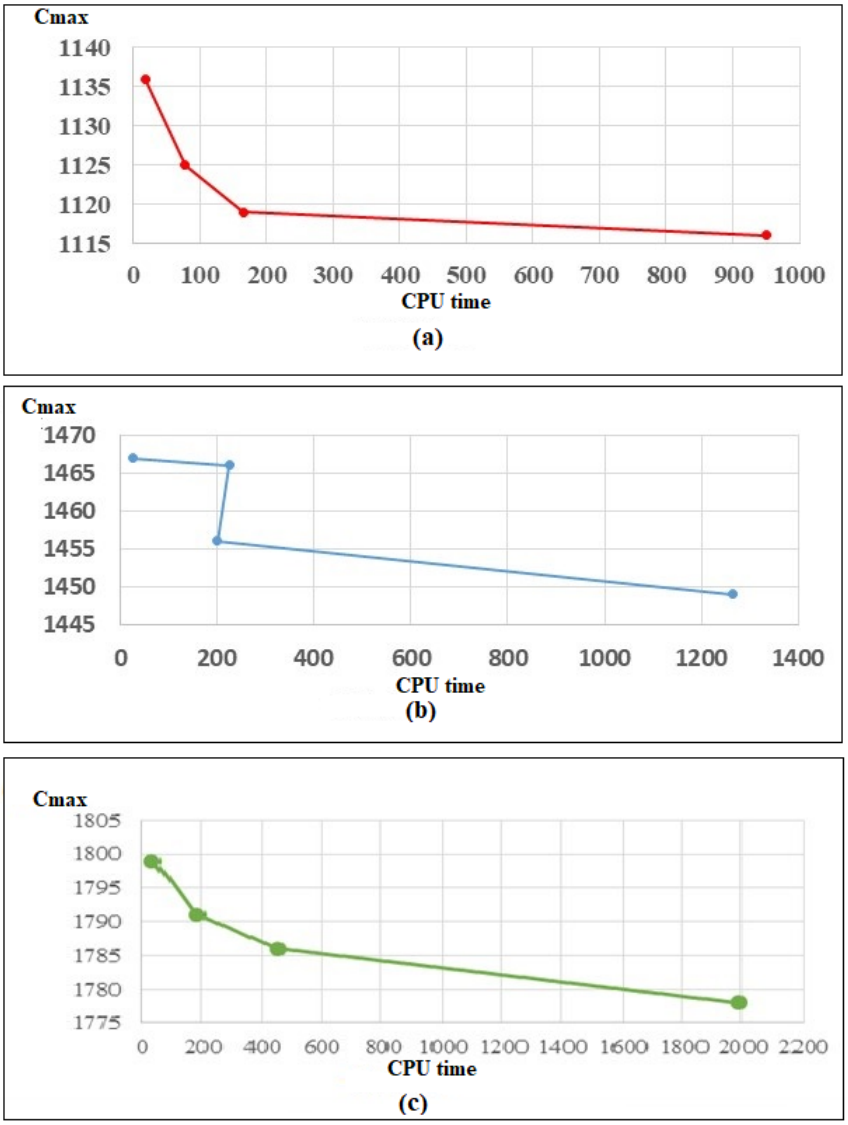

FIGURE 6.5 – Simulations de l'heuristique IGIRT pour 10 x A-I-P, (b) 5 x B-E-L-T-A-I-P et (c) 10 x B-E-L-T

Dans la Figure 6.5 (section "a"), nous remarquons que l'approche IGIRT nécessite environ 200 sec pour se stabiliser et commencer à converger, donc c'est seulement après 100 000 itérations que l'heuristique commence à converger vers une solution optimale pour les 10 x

A-I-P. Ce qui est le même cas pour le scénario 5 x B-E-L-T-A-I-P, qui nécessite le même temps de calcul. Cependant, la convergence est moins harmonieuse que dans le scénario précédent.
Comme indiqué dans la section "b" de la Figure 6.5, le C_{max} diminue considérablement après 200 sec d'itérations avant de converger progressivement vers une solution optimale.
Quant à l'ordre de production de 10 x B-E-L-T (Figure 6.5 (section "c")), sa convergence vers un optimum se fait de manière plus progressivement comparé aux deux ordres de production précédents, mais l'IGIRT nécessite plus de temps pour commencer à converger. En effet, les autres instances sur lesquels l'approche fut testée nécessite un temps moyen avoisinant les 200 secondes pour converger, alors que pour l'instance 10 x B-E-L-T nécessite plus de 400 secondes. Ce phénomène peut être expliqué par le nombre de tâches traitées dans l'atelier, car plus les travaux sont différents, plus le temps de transport est important, donc la procédure prend plus de temps et d'itérations pour se stabiliser.
Nous pouvons remarquer dans les sections "a" et "b" de la Figure 6.5 que le comportement de l'approche IGIRT avec les instances de grandes tailles 10 x A-I-P et 10 x B-E-L-T est assez similaire. Bien que l'approche IGIRT nécessite plus de temps CPU pour atteindre un optimum global avec l'ordre de fabrication 10 x B-E-L-T, la convergence se fait pratiquement de la même manière pour les deux scénarios à partir de 200 sec d'itération contrairement à l'ordre de fabrication 5 x B-E-L-T-A-I-P.
Cette convergence plus harmonieuse peut être expliquée par le nombre de types de jobs à gérer. En effet, avec les instances n x A-I-P et n x B-E-L-T (n $\in \mathbb{N}$) il existe respectivement 3 et 4 types de jobs différents contrairement à l'instance m x B-E-L-T-A-I-P où il existe sept types de jobs différents (m $\in \mathbb{N}$).

6.6 Comparaison entre IGIT et IGIRT sur les grandes instances

Le tableau 6.3 compare les *makespans* (C_{max}) obtenus avec les deux approches proposées, IGIT et IGIRT. La première colonne indique l'instance de grande taille sur laquelle le test de performance a été effectué. Les colonnes deux et trois présentent les C_{max} obtenus avec nos deux approches. la quatrième colonne indique l'écart entre les deux solutions (gap) et les deux dernières colonnes présentent le temps CPU nécessaire à chacune des deux approches proposées pour l'obtention de ces solutions.

TABLE 6.3 – IGIT VS IGIRT

Produit	C_{max} **IGIRT**	C_{max} **IGIT**	**gap**	**CPU time IGIRT**	**CPU time IGIT**
10 x A-I-P	1116	1653	**32.49%**	949.689	508.731
10 x B-E-L-T	1778	2670	**33.41%**	1988.558	1127.886
5 x B-E-L-T-A-I-P	1449	2171	**33.26%**	1264.696	595.277

L'étude comparative effectuée dans le tableau 6.3 indique de manière implicite les forces et les faiblesses de l'intégration de la RCL. En effet, les gains apportés par rapport à l'approche "IGIT" sont considérables. Cependant, bien que l'heuristique IGIT soit moins efficace que l'IGIRT, au vu de la différence de C_{max} entre ces deux méthodes, une faiblesse subsiste dans le temps de calcul nécessaire (temps CPU) pour obtenir ces solutions. Nous remarquons, en effet que l'approche IGIRT prend beaucoup plus de temps à obtenir une bonne solution que l'approche IGIT et cela malgré le même nombre d'itérations. Cette différence notable est facilement explicable par la construction de la RCL, car après chaque affectation d'une opération à une machine, les fonctions fitness des autres opérations sont réévaluées afin d'intégrer la liste des opérations candidates à l'insertion. Cette étape de sélection et d'intégration à chaque itération de l'IGIRT consomme beaucoup de temps CPU.
Dans un ordonnancement off-line cela n'est pas très problématique car la principale mesure de performance reste le *makespan*. Cependant, dans un ordonnancement dynamique sujet à des imprévus internes et externes, le temps nécessaire au calcul et au recalcul d'une solution prend plus d'ampleur.
Ainsi, nous pouvons déduire d'une manière générale que malgré les performances nettement supérieures de l'IGIRT sur l'IGIT, cette dernière prend moins de temps pour converger vers un optimum et consomme, par conséquent, moins de temps à l'obtention d'une solution acceptable.

6.7 Conclusion

Dans ce chapitre, une nouvelle variante de l'approche heuristique IGIT a été proposée pour le problème du job shop flexible avec contrainte de temps de transport.
Cette variante, nommée IGIRT, intègre une technique d'insertion randomisée basée sur une liste restreinte de candidats potentiels à l'insertion (RCL).
L'approche IGIRT a été comparée avec les approches précédemment testées sur ce jeu d'instances et validée avec succès.
Les expérimentations indiquent deux points importants. En premier lieu, l'approche IGIRT est plus performante que les méthodes MILP, PF et IGIT et de nouvelles meilleures solutions ont été trouvées pour les instances 2 x A-I-P, 3 x A-I-P et B-E-L-T-A-I-P. En second lieu, l'approche IGIRT permet contrairement aux méthodes MILP et PF de traiter des instances

de grandes tailles dans des temps raisonnables.
Nous avons expliqué le lien entre l'efficacité de l'approche IGIRT et l'ajout de la notion de RCL qui sélectionne les opérations les plus aptes à l'affectation aux machines et permet de séquencer de manière optimale ces opérations sur les machines en réduisant au maximum les temps d'inactivité.
La comparaison des deux approches proposées, en fin de chapitre, nous a permis de faire des analyses pouvant ouvrir la porte à des futures perspectives intéressantes. Car bien que l'IGIRT soit plus performante que l'approche IGIT, cette dernière est plus réactive, car l'approche IGIRT prend un temps considérable à converger ce qui peut être considérablement désavantageux dans un contexte de production réel pour cela nous envisageons dans nos futurs travaux de trouver un compromis entre la réactivité de l'approche IGIT et l'efficacité et l'efficience de l'IGIRT.

Conclusion générale et Perspectives

L'ordonnancement dans les systèmes de production flexibles est un problème d'optimisation NP-difficile et représente une source importante de travaux de recherches. Parmi les ateliers flexibles, le job shop flexible est considéré comme l'un des ateliers les plus étudié par la communauté de l'optimisation combinatoire et la recherche opérationnelle. Le problème d'ordonnancement dans le job shop flexible (FJSP) est une extension du problème d'ordonnancement dans le job shop classique où une opération peut être effectuée par un ensemble de ressources (flexibilité partielle) ou la totalité des ressources (flexibilité totale) présente dans l'atelier de production.

Bien qu'étant une contrainte d'une grande importance compte tenu des répercussions directes sur le temps de production totale, la contrainte concernant les temps de transport entre les ressources est très souvent négligée. Par conséquent, nous nous intéressons dans notre thèse à ce cas particulier du job shop flexible.

Pour cela, nous avons présenté dans notre thèse deux approches heuristiques pour la résolution du problème d'ordonnancement dans les systèmes de production de type job shop flexible en prenant en compte la contrainte des temps de transport entre les ressources.
La première approche, nommé **I**terated **G**reedy **I**nsertion **T**echnique (IGIT), est le résultat de l'hybridation de deux stratégies basées sur une technique d'insertion et une heuristique gloutonne itérée où chaque itération est composée de phases de construction et déconstruction (partielle ou totale) de la solution jusqu'à ce qu'elle ne peut plus être améliorée. Quant à la seconde approche, nommé **I**terated **G**reedy **I**nsertion **R**andomized **T**echnique (IGIRT), il s'agit d'une amélioration de la première approche. Cette amélioration consiste en l'ajout d'une technique d'insertion aléatoire à partir d'une liste restreintes d'opérations candidates nommé RCL. Cette liste comporte un nombre restreint d'opérations ayant le meilleur fitness, puis l'une d'entre elles est sélectionnée aléatoirement et affectée à la ressource adéquate permettant ainsi de réduire considérablement le temps total de production.
Les performances des deux approches IGIT et IGIRT ont été évaluées sur les instances du

Benchmark de la cellule flexible de l'AIP-PRIMECA de l'université de Valenciennes et du Hainaut Cambrésis [151].

Les résultats encourageants obtenus sur des instances de tailles réduites montrent l'efficacité et l'efficience des deux approches proposées. Ces résultats nous ont encouragé à passer à l'échelle et traiter des instances de plus grandes tailles (pouvant inclure jusqu'à 600 opérations). L'évaluation des performances des deux approches sur ces types d'instances nous a permis de déduire une plus grande efficacité de la part de l'IGIRT contrairement à l'IGIT avec un gain avoisinant les 34Plusieurs perspectives de recherches intéressantes peuvent être étudiées à court, moyen et long termes.

Perspectives

- **Perspectives à court terme :**
  1. Bien que les Benchmarks standards dédiés aux FMS et FJSP de Brandimarte [95] et Kacem [101] ne prennent pas en compte les temps de transport, il est intéressant de tester nos deux approches sur ces instances afin de consolider l'efficacité de l'IGIT et IGIRT.
  2. Intégrer une méthode de recherche locale à chacune des méthodes IGIT et IGIRT afin de développer une méta-heuristique pouvant traiter des problèmes plus génériques.
- **Perspectives à moyen terme :**
  1. Repenser les approches proposées pour intégrer la notion de multi-objectif dans le but d'optimiser d'autres objectifs que le C_{max}.
- **Perspectives à long terme :**
  1. Proposer une architecture générique (applicable à différents cas d'études) basée sur l'approche IGIRT pour un ordonnancement optimisé et réactif du système capable de gérer les perturbations internes (pannes, etc.) et externes (évolution de la demande du marché, pénuries d'approvisionnement, etc.), tels que les architectures ORCA-FMS [171] et Pollux [175].
  2. Prendre en compte d'autres activités de production simultanément avec la fonction d'ordonnancement, tels que la maintenance [176], la planification des processus [177] ou le contrôle des lots et des stocks [178].
  3. Proposer une hyper-heuristique en utilisant les deux heuristiques IGIT et IGIRT.

Annexes

Annexe A

Les principales métaheuristiques

Une métaheuristique est un algorithme d'optimisation qui vise à résoudre des problèmes complexes pour lesquels il n'existe pas de méthode classique efficace. Les métaheuristiques recherchent la meilleure solution d'un problème en manipulant une ou plusieurs solutions de telle sorte que l'on passe, après un certain nombre d'itérations, d'une solution mauvaise à une solution optimale ou proche de l'optimale. Le principal problème est le choix de la métaheuristique en elle-même car aucune règle n'a été établie et on ne peut savoir à priori quelle métaheuristique correspond le mieux à la résolution d'un problème donné. Ainsi, afin de choisir, il est souvent recommandé de comparer plusieurs métaheuristiques pour un même problème. D'une manière générale, elles fonctionnent sur beaucoup de problèmes différents car elles ne nécessitent pas de connaissances particulières sur le problème qu'elles traitent. De plus, on peut rencontrer ces méta heuristiques pour des problèmes à variables discrètes, continues voire mixtes. Pour finir, notons que ces méta heuristiques peuvent être combinées avec d'autres méthodes, on parlera dans ce cas d'hybridation.

Bien qu'il existe un nombre important, et que de nouvelles métaheuristiques ne cessent d'être intégrées à la littérature d'années en années, les plus connues sont :

Les algorithmes génétiques (Genetic Algorithm : GA)

Initialement proposés par [179] et popularisés plus tard par [180], les algorithmes génétiques sont des algorithmes stochastiques itératifs fondés sur le principe Darwinien et des techniques dérivées de la génétique et des mécanismes de la sélection naturelle. Le but des AG est d'optimiser une fonction prédéfinie, appelée *fonction fitness*. Pour atteindre cet objectif, l'AG travaille sur une population d'individus, où chaque individu (ou chromosome) représente une solution possible du problème donné. Le chromosome est composé d'éléments,

appelés gènes, dont les valeurs sont appelées allèles. La qualité du codage du chromosome (binaire, réel, arbre,...) est l'un des principaux facteurs responsables du succès de l'AG [73].

L'ensemble des individus traités simultanément par l'AG constitue une population, l'algorithme itère et passe d'une génération à l'autre jusqu'à ce qu'un critère d'arrêt soit atteint. Durant chaque génération, un ensemble d'opérateurs (sélection, croisement et mutation) est appliqué à la population pour engendrer une nouvelle génération.
La figure A.1 illustre le fonctionnement général d'un algorithme génétique.

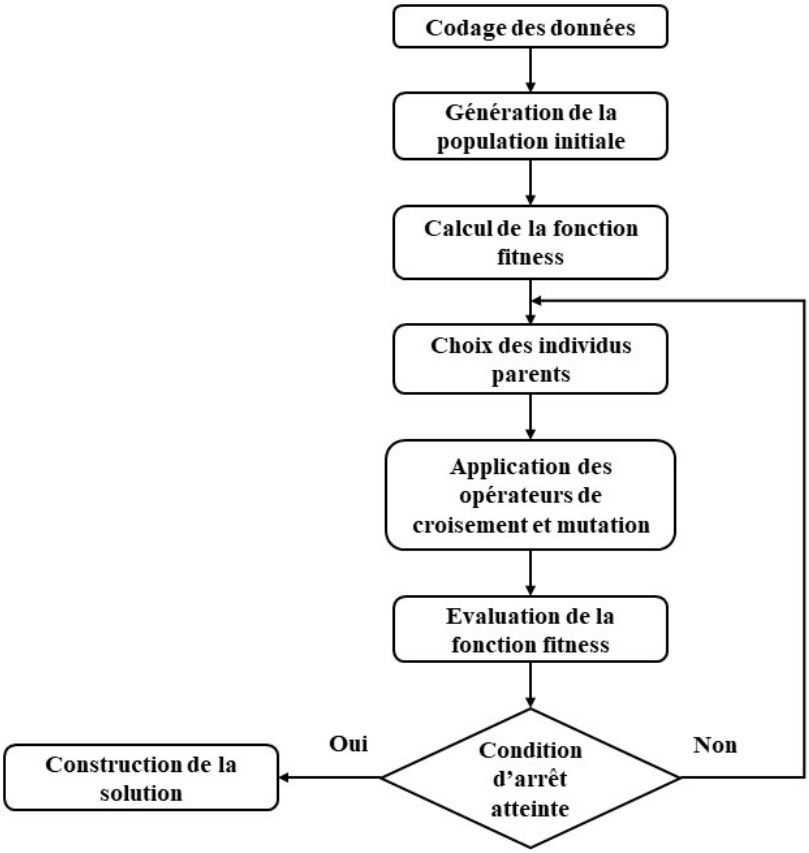

FIGURE A.1 – Fonctionnement général d'un algorithme génétique

Le recuit simulé (Simulated Annealing : SA)

L'algorithme du récuit simulé a été proposé pour la première fois par [181] et est inspiré d'un processus utilisé en métallurgie, où des cycles de refroidissement lent et de réchauffage (récuit) sont alternés dans le but d'obtenir un état solide stable du matériau avec énergie minimale.

En optimisation cette technique peut être transposée de la manière suivante : la fonction objective à minimiser représente l'énergie E du matériel et la température T représente un paramètre fictif servant à contrôler l'algorithme.
Partant d'un état donné du système, on obtient un autre état en le modifiant. Si la fonction objectif est améliorée, on dit alors qu'on a fait baisser l'énergie du système. Sinon, si la fonction objectif est dégradée, elle peut être acceptée avec une probabilité.

L'optimisation par colonies de fourmis (Ant Colony Optimization : ACO)

Les algorithmes de colonies de fourmis forment une classe de métaheuristiques initialement proposés dans les années 90 par [182] [183]. Le premier algorithme de ce type appelé *Ant System* a été conçu par [184].
Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de pistes observés dans les colonies de fourmis. L'objectif est qu'un ensemble de fourmis cherchent le chemin "optimal" en partant de leur nid jusqu'à la nourriture. Ainsi, chaque fourmi communique indirectement via des modifications dynamiques de leur environnement (dépôt de phéromone sur leur chemin) et construisent ainsi une solution à un problème, en s'appuyant sur leur expérience collective. Par conséquent, certains chemins moins optimaux s'effacent (évaporation des phéromones) alors que de meilleurs chemins se tracent, renforcés par les traces de phéromones laissées par les autres fourmis lors de leurs passages. Parmi les travaux les plus notables et le plus souvent évoqués dans la littérature ayant utilisé l'ACO pour la résolution du FJSSP nous pouvons citer [146] [185] [113] [186]

L'optimisation par essaims particulaires (Particle Swarm Optimization : PSO)

Proposée par Russel Eberhart ingénieur en électricité) et James Kennedy (socio psychologue) [187]. Le PSO est un algorithme inscrit dans la famille des algorithmes évolutionnaires et s'inspire grandement de l'observation des relations *grégaires* des oiseaux migrateurs, qui lors de leurs migrations doivent optimiser leurs déplacements en termes d'énergie dépensée, de distance, de temps, etc.
L'individu dans l'algorithme du PSO est nommé *particule* et l'ensemble des individus, *essaim*.

Pour être en mesure d'utiliser le PSO, il est indispensable de définir un espace de recherche (composé de particules) et une fonction objectif à optimiser. Ainsi, le fonctionnement du PSO consiste alors à déplacer ces particules de telle sorte qu'elles trouvent l'optimum. Pour cela les particules doivent disposer :

- Des coordonnées relatives à leurs positions, avec comme condition qu'elles soient comprises dans l'espace de définition.

- De la meilleure position rencontrée.
- De la meilleure position rencontrée par leur voisinage et le résultat de leur fonction objectif.
- De leur vitesse qui leur permet de se déplacer et de changer de position au fil des itérations.
- D'un voisinage, qui consiste en un sous-ensemble de particules qui interagit directement avec la particule (surtout celle possédant la meilleure position).

La recherche tabou (Taboo Search : TS)

Créée par Glover en 1986 [188] [189] et formalisée en 1989.
La recherche tabou consiste à garder en mémoire les dernières solutions visitées et à interdire le retour vers celles-ci pour un nombre fixé d'itérations. Le but étant de donner assez de temps à l'algorithme pour lui permettre de sortir d'un minimum local. Ainsi, cette méthode conserve à chaque étape une liste T de solutions « taboues », vers lesquelles il est interdit de se déplacer momentanément. L'espace nécessaire pour enregistrer un ensemble de solutions taboues peut s'avérer important en espace mémoire. Pour cette raison, il est parfois préférable d'interdire uniquement un ensemble de mouvements qui ramèneraient à une solution déjà visitée. Ces mouvements interdits sont appelés mouvements tabous [36]. Des versions plus raffinées de l'algorithme Tabou ont été présentées de manière détaillée dans la littérature. Il a été démontré par Benbouzid [190], que contrairement à la méthode du Recuit Simulé, il n'existe pas de résultat théorique garantissant la convergence de l'algorithme vers une solution optimale du problème traité. La raison principale de cet état de fait tient dans la nature même de la méthode. Celle-ci étant hautement adaptative et modulable, son analyse par les outils mathématiques traditionnels n'est pas concluante. Ce qui compte c'est de visiter au moins une solution optimale ou une solution de bonne qualité en cours de recherche [191].

La méthode GRASP

La méthode GRASP (Greedy Randomized Adaptive Search Procedure) est une métaheuristique utilisée dans les problèmes d'optimisation combinatoire. Elle fut introduite pour la première fois par Feo & Resende [192]. Cet algorithme consiste en des itérations constituées de constructions successives à partir d'une solution gloutonne aléatoire ainsi que des améliorations itératives subséquentes en utilisant une méthode de recherche locale.
Les solutions aléatoires gloutonnes sont générées en ajoutant des éléments à l'ensemble des solutions du problème à partir d'une liste d'éléments classés par une fonction gloutonne en fonction de la qualité de la solution atteinte. Pour obtenir une variabilité dans l'ensemble candidat des solutions gloutonnes, des éléments candidats bien classés sont souvent placés dans une liste restreinte de candidats (également appelée RCL), et choisis aléatoirement lors de la construction de la solution.

Annexe B

Description détaillée de la notation ($\alpha \mid \beta \mid \gamma$)

Le tableau B.1 explique de manière détaillée chacun des champs de la notation de [75] et les valeurs qu'ils peuvent prendre.

TABLE B.1 – Définition détaillée des champs de la notations ($\alpha \mid \beta \mid \gamma$)

Champ	**Composantes du champ**	**Description**	**Valeur**
α	α_1	Type d'atelier	**0** : atelier à machine unique / $p_{1j} = p_j$ **P :** atelier à machines parallèles identiques / $p_{ij} = p_j$ (i=1,...,m) **Q :** Atelier à machines parallèles uniformes **R :** Atelier à machines parallèles avec temps d'exécution variant d'une machine à l'autre **O :** open shop **F :** Flow shop **J :** Job shop **FF :** Flexible flow shop **FJ :** Flexible job shop
	α_2	Nombre de machines	

TABLE B.1 – Définition détaillée des champs de la notations ($\alpha \mid \beta \mid \gamma$)

Champ	**Composantes du champ**	**Description**	**Valeur**
	β_1	Job/opération avec préemption (une opération peut être interrompue en cours d'exécution puis reprise ultérieurement)	**0** (absence de préemption), **pmtn** (opérations préemptives)
	β_2	Contraintes de précédence	**0 :** absence de relation de précédence **prec :** les opérations d'un job doivent respecter la relation de précédence. **chain :** le job a au plus un prédécesseur et au plus un successeur. **tree :** relation de précédence entre opérations sous forme d'arbre. **intree :** le job a au plus un successeur. **outree :** le job a au plus un prédécesseur
	β_3	Restriction sur les durées opératoires	
	β_4	Due date	
	β_5	Traitement par lots (Batch) ou par catégories (families processing)	
β	β_6	Nombre d'opérations d'un job	
	β_7	Jobs et opérations prioritaires (cas du jobshop)	
	β_8	Disponibilité de la machine (appelée aussi Panne) : une machine n'est pas disponible en permanence	0, avail (ou brkdwn)

TABLE B.1 – Définition détaillée des champs de la notations ($\alpha \mid \beta \mid \gamma$)

Champ	**Composantes du champ**	**Description**	**Valeur**
	β_9	Existence de ressources additionnelles/auxiliaires (cas du jobshop)	
	β_{10}	Buffers (cas du jobshop)	
	β_{11}	Temps de configuration (Setup times) (cas du jobshop)	
	β_{12}	Recirculation : dans le cas où un job visite une machine plus d'une fois (cas du job shop ou job shop flexible)	0, rcrc
	β_{13}	Non-attente (ou no-wait) : Pas d'attente autorisée entre deux opérations du même job (cas du flow shop)	0, nwt
	β_{14}	Permutation : Les files d'attente devant chaque machine fonctionne selon la règle du FIFO (l'ordre par lequel les jobs passent sur la première machine correspond à celui du système entier (cas du flow shop))	0, prmu
γ	γ	La fonction objectif à optimiser	voir 2.1

Annexe C

La myopie dans les différents domaines

- **Médecine :** La myopie, du grecque *muôpia*, qui signifie "à courte vue", est un trouble de la vision qui se caractérise par une perte de la netteté visuelle au fur et à mesure que la distance entre l'œil et l'objet augmente.
  Il existe différentes myopie :
  - Myopie axiale : l'œil est trop long, la focalisation de rayons provenant de l'infini se fait avant la rétine.
  - Myopie d'indice (de puissance) : liée à une augmentation de l'indice de réfraction du cristallin à cause d'une cataracte. Contrairement à la myopie axiale, les rayons se focalisent avant la rétine.

  L'image d'un point ressemble plus à une tache sur la rétine qu'un point, la perception d'un objet éloigné devient alors floue (Figure C.1). Plus l'objet est éloigné, plus celui-ci est flou.

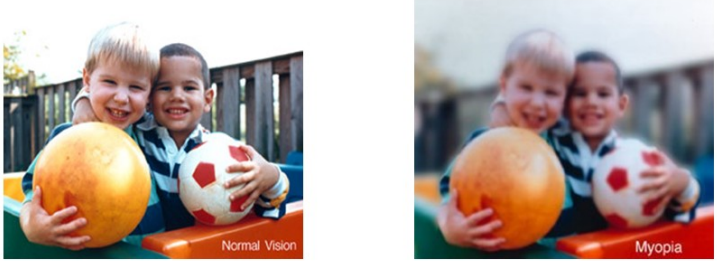

FIGURE C.1 – Œil normal VS œil myope

- **Économie :** ce type de myopie fait référence à une plus grande sensibilité aux pertes qu'aux gains (aversion aux pertes). Cette situation résulte du fait que certains décideurs agissent de manière "myope" lors de l'évaluation des opportunités d'investissement et essayent d'éviter à tout prix les pertes financières lors de leurs choix [193]. Ce type de myopie traite des décisions dans le temps et pourrait être qualifié donc de *temporelle* [32], car elle est liée à la volonté de suivre un planning à long terme.

- **Marketing :** deux types de myopie dans le domaine du markketing peuvent être distinguées [194] :
  - Myopie des capacités (capability myopia) : elle représente l'incapacité d'une entreprise à se reconfigurer, à faire de nouvelles propositions et à proposer de nouvelles solutions afin de s'adapter aux changements de l'environnement.
  - Myopie des limites (boundary myopia) : caractérise une entreprise en marketing qui a tendance à toujours communiquer et travailler avec le même groupe d'entités en ignorant ou en limitant les interactions avec les nouveaux groupes qui se créent.

  Ces deux types de myopie sont de type social, car elles déclarent qu'une organisation décidera de s'entourer d'une certaine quantité d'informations et de rejeter d'autres sources de données, même provenant d'entités fortement associées ce qui empêche l'entreprise de se développer ou d'atteindre d'autres marchés.

- **Robotique Mobile :** définie par l'incapacité de prévoir des cycles (répétition d'un ensemble d'actions), des oscillations et des minima locaux lors de trajectoires ou déplacements de robots mobiles [195].

Bibliographie

[1] Damien Trentesaux. Distributed control of production systems. *Engineering Applications of Artificial Intelligence*, 22(7) :971–978, October 2009.

[2] Hitoshi Tsubone and Mitsuyoshi Horikawa. A Comparison Between Machine Flexibility and Routing Flexibility. *International Journal of Flexible Manufacturing Systems*, 11(1) :83–101, February 1999.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[4] Richard Walter Conway, William L. Maxwell, and Louis W. Miller. *Theory of scheduling*. Addison-Wesley Pub. Co., December 1967.

[5] Weijun Xia and Zhiming Wu. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 48(2) :409–425, March 2005.

[6] Parviz Fattahi, Mohammad Saidi Mehrabad, and Fariborz Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3) :331–342, July 2007.

[7] Azzedine Bekkar, Oualid Guemri, Abdelghani Bekrar, Nassima Aissani, Bouziane Beldjilali, and Damien Trentesaux. An Iterative Greedy Insertion Technique for Flexible Job Shop Scheduling Problem. *IFAC-PapersOnLine*, 49(12) :1956–1961, January 2016.

[8] Azzedine Bekkar, Ghalem Belalem, and Beldjilali Bouziane. Iterated Greedy Insertion Approaches for the Flexible Job Shop Scheduling Problem with transportation times constraint. *International Journal of Manufacturing Research*, 14(1) :43–66, 2019.

[9] IMS2020. Supporting global research for 2020 manufacturing vision. www.ims2020.net, Mis en ligne en 2011, consulté le 02 Juin 2018.

[10] Pascal Blanc, Isabel Demongodin, and Pierre Castagna. A holonic approach for manufacturing execution system design : An industrial application. *Engineering Applications of Artificial Intelligence*, 21(3) :315–330, April 2008.

[11] A.J. Van Looveren, L.F. Gelders, and L.N. Van Wassehnove. A review of FMS planning models. In *A. Kusiak*, Modelling and design of flexible manufacturing systems. Elsevier Science Publishers, 1986.

[12] Paulo Jorge Pinto Leitão. *An Agile and Adaptive Holonic Architecture for Manufacturing Control*. PhD Thesis, Polytechnic Institute of Bragança, Portugal, January 2004.

[13] Khaled Ghédira. *Logistique de la production : approches de modélisation et de résolution*. Editions TECHNIP, 2006.

[14] Mohamed Essafi. *Conception et optimisation d'allocation de ressources dans les lignes d'usinage reconfigurables*. PhD Thesis, École Nationale Supérieure des Mines de Saint-Étienne, December 2010.

[15] Imad Chalfoun. *Conception et déploiement des Systèmes de Production Reconfigurables et Agiles (SPRA)*. PhD Thesis, Université BLAISEPASCAL-CLERMONTII, September 2014.

[16] José Eloundou. *Modélisation Multi-contraintes d'un Système de Production Flexible*. PhD Thesis, Université de Normandie, July 2016.

[17] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel. Reconfigurable Manufacturing Systems. *CIRP Annals*, 48(2) :527 – 540, 1999.

[18] Yoram Koren and Moshe Shpitalni. Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 29(4) :130 – 141, 2010.

[19] FMS, système de fabrication flexible. http ://www.logistiqueconseil.org/Articles/Gestion-production/Fms.htm, 2016-12-03.

[20] Andrea Krasa Sethi and Suresh Pal Sethi. Flexibility in manufacturing : A survey. *International Journal of Flexible Manufacturing Systems*, 2(4) :289–328, 7 1990.

[21] Mattias Hallgren and Jan Olhager. Flexibility configurations : Empirical analysis of volume and product mix flexibility. *Omega*, 37(4) :746–756, August 2009.

[22] Pawel Wojakowski. Research study of state-of-the-art algorithms for flexible job-shop scheduling problem. *Czasopismo Techniczne*, 2015.

[23] Jim Browne, Didier Dubois, Keith Rathmill, Suresh P. Sethi, and Kathryn E. Stecke. Classification of flexible manufacturing systems. *The FMS magazine*, 2(2) :114–117, 1984.

[24] B. L. MacCarthy and Jiyin Liu. A new classification scheme for flexible manufacturing systems. *International Journal of Production Research*, 31(2) :299–309, February 1993.

[25] Marino Widmer. *Modèles mathématiques pour une gestion efficace des ateliers flexibles*. PPUR presses polytechniques, 1991.

[26] Ronald G. Askin and Charles R. Standridge. *Modeling and analysis of manufacturing systems*. John Wiley & Sons, Ed., 1993.

[27] Damien Trentesaux. *Pilotage hétérarchique des systèmes de production*. PhD thesis, Université de Valenciennes et du Hainaut-Cambresis, 2002.

[28] Ray Bolton and Stephen Tyler. PQLI Engineering Controls and Automation Strategy. *Journal of Pharmaceutical Innovation*, 3(2) :88–94, June 2008.

[29] Patrick Pujo. *Isoarchic control for manufacturing systems.* Habilitation à diriger des recherches, Université Paul Cézanne - Aix-Marseille III, December 2009.

[30] M K Senehi and Thomas R Kramer. A Framework for Control Architectures. *International Journal of Computer Integrated Manufacturing*, 11(4) :347–363, July 1998.

[31] Cyrille Pach. *ORCA : Architecture hybride pour le contrôle de la myopie dans le cadre du pilotage des Systèmes Flexibles de Production.* PhD thesis, Université de Valenciennes et du Hainaut-Cambresis, 2013.

[32] Gabriel Zambrano, Cyrille Pach, Emmanuel Adam, Thierry Berger, and Damien Trentesaux. Myopic behaviour in heterarchical control of fms. In *International Conference on Industrial Engineering and Systems Management IESM*, 2011.

[33] Q. Zhang, H. Manier, and M.-A. Manier. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, 39(7) :1713–1723, July 2012.

[34] Stephen C. Graves. A review of production scheduling. *Operations Research*, 29(4) :646–675, 1981.

[35] Michael L. Pinedo. *Scheduling, theory, algorithms, and systems.* Prentice Hall, Englewood Cliffs, 1995.

[36] Pierre Lopez and Roubellat François. *Ordonnancement de la production.* Ed. Hermès Science Publications., France, 2001.

[37] Michael L. Pinedo. *Scheduling.* Springer US, Boston, MA, 2012.

[38] Norme AFNOR. Organisation et gestion de la production industrielle – concepts fondamentaux de la gestion de production. http ://www.afnor.fr/portail.asp, Mis en ligne en Décembre 1991, consulté le 23 Mai 2018.

[39] Mohand Larabi. *Le problème de job-shop avec transport : modélisation et optimisation.* PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, December 2010.

[40] Toumi F. Tangour. *Ordonnancement Dynamique dans les Industries Agroalimentaires.* PhD Thesis, Université des Sciences et Technologies de Lille, 2007.

[41] Patrick Esquirol and Pierre Lopez. *L'ordonnancement.* Economica, 1999.

[42] S. Afshin Mansouri, Emel Aktas, and Umut Besikci. Green scheduling of a two-machine flowshop : Trade-off between makespan and energy consumption. *European Journal of Operational Research*, 248(3) :772 – 788, 2016.

[43] Hadi Mokhtari and Aliakbar Hasani. An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers & Chemical Engineering*, 104 :339 – 352, September 2017.

[44] Zeyi Sun, Stephan Biller, Fangming Gu, and Lin Li. Energy Consumption Reduction for Sustainable Manufacturing Systems Considering Machines With Multiple-Power States. (44311) :99–103, 2011.

[45] Shaohua Hu, Fei Liu, Yan He, and Tong Hu. An on-line approach for energy efficiency monitoring of machine tools. *Journal of Cleaner Production*, 27 :133 – 140, may 2012.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

[46] T. Hsu, R. Dupas, D. Jolly, and G. Goncalves. Evaluation of mutation heuristics for solving a multiobjective flexible job shop by an evolutionary algorithm. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 5, Oct 2002.

[47] Guohui Zhang, Xinyu Shao, Peigen Li, and Liang Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4) :1309–1318, May 2009.

[48] Li-Ning Xing, Ying-Wu Chen, and Ke-Wei Yang. Multi-objective flexible job shop schedule : Design and evaluation by simulation modeling. *Applied Soft Computing*, 9(1) :362–376, January 2009.

[49] Jose Fernando Jimenez, Abdelghani Bekrar, Adriana Giret, Paulo Leitão, and Damien Trentesaux. A dynamic hybrid control architecture for sustainable manufacturing control. *IFAC-PapersOnLine*, 49(31) :114–119, January 2016.

[50] Brian J Ritzel, J Wayland Eheart, and S Ranjithan. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research*, 30(5) :1589–1603, 1994.

[51] A. Charnes and W. W. Cooper. *Management models and industrial applications of linear programming. Vol. I, II.* John Wiley and Sons, New York, NY, 1961.

[52] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[53] Michael P. Fourman. Compaction of symbolic layout using genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 141–153, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[54] Frank Kursawe. A variant of evolution strategies for vector optimization. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, PPSN I, pages 193–197, Berlin, Heidelberg, 1991. Springer-Verlag.

[55] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization : FormulationDiscussion and Generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[56] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, June 1994.

[57] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3) :221–248, Sept 1994.

[58] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE transactions on evolutionary computation*, 6(2) :182–197, April 2002.

[59] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i : Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4) :577–601, Aug 2014.

[60] Joshua Knowles and David Corne. The pareto archived evolution strategy : A new baseline algorithm for pareto multiobjective optimisation. In *Congress on Evolutionary Computation (CEC99)*, volume 1, pages 98–105, 1999.

[61] David Corne, Joshua D. Knowles, and Martin J. Oates. The pareto envelope-based selection algorithm for multi-objective optimisation. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 839–848, London, UK, UK, 2000. Springer-Verlag.

[62] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. Pesa-ii : Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 283–290, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[63] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms : a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4) :257–271, November 1999.

[64] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2 : improving the strength pareto evolutionary algorithm. *TIK-Report*, 103, 2001.

[65] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation*, volume 535, pages 3–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[66] Joc Cing Tay and Nhu Binh Ho. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, 54(3) :453–473, April 2008.

[67] Zengqiang Jiang, Le Zuo, and Mingcheng E. Study on multi-objective flexible job-shop scheduling problem considering energy consumption. *Journal of Industrial Engineering and Management*, 7(3) :589–604, June 2014.

[68] Anthony Caumond. *Le problème de jobshop avec contraintes : modélisation et optimisation.* PhD Thesis, Université Blaise Pascal - Clermond Ferrand II, December 2006.

[69] Atif Shahzad. *Une Approche Hybride de Simulation-Optimisation Basée sur la fouille de Données pour les problèmes d'ordonnancement.* PhD Thesis, École polytechnique de l'Université de Nantes, 2011.

[70] Mohamed Habib Zahmani. *Contribution à l'ordonnancement dynamique : proposition d'une approche guidée par effet de Simulation/Datamining.* PhD Thesis, Université Abdelhamid Ibn Badis Mostaganem, March 2018.

[71] Imen Mhedhbi Brinis. *Ordonnancement d'ateliers de traitements de surfaces pour une production mono-robot/multi-produits. Résolution et étude de la robustesse.* PhD Thesis, École Centrale de Lille, May 2014.

[72] Carl Adam Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390, 1962.

[73] Hela Boukef. *Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques : optimisation par algorithmes génétiques et essaims particulaires.* PhD Thesis, École Centrale de Lille et l'École Nationale d'Ingénieurs de Tunis, July 2009.

[74] Asma Karray. *Contribution à l'ordonnacement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides.* PhD Thesis, École Centrale de Lille et l'École Nationale d'Ingénieurs de Tunis, July 2011.

[75] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling : a Survey. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.

[76] Mohamed Habib Zahmani and Baghdad Atmani. Extraction of dispatching rules for single machine total weighted tardiness using a modified genetic algorithm and data mining. *International Journal of Manufacturing Research*, 13(1) :1–25, 2018.

[77] S. S. Sankar, S. G. Ponnambalam, V. Rathinavel, and M. S. Visveshvaren. Scheduling in parallel machine shop : an ant colony optimization approach. In *2005 IEEE International Conference on Industrial Technology*, pages 276–280, Dec 2005.

[78] Atif Shahzad. *Une Approche Hybride de Simulation-Optimisation Basée sur la fouille de Données pour les problèmes d'ordonnancement.* PhD thesis, École polytechnique de l'Université de Nantes, 2011.

[79] K. Ueda, I. Hatono, N. Fujii, and J. Vaario. Line-less Production System Using Self-Organization : A Case Study for BMS. *CIRP Annals - Manufacturing Technology*, 50(1) :319–322, January 2001.

[80] D.Y. Sha and Cheng-Yu Hsu. A new particle swarm optimization for the open shop scheduling problem. *Computers & Operations Research*, 35(10) :3243–3261, October 2008.

[81] Kun Fan, Yafei Zhai, Xinning Li, and Meng Wang. Review and classification of hybrid shop scheduling. *Production Engineering Research and Development*, 12(5) :597–609, October 2018.

[82] T.S. Arthanari and K.G. Ramamurthy. An extension of two machine sequencing problem. *Opsearch*, 8 :10–22, 1971.

[83] Hong Wang. Flexible flow shop scheduling : optimum, heuristics and artificial intelligence solutions. *Expert Systems*, 22(2) :78–85, May 2005.

Rapport-Gratuit.com

[84] Teruo Masuda, Hiroaki Ishii, and Toshio Nishida. The mixed shop scheduling problem. *Discrete Applied Mathematics*, 11(2) :175–186, June 1985.

[85] Andrea Rossi, Sauro Soldani, and Michele Lanzetta. Hybrid stage shop scheduling. *Expert Syst. Appl.*, 42(8) :4105–4119, May 2015.

[86] V. Nguyen and H. P. Bao. An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm. *Procedia Computer Science*, 95 :475 – 482, 2016.

[87] Zongyan Lou, Gaoxiang ; Cai. Improved hybrid immune clonal selection genetic algorithm and its application in hybrid shop scheduling. *Cluster Computing*, pages 1–11, 3 2018.

[88] Michael R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2) :117–129, 1976.

[89] Peter Brucker and Rainer Schlie. Job-shop scheduling with multi-purpose machines. *Computing*, 45(4) :369–375, 1990.

[90] Pierre Lopez and François Roubellat. *Production scheduling (Control systems, robotics and manufacturing series)*. London, 2008.

[91] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3) :268–308, 2003.

[92] Jean-Charles Boisson. *Modélisation et résolution par métaheuristiques coopératives : de l'atome à la séquence protéique*. PhD thesis, Université Lille 1, 2008.

[93] Vittorio Maniezzo, Thomas Stützle, and Stefan Voss, editors. *Matheuristics : hybridizing metaheuristics and mathematical programming*. Number v. 10 in Annals of information systems. Springer, New York, 2009.

[94] M. Yazdani, M. Amiri, and M. Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37(1) :678–687, January 2010.

[95] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3) :157–183, 1993.

[96] Johann Hurink, Bernd Jurisch, and Monika Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4) :205–215, December 1994.

[97] Stéphane Dauzère-Pérès and Jan Paulli. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70 :281–306, 1997.

[98] Monaldo Mastrolilli and Luca Maria Gambardella. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3(1) :3–20, January 2000.

[99] Yazid Mati, Nidhal Rezg, and Xiaolan Xie. An integrated greedy heuristic for a flexible job shop scheduling problem. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 4, pages 2534–2539. IEEE, 2001.

[100] Yazid Mati, Nidhal Rezg, and Xiaolan Xie. Greedy Heuristic and Genetic Algorithms for the Multi-resource Shop Scheduling with Resource Flexibility. In *Proc. 8th International Workshop on Project Management and Scheduling (PMS2002), Valencia, Spain.* Citeseer, 2002.

[101] Imed Kacem, Slim Hammadi, and Pierre Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(1) :1–13, 2002.

[102] Imed Kacem, Slim Hammadi, and Pierre Borne. Pareto-optimality approach for flexible job-shop scheduling problems : hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and computers in simulation*, 60(3) :245–276, 2002.

[103] Yeo Keun Kim, Kitae Park, and Jesuk Ko. A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers & Operations Research*, 30(8) :1151–1171, July 2003.

[104] F. T. S. Chan, T. C. Wong, and L. Y. Chan. Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, 44(11) :2071–2089, 2006.

[105] Hongbo Liu, Ajith Abraham, Okkyung Choi, and Seong Hwan Moon. Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 197–204. Springer, 2006.

[106] Jie Gao, Linyan Sun, and Mitsuo Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9) :2892–2907, September 2008.

[107] F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research*, 35(10) :3202–3212, October 2008.

[108] Mitsuo Gen, Jie Gao, and Lin Lin. Multistage-Based Genetic Algorithm for Flexible Job-Shop Scheduling Problem. In Mitsuo Gen, David Green, Osamu Katai, Bob McKay, Akira Namatame, Ruhul A. Sarker, and Byoung-Tak Zhang, editors, *Intelligent and Evolutionary Systems*, volume 187, pages 183–196. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[109] Guohui Zhang, Xinyu Shao, Peigen Li, and Liang Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Comput. Ind. Eng.*, 56(4) :1309–1318, May 2009.

[110] Jun-qing Li, Quan-ke Pan, and Yun-Chia Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Comput. Ind. Eng.*, 59(4) :647–662, November 2010.

[111] A. Bagheri, M. Zandieh, Iraj Mahdavi, and M. Yazdani. An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Gener. Comput. Syst.*, 26(4) :533–541, April 2010.

[112] L. De Giovanni and F. Pezzella. An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. *European Journal of Operational Research*, 200(2) :395–408, January 2010.

[113] Li-Ning Xing, Ying-Wu Chen, Peng Wang, Qing-Song Zhao, and Jian Xiong. A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems. *Applied Soft Computing*, 10(3) :888–896, June 2010.

[114] Ghasem Moslehi and Mehdi Mahnam. A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1) :14–22, January 2011.

[115] Ahmed Azab and B. Naderi. Greedy Heuristics for Distributed Job Shop Problems. *Procedia CIRP*, 20 :7–12, 2014.

[116] S. Karthikeyan, P. Asokan, S. Nickolas, and Tom Page. A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems. *Int. J. Bio-Inspired Comput.*, 7(6) :386–401, November 2015.

[117] T. Zhang and O. Rose. Scheduling in a flexible job shop with continuous operations at the last stage. *Journal of Simulation*, 10(2) :80–88, May 2016.

[118] H. Xu, Z.R. Bao, and T. Zhang. Solving dual flexible job-shop scheduling problem using a Bat Algorithm. *Advances in Production Engineering & Management*, 12(1) :5–16, March 2017.

[119] S. Kavitha, P. Venkumar, N. Rajini, and P. Pitchipoo. An efficient social spider optimization for flexible job shop scheduling problem. *Journal of Advanced Manufacturing Systems*, 17(2) :181–196, 2018.

[120] Nassima Keddari, Nasser Mebarki, Atif Shahzad, and Zaki Sari. Solving an Integration Process Planning and Scheduling in a Flexible Job Shop Using a Hybrid Approach. In Abdelmalek Amine, Malek Mouhoub, Otmane Ait Mohamed, and Bachir Djebbar, editors, *Computational Intelligence and Its Applications*, volume 522, pages 387–398. Springer International Publishing, Cham, 2018.

[121] *Shop-Scheduling Problems with Transportation.* Knust, sigrid, Fachbereich Mathematik/Informatik Universität Osnabrück, 1999.

[122] Johann Hurink and Sigrid Knust. Tabu search algorithms for job-shop problems with a single transport robot. *European Journal of Operational Research*, 162 :99–111, April 2005.

[123] Johann Hurink and Sigrid Knust. Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Applied Mathematics*, 112(1) :199 – 216, 2001.

[124] Philippe Lacomme, Mohand Larabi, and Nikolay Tchernev. A Disjunctive Graph for the Job-Shop with Several Robots. In *The 3rd Multidisciplinary International Scheduling Conference : Theory & Applications (MISTA)*, volume 20, pages 285 – 292, Paris, France, August 2007.

[125] Philippe Lacomme, Mohand Larabi, and Nikolay Tchernev. System generation schedules for transportation problem in job-shop environment. In *Actes du Colloque international sur l'Optimisation et les Systèmes d'Information*, pages 281–392, Tizi-Ouzou, Algérie, June 2008.

[126] Philippe Lacomme, Mohand Larabi, and Nikolay Tchernev. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics*, 143(1) :24–34, May 2013.

[127] H.M. Afsar, P. Lacomme, L. Ren, C. Prodhon, and D. Vigo. Resolution of a Job-Shop problem with transportation constraints : a master/slave approach. *IFAC-PapersOnLine*, 49(12) :898–903, 2016.

[128] Chengkuan Zeng, Jiafu Tang, and Chongjun Yan. Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles. *Applied Soft Computing*, 24 :1033–1046, November 2014.

[129] Myriam Leonor Niño López, Henry Lamos Díaz, Paula Julieth Jaimes Sanmiguel, and Julieth Vanessa Rivera González. Transfer batch size impact on a job shop environment performance. *International Journal of Manufacturing Research*, 12(3) :318–337, 2017.

[130] Ümit Bilge and Gündüz Ulusoy. A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in an FMS. *Operations Research*, 43(6) :1058–1070, 1995.

[131] V.A. Strusevich. A heuristic for the two-machine open-shop scheduling problem with transportation times. *Discrete Applied Mathematics*, 93(2-3) :287–304, 1999.

[132] Tamer F. Abdelmaguid, Ashraf O. Nassef, Badawia A. Kamal, and Mohamed F. Hassan. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 42(2) :267–281, 2004.

[133] H. Allaoui and A. Artiba. Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. *Computers & Industrial Engineering*, 47(4) :431–450, December 2004.

[134] L. Deroussi, M. Gourgand, and N. Tchernev. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 46(8) :2143–2164, April 2008.

[135] A. Caumond, P. Lacomme, A. Moukrim, and N. Tchernev. An MILP for scheduling problems in an FMS with one vehicle. *European Journal of Operational Research*, 199(3) :706–722, December 2009.

[136] Mourad Boudhar and Amina Haned. Preemptive scheduling in the presence of transportation times. *Computers & Operations Research*, 36(8) :2387–2393, 2009.

[137] B. Naderi, M. Zandieh, A. Khaleghi Ghoshe Balagh, and V. Roshanaei. An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications*, 36(6) :9625–9633, 2009.

[138] B. Naderi, A. Ahmadi Javid, and F. Jolai. Permutation flowshops with transportation times : mathematical models and solution methods. *The International Journal of Advanced Manufacturing Technology*, 46(5–8) :631–647, 2010.

[139] Fatima El Khoukhi, Tarik Lamoudan, Jaouad Boukachour, and Ahmed El Hilali Alaoui. Ant Colony Algorithm for Just-in-Time Job Shop Scheduling with Transportation Times and Multirobots. *ISRN Applied Mathematics*, 2011 :1–19, 2011.

[140] Erol Rizvan, Sahin Cenk, Baykasoglu Adil, and Kaplanoglu Vahit. A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. *Applied Soft Computing*, 12(6) :1720–1732, 2012.

[141] Qiao Zhang, Hervé Manier, and Marie-Ange Manier. A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *International Journal of Production Research*, 52(4) :985–1002, February 2014.

[142] Medikondu Nageswararao, K. Narayanarao, and G. Ranagajanardhana. Simultaneous scheduling of machines and agvs in flexible manufacturing system with minimization of tardiness criterion. *Procedia Materials Science*, 5 :1492–1501, 2014.

[143] Z. Liu, S. Ma, Y. Shi, and H. Teng. Solving multi-objective flexible job shop scheduling with transportation constraints using a micro artificial bee colony algorithm. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 427–432, June 2013.

[144] Sajad Karimi, Zaniar Ardalan, B. Naderi, and M. Mohammadi. Scheduling flexible job-shops with transportation times : Mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*, 41 :667–682, January 2017.

[145] Andrea Rossi. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *International Journal of Production Economics*, 153 :253–267, July 2014.

[146] Andrea Rossi and Gino Dini. Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. *Robotics and Computer-Integrated Manufacturing*, 23(5) :503–516, October 2007.

[147] L Deroussi and S Norre. Simultaneous scheduling of machines and vehicles for the flexible job shop problem. In *International conference on metaheuristics and nature inspired computing*, pages 1–2. Djerba Island Tunisia, 2010.

[148] Paul P Pandian, S Saravana Sankar, SG Ponnambalam, and M Victor Raj. Scheduling of automated guided vehicle and flexible jobshop using jumping genes genetic algorithm. *American Journal of Applied Sciences*, 9(10) :1706–1720, 2012.

[149] Houssem Eddine Nouri, Olfa Belkahla Driss, and Khaled Ghédira. Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model. *Computers & Industrial Engineering*, March 2016.

[150] Derya Deliktas, Orhan Torkul, and Ozden Ustun. A flexible job shop cell scheduling with sequence-dependent family setup times and intercellular transportation times using conic scalarization method. *International Transactions in Operational Research*, May 2017.

[151] D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, and J. Barbosa. Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice*, 21(9) :1204–1225, 2013.

[152] Imran Ali Chaudhry and Abid Ali Khan. A research survey : review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3) :551–591, May 2016.

[153] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2) :278–285, January 1993.

[154] Hoda A. ElMaraghy. Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, 17(4) :261–276, October 2005.

[155] Adil Baykasoğlu and Lale Özbakır. Analysing the effect of flexibility on manufacturing systems performance. *Journal of Manufacturing Technology Management*, 19(2) :172–193, 2008.

[156] El-Ghazali Talbi. *Metaheuristics : from design to implementation.* Wiley, Hoboken, NJ, 2009.

[157] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated Local Search. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, number 57 in International Series in Operations Research & Management Science, pages 320–353. Springer US, 2003. DOI : 10.1007/0-306-48056-5_11.

[158] Rubén Ruiz and Thomas Stützle. An iterated greedy algorithm for the flowshop problem with sequence dependent setup times. In *The 6th Metaheuristics International Conference*, pages 817–826, 2005.

[159] Rubén Ruiz and Thomas Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3) :2033–2049, March 2007.

[160] Jose M. Framinan and Rainer Leisten. A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. *OR Spectrum*, 30(4) :787–804, October 2008.

[161] Kuo-Ching Ying and Hui-Miao Cheng. Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Systems with Applications*, 37(4) :2848–2852, April 2010.

[162] Bahman Naderi, Seyed Mohammad Taghi Fatemi Ghomi, and others. An Iterated Greedy Algorithm for Flexible Flow Lines with Sequence Dependent Setup Times to Minimize Total Weighted Completion Time. *Journal of Optimization in Industrial Engineering*, 2(3) :33–37, 2010.

[163] Mary E. Kurz and Ronald G. Askin. Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, 159(1) :66 – 82, 2004.

[164] Gerardo Minella, Rubén Ruiz, and Michele Ciavotta. Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operations Research*, 38(11) :1521–1533, November 2011.

[165] B. Naderi, Shadi Rahmani, and Shabnam Rahmani. A Multiobjective Iterated Greedy Algorithm for Truck Scheduling in Cross-Dock Problems. *Journal of Industrial Engineering*, 2014 :1–12, 2014.

[166] Marco Pranzo and Dario Pacciarelli. An iterated greedy metaheuristic for the blocking job shop scheduling problem. *Journal of Heuristics*, 22(4) :587–611, August 2016.

[167] Korhan Karabulut. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. *Computers & Industrial Engineering*, 98 :300–307, August 2016.

[168] Jérémie Dubois-Lacoste, Federico Pagnozzi, and Thomas Stützle. An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem. *Computers & Operations Research*, 81 :160–166, May 2017.

[169] Holger H. Hoos and Thomas Stützle. *Stochastic local search : foundations and applications*. Elsevier, Amsterdam, 2005. OCLC : 249889922.

[170] Michael Pinedo. *Scheduling : Theory, Algorithms, and Systems*. Prentice Hall, 2002.

[171] Cyrille Pach, Thierry Berger, Thérèse Bonte, and Damien Trentesaux. ORCA-FMS : a dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. *Computers in Industry*, 65(4) :706–720, May 2014.

[172] N. Zbib, C. Pach, Y. Sallez, and D. Trentesaux. Heterarchical production control in manufacturing systems using the potential fields concept. *Journal of Intelligent Manufacturing*, 23(5) :1649–1670, October 2012.

[173] J.Pirie Hart and Andrew W. Shogan. Semi-greedy heuristics : An empirical study. *Oper. Res. Lett.*, 6(3) :107–114, July 1987.

[174] Fred Glover and Manuel Laguna. *Tabu search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[175] Jose-Fernando Jimenez, Abdelghani Bekrar, Gabriel Zambrano-Rey, Damien Trentesaux, and Paulo Leitão. Pollux : a dynamic hybrid control architecture for flexible job shop systems. *International Journal of Production Research*, 55(15) :4229–4247, 2017.

[176] Fatima Benbouzid Sitayeb, Sid Ali Guebli, Yassine Bessadi, Christophe Varnier, and Noureddine Zerhouni. Joint scheduling of jobs and Preventive Maintenance opera-

tions in the flowshop sequencing problem : a resolution with sequential and integrated strategies. *International Journal of Manufacturing Research*, 6(1) :30–48, 2011.

[177] W D Li and L Gao. Intelligent and cooperative manufacturing planning. *International Journal of Manufacturing Research*, 5(1) :39–57, 2010.

[178] Deyun Wang, Xiaohan Zhao, and Kejun Zhu. Single machine serial batching and scheduling problem with deteriorating jobs for production-distribution supply chain under discrete due-date constraints. *International Journal of Manufacturing Research*, 10(4) :346–370, 2015.

[179] John H Holland. Adaptation in natural and artificial systems. an introductory analysis with applications to biology, control and artificial intelligence. *Ann Arbor : University of Michigan Press, 1975*, 1975.

[180] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[181] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598) :671–680, May 1983.

[182] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[183] Marco Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[184] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1) :29–41, February 1996.

[185] Li-Ning Xing, Ying-Wu Chen, and Ke-Wei Yang. Double Layer ACO Algorithm for the Multi-Objective FJSSP. *New Generation Computing*, 26(4) :313–327, August 2008.

[186] Rong-Hwa Huang, Chang-Lin Yang, and Wei-Che Cheng. Flexible job shop scheduling with due window—a two-pheromone ant colony approach. *International Journal of Production Economics*, 141(2) :685 – 697, 2013.

[187] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium*, pages 39–43, Oct 1995.

[188] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5) :533–549, January 1986.

[189] Fred Glover. Tabu search—part II. *ORSA Journal on Computing*, 2(1) :4–32, 1990.

[190] Fatima Benbouzid Sitayeb. *Contribution à l'étude de la performance et de la robustesse des ordonnancements conjoints production/maitenance - Cas du flowshop*. PhD Thesis, Université de Franche-Comté des Sciences et Techniques, June 2005.

[191] Noria Taghezout. *Conception et Développement d'un système multi-agent d'Aide à la décision pour la gestion de production dynamique.* PhD Thesis, Université de Toulouse III - Paul Sabatier, 2011.

[192] Thomas A. Feo and Mauricio G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2) :67 – 71, 1989.

[193] R. H. Thaler, A. Tversky, D. Kahneman, and A. Schwartz. The effect of myopia and loss aversion on risk taking : An experimental test. *Quarterly Journal of Economics*, 112(2) :647–661, may 1997.

[194] Kevin Johnston. Extending the marketing myopia concept to promote strategic agility. *Journal of Strategic Marketing*, 17(2) :139–148, April 2009.

[195] M.J. Mataric. Minimizing complexity in controlling a mobile robot population. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, volume 1, pages 830–835, May 1992.

Résumé

L'ordonnancement, point névralgique du pilotage de production, consiste à affecter les tâches aux machines et séquencer les opérations sur les ressources. Les méthodes d'ordonnancement sont de plus en plus complexes et sophistiqués afin de répondre à l'objectif premier qui est de minimiser le temps total de production tout en respectant des contraintes de plus en plus avancées, telle que le coût de production, la qualité du produit fini, la répartition équitable des opérations sur machines pour éviter toute surexploitation entrainant la détérioration de la ressource, ou plus récemment réduire la consommation d'énergie et l'émission du CO2 dans un but environnemental. Nous proposons dans cette thèse deux nouvelles approches en combinant des heuristiques gloutonnes itérées et des techniques d'insertion pour l'ordonnancement dans un job shop flexible en prenant en considération les temps de transport entre chaque machine. Les deux méthodes proposées ont été testées sur un benchmark qui prend en compte les temps de transport entre les machines et qui représente l'instanciation d'un système de production flexible réel : « AIP-PRIMECA » de l'université de Valenciennes. Les résultats des heuristiques proposées sont comparés à ceux de la méthode exacte "MILP" et de la méthode des champs potentiels. Les résultats obtenus sont prometteurs et les deux méthodes permettent de traiter des problèmes de grandes tailles.

Mots clés : Job shop flexible, Ordonnancement, Production, Optimisation, Heuristiques d'insertion gloutonnes.

Abstract

Scheduling, hotspot in production control, consists in assigning tasks to machines and sequencing operations on resources. Scheduling methods are increasingly complex and sophisticated in order to meet the primary objective of minimizing total production time while respecting more and more advanced constraints, such as the cost of production, the quality of the finished product, the equitable distribution of the operations on machines to avoid overexploitation leading to the deterioration of the resource, or more recently to reduce energy consumption and CO2 emissions for environmental purposes. In this thesis, we propose two new approaches by combining iterated greedy heuristics and insertion techniques to schedule tasks in a flexible Job Shop taking into account the transportation time between each machine. The two proposed methods have been tested on a benchmark that takes into account the transport time between machines and represents the instantiation of a real flexible production system: "AIP-PRIMECA" from the University of Valenciennes. The results of the proposed heuristics are compared with those of the exact method "MILP" and the method of the potential fields. The results obtained are very promising and the two methods make it possible to treat instances of large sizes.

Keywords: Flexible job shop, Scheduling, Production, Optimization, Iterated Greedy Heuristics.

ملخص

تتألف الجدولة، وهي نقطة ساخنة في التحكم في الإنتاج، من تعيين المهام إلى الآلات وعمليات التسلسل على الموارد.

طرق الجدولة معقدة ومعقدة بشكل متزايد من أجل تلبية الهدف الأساسي للتقليل من وقت الإنتاج الإجمالي مع احترام المزيد والمزيد من القيود المتقدمة، مثل تكلفة الإنتاج والجودة من المنتج النهائي، التوزيع العادل لعمليات الماكينة لتجنب الإفراط في الاستغلال مما يؤدي إلى تدهور المورد، أو في الآونة الأخيرة للحد من استهلاك الطاقة وانبعاثات CO2 لأغراض بيئية.

في هذه الأطروحة، نقترح نهجين جديدين من خلال الجمع بين الأساليب الجراحية المتكرّرة وتقنيات الإدراج لجدولة المهام في ورشة عمل مرنة لورشة العمل مع الأخذ في الاعتبار وقت النقل بين كل جهاز.

تم اختبار الطريقتين المقترحتين على مقياس مرجعي يأخذ في الاعتبار وقت النقل بين الآلات ويمثل إنشاء نظام إنتاج مرن حقيقي: "AIP-PRIMECA" من جامعة فالنسيان.

يتم مقارنة نتائج الاستدلال المقترحة مع تلك الطريقة الدقيقة "MILP" وطريقة المجالات المحتملة. النتائج التي تم الحصول عليها واعدة للغاية وكلا الطريقتين تجعل من الممكن علاج الحالات الكبيرة.

الكلمات الرئيسية: ورشة عمل مرنة، جدولة، إنتاج، تحسين، خوارزميات الإدراج الجشع.

Résumé

L'ordonnancement, point névralgique du pilotage de production, consiste à affecter les tâches aux machines et séquencer les opérations sur les ressources. Les méthodes d'ordonnancement sont de plus en plus complexes et sophistiqués afin de répondre à l'objectif premier qui est de minimiser le temps total de production tout en respectant des contraintes de plus en plus avancées, telle que le coût de production, la qualité du produit fini, la répartition équitable des opérations sur machines pour éviter toute surexploitation entrainant la détérioration de la ressource, ou plus récemment réduire la consommation d'énergie et l'émission du CO2 dans un but environnemental. Nous proposons dans cette thèse deux nouvelles approches en combinant des heuristiques gloutonnes itérées et des techniques d'insertion pour l'ordonnancement dans un job shop flexible en prenant en considération les temps de transport entre chaque machine. Les deux méthodes proposées ont été testées sur un benchmark qui prend en compte les temps de transport entre les machines et qui représente l'instanciation d'un système de production flexible réel : « AIP-PRIMECA » de l'université de Valenciennes. Les résultats des heuristiques proposées sont comparés à ceux de la méthode exacte "MILP" et de la méthode des champs potentiels. Les résultats obtenus sont prometteurs et les deux méthodes permettent de traiter des problèmes de grandes tailles.

Mots clés :

Système de production flexible; Job shop flexible; Ordonnancement ; Production; Architectures de pilotage; Transport; Optimisation; Techniques d'insertion; Heuristiques gloutonnes itérées; cellule flexible AIP-PRIMECA.