الجـــــــــــــــــــــمهـوريـــــة الجـــــــــــــــزائـريـة الديمـــــــقـراطيـة الـشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــــعـليـم العـــــــالي و البـــــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد– تـلمســـــــــــــــان –

Université Aboubakr Belkaïd– Tlemcen –

Faculté de TECHNOLOGIE

THESE

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

En : Automatique

Spécialité : Modélisation et commande des processus

Par : Mlle. KEDDARI Nassima

Sujet

INTEGRATION DE LA PLANIFICATION DES PROCESSUS ET L'ORDONNANCEMENT D'UN SYSTEME DE PRODUCTION PAR LES METAHEURISTIQUES

Soutenue publiquement, le 03 / 06 /2020 , devant le jury composé de :

Dr. GHOMRI Latéfa	MCA	Université de Tlemcen	Président
Dr. MELIANI Sidi Mohammed	MCA	Université de Tlemcen	Directeur de thèse
Dr. HASSAM Ahmed	MCB	Université de Tlemcen	Invité
Pr. HACHEMI Khalid	Professeur	Université d'Oran 2	Examinateur 1
Dr. SOUIER Mehdi	MCA	ESM Tlemcen	Examinateur 2
Dr. BENYAHIA Boumediene	MCA	Université de Tlemcen	Examinateur 3

Remerciements

Tout d'abord, je tiens à remercier très vivement mon directeur de thèse Monsieur MELIANI Sidi Mohammed et mon encadrant Monsieur HASSAM Ahmed, Maitres de conférence à l'université de Tlemcen, pour leurs conseils et encouragements tout au long de ce projet.

Mes remerciements les plus sincères à Monsieur MEBARKI Nasseur, Maitre de conférence à l'école centrale de Nantes, pour m'avoir accueillie au sein de l'institut technologique (IUT) et pour toute l'aide et la disponibilité qu'il m'a offert afin de mener ce travail en terme.

Je veux aussi remercier mon ancien directeur de thèse Monsieur SARI Zaki, Professeur à l'université de Tlemcen pour son aide et sa disponibilité.

J'exprime, également, toute ma gratitude à Madame GHOMRI Latéfa, Maître de conférence à l'université de Tlemcen pour avoir accepté être le président de jury de ma soutenance.

J'adresse tous mes remerciements à Monsieur HACHEMI Khalid, Professeur à l'université de d'Oran, Monsieur SOEIR Mehdi, Maître de Conférence à l'école supérieure de Tlemcen et Monsieur BENYAHIA Boumediene, Maître de conférence à l'université de Tlemcen, pour m'avoir fait l'honneur d'accepter d'être rapporteurs de ma thèse.

Un grand merci à ma famille et mes proches, vous m'avez toujours encouragé à terminer ce que j'entreprends et aidé dans les moments les plus difficiles.

Merci infiniment.

A mes parents, maman et mon très cher papa

A ma sœur Dalila et mes frères Amine et Abdessamad

A mon fiancé Mohammed

Sommaire

CHAPITRE 5. Une approche hybride pour la planification des processus et l'ordonnancement intégré

Liste des Figures

Liste des Tableaux

Liste des abréviations

ABC	Artificial bee colony / Colonie d'abeilles artificielles
ACO	Ant colony optimization
OCF	Optimisation des colonies de fourmis
ACI	Algorithme compétitif impérialiste
AI	Artificial Intelligence
	Intelligence artificielle
AIS-FLC	Algorithm fuzzy logic controller
	Contrôleur de logique floue appelé algorithme
AIS	Artificial immune system
	Système immunitaire artificiel
AHC	Algorithme de hill climbing
AL+CGA	Approach by localization Genetic algorithm
	Approche par localisation d'algorithme génétique
BBO	Biogeography-based optimization
	Optimisation basée sur la biogéographie
B&B	Branch and Bound
SEP	Méthodes par séparation et évaluation
CAD	Computer-aided design
	Conception assistée par ordinateur
CAPP	Computer-aided process planning
	Les systèmes de planification des processus assistée par ordinateur
CLPP	Closed loop process planning
	Planification de processus en boucle fermée
CIM	Computer Integrated Manufacturing
	Fabrication intégrée par ordinateur
CPU	Central processing unit
	Unité centrale de traitement
CRO	Chemical Reaction Optimization
	Optimisation de la réaction chimique

DSS	Decision support system
	Système d'aide à la décision
ECT	Earliest completion time first
	Heure de fin au plus tôt en premier
EDD	Earliest due date
	Date d'échéance au plus tôt
ESA	Evolutionary search approach
	Approche de recherche évolutive
ESCSA	Enhanced Swift Conveging Simulated Annealing
	Convoyage rapide amélioré Recuit simulé
FIFO	Premier entré, premier sorti
	First-in first-out
FJS	Flexible job shop
JSF	Job shop flexible
FMS	Flexible manufacturing system
	Les systèmes de fabrication flexibles
GA	Genetic algorithm
AG	Algorithme génétique
GP	Genetic programming
PG	Programmation génétique
GPU	Graphics Processing Unit
	Unité de traitement graphique
GRASP	Greedy randomized adaptive search procedure
	Procédure de recherche adaptative randomisée gourmande
HS	Harmony Search
	Recherche d'harmonie
HM	Hollonic mating
	Accouplement hollonique
IAI	Imunitive artificial intelligence
	Intelligence artificielle impériale

IPPS	Integrated process planning and scheduling
	Planification des processus et ordonnancement intégrés
IPPS-FJS	Integration of process planning and scheduling in a flexible job shop
	L'intégration de la planification des processus et de l'ordonnancement dans un job shop flexible
IGA	Improved Genetic Algorithm
	Algorithme génétique amélioré
ITS	Integer tabu search
	Recherche tabou intégrée
ISA	Integer simulated annealing
	Recuit simulé intégré
JSP	Job shop problem
PJS	Problème job shop
LS	Local search
RL	Recherche locale
MA	Memetic Algorithms
AM	Algorithmes mémétiques
MATSLO	Multi Agent model Tabu Search Local Optimization Approach
	Modèle multi-agents Recherche Tabou Optimisation de la Recherche Locale
MATSPSO	Multi Agent model Tabu Search Local particle swarm Optimization
	Modèle Multi-Agents Recherche Tabou Optimisation des Essaims de Particules
MACRO’	Multi Agent model Chemical Reaction Optimization
	Modèle Multi-Agents Optimisation de la Réaction Chimique
MAS	Multi Agent System
	Système Multi-Agents
MIP	Mixed-integer programming
	Programmation en mode mixte
MILP	Mixed-integer linear programming
PLNE	Programmation linéaire à nombres entiers mixtes

MPP	Multiples plans de processus
MOPSO+LS	Multi Objectif Particle Swarm Optimization
	Optimisation multi-objectifs des essaims de particules
NLPP	Non-linear process planning
	Planification des processus non linéaires
NP-hard	Non-deterministic polynomial-time hard
	Temps Polynômial non déterministe
OF	Operation flexibility
	Flexibilité d'opération
PF	Processing flexibility
	Flexibilité de traitement
SF	Sequencing flexibility
	Flexibilité de séquencement
SA	Simulated annealing
RS	Recuit simulé
SBH	Shifting bottleneck heuristic
	Heuristique de goulot d'étranglement
SPT	Shortest processing time
	Temps de traitement le plus court
SPP	Scheduling Sub-Problem
	Ordonnancement de sous-problème
TS	Tabu search
RT	Recherche tabou
VNS	Variable neighborhood search
VRR	Variable de voisinage de recherche
	Flexible job shop problem
	Problème Job Shop Flexible
PL/LP	Linear programing
	La programmation linéaire
PSO	Particulair Swarm Optimization.

OEP	Algorithme optimisation par essaims particulaires
SIA	Artificial immune system
	Système immunitaire artificiel
SEA	Symbiotic Evolutionary Algorithm
	Algorithme évolutionnaire symbiotique
GT	Giffler and Thompson algorithm
	Algorithme de Giffler et Thompson
HMS	Holonic manufacturing system
	Système de fabrication holonique
TAD	Tool access direction
	Direction d'accès à l'outil

Introduction générale

Des systèmes de production de plus en plus complexes sont mis en œuvre dans le milieu industriel, afin de satisfaire au mieux, qualitativement et/ou quantitativement, une demande provenant d'un marché incertain. Assurer une grande compétitivité devient une nécessité pour les entreprises ; et ce, afin de faire face à la concurrence.

Le concept d'industrie 4.0 favorise l'intégration de tous les aspects de la production pour une plus grande efficacité. L'usine du futur sera une production intelligente, globale et flexible, où l'internet des objets, la réalité augmentée, l'automatisation et l'intelligence artificielle permettront d'adapter rapidement le système de production à un environnement en constante évolution. Cela nécessite un niveau élevé d'intégration d'entreprise.

Les systèmes flexibles de production (FMS) sont développés pour relever ces défis, car ils ont la capacité de réagir rapidement à la dynamique du marché (Chan et Chan 2004), et d'atteindre une productivité élevée pour répondre aux besoins concurrentiels actuels (MacCarthy et Liu 1993). L'amélioration de la gestion de production est alors, à la base de toute tentative de changement (Jav, 2004).

Cependant, dans la plupart des systèmes de production, deux fonctions distinctes sont toujours gérées indépendamment pour gérer la production : la fonction de planification des processus et la fonction de l'ordonnancement. L'ordonnancement et la planification des processus sont deux composantes clés du système de production (Mohammadi et al 2012) (Jain et al 2006).

La planification du processus détermine la manière dont un produit sera fabriqué, du produit initial au produit fini, en d'autres termes, un plan de processus spécifie les ressources de production et les opérations/routes techniques qui sont nécessaires pour fabriquer un produit (Kumar et Rajotia, 2006). Typiquement, un job (tâche) peut avoir plusieurs plans de processus alternatifs.

L'ordonnancement est une autre fonction de production qui vise en effet à organiser le fonctionnement du système de production, et qui trouve un mappage entre les jobs et les ressources pour atteindre certains critères pertinents. Par conséquent, l'ordonnancement est basé sur une planification de processus fixe.

Clairement, les deux fonctions sont interdépendantes, l'ordonnancement reçoit des plans de processus comme input. Toutefois, dans les approches traditionnelles, ces deux fonctions sont exécutées séquentiellement dans les systèmes de production en tant qu'activités de pré-production (Jain et al 2006). Cette approche séquentielle crée souvent des obstacles à l'amélioration de la productivité des systèmes de production et il est difficile de fournir une réactivité aux incertitudes de production (Shao et al, 2009; Lian et al, 2012). Au même temps, il peut apporter quelques autres problèmes, tels que les conflits d'objectifs entre la planification des processus et l'ordonnancement, le déséquilibre de charge des ressources de production et l'infaisabilité du plan de processus (Lv et Qiao, 2013; Li et al, 2012a; Shao et al, 2009).

Bien que, l'échec critique de ce paradigme séquentiel apparaît : un plan de traitement prédéterminé peut ne pas être le meilleur pour la planification. Pour résoudre ce problème, l'ordonnancement et la planification des processus doivent être considérés simultanément (Moon & Seo, 2005).

La planification des processus et l'ordonnancement intégrés *(IPPS ; Integration of Process Planning and Scheduling)* est le concept d'effectuer ces deux fonctions de façon concourante avec les objectifs d'éliminer ou réduire les conflits d'objectifs entre la planification des processus et l'ordonnancement, de réduire le makespan et les pièces dans le processus, de perfectionner l'utilisation des ressources et d'améliorer la flexibilité pour s'adapter à des incertitudes irrégulières dans les ateliers d'usinage (Lee et Kim, 2001;. Wan et al, 2013). Chryssolouris et Chan (1984) ont été les premiers à proposer l'idée préliminaire de l'intégration de la planification des processus et de l'ordonnancement.

L'IPPS devient un sujet de recherche brûlant et attire de plus en plus d'attention de la part des chercheurs et des ingénieurs. Ces avantages sont d'accroître la faisabilité et l'optimalité de la production en combinant ces deux fonctions (Tan, 1998 ; Tan et Khosnevis 1997; Li et McMahon, 2007). La productivité est l'une des mesures de performance les plus importantes dans tous les systèmes de production, le makespan est reconnu comme un moyen efficace à cet égard (Pinedo 2008 ; Seidel et Arndt 1998). Le problème d'ordonnancement avec l'objectif makespan assure une meilleure utilisation des ressources avec un coût supplémentaire nul, ce qui rend la minimisation des coûts inévitables dans le cadre de la gestion de production.

Vu la diversité des systèmes de production au niveau opérationnel, les problèmes étudiés en ordonnancement sont extrêmement divers. Parmi les nombreux ouvrages de référence qui ont été publiés, on trouve (Conway et al. 1967), (Baker 1974), (Rinnooy Kan 1976), (French 1982), (Carlier et Chrétienne 1988), (Tanaev et al. 1994a) (Brucker 1998), (Lopez et Esquirol 1999), (Lopez et Roubellat 2001), (Leung 2004), (Blazewicz et al. 2007). Les modèles d'ordonnancement diffèrent selon les technologies utilisées et les contraintes auxquelles est soumis le système. Théoriquement, la littérature a l'habitude de distinguer trois types d'atelier :

- Les ateliers de type flow-shop. Dans ce type d'atelier, la gamme opératoire est la même pour tous les jobs.
- Les ateliers de type job-shop. Dans ce type d'atelier, les gammes opératoires des jobs sont fixées et peuvent être différentes d'un job à un autre.
- Les ateliers de type open-shop. Dans ce type d'atelier, les gammes opératoires des jobs ne sont pas fixées.

Dans cette thèse, nous nous sommes intéressés aux problèmes de type job-shop flexible. Il est parmi les problèmes d'ordonnancement les plus difficiles et les plus étudiés. Le problème d'ordonnancement de type job shop classique *(PJS)* généralement assume qu'il existe un seul plan de processus réalisable pour chaque job et que chaque job à une séquence d'opérations fixe (par exemple, $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow ...$). Cela implique que l'opération O_{i+1} doit être programmée après O_i et avant O_{i+2} dans tous les cas. Afin d'augmenter la capacité de production, il est possible de mettre en parallèle des machines qui peuvent exécuter les mêmes opérations .Dans ce cas, on parle de système de production de type job shop flexible *(PJSF)*. Le job-shop flexible (hybride) est une extension du problème d'ordonnancement de type job-shop classique pour lequel une opération peut être traitée par l'une des machines d'un étage.

Tandis que, le problème d'intégration est un problème plus flexible car la plupart des jobs peuvent comporter un grand nombre des plans de processus réalisables reposant sur des

permutations d'opération différentes. L'intégration de la planification des processus et de l'ordonnancement dans un job shop flexible (IPPS-FJS) prend en compte les machines alternatives pour les opérations, appelée flexibilité d'opération (OF), et la séquence des opérations alternatives, appelée flexibilité de séquencement (SF).

Ce problème est en fait le problème d'ordonnancement de job shop flexible avec les plans de processus alternatifs (Özgüven et al 2010). Ce système est aussi appelé JSP-MPP problème d'ordonnancement de job shop avec les multiples planifications des processus. Cependant, il est traditionnellement appelé problème IPPS.

Comme l'ordonnancement et la planification des processus sont individuellement non polynomiales (NP), le problème qui en résulte est également difficile à résoudre (Gen et al., 2009). La complexité combinatoire du problème IPPS explose très rapidement, ce qui interdit généralement l'utilisation de méthodes exactes pour le résoudre. D'autre part, les méthodes heuristiques fournissent des solutions bonnes ou presque optimales dans un délai raisonnable (Brucker 2007). Les métaheuristiques sont parmi les méthodes heuristiques les plus efficaces. Une métaheuristique est un algorithme d'optimisation stochastique capable de résoudre une large gamme de problèmes sans nécessiter de modifications profondes de l'algorithme (Adekanmbi et Green, 2015).

Le problème de l'IPPS à l'étude comporte trois dimensions décisionnelles :

1- Décision pour l'affectation du plan de processus ;

2- Décision pour l'affectation des machines ;

3- Décision pour l'ordonnancement.

L'idée traditionnelle est de résoudre le problème à l'aide d'une approche en deux phases : sélectionner d'abord le plan de processus, puis assigner et planifier les opérations (Weintraub et al 1999 ; Li et al 2010). C'est-à-dire qu'un plan de processus pour chaque job est déterminé, puis les jobs avec des plans de processus fixes sont planifiés.

Objectif de thèse

Pour résoudre ce type de problème, nous avons opté pour l'hybridation des métaheuristiques, vu qu'elles sont efficaces et fiables. La principale contribution de cette thèse est de proposer une méthode hybride efficace pour évaluer et explorer l'espace de recherche de façon intelligente dans un délai raisonnable afin de résoudre le problème IPPS. La méthode proposée est évaluée à l'aide d'un modèle de production tiré de la littérature, l'objectif étant de minimiser le makespan Cmax. Cette méthode hybride, décrite plus en détail précisément dans les chapitres 4 et 5.

Plan de lecture

Ce mémoire est organisé en cinq chapitres.

Le chapitre 1 d´écrit l'ordonnancement de production. Nous rappelons donc les structures Classiques des problèmes d'ordonnancement d'ateliers et nous introduisons la notation de la planification des processus. Nous abordons ensuite les aspects de représentations des problèmes de la planification des processus et l'ordonnancement intégrés. Nous d´écrivons à la fin le contexte de l'étude et la caractérisation du problème étudié. Cette

section donne un aperçu de l'IPPS. Dans un premier temps, le besoin d'IPPS est identifié. Diverses approches fondamentales de l'IPPS ainsi que leurs avantages et leurs inconvénients sont discutés.

Le chapitre 2 est consacré à la description des méthodes hybrides proposées pour résoudre des problèmes d'ordonnancement de type job shop flexible (FJSP) et son intégration avec la planification des processus. Il donne un panorama sur les différents types d'hybridation utilisée pour résoudre les problèmes mono objectifs.

Dans le chapitre 3, nous élaborons un état de l'art des techniques de résolution couvrant les travaux de recherches menés sur les problèmes de planification des processus et d'ordonnancement d'atelier de type job shop/ job shop flexible, en accordant une importance particulière aux méthodes utilisées dans notre recherche. Les contributions de divers chercheurs sur l'IPPS sont aussi discutées.

Dans le chapitre 4, il s'agit de l'application d'une approche basée sur l'hybridation de trois techniques d'optimisation sur les problèmes d'ordonnancement de type job shop flexible sans intégration de la planification des processus.

L'approche proposée est une méthode hybride efficace permettant d'évaluer et d'explorer intelligemment l'espace de recherche. Cette approche hybride (AH), combine l'heuristique shifting bottleneck SBH pour la génération de la solution initiale, la recherche tabou RT pour l'exploration et la recherche locale et l'exploitation en utilisant l'algorithme de kangourou AK pour la recherche globale. Cette méthode combine les avantages des trois méthodes.

Le chapitre 5 étudie le problème de la planification des processus et l'ordonnancement de job shop intégré, en appuyant sur l'utilisation de la même approche hybride afin de tirer l'intérêt de cette intégration. Nous exposons dans cette partie l'approche hybride décrit précédemment. Nous terminons ce chapitre par une synthèse des résultats que nous avons conduits pour évaluer et comparer les méthodes proposées.

Nous conclurons le manuscrit en synthétisant les principales contributions présentées, et en dénombrant de nouvelles perspectives qui peuvent découler de cette étude.

CHAPITRE 1

Contexte du travail : Planification des processus et ordonnancement dans un job shop flexible

Ce chapitre est consacré aux aspects et la nécessité d'intégration de la planification des processus et l'ordonnancement dans un atelier de production de type job shop/ job shop flexible. Il regroupe les notions et notations nécessaires à la compréhension de ce mémoire.

1. Introduction

Dans l'environnement complexe des systèmes de production actuel, la capacité de produire d'une manière efficace et réaliste devient un facteur vital pour la gestion d'une entreprise manufacturière. Il est communément reconnu, que la planification des processus et l'ordonnancement sont deux fonctions clés dans la gestion d'un système de production qui ont une influence sur l'ensemble du processus de production. Par conséquent, tant la planification des processus que l'ordonnancement impliquent l'affectation des ressources et sont étroitement liés, leur intégration effective est très importante pour accroître la productivité et l'efficacité globale d'un atelier de production.

Bien que la planification des processus et l'ordonnancement de la production soient étroitement liés, nous remarquons souvent que ces deux fonctions sont exécutées séparément. La séparation de la planification des processus et de l'ordonnancement de la production entraîne de nombreux problèmes dus à des objectifs contradictoires ou à l'incapacité de communiquer les changements dynamiques dans l'atelier de production. Il en résulte un système de production qui manque de flexibilité et d'adaptabilité (Nasr et Elsayed 1990).

Dans cette étude, nous mettons l'accent sur l'affectation des machines et la planification des jobs dans un système intégré pour surmonter ces problèmes. De plus, nous rendons le système plus flexible en permettant des routages et des séquences alternatifs pour les opérations et les jobs respectivement.

En effet, ces problèmes se composent de trois grands processus :

Affectation : C'est le processus d'affectation d'une machine à chaque job à exécuter.
Séquencement : C'est le processus d'attribution d'un ordre d'exécution des jobs sur chaque machine.
Ordonnancement : C'est le processus d'attribution d'une date de début et de fin à chaque job.

Cependant, ces trois processus peuvent être traités comme des problèmes distincts, connus sous le nom de problème d'affectation, de problème de séquençage et de problème d'ordonnancement respectivement (French, 1982).

2. Concept associés

2.1. Planification du processus 'Process Planning'

La planification des processus est définie comme "la fonction qui établit la séquence de processus de production à utiliser pour convertir une pièce de sa forme initiale en une pièce de qualité, où la séquence du processus comprend la description du processus, les paramètres du processus et la sélection de l'équipement/l'outil et/ou de la machine-outil" Il s'agit de " la détermination systématique des méthodes par lesquelles un produit est fabriqué de manière économique et compétitive " (Sobayko 2018).

Dans le cadre d'un plan de processus, les ressources de production doivent être affectées aux opérations sélectionnées. Ensuite, les opérations doivent être planifiées sur les ressources allouées afin d'optimiser le critère de performance considéré. La planification de processus

comprend également les contraintes de précédence technologique entre les opérations et le routage au cours du processus de fabrication (Khoshnevis et Chen 1991, Park et Choi 2006). Les systèmes de planification des processus assistée par ordinateur (CAPP) jouent un rôle clé dans l'intégration des systèmes de conception et de fabrication en fonction des ressources disponibles et contraintes de conception (Ahmad et al. 2001).

Les diverses activités qui sont exécutées par les fonctions de planification des processus sont montré dans la figure 1.1 (Alting et Zhang 1989 ; Chryssolouris 2005).

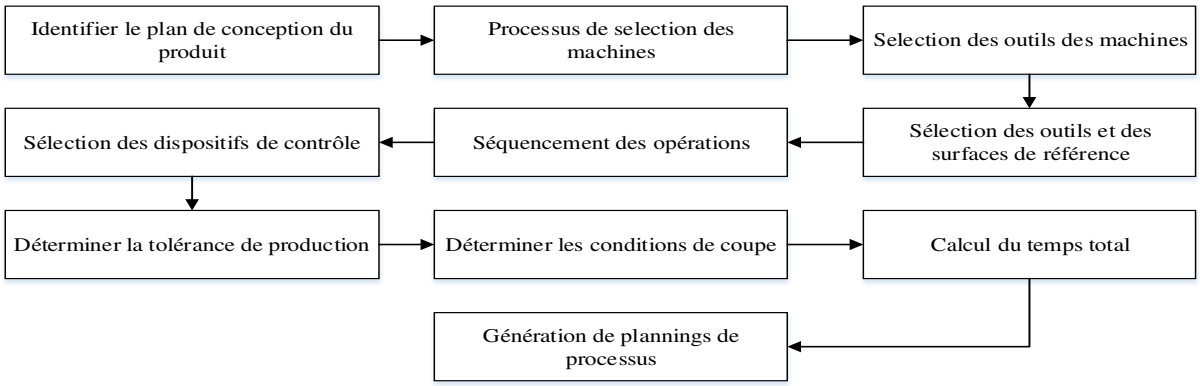


Figure. 1.1. Les activités de la planification des processus (Alting et Zhang 1989)

2.1.1. Plan de processus flexible

Le résultat de la planification du processus, est ce qu'on appelle un plan de processus ; est l'information requise pour transformer de la matière première ou produit semi-fini en un produit final (Li 2013).

Le plan de processus de chaque job, qui comprend l'identification de machines et d'outils de coupe, ainsi que de dispositifs de fixation et même les outils de mesure pour tous les processus, est reçu par le système d'ordonnancement en tant qu'entrée, puis les opérations sont assignées à des machines spécifiées pendant des moments appropriés lorsque les relations de précédence dans les plans de processus sont satisfaites. Autrement dit, un plan de processus spécifie les ressources de production et les opérations/routages nécessaires pour fabriquer un produit (un job/une tâche).

Typiquement, un job peut avoir un ou plusieurs plans de processus alternatifs. Avec les jobs comme inputs, une tâche d'ordonnancement consiste à ordonnancer les opérations de tous les jobs sur les machines pendant que les relations de précédence dans les plans de processus sont satisfaites.

En raison de la flexibilité de la production, il est possible de générer de nombreux plans de processus réalisables pour chaque job. Le plan de processus choisi est celui qui a été optimisé pour l'application mais ce n'est peut-être pas le meilleur. Il existe trois types de flexibilité dans la planification du processus : la flexibilité opérationnelle, la flexibilité de séquencement et la flexibilité de traitement (Benjaafar et Ramakrishnan 1996).

- La flexibilité des opérations (OF Operation Flexibility) également appelée flexibilité de routage, concerne la possibilité d'effectuer une opération sur d'autres machines, avec

éventuellement des temps de traitement et des coûts distincts. Ce type est souvent appelé flexibilité de routage (Lin et al 1991).

- La flexibilité de séquencement (SF Sequencing Flexibility) correspond à la possibilité de changer l'ordre dans lequel les opérations de production requises sont effectuées.
- La flexibilité de traitement (PF processing flexibility) est déterminée par la possibilité de produire la même caractéristique de production avec opérations alternatives ou séquences d'opérations.

Le premier et le deuxième type concernent respectivement des machines alternatives et des séquences alternatives, mais les opérations à effectuer sont fixées. En tenant compte de ces flexibilités, on peut obtenir de meilleures performances en termes de temps d'écoulement moyen (mean flow time), de débit (throughput) et utilisation de la machine (Lin et al 1991).
Il existe de nombreuses méthodes utilisées pour décrire ces types de flexibilité tels que les réseaux de Petri et les graphiques ET/OU (And/Or). Ici, nous adoptons les graphiques ET/OU proposée par (Ho et Moodie 1996) pour représenter les plans de processus alternatifs et les ordonnancements.

Il y a trois types de nœuds dans le réseau : le nœud de départ, nœud intermédiaire et nœud final (voir Figure 1.2) :

- Un nœud de départ et un nœud d'arrivée sont fictifs, et, respectivement, indiquent le début et l'achèvement du processus de fabrication d'un job.
- Un nœud intermédiaire représente une opération. Il contient les machines alternatives qui peuvent effectuer l'opération et les temps de traitement nécessaires à l'opération en fonction des machines.
- Les flèches qui relient les nœuds représentent la précédence entre eux. Les relations 'OR' sont utilisées afin de décrire la flexibilité de traitement qui permet de compléter la même caractéristique de production avec des procédures d'opération différentes. Si les liens qui suivent un nœud sont reliés par un symbole 'OR' (appelé connecteur OR), nous n'avons besoin de franchir qu'un seul des liens OR (les liens reliés par un connecteur OR sont appelés des liens OR). Un parcours d'opération qui commence avec une liaison OR et se termine lorsqu'il fusionne avec les autres parcours s'appelle un parcours de liaison OR. La fin du chemin de la liaison OR est également indiquée par un connecteur OR. Pour les liens qui ne sont pas connectés par des connecteurs OR, tous devraient être visités.

Dans le réseau de la Figure 1.3, les chemins {12} et {13 ; 14} sont deux chemins de liaison OR. Un chemin de liaison OR peut bien sûr contenir les autres chemins de liaison OR, par exemple les chemins {6 ; 7} et {8}. Plus de détails peuvent être trouvés dans (Ho and Moodie 1996).

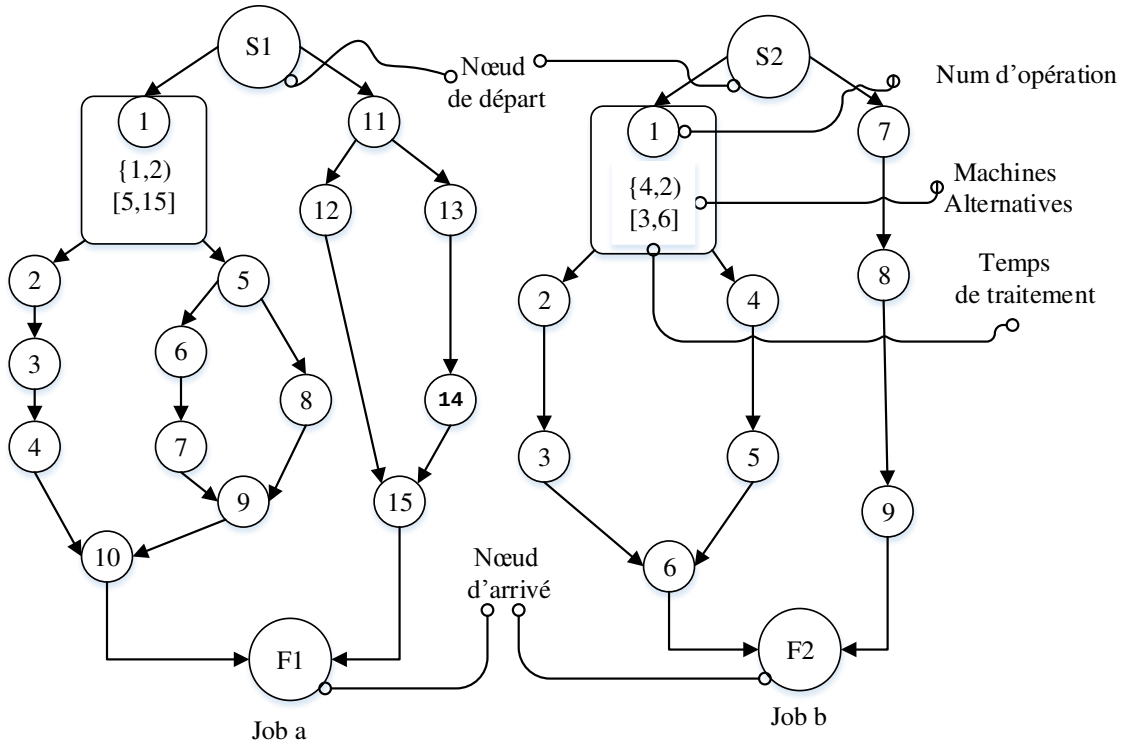


Figure 1.2. Le réseau du plan de processus flexible

2.2. Ordonnancement

L'intérêt principal des ateliers de production est d'amplifier la productivité tout en augmentant la flexibilité et la réactivité. L'ordonnancement est une autre fonction décisionnelle importante qui aide à améliorer la productivité du système de production. C'est un processus d'assigner un ensemble de tâches/jobs aux ressources/machines sur une période donnée (Pinedo 2012) indiqué dans les plans de processus afin de satisfaire à la mesure de performance ou les objectifs d'ordonnancement tels que le temps de réalisation des taches, le débit, le retard des pièces, une utilisation équilibrée des machines, la date d'échéance ou l'inventaire.

2.2.1. Éléments du problème d'ordonnancement

Un problème d'ordonnancement est un problème typique représentatif d'une classe de problèmes d'optimisation combinatoire composée d'un ensemble de tâche *O*, d'un ensemble de ressources *M*, d'un ensemble d'objectifs f et d'un ensemble de contraintes *C*.

2.2.1.1. Les tâches

Une tâche (task en anglais) est une entité élémentaire localisée dans le temps par une date de début s_i (start time) et de fin c_i (completion time), dont la réalisation nécessite une certaine durée p_i, et qui mobilise certaines ressources. En ordonnancement d'atelier, on exprime aussi la tâche par l'opération. On notera O_i une opération *i*. Dans notre cas de job shop/ job shop flexible, chaque job J se compose d'un nombre d'opérations *n*. Dans la phase de planification, le planificateur s'intéressera au temps requis pour exécuter l'opération sur une machine donnée, la quantité et le type de machine que cette opération utilise. Cependant

que l'ordonnanceur va lui s'intéresser au procédé de fabrication de l'opération. Généralement trois paramètres déterminent une tâche : (Bil 2006)

- Le temps d'exécution p_i (processing time) : temps de traitement.
- La date de disponibilité r_i (release time) : la date de début au plus tôt (date de disponibilité).
- La date d'échéance d_i (due date) : la date de fin au plus tard (date d'échéance).

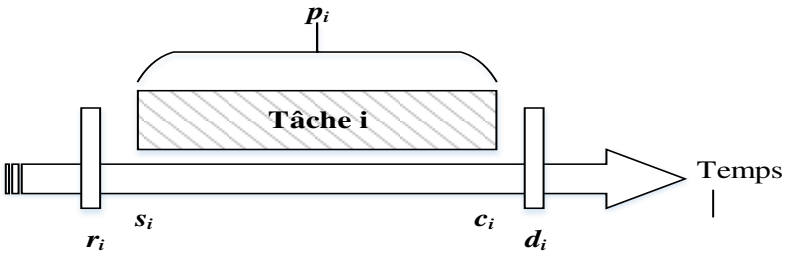


Figure 1.3. Caractéristiques d'une tâche i

2.2.1.2. Les ressources

La ressource est un moyen technique ou humain destiné pour l'exécution d'une tâche, en quantité limitée durant des intervalles de disponibilité. Les tâches sont en concurrence pour consommer une ou plusieurs ressources. Plusieurs types de ressources sont à distinguer.

- Une ressource est renouvelable si après avoir été assignée à une ou plusieurs tâches, elle est à nouveau disponible avec la même capacité (les machines, les hommes, l'équipement en général).
- Par contre, elle est consommable (matières premières, budget) : si après avoir été assignée à une ou plusieurs tâches, elle est disponible avec une capacité inférieure ou nulle.
- On distingue par ailleurs les ressources disjonctives (ou non partageables) qui ne permettent d'exécuter qu'une seule tâche à la fois.
- Certaines ressources peuvent être utilisées par plusieurs tâches simultanément, ce type concerne les ressources cumulatives (ou partageables)

En ordonnancement d'atelier, le terme machine est généralement utilisé à la place du terme ressource. Dans le cadre de cette étude, nous ne prendrons en compte que le cas de ressources renouvelables disjonctives. Une machine k sera notée M_k. L'indice k sera utilisé exclusivement pour les machines. Le nombre de machines dans un problème donné sera noté m.

2.2.1.3. Les contraintes

Une contrainte représente des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision impliquées dans le problème. La solution d'un problème d'ordonnancement doit toujours satisfaire aux contraintes données. Une solution qui contrevient aux contraintes n'est pas une solution réalisable (Lopez et Esquirol 1999).

- **Les contraintes temporelles**

Les contraintes temporelles sont généralement liées à la fenêtre d'exécution d'un job (ou d'opérations) à l'horizon temporel. Elles se répartissent en deux ensembles : les contraintes de temps alloué et les contraintes de cohérence technologique.

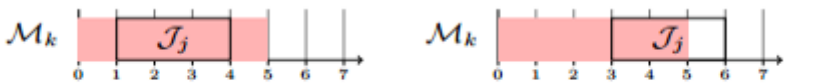
- ✓ Les contraintes de temps alloué reflètent la non disponibilité continue des ressources. Elles sont issues d'impératifs de gestion et relatives aux dates limites des tâches ou du projet (délais de livraisons, disponibilité des approvisionnements). Une opération peut également posséder une date de fin impérative (ou date due), noté d_i à laquelle l'opération doit être achevée. Un ordonnancement réalisable doit donc satisfaire les contraintes suivantes :

$$\forall i,\ r_i \leq s_i\ c \leq d_i$$

- ✓ Les contraintes de cohérence technologique, ou contraintes de gammes, modélisent les relations de précédence entre les différentes tâches : une opération O_j ne peut s'exécuter qu'après une opération O_i . Une telle contrainte s'exprime donc ainsi :

$$c_i \leq s_j$$

La figure 1.4 illustre l'exécution d'une tâche/job Ji avec temps r_i=*0 et* d_i=5 avec deux types d'ordonnancement, le premier réalisable (Figure 1.4(a)) par rapport à cette contrainte, alors qu'un ordonnancement irréalisable est illustré dans (Figure 1.4(b)).



a. Ordonnancement réalisable b. Ordonnancement irréalisable

Figure 1.4. Illustration des deux types d'ordonnancement selon la contrainte temporelle.

- ➢ **Les contraintes de ressources**

Les caractéristiques des ressources peuvent induire des contraintes indiquant les conditions de passage des tâches tout au long de leur exécution (Baptiste 1998). Ces ressources sont disponibles en quantité limitée (contraintes d'utilisation de machine, d'équipe d'ouvriers, etc.). On parle généralement de contraintes de partage. Dans notre cas de ressource disjonctive renouvelable, ce type de contraintes exprime la limitation de sa capacité. Soit m_i la machine utilisée par l'opération O_i, on peut alors exprimer cette contrainte ainsi :

$$\forall (i, j),\ m_i = m_j \Rightarrow c_i \leq s_j\ ou\ c_j \leq s_i$$

2.2.1.4. Les objectifs

Un objectif reflète les caractéristiques souhaitées dans la solution à trouver, pour un problème d'ordonnancement en exécutant un ensemble donné de tâches sur les machines dans un certain ordre.

2.2.1.4.1. Satisfaction de contraintes

Nous sommes tous désireux de trouver une solution à un problème d'ordonnancement. Tout d'abord il faut trouver une solution satisfaisant toutes les contraintes autrement dit, une solution réalisable et faisable. Les contraintes expriment par exemple, les dates de fin impératives ($\tilde{d}_i$) ne peuvent être prises en compte clairement, et peuvent même engendrer des

problèmes sans solution. Tandis que, certaines contraintes ne peuvent être prises en compte facilement lors de la construction d'un ordonnancement.

2.2.1.4.2. Optimisation d'un objectif

L'objectif de l'ordonnancement est d'optimiser (maximiser ou minimiser) une fonction d'évaluation en respectant un certain nombre de contraintes. Dans le cas d'un problème de planification d'un seul objectif, le but est alors de trouver un ordonnancement réalisable de meilleure qualité possible. Cet objectif reflète le type de solution recherché. La majorité de la documentation sur l'ordonnancement aborde les problèmes avec un seul objectif. Il existe un grand nombre d'objectifs. Le tableau 1.1 décrit les principaux objectifs de la littérature, ainsi que leurs caractéristiques.

Nom	Notation	Formule	Régulier	Type
Makespan	C_{max}	$\max_i C_i$	oui	minmax
Retard algébrique maximum	L_{max}	$\max_i L_i$	oui	minmax
Retard maximum	T_{max}	$\max_i T_i$	oui	minmax
Retard total	$\sum T_i$	$\sum_i T_i$	oui	minsum
Nombre d'opérations en retard	$\sum U_i$	$\sum_i U_i$	oui	minsum
Retard total absolu	$\sum \lvert L_i \rvert$	$\sum_i \lvert L_i \rvert$	non	minsum
Charge totale des machines	W_T			
Charge de travail des machines critiques	W_M			
Objectif régulier quelconque	f	$f\ (C_1, \dots, C_n)$	oui	-
Objectif régulier quelconque de type minmax	f_{max}	$\max_i f_i\ (C_i)$	oui	minmax
Objectif régulier quelconque de type minsum	$\sum f_i$	$\sum_i f_i(C_i)$□	oui	minsum

Tableau 1.1 : Quelques fonctions objectifs pour l'ordonnancement

Les objectifs sont classés selon deux caractéristiques principales, ces caractéristiques permettent de déduire des propriétés sur les ordonnancements recherchés :

- La régularité de l'objectif : S'il s'agit d'une fonction monotone des dates de fin des opérations (C_i).
- Le type d'opérateur : Somme, maximum, minimum… utilisé par la fonction.

Ainsi, il est possible de catégoriser les objectifs en fonction d'un ordonnancement donné (Lopez et Esquirol 1999) :

- Les objectifs liés au temps : on trouve par exemple, la minimisation du temps total d'exécution, des durées totales de réglage ou des retards par rapport aux dates de livraison, du temps moyen d'achèvement.

- Les objectifs liés aux ressources : par exemple, maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour exécuter un ensemble de tâches.
- Les objectifs liés au coût : ces objectifs sont généralement de minimiser les coûts de production, de stockage, de transport, etc.

2.2.1.4.3. Optimisation multiobjectif

En général, les problèmes d'ordonnancement à objectif unique font la majeure partie dans la littérature du domaine de l'ordonnancement de machines. Cependant, il est souvent souhaitable d'avoir des préférences multiples pour une solution trouvée. Ces préférences peuvent conduire à des objectifs contradictoires. Il est nécessaire de trouver une solution de compromis qui satisfasse simultanément tous les objectifs fixés, bien que cela ne soit pas optimal par rapport à un seul objectif. L'optimisation multi-objectifs (MO) est le processus d'optimisation simultanée de deux ou plusieurs d'objectifs sous réserve de certaines contraintes.

La considération de multiples objectifs pose un problème ; comment savoir si une solution est meilleure qu'une autre ? Pour pallier ce problème, on définit la notion de dominance entre un ensemble de solutions. De manière analogue, un sous ensemble de solutions est dit dominant, s'il possède au moins un optimum d'un objectif donné.

Ainsi, pour un ensemble de solutions donné, on peut séparer les solutions dominées des solutions non dominées. Les solutions non dominées sont les solutions intéressantes du point de vue des objectifs considérés. La figure 1.5 illustre graphiquement, dans un espace de deux objectifs, des solutions dominées et non dominées.

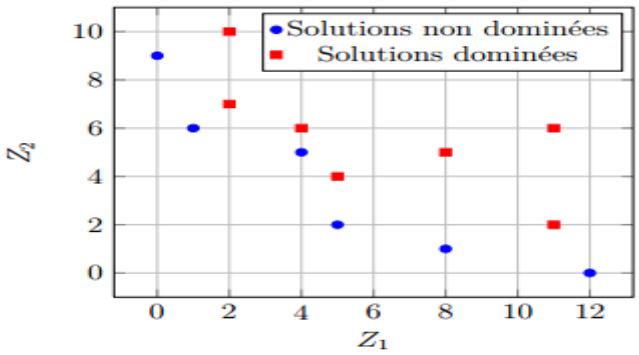


Figure 1.5. Solutions dominées et solutions non dominées dans un espace de deux objectifs à minimiser

2.2.1.5. Les classes d'ordonnancement

L'espace de l'ordonnancement peut être divisé en sous-classes en fonction de la manière dont les différentes opérations sont exécutées sur l'horizon temporel. Pour cela, il est impératif d'exploiter le temps au maximum afin de générer des solutions.

2.2.1.5.1. Ordonnancement semi-actif

Ce type d'ordonnancement est aligné à gauche où aucune opération ne peut être lancée plus tôt sans modifier les séquences de traitement. Dans un tel ordonnancement, la seule façon d'améliorer le makespan est de réorganiser la séquence d'opérations sur les machines. Sur le diagramme de Gantt, une telle situation peut être vue comme un ordonnancement où aucun glissement d'un bloc à sa gauche n'est possible.

2.2.1.5.2. Ordonnancement actif

Un ordonnancement réalisable est actif si aucune opération ne peut être traitée plus tôt sans retarder la date de début d'une autre opération. Cela signifie que l'ordonnancement est calé vers la gauche et il n'y a pas assez d'espace disponible entre deux jobs où un autre job peut être inséré sans violer des contraintes. Un ordonnancement actif est également un ordonnancement semi-actif.

2.2.1.5.3. Ordonnancement sans délai

Un ordonnancement sans délai est un ordonnancement où une machine n'est jamais laissée inactive alors qu'elle pourrait l'être. Cela signifie que s'il y a une opération dans la file d'attente d'une machine, la machine exécute une opération. Les ordonnancements à règles de priorité génèrent généralement de tels ordonnancements. Un ordonnancement sans délai est également un ordonnancement actif.

La figure 1.6 présente les relations entre les classes d'ordonnancements. Les ordonnancements sans délai sont inclus dans les ordonnancements actifs qui sont inclus dans les ordonnancements semi-actifs et qui sont tous réalisables.

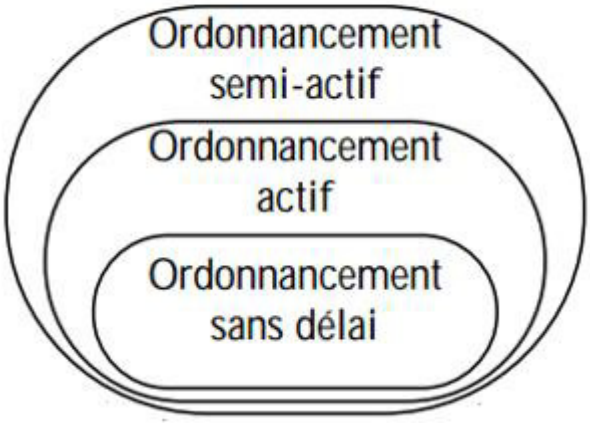


Figure 1.6. Les relations d'inclusion entre les classes d'ordonnancement

Considérons un problème à une machine et à trois opérations représenté dans la figure 1.7 :

- L'ordonnancement (a) est un ordonnancement semi-actif car toutes les opérations sont exécutées au plus tôt. Il n'est pas actif car on peut déplacer l'opération J_1 au début de l'ordonnancement sans retarder aucune opération. Comme il n'est pas actif, il n'est pas sans délai.
- L'ordonnancement (b) est un ordonnancement actif car toutes les opérations sont exécutées au plus tôt et on ne peut avancer aucune opération sans en retarder une autre. Il n'est pas sans délai car, à $t = 0$, l'opération J_1 peut être exécutée.

- L'ordonnancement (c) est un ordonnancement sans délai car, à chaque instant où une opération peut être exécutée, une opération est exécutée.

i	p_i	r_i
1	1	0
2	3	1
3	2	2

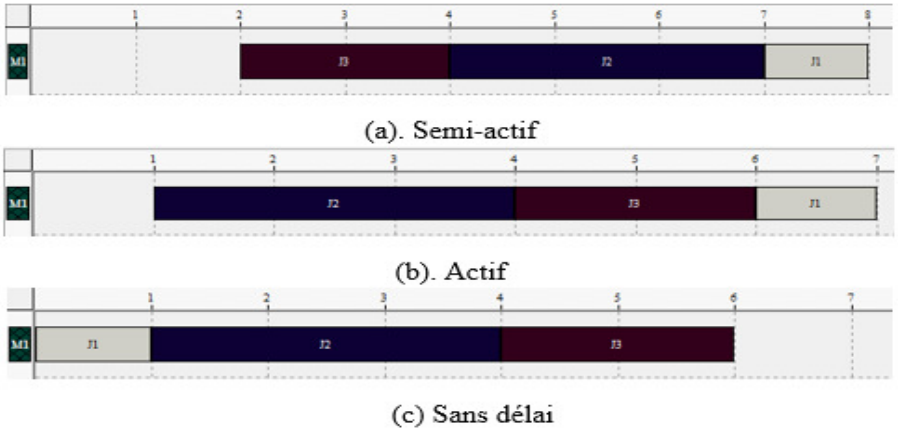


Figure 1.7. Exemples d'ordonnancements semi-actif, actif et sans délai

2.2.2. Classification des problèmes d'ordonnancement

Cette classification permet d'identifier un problème d'ordonnancement, et ce, sans ambiguïté. La notation utilisée fut d'abord proposée par (Ronald et al 1979) puis étendue par (Bru 1998). Un problème se décrit par trois champs de la manière suivante : α|β |γ.

2.2.2.1. Le champ α

Le champ α décrit l'environnement de machines. Il se compose de deux sous champs concaténés α_1 et α_2, où α_1 est le type d'atelier et α_2 est le nombre de machines dans le problème et il est facultatif. Si le α_2 est vide, le nombre de machines est une variable du problème.

Tout d'abord, les problèmes d'ordonnancements sont classés suivant le parcours des produits sur les différentes machines, c'est-à-dire suivant les caractéristiques des contraintes de précédences entre opérations. Le tableau suivant 1.2 donne une brève description des ateliers de production. Ensuite, les principales notations du champ α1 sont énumérées dans le tableau 1.3.

2.2.2.2.Le champ β

Le champ β permet de décrire des contraintes additionnelles au problème. Il est constitué d'une suite de sous champs séparés par des virgules. Les principaux sous champs sont décrits dans le tableau 1.4

2.2.2.3.Le champ γ

Le champ γ décrit l'objectif à optimiser. Le tableau 1.5 décrit les principales valeurs de ce champ.

Atelier	Description
Machine unique	L'atelier se compose d'une seule machine et toutes les pièces doivent être traitées sur cette machine. C'est le type de configuration d'atelier le plus simple.
Flow shop	Ateliers à cheminement unique, il contient le nombre "m" de machines en série, dans lesquelles toutes les pièces doivent être usinées par le même routage, bien qu'il ne soit pas nécessaire que toutes les pièces soient affectées à toutes les machines.
Job shop	Ateliers à cheminements multiples, il se compose de "n" nombre de pièces à usiner sur "m" nombre de machines et chaque pièce a son propre routage à traiter.
Open shop	Ateliers à cheminements libres, il contient un nombre "m" de machines et le routage des pièces à travers la machine n'est pas limité. Les pièces peuvent utiliser les machines dans n'importe quel ordre. Cependant, le planificateur est autorisé à déterminer un routage pour chaque pièce et chaque pièce peut avoir un routage différent.
Mixed-shop	Il se compose d'un nombre "m" de machines et certaines pièces ont leur propre routage (fixe) et d'autres pas.

Tableau 1.2. Description des configurations d'atelier

Variante	Notations	Description
α_1	Ø	Machine unique, la valeur du champ α est alors « 1 ».
	P	Machines parallèles identiques.
	Q	Machines parallèles proportionnelles.
	R	Machines parallèles non reliées.
	F	Flow shop.
	J	Job shop.
	O	Open shop.
	FH	Flow shop hybride.
	JG	Job shop généralisé.
	OG	Open shop généralisé.
	XAG	Problème de type X (avec X ∈ {P, Q, R, F, J, O}) avec affectation générale.
α_2	{}	Nombre de machines variable
	{m}	Problème à m Machines

Tableau 1.3. Principales valeurs du champ α_1 et α_2

Notations	Description
d_i	Les opérations ont des dates de fin souhaitées. Généralement, cet indicateur peut être omis car l'existence de telles dates se retrouve dans la fonction objectif utilisée, comme par exemple L_{max}
$\tilde{d}_i$	Les opérations ont des dates de fin impératives
pmtn	La préemption des opérations est possible
prec	Contraintes de précédences quelconques entre opérations
r_i	Les opérations ont des dates de début au plus tôt.
S_{sd} (R_{sd})	L'exécution d'une opération est précédée (resp. suivie) d'un temps de montage (resp. démontage) dépendant de la séquence des travaux jobs sur la machine. Ce temps mobilise uniquement la ressource.

Tableau 1.4. Principaux sous champs du champ β

$C_{max} = max_{i\in\{0..n\}} C_i$	Makespan : la plus grande date de fin
$\sum w_i C_i$	La date de fin pondérée des jobs
$T_{max} = max_{i\in\{0..n\}} (C_i - d_i , 0)$	Le retard maximum des jobs
$\sum w_i T_i$	Le retard pondéré des jobs
$\sum T_i$	Le retard total
$L_{max} = max_{i\in\{0..n\}} (C_i - d_i)$	Le retard algébrique maximum des travaux
$\sum w_i L_i$	Le retard algébrique pondéré des travaux
$\sum \lvert L_i \rvert$	Le retard total absolu
$\sum U_i = \lvert(J_i \mid \text{if } C_i > d_i)$	Le nombre de jobs en retard
-	L'objective est de trouver une solution admissible

Tableau 1.5. Principales notations du champ γ

2.2.3. L'activité principale de l'ordonnancement des jobs

D'un côté, les plans de production sont obtenus par les systèmes d'ordonnancement et des systèmes de planification supérieure et après divisés en niveau de l'opération de traitement pour construire le plan d'ordonnancement de l'atelier selon lequel les jobs au niveau d'atelier sont spécifiquement et raisonnablement affectées à chaque unité de production et les instructions de l'ordonnancement sont envoyées aux systèmes de contrôle de l'atelier.

D'un autre côté, le système d'ordonnancement obtient les informations de traitement depuis les systèmes de contrôle en temps réel, et se préoccupe des perturbations aléatoires exprimés par les incertitudes telles que le changement de l'ordre, pour adapter la planification de la production et considérer un ré-ordonnancement lorsque cela est nécessaire. De plus, il

offre des rétroactions des informations de progression d'exécution pour les systèmes de planification supérieurs afin de commander efficacement le traitement des jobs.

En théorie, l'ordonnancement des jobs doit satisfaire aux exigences suivantes :

- Garantir la livraison du produit ;
- Diminuer le temps d'attente pour les personnels et les équipements ;
- Optimiser le temps de traitement de la pièce pour être le plus court ;
- Réduire le nombre de produits en cours de traitement et le temps de stationnement ;
- Procéder un contrôle de la production.

3. La planification des processus et l'ordonnancement

La planification des processus est souvent exécutée avant l'ordonnancement. L'une des hypothèses courantes de la planification des processus est que les ressources ont une capacité infinie et qu'elles sont disponibles (Huang 1995). Cependant, la disponibilité des ressources change généralement avec le temps. Il y a souvent un délai entre la planification des processus et les phases d'ordonnancement qui aboutissent à des plans de processus irréalisables et qui nécessitent de modifier jusqu'à 30 % des plans de processus existants. (Zhang et al. 2010).

Cette méthode de séquencement ne tient pas compte de la flexibilité de planification des processus qui est essentielle pour le système de production et qui influence toujours directement les performances de l'ordonnancement. En outre, cela pourrait également apporter quelques d'autres problèmes, tels que les conflits objectifs entre l'ordonnancement et la planification des processus, déséquilibre de la production machines et l'impossibilité d'avoir un plan de processus après qu'il a été élaboré pour le système de production pour la production est de plus en plus dynamique par nature. La connexion entre les deux fonctions est maintenu par les opérations des jobs, qui est illustré dans la figure 1.8.

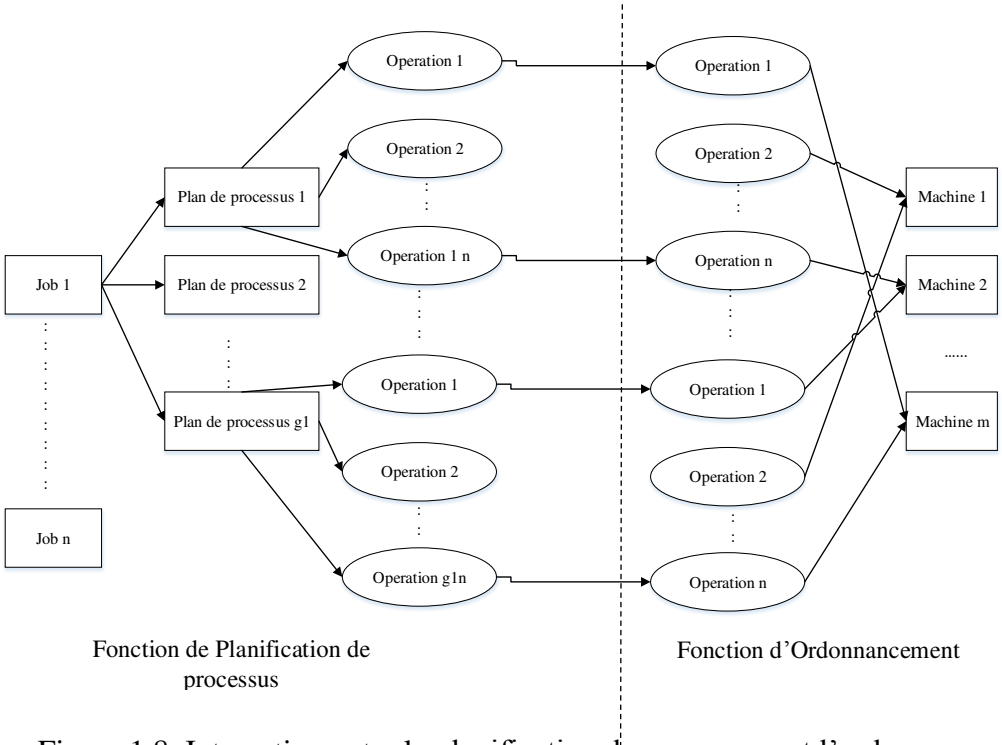


Figure 1.8. Interaction entre la planification des processus et l'ordonnancement

3.1. La relation entre la planification de processus et l'ordonnancement

Les deux fonctions de la planification des processus et de l'ordonnancement sont étroitement liées l'une à l'autre. L'optimisation de l'ordonnancement dépend du résultat de la planification des processus. L'intégration de la planification des processus et de l'ordonnancement est donc indispensable pour une exploration efficace des ressources de production (Kim 2003).

✓ Le partage de données existe entre la planification des processus et l'ordonnancement

Tout d'abord, le plan du processus doit être élaboré, il précise également les contraintes de précédence entre les opérations et le routage pendant le processus de production ; L'ordonnancement sera ensuite réalisé afin d'assigner les ressources de production aux opérations précisées selon le plan dans le but de maximiser ou de minimiser une fonction objectif désirée tout en satisfaisant les contraintes distinctes de production (Jain et al. 2006).

✓ Tous les deux impliquent l'allocation des ressources et leurs fonctions sont complémentaires.

En outre, tous les deux sont responsables de la répartition, l'identification et de l'utilisation efficace des ressources (Phanden et al. 2011). La planification du processus est indépendant du temps, alors que l'ordonnancement est considéré comme une activité dépendant du temps (Wu et al. 2002).

3.2. La Nécessité d'intégration la planification des processus et l'ordonnancement

L'environnement de production actuel est très différent de l'environnement traditionnel. Il se caractérise par des délais de livraison plus courts, des normes de qualité exigeantes, une plus grande variété de pièces et des coûts compétitifs. Dans un tel environnement de production, il est difficile d'obtenir un résultat satisfaisant en utilisant l'approche traditionnelle de la planification et de l'ordonnancement des processus en raison des facteurs suivants (Baykasoğlu et Özbakır, 2009 ; Li et al 2010 ; Phanden et al 2013).

- Le planificateur des processus supposent des ressources illimitées dans l'atelier et planifient le processus alternatif le plus recommandé (Usher et Fernandes 1996). Cela peut conduire les planificateurs à privilégier la sélection répétée des machines désirables. Par conséquent, les plans de processus générés sont quelque peu irréaliste et ne peut pas être facilement exécutée dans l'atelier. Par conséquent, les plans de processus optimaux qui en résultent sont souvent irréalisables lorsqu'ils sont mis en œuvre dans la pratique lors de la dernière étape.
- Même si, au cours de la phase de planification, les planificateurs de processus considèrent les ressources actuelles de l'atelier, les contraintes prises en compte dans la phase de planification du processus ont peut-être déjà beaucoup changé grâce au délai entre la phase de planification et la phase d'exécution. Ce qui peut mener à l'optimisation de la faisabilité du plan de processus. Les investigations ont montré que 20 à 30 % des plans de production totaux au cours d'une période donnée doivent être

modifiés pour s'adapter à l'évolution dynamique d'un environnement de production (Kumar et Rajotia 2003).

- Les plans d'ordonnancement sont souvent déterminés après les plans de processus. Dans la phase d'ordonnancement, les planificateurs de l'ordonnancement doivent prendre en compte les plans de processus déterminés. Des plans de processus fixes peuvent influencer les plans d'ordonnancement de se retrouver avec une charge de ressources très déséquilibrée et de créer des goulots d'étranglement superflus (bottleneck).
- Dans la plupart des cas, tant pour la planification des processus que pour l'ordonnancement, une technique d'optimisation à critère unique est utilisée pour déterminer la solution optimale (Kumar et Rajotia 2003). De plus, la planification et l'ordonnancement du processus peuvent avoir des objectifs contradictoires. La planification des processus met l'accent sur l'aspect technologique d'une tâche, tandis que l'ordonnancement implique les aspects temporels d'entre eux. S'il n'y a pas de coordination appropriée, cela peut créer des conflits.
- Les lacunes de l'approche traditionnelle peuvent être contournées par la considération d'une approche intégrée de la planification des processus et de l'ordonnancement. L'intégration des deux phases en un seul problème d'optimisation, en considérant les contraintes des deux domaines simultanément, peut théoriquement aboutir à une optimisation globale optimale mais cela augmente considérablement l'espace de la solution (Chryssolouris et Chen 1985; Sundaram et Fu 1988; Saygin et Kilic 1999; Lee et Kim 2001; Kumar et Rajotia 2003).
- Le développement des produits et la production seront étroitement intégrés conformément à la vision 4.0 de l'industrie. (Lasi et al. 2014). L'intégration de ces deux fonctions est donc essentielle pour parvenir à une manufacture incorporée et pour écarter l'approche manufacturière conventionnelle.

3.3. Intégration de planification des processus et ordonnancement IPPS

Pour surmonter les problèmes ci-dessus, la meilleure façon est de fusionner les fonctions de planification des processus et d'ordonnancement en un seul système 'IPPS'. L'IPPS est le concept permettant d'effectuer la planification des processus et l'ordonnancement de manière concourante avec les objectifs d'éliminer ou réduire les conflits d'ordonnancement, de réduire les temps d'écoulement et le travail dans le processus et d'améliorer l'utilisation des ressources (Lee et Kim, 2001; Wan et al, 2013).

Par l'intégration de ces deux fonctions, l'IPPS peut fournir de meilleurs plans des processus et l'ordonnancement que les systèmes de production traditionnels. Les capacités de l'IPPS sont d'augmenter la faisabilité de la production et l'optimalité en combinant à la fois des problèmes de la planification des processus et de l'ordonnancement (Wong et al 2006).

L'intégration entre les activités de ces deux fonctions s'effectue par le biais de deux modules de contrôle ; le générateur de plan de processus et l'ordonnanceur. Les activités à l'intérieur de chaque module se déroulent sur des périodes de temps différentes, comme le montre la figure 1.9 La génération du plan de processus est exécutée dès que la conception du produit est terminée. L'ordonnancement est effectué juste avant le début de la production.

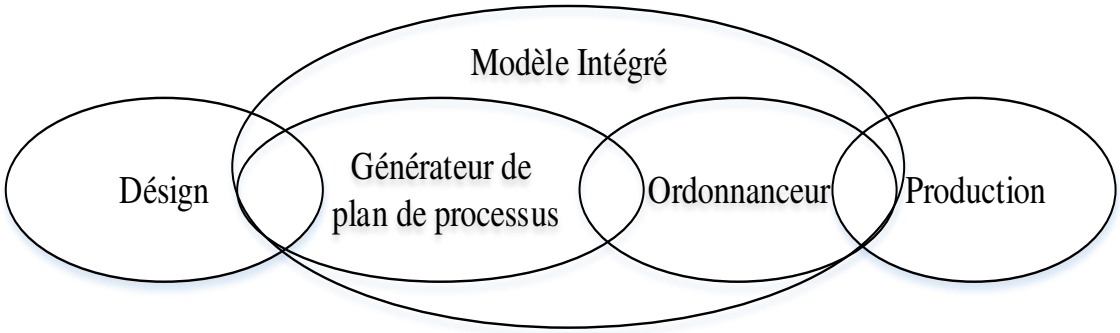


Figure 1.9. Domaine du modèle d'intégration

4. Formulation et analyse du problème d'intégration

Les systèmes de production peuvent avoir des configurations différentes et être de types différents selon les produits fabriqués (Pinedo 2008). Dans la pratique, l'environnement de job shop *JS* et job shop flexible *JSF* jouent un rôle important (Pinedo 2008, Scrich et al. 2004, Brandimarte 1993).

4.1. Problème d'ordonnancement de Job Shop PJS

Dans le domaine d'ordonnancement de la production, le problème d'ordonnancement job shop (*PJS*) est l'un des problèmes les plus importants. Un grand nombre de petites et moyennes entreprises sont toujours exploités dans des jobs shops (Chryssolouris 2005). De plus, l'analyse de l'ordonnancement dans un job shop fournit une vision de qualité de la solution d'ordonnancement que l'on rencontre dans de plus en plus de systèmes réalistes et compliqués (Kutanglu et Sabuncuoglu 1999).

Le flux de travail dans cet atelier n'est pas unidirectionnel, chaque machine de l'atelier peut être caractérisée par les flux d'entrée et de sortie du travail. Un job shop ne suppose qu'une seule machine disponible pour chaque opération et un plan de processus réalisable pour chaque job (séquence d'opération, par exemple, $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow ...$) (Qiu et Lau 2014). Cela implique qu'aucune flexibilité dans le plan de processus n'est prise en compte, l'opération O_{i+1} doit être programmée après O_i et avant O_{i+2} dans tous les cas. Un exemple d'un job shop classique est illustré dans le tableau 1.6.

La flexibilité des plans de processus alternatifs et les machines parallèles indépendantes sont avantageuses pour l'optimisation des problèmes d'ordonnancement de type job shop, mais entre-temps accroître la complexité du problème (Zhao 2015).

4.2. Problème d'ordonnancement de Job Shop Flexible PJSF

Le problème job shop flexible (*PJSF*) est une généralisation de job shop et de l'environnement machine parallèle, qui fournit une approximation plus proche d'une large gamme de systèmes de production réels. En particulier, il existe un ensemble de postes de travail dans un atelier flexible. Chaque poste de travail possède un ensemble des machines parallèles dont l'efficacité peut varier.

Selon (Pinedo 2008), nous considérons un environnement d'usinage composé d'un certain nombre de groupes de machines différents $M = \{M_1, \ldots, M_{|m|}\}$. Il existe deux types de groupes de machines : les groupes de machines identiques désignés par P_m, et les groupes de machines non associées désignés par R_m. Si les groupes de machines sont du type R_m, le temps de traitement d'une opération peut varier en fonction de la combinaison de la machine sélectionnée et de l'opération elle-même.

Job_i	O_{ij}	Machines (P_{ijk})	Plan de Processus (Flexibilité de séquence)
J_1	O_{11}	$M_1(6)$	1: (O_{11}- O_{12}- O_{13})
	O_{12}	$M_2(5)$	
	O_{13}	$M_3(4)$	
J_2	O_{21}	$M_1(3)$	1: (O_{21}- O_{22}- O_{23})
	O_{22}	$M_2(7)$	
	O_{23}	$M_3(6)$	
J_3	O_{31}	$M_1(7)$	1: (O_{31}- O_{32}- O_{33})
	O_{32}	$M_3(5)$	
	O_{33}	$M_2(4)$	

Tableau 1.6. Instance d'un PJS

L'ordonnancement des jobs dans le cadre du *PJSF* peut être catégorisé en deux sous-problèmes (Xia et Wu 2005) :

- Un sous-problème de routage, c'est-à-dire l'affectation de chaque opération à une machine à partir d'un ensemble de machines alternatives.
- Un sous-problème d'ordonnancement, qui consiste à séquencer les opérations affectées sur toutes les machines afin d'obtenir une planification possible optimisant une fonction objective prédéfinie.

Basé sur la flexibilité, Kacem et al (2002) ont classé ce problème en deux sous-problèmes :

- Total PJSF : Chaque opération peut être traitée sur n'importe quelle machine "M" de l'atelier.
- Partielle PJSF : Certaines opérations ne sont réalisables que sur une partie des machines "M" disponibles dans l'atelier.

En général, la taille du *PJS* et *PJSF* est donnée en m × n. Le *PJS* présenté dans le tableau précédent (1.6) peut se transformer en un *PJSF* illustré dans le tableau 1.7.

4.3. Définition du problème intégré IPPS

Le problème considéré ici est celui de *PJSF* généralisée avec des plans de processus alternatifs qui sont reconnus sur la base des relations de précédence entre les opérations (*FJSP-APP)*. Ce système est aussi appelé *JSP-MPP* (problème d'ordonnancement de job

shop avec les multiples planifications des processus). On trouve aussi l'*IPPS* 'planification des processus et ordonnancement intégré'. L'appellation *IPPS* est considérée dans cette thèse.

Job_i	**O_{ij}**	**Machines Alternatives (P_{ijk}) (Flexibilité de Routage)**	**Plan de Processus (Flexibilité de séquence)**
J_1	O_{11}	{$M_1(6)$, $M_2(6)$}	1: (O_{11}- O_{12}- O_{13})
	O_{12}	{$M_2(5)$, $M_1(6)$, $M_3(6)$}	
	O_{13}	{$M_3(4)$}	
J_2	O_{21}	{$M_1(3)$, $M_3(4)$}	1: (O_{21}- O_{23}- O_{22})
	O_{22}	{$M_2(7)$}	
	O_{23}	{$M_3(6)$, $M_1(6)$, $M_2(7)$}	
J_3	O_{31}	{$M_1(7)$, $M_3(8)$}	1: (O_{31}- O_{32}- O_{33})
	O_{32}	{$M_3(5)$, $M_1(5)$, $M_2(6)$}	
	O_{33}	{$M_2(4)$}	

Tableau 1.7. Une instance d'un PJSF

Le problème de l'*IPPS* peut être considéré comme (Kim et al. 2003) : étant donné un ensemble de *n* jobs *J= {1, 2,…, n}* qui doivent être exécutés sur *m* machines *M= {1, 2,.., m}*. L'objectif est de sélectionner un plan de processus alternatif pour chaque job.
Chaque job a des séquences d'opérations alternatives (flexibilité de séquence) possibles nécessitant des machines alternatives (flexibilité d'opérations) afin de minimiser le makepan ou toute autre fonction objective en sélectionnant un plan de processus approprié ainsi que les machines pour chaque job avec un ordonnancement complet qui satisfait toutes les contraintes de précédence. Le *PJS* et *PJSF* présentés dans les tableaux précédents (1.6 et 1.7 respectivement) peuvent se transformer en un *IPPS* illustré dans le tableau 1.8.

Dans la flexibilité d'opération (routage alternatif, flexibilité de machines (Hutchinson et Pughoeft 1994), il y a un choix de machines sur lesquelles effectuer les opérations. Ces machines sont flexibles et peuvent être identiques ou distinctes. En considérant le routage alternatif, les solutions possibles pour le problème d'ordonnancement deviennent très vastes.

D'autre part, dans la flexibilité de séquence (plan de processus alternatif), il y a un choix de différentes combinaisons de séquences, dans lesquelles les opérations d'un job peuvent être effectuées telles qu'elles satisfassent les contraintes de précédence.

Le but est de trouver une séquence d'opérations et la séquence de machines-outils correspondante pour chaque job. *IPPS* vise à éviter des plans de processus irréalisables en prenant des décisions de telle sorte que le résultat de l'exécution d'ordonnancement soit anticipé.
Alors, le problème *IPPS* peut être considéré comme un problème d'ordonnancement job shop flexible (FJSP) plus divers routages, et donc, le problème est un problème fort non déterministe de temps polynomial (NP) – difficile. La flexibilité des plans de processus

alternatifs et les machines indépendantes parallèles sont avantageuses pour l'optimisation du problème de job shop mais en même temps accroître la complexité du problème.

Job_i	O_{ij}	Machines Alternatives (P_{ijk}) (Flexibilité de Routage)	Plan de Processus (Flexibilité de séquence)
J_1	O_{11}	{M_1(6), M_2(6)}	1: (1-2-3) 2: (1-3-2)
	O_{12}	{M_2(5), M_1(6), M_3(6)}	
	O_{13}	{M_3(4)}	
J_2	O_{21}	{M_1(3), M_3(4)}	1: (4-5-6) 2: (4-6-5) 3: (6-4-5)
	O_{22}	{M_2(7)}	
	O_{23}	{M_3(6), M_1(6), M_2(7)}	
J_3	O_{31}	{M_1(7), M_3(8)}	1: (7-8-9) 2: (7-9-8)
	O_{32}	{M_3(5), M_1(5), M_2(6)}	
	O_{33}	{M_2(4)}	

Tableau 1.8. Une instance d'un IPPS

.

4.3.1. Exemple d'un IPPS

Pour mieux comprendre ce type de problème, le tableau 1.9 illustre un type d'atelier de fabrication de métaux, chaque job doit réaliser trois opérations : fraisage - perçage - alésage, avec flexibilité d'opération et de séquencement.

Operation	Temps de traitement sur les machines alternatives			Plans de Processus
	M1	M2	M3	
Fraisage (Milling)	3	–	4	
Perçage (Drilling)	3	1	–	Fraisage– Perçage – Alésage
Alésage (Boring)	–	5	2	Perçage – Fraisage – Alésage

Tableau 1.9. Exemple d'un IPPS

Tout d'abord, la flexibilité des opérations est indiquée. Nous pouvons voir la possibilité d'effectuer des opérations sur des machines alternatives avec des temps de traitement différents, par exemple, l'opération de fraisage peut être effectuée sur la machine M_1 ou la machine M_3 avec un temps de traitement de 3 et 4 respectivement. Ensuite, la flexibilité de séquencement est illustrée.

Différents plans de processus donnent la possibilité d'échanger la séquence entre les opérations du même job ; par exemple, le fraisage avant le perçage ou après. Cette flexibilité

de séquence est possible grâce aux propriétés du métal : il n'y a pas de contrainte de priorité entre les opérations de fraisage et de perçage du métal, alors que ces opérations sont des conditions préalables pour l'opération d'alésage.

4.4. Avantages de l'intégration

L'intégration des deux fonctions engendre plusieurs avantages comprenant :

1. La nature dynamique des systèmes de production réels qui est causée par plusieurs événements tels que la panne d'une machine, l'arrivée d'un nouveau job, l'annulation d'un job et l'ajout d'une nouvelle machine peut rendre un plan de processus infaisable. Par conséquent, il est nécessaire de modifier fréquemment les plans en raison du changement de statut de l'atelier. L'intégration de la planification des processus à l'ordonnancement réduit le délai entre la planification de la production et l'exécution du plan, ce qui permet de générer un plan plus réaliste sans avoir à le modifier fréquemment (Jain et al 2006).

2. La planification des processus met l'accent sur les exigences technologiques d'un job sans tenir compte de la concurrence qui existe entre les jobs pour les ressources, tandis que l'ordonnancement implique les aspects temporels et le partage des ressources de tous les jobs (Jain et al 2006). Par conséquent, ces fonctions peuvent avoir des objectifs différents et parfois contradictoires. Le problème IPPS offre la possibilité de prendre en compte simultanément ces critères contradictoires, ce qui peut entraîner une réduction des coûts de production, l'élimination des goulots d'étranglement, une utilisation équilibrée de la machine et une amélioration de la productivité des installations (Li et al 2010).

3. IPPS peut contribuer efficacement à la réalisation de concepts et de paradigmes de production modernes qui ont été développés pour améliorer la flexibilité, la productivité et l'agilité des systèmes de fabrication. Plusieurs études ont abordé la mise en œuvre de l'idée IPPS dans un cadre de fabrication moderne comme l'ingénierie simultanée (Morad et Zalzala 1999) holonic manufacturing system (Zhao et al 2010) CIM (Kim et Egbelu 1999) et système de production flexible (FMS).

5. Représentation graphique

5.1.Diagramme de Gantt

Le diagramme de Gantt développé tient son nom de son créateur Henry Gantt (1861-1919), est le moyen le plus simple et largement utilisé pour présenter graphiquement l'ordre d'exécution et les temps d'exécution des tâches sur les machines à travers l'horizon temporel. Ce diagramme peut être orienté machine ou job.

Dans cette étude, tous les diagrammes de Gantt seront axés sur la machine. On obtient le diagramme de la figure 1.10 si on suppose une solution semi-active pour l'exemple donné dans le tableau 1.6.

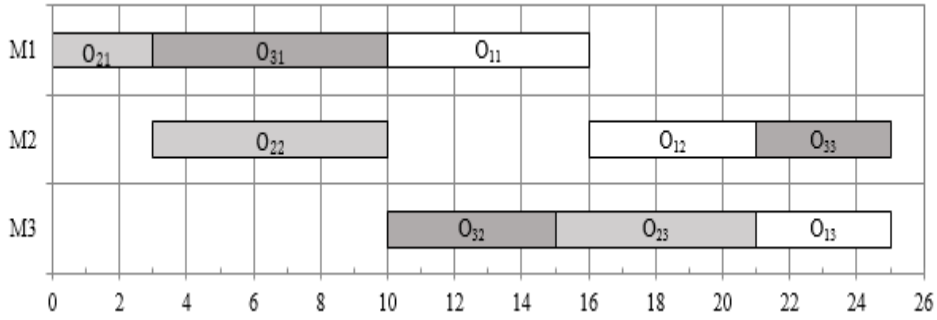


Figure 1.10. Le diagramme de gantt de la solution de l'instance JS

5.2.Modélisation par graphe disjonctif

Bien que le diagramme de Gantt soit un excellent outil de suivi d'un ordonnancement, il est incapable de représenter le problème lui-même.
La Représentation graphique disjonctive est en mesure de représenter efficacement un problème d'ordonnancement et en particulier les problèmes de Job Shop. Présentée pour la première fois par Roy et Vincke (1981), cette formulation a été à l'origine des premières méthodes de résolution du problème (Pen, 1994) (Jai, 1998).

- X : Ensemble des nœuds : chaque opération O (i, j) correspond à un nœud, avec deux nœuds modélisant deux opérations fictives. S (source : elle est réalisée avant toutes les autres opérations,) et P (puits : il est réalisé après toutes les autres opérations,) indiquant le début et la fin de l'ordonnancement.
- C : Ensemble des arcs conjonctifs (→) représentant les contraintes de précédence (conjonctifs) entre les opérations du même job (gammes opératoires) Chaque arc a un poids, égal au temps de traitement de l'opération correspondant au nœud à partir duquel l'arc émet.
- D : Ensemble d'arcs conjonctifs opposés dits arcs disjonctifs (↔), représente les contraintes de capacité des machines pour les opérations partageant une même machine.

Certaines notions bien évidentes sur ce graphe, sont importantes :

- Le chemin critique : Il est remarquable car il présente le chemin le plus long entre le nœud de départ S et le nœud puits P. De plus, il définit le makespan qui est égal à la somme des temps de traitement des opérations qui le compose.
- Opération critique : Elle se trouve sur le chemin critique. Dans un ordonnancement actif, cette opération provoque l'augmentation du makespan. En conséquence, elle ne doit subir aucun retard.
- Bloc critique : Il présente une suite d'opérations critiques faisant référence à la même machine. Cette notion de bloc a permis de définir des systèmes de voisinage très utile pour le Job-Shop (Grabowski et al. 1996).

Un exemple d'un graphe disjonctif est illustré dans la figure 1.11.

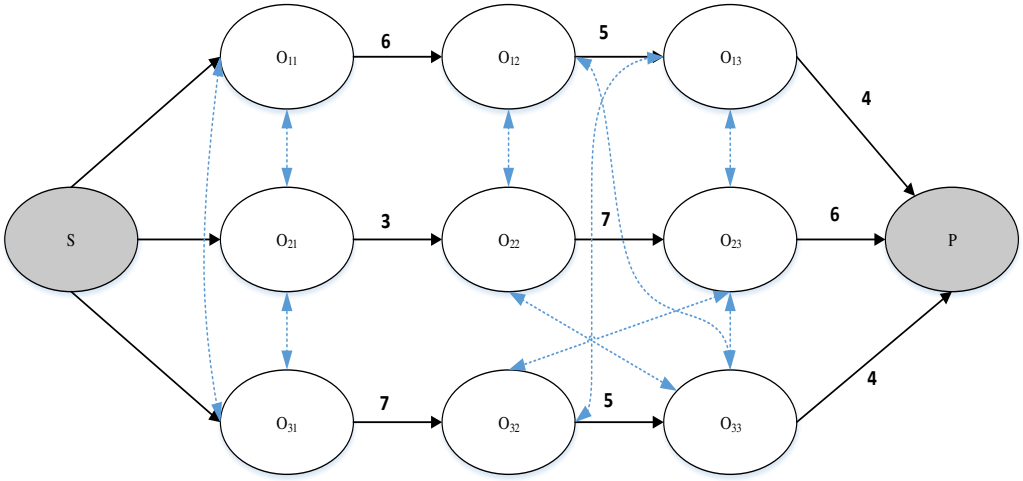


Figure 1.11 Le graphe disjonctif pour le problème job shop du tableau 1.6

6. Conclusion

Dans ce chapitre, les notions de base du domaine de l'intégration de la planification des processus et l'ordonnancement sont présentées. Plus précisément, le problème du job shop (JS), job shop flexible (JSF) et le problème d'intégration de la planification des processus et l'ordonnancement (IPPS) sont présentés avec ses variantes et ses techniques de représentation.

L'ordonnancement est l'une des questions les plus importantes dans les systèmes de production. Il établit le calendrier de l'allocation de ressources pour exécuter un ensemble de jobs. Il est limitée par un système de décision qui joue un rôle important dans la plupart des décisions du l'industrie manufacturière et les services.
Le problème de l'ordonnancement dans les JSF suscite beaucoup d'intérêt dans les études scientifiques en raison de sa grande importance dans la pratique. Compte tenu des ressources de fabrication limitées et de la concurrence féroce entre les différentes entreprises, il devient difficile d'utiliser les ressources de la meilleure façon possible et de respecter les critères de performance souhaités.

Le JS classique exige le séquencement des opérations sur des machines fixes, alors que dans le JSF, une opération peut être traitée sur un ensemble de machines capables. C'est pourquoi, dans le cadre du JSF, nous ne nous occupons pas seulement du séquencement, mais également de l'affectation des opérations aux machines appropriées (routage).
Le JSF est donc plus complexe que le JS, comme il considère la détermination de l'affectation des machines pour chaque opération. Dans ce domaine, il a été prouvé être un problème NP difficile

D'autre part, les plans de processus fixes conduisent souvent à des plannings qui aboutissent à des chargements de ressources très déséquilibrés, créent des goulots d'étranglement superflus et conduisent à une utilisation globale plus faible des ressources et à de mauvaises performances de livraison à temps. La séparation de la planification et de l'ordonnancement des processus introduit également des chevauchements ou des duplications partielles dans les efforts de solution. Par conséquence, le plan de processus d'origine doit être

modifié pour s'adapter aux changements dans l'atelier. Comme le JSP et JSFP sont NP-hard, le problème IPPS est un problème NP-hard.

Dans un second temps, nous avons présenté la caractérisation du problème d'ordonnancement d'atelier en dévoilant sa définition, et spécifiant notamment les différents éléments, ainsi que les approches de classification qui se basent sur différents paramètres portant sur différentes aspects du problème.

Dans la fin, nous avons abordé une modélisation sous forme de graphe disjonctif car elle accorde un bon compromis entre facilité d'utilisation et intégration des critères nécessaires â la suite de notre étude.

Dans le chapitre suivant, nous présentons un examen des classes de méthodes de résolution, qui ont beaucoup attiré les chercheurs en raison de leur utilisation étendue dans les systèmes en temps réel et de leur facilité de mise en œuvre.

CHAPITRE 2
Techniques d'optimisation

Ce chapitre introduit les méthodes de résolution utilisées tout au long de ce mémoire ainsi que les définitions principales. Certaines approches d'ordonnancement connues dans la littérature sont également abordées. Pour tenter de résoudre de manière approchée un problème NP — complet, il existe plusieurs méthodes possibles, méthodes de nature radicalement différente.

La première section concerne les méthodes exactes qui tentent de trouver des solutions optimales aux problèmes d'optimisation. Ensuite, la deuxième section présente les méthodes d'approximation qui sont utilisées comme solution de rechange.

1. Introduction

Au cours des dernières décennies, des recherches approfondies ont été menées pour résoudre le problème d'ordonnancement des ateliers job shop (PJS)/ job shop flexible (PJSF). Les chercheurs ont utilisé diverses approches pour tenter d'intégrer les fonctions de planification des processus et d'ordonnancement.

Diverses techniques, allant des méthodes exactes aux techniques hybrides, ont été utilisées dans cette recherche. Une grande partie de la littérature détaille la pléthore de méthodes qui peuvent ou qui ont été appliquées pour résoudre le problème d'intégration.

Ces méthodes peuvent être classées en trois grandes catégories de méthodes exactes, heuristiques et métaheuristiques. Un aperçu des différentes méthodes est présenté dans cette section.

2. Méthodes de résolution

L'ordonnancement de job shop/job shop flexible est bien connu comme un problème d'optimisation combinatoire NP-hard le plus difficile à résoudre (Mitchell 2002).

Différentes approches existent dans la littérature pour la résolution de ce problème, telles que la programmation dynamique 'Dynamic Programing', les règles de priorités 'Dispatching Rules', la méthode de séparation et évaluation 'branch and bound', et shifting bottleneck heuristique (Bagchi, 1999).

Un vaste champ de recherche, qui fait actuellement l'objet de nombreuses études approfondies, concerne l'utilisation des métaheuristiques comme recuit simulé 'Simulated Annealing', la recherche tabou ''Tabu-Search' optimisation par les essaims partuculaires 'Particle Swarm Optimization', les colonies de fourmis 'Ant Colony Optimisation', et les algorithmes génétiques 'Genetic Algorithm' (Yousf et al 2011).
La figure suivante (2.1) présente les différentes classes de ces méthodes de résolution.

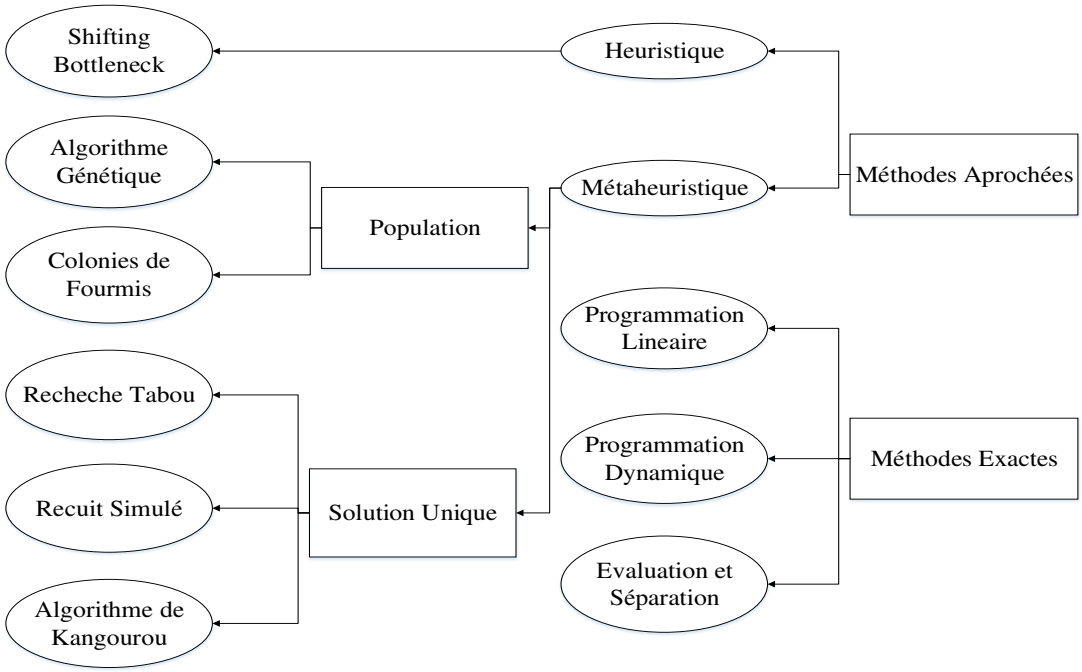


Figure 2.1 Classifications des méthodes de résolution

2.1 Méthodes de résolution exactes

Les méthodes exactes sont garanties pour trouver une solution optimale d'un problème d'optimisation combinatoire de taille finie, grâce à une exploration intelligente de l'espace des solutions dans un temps limité. Cependant, pour un problème typique d'optimisation combinatoire comme JS/JSF, il n'existe aucun algorithme pour résoudre ces problèmes en temps polynomial (Conway et al. 1967 ; French 1982).

Les méthodes exactes perdent leur efficacité dès que la taille des problèmes devient importante. Donc, pour résoudre les problèmes de moyennes et grandes tailles, il est indispensable de développer d'autres approches.
Les méthodes les plus couramment utilisées sont exposées dans la suite.

2.1.1. Les méthodes par séparation et évaluation (SEP / B&B 'Branch and Bound')

La majorité des chercheurs dans le domaine de l'ordonnancement de job shop et job shop flexible utilise la méthode de Séparation et évaluation "Branch-and-Bound" comme méthode de résolution exacte. Elle est aussi appelée méthode arborescente. Cette méthode consiste en une évaluation partielle et progressive des solutions du problème. Elles répartissent l'espace des solutions en sous-ensembles de moins en moins petits, la plupart étant écartés par des calculs de bornes.

Ces méthodes sont très sensibles aux instances individuelles et aux limites initiales utilisées. Par conséquent, ils sont considérés inadéquats pour des grandes instances. Par contre, elles peuvent pallier le manque d'algorithmes polynomiaux pour des problèmes de taille moyenne.

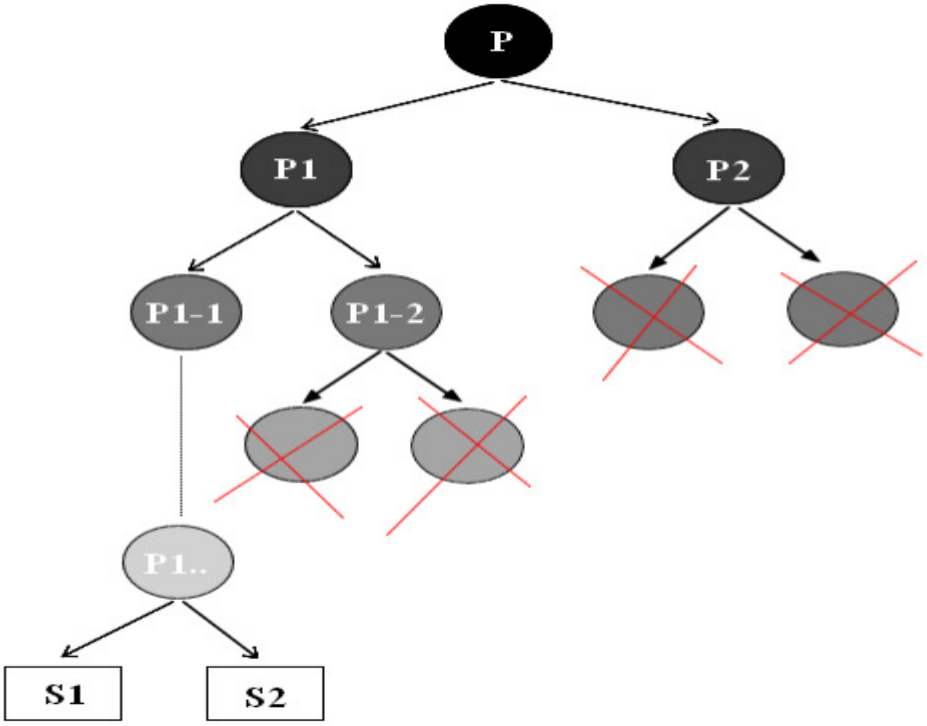


Figure 2.2 Représentation de la méthode séparation et évaluation

2.1.2. La programmation dynamique

La programmation dynamique est une technique mathématique développée par (Bellman 1957). Cette technique aide à prendre des décisions successives indépendantes basées essentiellement sur la décomposition des problèmes avec contraintes dont la fonction objectif est résolu localement de façon optimale (Gondran et Minoux 1984). Il n'y a pas de formalisme mathématique standard, la solution optimale est obtenue en calculant les solutions des sous-problèmes les plus simples, pour ensuite en déduire étape par étape les solutions de l'ensemble.

Quatre phases ont été distinguées pour décomposer un programme dynamique (Cormen et al. 1990) :

1) La définition d'une étape et d'un état ;
2) La définition de la relation de récurrence ;
3) La détermination des ensembles des états de chaque étape ;
4) La construction d'une solution optimale par une procédure retour arrière (backtracking).

Dans le domaine de l'ordonnancement, l'application de cette technique reste limitée. Elle est utilisée pour développer des algorithmes polynomiaux et pseudo polynomiaux pour la résolution des problèmes à une machine et à machines parallèles (Lawler 1973 ; Schrage et Baker 1978 ; Carlier et Chrétienne 1988).

2.1.3. Programmation linéaire

La programmation linéaire (PL/LP 'linear programing') est l'une des méthodes classique du domaine de la recherche opérationnelle. Il représente un outil très performant pour la modélisation mathématique d'un problème d'optimisation avec des contraintes exprimées en inégalités et la fonction objectif en équation par rapport à plusieurs variables sous la forme d'un programme mathématique.

Cette approche est généralement limitée par sa capacité de stockage (Brusco et Stahl 2005). L'algorithme simplex est l'algorithme le plus utilisé en pratique, bien que sa complexité théorique soit exponentielle (Dantzig 1951).

Selon l'ensemble de définition des variables, on distingue trois types pour cette méthode :

- La programmation linéaire continue : Les variables appartiennent à l'ensemble des réels.
- La programmation linéaire en nombres entiers (PLNE) : Les variables sont des entiers.
- La programmation linéaire mixte : Deux types des variables sont distinguées ; réelles et des variables entières.

Tandis que, la majorité des problèmes d'ordonnancement, qui sont NP -hard, ne peuvent pas être résolus de manière optimale par des programmes avec des variables entières ou mixtes (réel et entier). Leur relaxation dans des programmes linéaires continus permet d'obtenir des bornes inférieures pour des problèmes de minimisation.

2.1.4. Techniques de relaxation

Ces techniques jouent un double rôle d'importance dans la résolution des problèmes d'optimisation combinatoire. Le principe de ces méthodes repose sur la relaxation d'un certain nombre de contraintes (contraintes de ressources, contraintes de priorité entre opérations), afin de rendre la résolution du problème plus facile. Ils fournissent de bornes limites inférieures pour augmenter l'efficacité des méthodes de séparation et évaluation.
Fisher et al. (1983) a proposé une approche pour un atelier job shop basée d'abord sur la relaxation des contraintes de ressources ; puis sur la relaxation des routages des jobs pour les petites instances sur lesquelles il est testé, cela donne de bons résultats.

2.2 Méthodes de résolution approchées

Les complexités de calcul requises par les méthodes exactes conduisent la recherche à des méthodes d'approximation. Ces procédures ne donnent généralement pas une solution optimale, mais elles aboutissent rapidement à une solution presque optimale. Aussi, elles offrent une alternative importante aux méthodes exactes pour l'analyse des problèmes d'optimisation combinatoire NP – difficile.

Les heuristiques et les métaheuristiques font partie de ces catégories que nous traitons dans les paragraphes qui suivent.

2.2.1. Heuristiques

Une méthode heuristique est une procédure susceptible de trouver une solution réalisable, rapide mais qui n'offre aucune garantie de qualité ou d'optimisation (Hillieran et Lieberman 2006). Toutes les solutions possibles ne sont pas considérées, car cela demanderait un temps infini, mais plutôt une partie de l'espace avec des solutions qui pourraient ou non être optimales.

L'espace de solution est recherché plus intelligemment, en éliminant les parties qui ne contiendront certainement pas de bonnes solutions et en se concentrant d'avantage sur les parties qui pourraient en inclure une bonne. Néanmoins, une méthode heuristique bien conçue peut souvent fournir une solution quasi-optimale ou indiquer l'inexistante de solution optimale. Cette méthode devrait également être suffisamment efficace pour qu'elle puisse traiter des problèmes importants dans un délai raisonnable.

La procédure est généralement un algorithme itératif entièrement développé, où à chaque itération, il s'efforce de trouver une solution meilleur que celle trouvée auparavant. Une fois terminé, lorsqu'un critère d'arrêt est satisfait, la solution fournie est la meilleure trouvée au cours d'une itération. Dans chaque itération, les méthodes de recherche utilisées sont établies par le bon sens pour s'adapter à un problème typique plutôt que pour être une méthode de solution générale (Rohitash Singh 2011).
Diverses heuristiques générales sont présentées ici.

2.2.1.1. Les heuristiques gloutonnes

Il s'agit d'un type d'heuristiques très fréquemment employées pour générer une solution initiale aux métaheuristiques. Ces techniques sont utilisées quand on peut construire la solution pas-à-pas, comme dans le cas du voyageur de commerce.

Un algorithme glouton effectue un choix à chaque fois en choisissant la solution optimale. Cependant, il ne revient jamais à un choix déjà effectué. Dans le cas d'un problème d'ordonnancement, cet algorithme choisit à chaque phase, l'opération de période de traitement minimale non encore ordonnancée dont la fonction objectif est basée sur la minimisation de la durée totale de production tout en respectant les contraintes relatives aux machines.

2.2.1.2. Shifting bottleneck heuristique 'SBH' (Goulot d'étranglement)

Le shifting bottleneck est une heuristique très efficace pour le problème de job shop dont la fonction objectif est le makepan. Le principe de cette heuristique est de déterminer itérativement une machine considérée comme bottleneck et d'ordonnancer de manière optimale cette machine uniquement. Pour chaque machine, un problème d'ordonnancement à machine $1|r_j|L_{max}$ est résolu à l'aide de la technique de séparation et évaluation proposée par (Carlier 1982). La machine bottleneck est celle qui a le L_{max} le plus élevé.

Ensuite, à l'aide d'un graphe disjonctif, les machines déjà ordonnancées sont re-séquencées pour inclure la séquence optimale de la machine bottleneck actuelle. Pour chaque opération O_{ij}, une date de lancement r_{ij} et une date d'échéance d_{ij} sont calculées itérativement : r_{ij} est la date de début au plus tôt de l'opération O_{ij}, calculée par rapport à ses prédécesseurs déjà ordonnancés ; d_{ij} est la dernière date d'échéance de l'opération O_{ij}.

Cette technique fonctionne sur deux types de voisinage efficace, qui utilisent l'échange de séquences d'opérations et l'affectation de nouvelles machines pour les opérations sur le chemin critique. Pour une présentation plus complète du SBH, le lecteur peut se référer à (Pinedo 2012).

La figure 2.3 fournit une représentation graphique de l'organigramme des principales étapes de construction de cette procédure (Pinedo 2012).

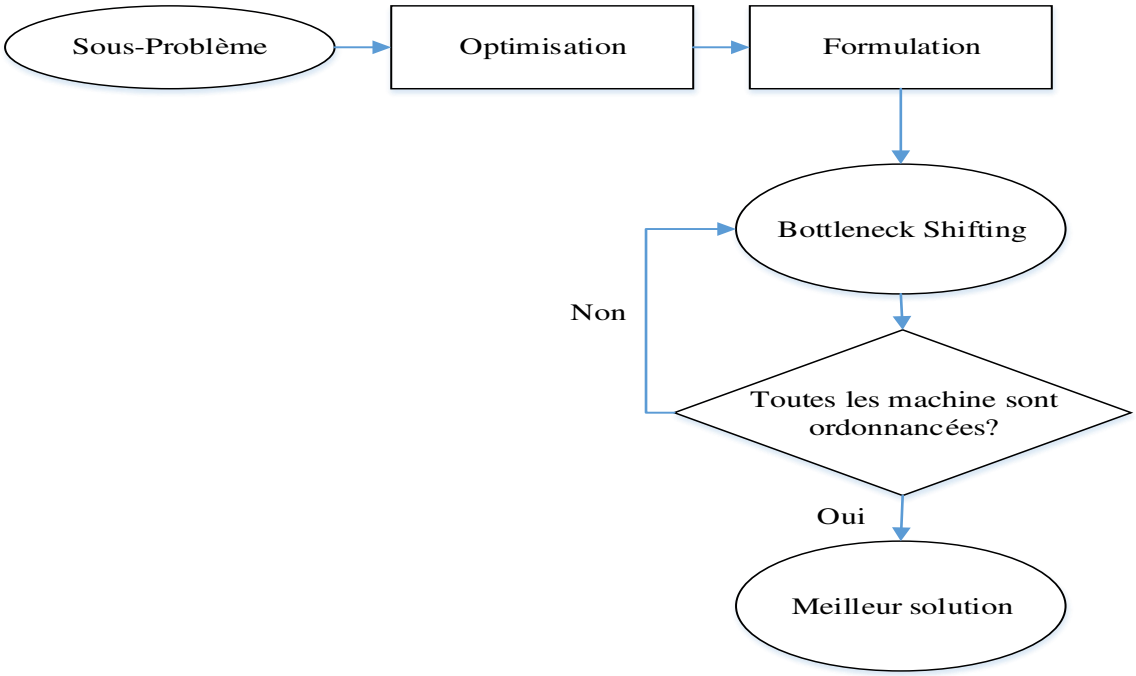


Figure 2.3 Organigramme de la procédure shifting bottleneck

2.2.1.3. Méthodes constructive

Les méthodes constructives consistent à construire une solution pour le problème à partir des données initiales. Deux méthodes sont utilisées dans ce contexte.

- Approche par opération qui consiste principalement sur les règles de priorité.
- Approche par machine qui consiste à trouver un séquencement sur les machines.

❖ **Les algorithmes de listes**

Un grand nombre de règles de répartition sont données dans la littérature et la recherche de nouvelles règles plus efficaces est un effort continu par les chercheurs.
Il a été argué qu'aucune règle de répartition ne peut satisfaire tous les critères d'optimisation ; il n'existe donc pas de règle de répartition magique à résoudre pour différents critères d'optimisation.

L'ordonnancement par listes est basé sur les différentes règles de priorité. Il est souvent appliqué dans la pratique pour générer des ordonnancements dans un délai relativement court (Kemppainen 2005 ; Pinedo 2008 ; Sobeyko et Monch 2016).

L'idée principale d'une telle approche est que des priorités différentes sont assignées aux jobs selon certaines règles de priorité. Ensuite, les jobs sont planifiés sur les machines en fonction de ces priorités. Les opérations ayant des priorités plus élevées sont ordonnancées en premier. Il peut y avoir plusieurs façons de sélectionner la machine pour chaque job. Généralement, une machine dont la disponibilité est la plus proche est sélectionnée.

En ce qui concerne les machines distinctes, nous modifions légèrement la procédure de sélection. En effet, la sélection d'une machine capable de terminer le job au plus tôt est la même que la sélection d'une machine ayant la disponibilité la plus précoce dans le cas de machines identiques. Il est à noter que cette approche tient compte des différentes vitesses des machines indépendantes disponibles dans les PJSF.

L'un des avantages de l'application de différentes règles de priorité est qu'un ordonnancement peut être calculé avec un effort de calcul relativement faible. En même temps, la performance de ces règles peut être médiocre compte tenu de la qualité de la solution pour de nombreuses classes de problèmes. Dans notre travail, nous nous référons à un certain nombre de règles de priorité qui sont les plus souvent citées dans la littérature (Tableau 2.1).

2.2.2. Metaheuristiques

Ces dernières années, les métaheuristiques sont de plus en plus utilisées pour résoudre les problèmes d'ordonnancement dans les ateliers de production offrant un bon compromis entre le coût de résolution et la qualité des solutions. Ces techniques sont des méthodes de recherche de voisinage basées sur l'idée de chercher les voisinages. Elles s'articulent autour de deux axes : la diversification et l'intensification dont chaque unes se caractérisent par un équilibre entre ces deux techniques (col et al 1996). Ne pas préserver cet équilibre conduit à une convergence trop rapide vers des minima locaux (manque de diversification) ou à une exploration trop longue (manque d'intensification) (Tai 2002).

➢ **La technique de diversification**

Pour garantir une meilleure résolution du problème, nous devons explorer plus d'espace de recherche. C'est pourquoi la phase de diversification doit être réalisée.

Lorsqu'il n'est pas possible de trouver une meilleure solution après un certain seuil α, la phase de diversification est déclenchée. Lorsque le nombre d'itérations est augmenté sans une meilleure solution après un certain seuil, cela signifie que la meilleure solution trouvée n'a pas été remplacée par un de ses voisins pendant un certain temps, ce qui est un signe que notre méthode est probablement piégée dans un optimum local. Dans ce cas, la décomposition doit être effectuée avec une autre solution initiale.

Un schéma de diversification peut être utilisé pour guider la recherche dans l'exploration d'autres régions qui pourraient ne pas être atteintes autrement. Plusieurs stratégies de diversification utiles peuvent être trouvées dans les récentes publications de livre de Glover et Laguna (1997). De tels schémas sont largement utilisés dans les méthodes basées sur la recherche tabou.

« La diversification fait référence à l'exploration de l'espace de recherche et, par conséquent, de l'espace de recherche »

- **La technique d'intensification**

La phase d'intensification est l'une des phases les plus importantes pour mieux exploiter l'espace de recherche en exécutant la fonction "On Wall" et "Inter Ineffective Collision". L'idée de cette fonction est de trouver une nouvelle solution à partir de la solution initiale. Pour développer une stratégie d'intensification, nous augmentons la taille du voisinage intéressant.

« Le terme intensification fait référence à l'exploitation de l'expérience de recherche accumulée »

Pour les problèmes complexes et de grande taille, on utilise les méta-heuristiques, où les méthodes de recherche combinent la procédure d'amélioration locale avec des stratégies plus avancées et intelligentes pour créer un processus capable de s'échapper de l'optimum local et d'effectuer une recherche robuste de l'espace de la solution. En théorie, ce processus de recherche pourrait se poursuivre indéfiniment, à moins que la valeur optimale du problème en question ne soit connue. Cependant, pour des raisons pratiques, un critère d'arrêt est établi qui détermine quand l'algorithme a trouvé une solution assez bonne. Les critères d'arrêt les plus courants sont :

- Après un nombre fixe d'itérations (ou un temps CPU fixe).
- Après un certain nombre d'itérations sans amélioration de la valeur de la fonction objectif.
- Lorsque l'objectif atteint une valeur seuil prédéfinie.

Dans le cadre de cette recherche, nous arrêtons l'algorithme après un nombre prédéfini d'itérations, sinon nous le déclarons explicitement.

Il existe différentes façons de classer et de décrire la métaheuristique. Les plus courants et utiles pour nous dans ce travail est de classer la métaheuristique en méthodes qui effectue une recherche à solution unique par rapport à une recherche sur la population.

Règle de priorité	Nomenclature	Description
FIFO	First In First Out	La prochaine opération ordonnancée est celle dont la date de disponibilité est antérieure à celles des autres opérations.
SPT	Shortest Processing Time	La prochaine opération ordonnancée est celle dont la durée est inférieure à celles des autres opérations non encore ordonnancées.
LPT	Longest Processing Time	La prochaine opération ordonnancée est celle dont la durée est supérieure à celles des autres opérations non encore ordonnancées.
EST	Earliest Starting Time	La prochaine opération ordonnancée est celle dont la date de début est antérieure à celles des autres opérations.
LST	Latest Starting Time	La prochaine opération ordonnancée est celle dont la date de début est postérieure à celles des autres opérations.
EFT	Earliest Finish Time	La prochaine opération ordonnancée est celle dont la date de fin est antérieure à celles des autres opérations.
LFT	Latest Finish Time	La prochaine opération ordonnancée est celle dont la date de fin est postérieure à celles des autres opérations.
EDD	Earliest due date	Définit les priorités de job à l'avance avant que l'ordonnancement ne soit exécuté.
ATC	Apparent tardiness cost	fixe les priorités de façon dynamique en fonction de l'état actuel de l'ordonnancement.
CR	Critical Ratio	Donne la priorité au job dont le rapport temps de traitement restant sur temps de traitement total est le plus faible.

Tableau 2.1 La description des règles de priorités les plus utilisées

2.2.2.1. Les méthodes de recherche locale (RL/LS 'Local Search')

S'appellent également métaheuristiques de voisinage ou à trajectoire (Blu et Rol, 2003). Le principe général de ces méthodes est de démarrer par une solution unique, et tentent de l'améliorer à chaque itération de leur progression. L'espace des solutions que l'on peut atteindre à partir d'une solution x est appelé voisinage N(x) de cette solution.
Cette catégorie regroupe plusieurs méthodes, les plus classiques et évidemment les plus populaires sont le Recuit Simulé et la Recherche Tabou.

2.2.2.2. Le Récuit Simulé (RS/SA 'Simulated Annealing')

La méthode de recuit simulé, inventée par (Kirkpatrick et al. 1983), basé sur la recherche locale. Il s'inspire du processus de recuit courant en métallurgie et dans l'industrie du verre pour résoudre les problèmes d'optimisation. La fonction objective du problème, similaire à l'énergie matérielle, est alors minimisée, introduisant une température fictive, qui est dans ce cas un paramètre de contrôle de l'algorithme.

La méthode commence par une solution initiale valable et continue l'exploration de l'espace d'états en exécutant des perturbations mineures sur la solution courante. Si la nouvelle solution obtenue est améliorée alors elle est maintenue. Si elle est détériorée par rapport au critère d'optimisation alors elle est maintenue avec une probabilité inversement proportionnelle au nombre d'itérations. Le recuit simulé a l'avantage de couvrir un espace de recherche plus grand notamment pour éviter la convergence prématurée vers un optimum local. La méthode de recuit simulé a été utilisée dans plusieurs problèmes de job shop.

2.2.2.3. La Recherche Tabou (RT/ TS 'Tabu Search')

Les algorithmes de recherche tabou (RT) proposée par (Glover 1977), sont parmi les approches les plus efficaces pour résoudre les PJS (Jain et Meeran, 1998). Ils utilisent une fonction mémoire pour éviter d'être piégés dans un optimum local (Zhang et al. 2009). Les structures de voisinage et les stratégies d'évaluation des déménagements jouent un rôle central dans l'efficacité et l'efficience du RT pour le PJS (Jain et al. 2000).

Le principe de la recherche tabou est d'explorer intelligemment l'espace de solutions du problème en évitant de s'enfermer dans un optimum local, grâce à deux stratégies spécifiques. La première stratégie est l'intensification qui force la recherche dans les domaines les plus prometteurs de l'espace de solution. La deuxième stratégie est la diversification, qui guide la recherche dans des nouvelles régions de l'espace.

Sa principale particularité est l'utilisation de mécanismes inspirés de la mémoire humaine.
Contrairement à la recherche tabou, le recuit simulé ne mémorise pas les configurations explorées précédemment. Cependant, la modélisation de la mémoire induit de multiples degrés de liberté qui rendent difficile une analyse mathématique rigoureuse.

En effet, c'est la seule métaheuristique utilisée pour résoudre des problèmes d'optimisation qui fonctionne avec une mémoire ou un ensemble de mémoires : la mémoire explicite qui permet de sauvegarder les solutions trouvées lors du processus de recherche et qui est la base de la stratégie de diversification, et la mémoire attributive qui sauvegarde les attributs tels que les permutations d'opérations qui permettent de passer d'une solution à l'autre.

Pour certains problèmes d'optimisation, la recherche tabou donne de bons résultats. De plus, dans sa forme de base, la méthode contient moins de paramètres de configuration que le recuit simulé, ce qui la rend simple à utiliser. Cependant, certains mécanismes tels que l'intensification et la diversification, apportent une complexité notable.
L'algorithme suivant décrit le schéma général de la recherche tabou.

Algorithme 2.1. Le pseudo code de la recherche tabou

- Variables et operateurs

s : Solution courante
s_0 : Solution initiale
N(s) : Ensemble de voisinage
TL: Liste taboue

Début
1. Trouver une solution initiale s_0 ; et poser s←s_0, i=0 et TL=∅ ;
2. Poser i=i+1 ;
3. Déterminer le sous-ensemble de voisinage $N^*(s_i) \subset N(s_i)$ tel que :
4. $\forall s_{i+1} \in N^*(s_i) : (s_i, s_{i+1}) \notin TL : ((s_i, s_{i+1})$: le mouvement de s_i à $s_{i+1})$;
5. Remplacer s_i par s_{i+1} tel que s_{i+1} est la meilleure solution de $N^*(s_i)$;
6. Mettre à jour TL ;
7. Si Critère d'arrêt n'est pas vérifié Alors aller à 3

Fin.

2.2.2.4. Algorithme de Kangourou (AK/KA'Kangaro Algorithm')

L'algorithme de Kangourou proposé par Gerard Fleury (1993), est une technique d'approximation fondée sur la descente stochastique, son principe est similaire à l'algorithme de recuit simulé mais avec une stratégie de recherche différente.

Il génère une solution en utilisant une procédure itérative qui contient deux parties : la procédure de descente stochastique et la procédure de saut (Serbencu et al 2007). Pendant la descente stochastique, AK cherche une solution qui minimise une fonction f(S) dans un voisinage N(S) de la solution actuelle S en utilisant une mutation d'opérateur local η1.
Si la nouvelle solution S'est meilleure que la solution précédente, elle est stockée et une nouvelle solution est explorée dans le même voisinage N(S').

L'algorithme tente d'améliorer la solution actuelle A fois, A étant le nombre maximum d'itérations pour améliorer la solution actuelle avant d'effectuer un saut. S'il n'est pas possible d'obtenir une amélioration supplémentaire, l'algorithme se déplace vers un autre voisinage avec une procédure de saut à l'aide d'un opérateur de mutation global η2. L'AK n'est en fait qu'une implantation spéciale de la descente stochastique, quand le nombre de sauts est nul (Belkadi 2006).

La méthode du kangourou présente l'avantage de ne pas perdre l'information relative aux optima locaux rencontrés, et il offre des solutions de bonnes qualités dans un temps de calcul modéré. Ainsi, il permet la recherche globale. Notamment, le fait d'effectuer des sauts permet à l'algorithme du kangourou de sortir d'une vallée c'est à dire d'un minimum local en sautant les barrières de potentiel (Talbi 2004). Cependant, il dévoile l'inconvénient comme le nombre de stationnements et de sauts nécessaire pour la recherche global.
Le pseudo code de l'algorithme et les différents paramètres constituant les deux phases de l'algorithme sont présentés ci-dessous.

Algorithme 2.2. Le pseudo code de l'algorithme de kangourou

- Variables et operateurs

A : Nombre maximal d'itérations sans l'amélioration de la solution courante
x : Etat courant.
f : Fonction objectif
x*: meilleur état rencontré à l'itération courante.
C : compteur d'itérations entre deux améliorations de la solution.
η1: Mutation uniforme locale
η2: Mutation uniforme globale
Début
Initialisations : solution actuelle : x ← x0 ; meilleure solution rencontrée : x*←x0 ;
Compteur de stationnement : C← 1 ;
Répéter
Si C < A alors Descente (x, x*,C);
Sinon Saut (x, x*, C);
Jusqu'à 'Critère d'arret';
Fin;

Procedure de Descente (x, x*, C) :

Répéter n_s fois :
1 : Appliquer la mutation η_1 à la solution courante :
$x_1 \leftarrow \eta_1(x)$;
2 : Si f (x_1) < f (x*) alors
Mettre à jour la meilleure solution rencontrée x* ← x_1 ;
C ← 0;
3 : Mettre à jour la solution courante x ← x_1 ;
4: C ← C +1

Procédure de Saut (x, x*, C) :

1 : Appliquer la mutation η_2 à la solution courante :
$x_1 \leftarrow \eta_2(x)$;
2 : Si f (x_1) < f (x) alors
C← 0 ;
x ← x_1 ;
3: C ← C+1;

Les mutations η1 et η2 ne sont pas certainement les mêmes, mais doivent respecter la propriété d'accessibilité de l'algorithme. Elles ont été choisies comme suit :

- *η1*: mutation uniforme locale. Elle permet d'effectuer un déplacement local ;

$\eta 1\ (x_i) = x_i + (2\gamma - 1)\ p$, avec p : taille maximum du pas, est un nombre réel ($0 < p < 1$).
Il est obtenu à partir d'une distribution uniforme sur [0,1].

- *η2*: mutation uniforme globale. Elle permet d'effectuer un saut vers un autre voisinage pour s'échapper d'un optimum local

$\eta 2\ (x_i) = \gamma$, avec γ est obtenu à partir d'une distribution uniforme sur [0,1].

✓ Algorithme à base de population

Les méthodes basées sur la population, comme leur nom l'indique, traitent à chaque itération de l'algorithme un ensemble de solutions plutôt qu'avec une solution unique. Ils offrent un moyen naturel pour l'exploration de l'espace de recherche. Pourtant, la performance finale dépend fortement de la manière dont la population est manipulée.

Les métaheuristiques à base de population les plus étudiées sont les algorithmes génétiques (AG/GA 'Evolutionary Algorithms') les colonies de fourmis (ACO/Ant Colony Optimization) et l'optimisation par les essaims particulaires (PSO/ Particulair Swarm Optimization).

2.2.2.5. Algorithmes génétiques (AG/ GA 'Genetic Algorithm'

Contrairement aux algorithmes de recherche locale tels que le recuit simulé et la recherche tabou qui n'ont besoin que d'une seule solution réalisable, les algorithmes génétiques considèrent une population de solutions réalisables appelée individus.

Les algorithmes génétiques par (Holland 1975), sont des algorithmes évolutionnaires qui construisent des solutions en combinant d'autres. Un ensemble d'un nombre défini de points dans l'espace de recherche, choisis au hasard ou par une heuristique, constitue la population initiale ; chaque individu (chromosome) de la population a une performance, qui mesure son degré d'adaptation à l'objectif recherché.

L'algorithme consiste à évaluer progressivement, par générations successives, la composition de la population, en maintenant sa taille constante et en trouvant des individus forts. L'objectif est d'améliorer globalement la performance des individus. Ils simulent le processus naturel d'évolution des espèces en adoptant deux lois qui les caractérisent : la transmission des caractères héréditaires à la reproduction et la loi de survie au sein de la population selon la théorie de Darwin.

Le but de la méthode est d'évaluer la population courante, la transition d'une génération à l'autre se fait en quatre phases : une phase de sélection, une phase de reproduction (ou variation), une phase d'évaluation des performances et une phase de remplacement.

- La sélection, qui favorise la reproduction et la survie des plus performants individus. Certains modes de sélection sont :
    - La sélection des tournois.
    - La sélection proportionnelle (Roulette) (Goldberg 1989).
    - La sélection de classement.
    - La sélection en régime permanent.
- La reproduction, qui permet le brassage, la recombinaison et les variations des caractères héréditaires des parents, pour former des progénitures d'un nouveau potentiel.
- L'opérateur de croisement, qui consiste à combiner les gènes de deux chromosomes et à créer de nouveaux individus, est appliqué dans un premier temps.
- Le deuxième opérateur impliqué dans le processus de génération des nouveaux individus est l'opérateur de mutation. C'est un opérateur de perturbation de chromosomes choisis au hasard qui modifie une partie de leurs caractéristiques.
- Le choix du codage, de la sélection, des opérateurs dépend du problème. La performance de l'algorithme génétique dépend de ses caractéristiques : la taille de la population, les probabilités de croisement et de mutation.
- Enfin, la phase de remplacement est effectuée. Elle consiste à sélectionner la nouvelle population.

L'algorithme est interrompu après un nombre fixe de générations, selon un critère d'arrêt.

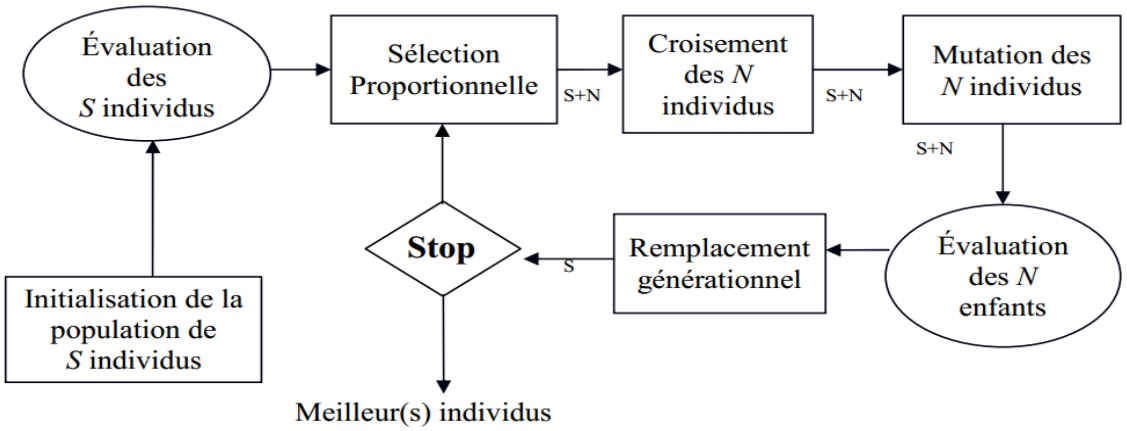


Figure 2.4. Le schéma de l'algorithme génétique (Dréo 2003)

2.2.2.6. Optimisation des Colonies de fourmis (OCF/ACO 'Ant Colonies Optimization')

La métaheuristique de l'optimisation de la colonie de fourmis Marco Dorigo (1990) a été inspirée par l'observation de plus court chemin des colonies de fourmis naturelles. La première application connue de l'ACO au JS est attribuée à Colorni et al (1994).

Au début, les fourmis explorent les environs de leur nid de façon aléatoire. Au fur et à mesure dès qu'une fourmi trouve une source de nourriture, elle évalue la quantité et la qualité de la nourriture et transporte une partie de cette nourriture jusqu'au nid.
Pendant le voyage de retour, la fourmi dépose un des traces de phéromones chimiques au sol. La quantité de phéromone déposée, qui peut dépendre de la quantité et de la qualité de l'aliment, guidera d'autres fourmis vers l'aliment source.

La communication indirecte entre les fourmis via les traînées de phéromones permet pour trouver le chemin le plus court entre leur nid et les sources de nourriture. Cette fonctionnalité des colonies de fourmis réelles sont exploitées dans les colonies de fourmis artificielles afin de résoudre les problèmes d'optimisation difficile.

Dans les algorithmes OCF, les traînées de phéromones chimiques sont simulées par un modèle paramétré probabiliste qu'on appelle le modèle des phéromones. Il se compose d'un ensemble de paramètres de modèle dont les valeurs sont appelées valeurs des phéromones. Ces valeurs servent de mémoire qui garde une trace du processus de recherche. L'ingrédient de base de l'optimisation des colonies de fourmis est une heuristique constructive qui est utilisée pour construire des solutions probabilistes en utilisant les valeurs des phéromones.

En général, l'approche d'optimisation des colonies de fourmis tente de résoudre un problème d'optimisation combinatoire en répétant les deux étapes suivantes :

- Les solutions sont construites à l'aide d'un modèle phéromone, c'est-à-dire une distribution de probabilité paramétrée sur l'espace de solution.
- Les solutions construites et, éventuellement, les solutions qui ont été construites au cours des itérations précédentes sont utilisées pour modifier les valeurs des phéromones d'une manière qui est considérée comme biaiser l'échantillonnage futur vers des solutions de haute qualité.

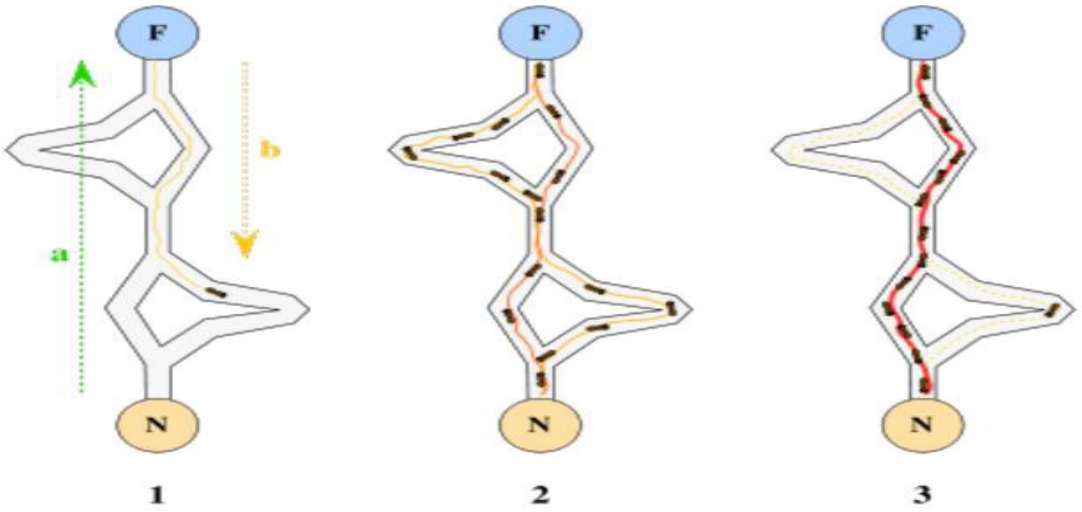


Figure 2.5. La structure de recherche du plus court chemin pour la technique OCF

2.2.2.7. Algorithme d'optimisation de l'essaim de particulaires (OEP/OPS : 'Particule Swarm Optimization')

Dans ces méthodes, proposées par Kennedy Eberhart (1995), chaque solution est un " oiseau " dans l'espace de recherche. Nous les appelons " particules ". Toutes les particules ont des valeurs de condition physique, qui sont évaluées par la fonction objectif, et des vitesses, qui dirigent le vol des particules.

Les particules " volent " dans l'espace du problème en suivant les particules optimales actuelles. OEP est initialisé avec un groupe de particules aléatoires (solutions) et recherche ensuite l'optimum en mettant à jour les générations. A chaque itération, chaque particule est mise à jour en suivant les deux " meilleures " valeurs.

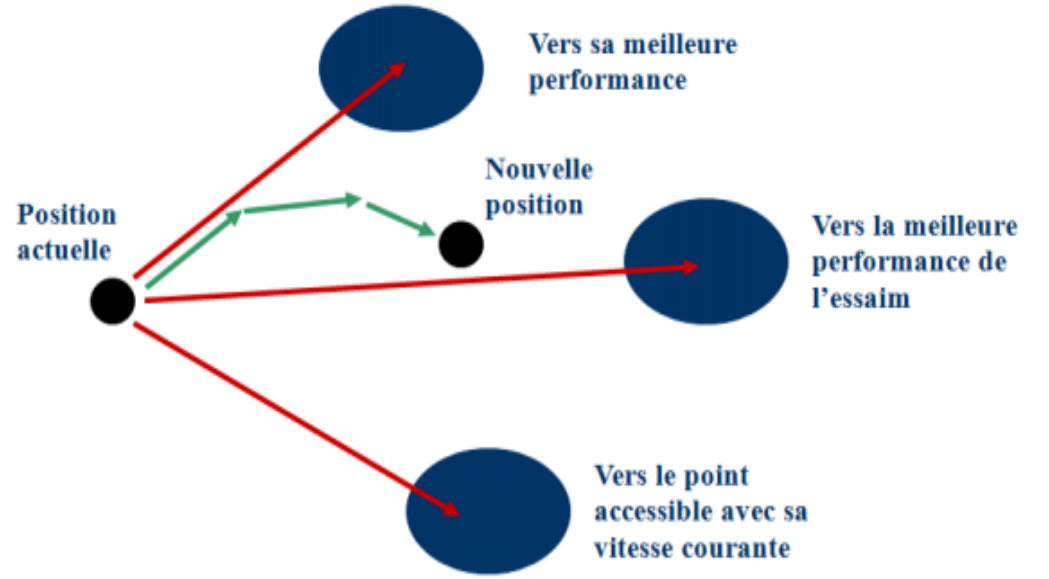


Figure 2.6. Schéma du principe du déplacement d'une particule (Cooren 2008)

2.3. Méthode hybride

Le concept de métaheuristique hybride n'a été communément accepté qu'au cours des trois dernières décennies, même si l'idée de combiner différentes stratégies et algorithmes métaheuristiques remonte aux années 1980. Ces techniques sont de plus en plus attirer l'intérêt académique.

Une méthode hybride est une méthode de recherche composée de deux ou plusieurs méthodes de recherche distinctes. Ce type d'algorithme produit généralement de bons résultats dans une grande variété de problèmes théoriques d'optimisation combinatoire (Grefenstette 1987).

Cette méthode tente d'exploiter les avantages respectifs élimine les faiblesses en combinant les différents concepts de métaheuristiques suivant une approche synergétique (Günther et Raidl 2006). En outre, la méthode hybridée peut engendrer des espaces de recherche plus efficace avec de nouvelles opportunités ce qui peut conduire à des méthodes plus puissantes et plus flexibles.

Par conséquent, L'hybridation n'offre pas que des avantages, ainsi selon le choix et le type de couplage de ses composants. Pour déterminer une méthode hybride efficace, il faut savoir caractériser les avantages et les limites de chaque méthode.

Aujourd'hui, on observe un intérêt commun généralisé à combiner des composants issus de différentes techniques de recherche et la tendance à concevoir des techniques hybrides est répandue dans les domaines de la recherche opérationnelle et de l'intelligence artificielle. L'intérêt consolidé autour de la métaheuristique hybride est également démontré par des publications sur les classifications, les taxonomies et des aperçus sur le sujet (Talbi 2002).

Talbi (2002) a présenté une taxonomie pour les méta-heuristiques hybrides. Dans cette classification, l'hybridation de métaheuristiques est étudiée sous deux angles : conception et mise en œuvre. Du point de vue de la conception, une classification hiérarchique dans laquelle trois classes principales sont identifiées selon leur architecture est décrite dans Figure 2.7.

- Hybridation séquentielle : Elle présente le type d'hybridation le plus populaire. Elle consiste à exécuter plusieurs méthodes de telle manière que le (ou les) résultat(s) d'une méthode serve(nt) de solution(s) initiale(s) à la suivante.
- Hybridation parallèle synchrone : Elle est obtenue par incorporation d'une méthode de recherche particulière dans un opérateur d'une autre.
- Hybridation parallèle asynchrone : Elle consiste à faire évoluer en parallèle plusieurs méthodes de recherche qui peuvent être identiques. Cette coévolution requiert un dispositif coordinateur qui permet d'assurer une bonne coopération des méthodes.

Il est bien évidemment possible de coordonner plusieurs stratégies d'hybridation au sein de la même méthode. Dans cette thèse, les méthodes de résolution des problèmes d'ordonnancement sont basées sur l'hybridation séquentielle des métaheuristiques.

Il existe bien d'autres méthodes importantes qui combinent entre les avantages des deux paradigmes par l'approche de l'hybridation. Parmi ces méthodes, on a les Algorithmes Mémétiques (Genetic Local Search), Scatter Search, GA/PM.

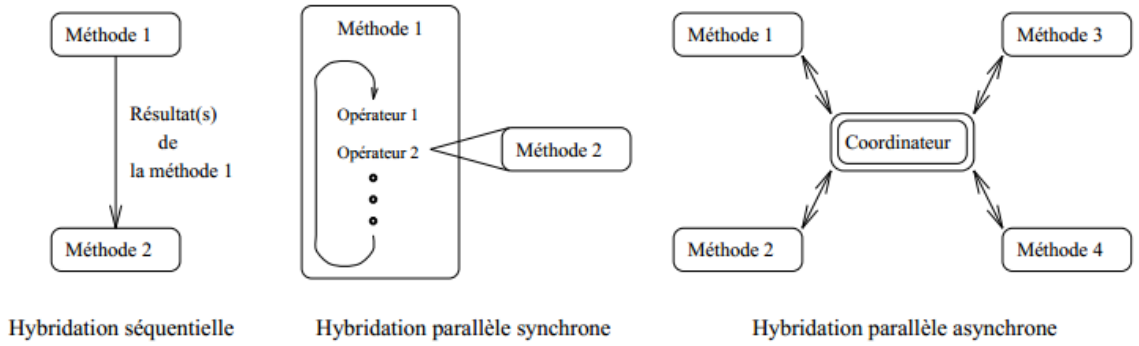


Figure 2.7. Techniques d'hybridation

3. Les fonctions de voisinage

Le voisinage d'une solution courante est composé de toutes les solutions obtenues en effectuant un déplacement élémentaire autour de cette solution. Une fonction de voisinage définie ce déplacement qui produit pour une solution donnée, un ou plusieurs voisins.

Toute fonction de génération de voisinage doit, dans certaines limites, respecter deux conditions qui sont : la faisabilité et L'accessibilité.

- La faisabilité désigne que la fonction de voisinage est capable de générer des solutions admissibles.
- L'accessibilité d'un optimum global est garantie si la solution optimale est achevée depuis un état initial admissible.

Plusieurs voisinages sont basés sur la notion de bloc critique. Rappelons en effet que le bloc critique est l'ensemble d'opérations critiques s'exécutant consécutivement sur une même machine. Ces voisinages, issus des travaux de plusieurs auteurs sont : Nl, N2, NA, RNA, NB et NS.

3.1. Les voisinages Nl et N2

Le voisinage N1 a été introduit pour la première fois par Van Laarhoven en 1988. N1 est un simple voisinage obtenu par inversion de deux opérations successives sur le chemin critique. De plus, il est justifié que ce voisinage vérifie la connectivité.

D'autre part, Le voisinage N2 présenté par Matsuo et al. (1988 ; Jai 1998), est dérivé de Nl, mais il réduit le nombre de voisinage. Cet avantage de réduction de voisinage vient au détriment de la caractéristique de connectivité, en effet N2 ne vérifie pas cette propriété (Sch, 2001). Le principe de N1 est interprété par l'exemple de la figure 2.8.

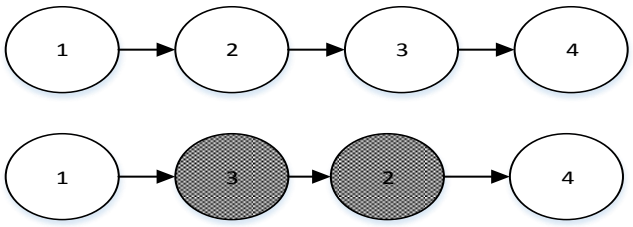


Figure 2.8. Illustration du principe voisinage N1

3.2. Les voisinage RNA et NA

Les voisinages NA et RNA sont proposés par Dell'Amico et Trubian (1993). Le voisinage NA peut être vu particulièrement comme une suite de Nl. Mais, au lieu d'inverser un seul arc dans un bloc, NA considère plus de trois opérations simultanément. Ce voisinage respecte la connectivité, puisque il s'agit d'une application consécutive de Nl, qui vérifie cette propriété. RNA est une variante de NA dans le même contexte de N2 avec Nl.
Dans l'exemple de la figure 2.9 à partir d'une solution initiale, le voisinage NA est engendré en inversant successivement trois arcs.

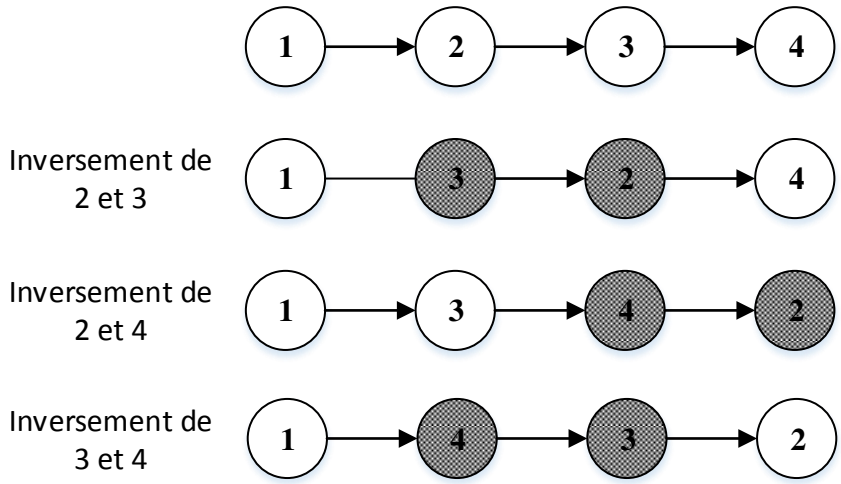


Figure 2.9. Illustration du principe de voisinage NA.

3.3. Le voisinage NB

Ce voisinage introduit par Grabowski et al. (Jai 1981) est fondé sur la structure de bloc critique. Le principe de NB comporte à déplacer une opération, soit au début, soit à la fin de son bloc critique. Tant que ce déplacement génère des ordonnancements faisables, jusqu'à aboutir la fin (respectivement le début) de son bloc. La figure 2.10 illustre deux voisins qui peuvent être engendrés de l'opération 3 en la déplaçant au début ou à la fin du bloc.

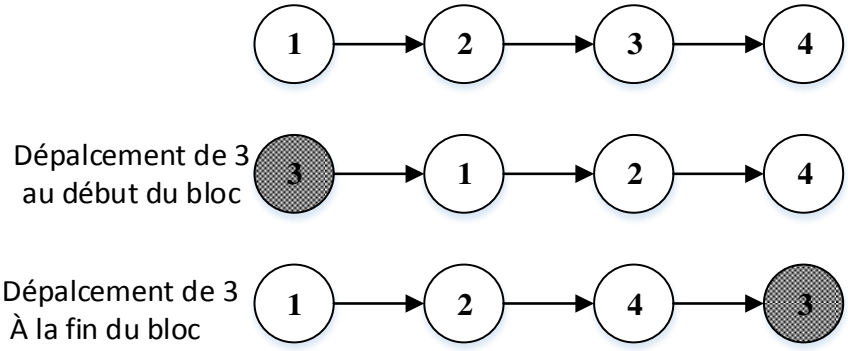


Figure 2.10. Illustration du principe de voisinage NB

3.4. Le voisinage NS

L'inconvénient majeur des voisinages précédents, et le plus grand nombre de voisins générés pour une solution donnée. L'impact de leur évaluation sera important sur la performance de la méthode de résolution.

Le voisinage NS proposé par Nowicki et Smutnicki est connu comme étant le voisinage le plus réduit. Il permet d'obtenir les résultats en temps relativement court (Wat et al 2006). L'approche de NS génère d'bord un seul chemin critique, sur ce chemin critique qui comprend b blocs, NS inverse le premier et le **dernier** arc de chaque bloc, sauf le premier et le dernier bloc où le reversement concerne respectivement le dernier et le premier arc seulement.

Sur l'exemple de la figure 2.11, le bloc est constitué de six opérations, trois voisinages sont possibles selon la localisation du bloc : au début, au milieu ou à la fin de l'ordonnancement

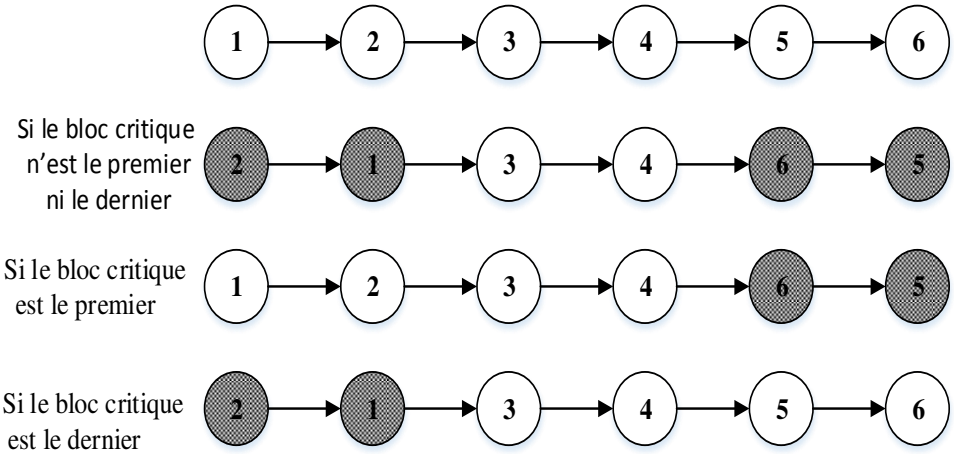


Figure 2.11. Illustration du principe de voisinage NS

4. Conclusion

Dans ce chapitre, nous avons détaillé les principales méthodes de résolution des problèmes d'ordonnancement. Pour chaque méthode, l'idée générale est rapportée. Ces méthodes étaient et restent le concept d'une recherche intense pour la résolution de divers problèmes NP-difficiles dont le problème d'ordonnancement fait partie.

La présentation générale menée au cours de ce chapitre sur les méthodes de résolution appuyant sur leurs définitions et l'explication de leurs principes de fonctionnement a permis de faire ressortir l'idée de base de chacune de ces méthodes pour conduire ensuite à la description de leur adaptation et application au problème de Job Shop et ses dérivées. Nous nous concentrons sur celles qui sont utilisées dans notre étude.

Parmi les différentes approches, on constate un intérêt croissant aux approches des métaheuristiques. Les métaheuristiques sont représentées essentiellement par les méthodes de voisinage Elles ont prouvé leur grande efficacité de fournir des solutions approchées de bonne qualité pour un grand nombre de problèmes.

Le chapitre suivant dresse un état de l'art de différentes méthodes utilisées pour la résolution des problèmes d'ordonnancement flexible avec et sans intégration de planification des processus abordés dans cette thèse.

CHAPITRE 3
Etat de l'art

Les chercheurs ont utilisé un grand nombre de techniques allant des méthodes exactes à diverses métaheuristiques pour résoudre le problème d'intégration de la planification des processus et de l'ordonnancement d'un côté et le problème d'ordonnancement de job shop flexible d'un autre côté. Ce chapitre tente de consolider ces recherches et présente les résultats. Leurs forces et faiblesses relatives sont évaluées en termes d'applicabilité pratique, de qualité des solutions et d'efficacité des algorithmes.

1. Introduction

Au cours des 25 dernières années, les chercheurs et les praticiens ont tenté de mettre au point des techniques/méthodes efficaces pour trouver des solutions. Une analyse critique de la documentation publiée révèle que la communauté des chercheurs accorde une attention digne de mention à l'IPPS et le PJSF. Ils ont enquêté sur les avantages d'intégrer l'ordonnancement avec différents fonctions de production, l'une d'entre elles est d'ordonnancer avec d'autres plans de processus (Zhao et Kops 1987 ; Tonshoff et Beckendorff 1989 ; Khoshnevis et al 1990 ;Tan et Khoshnevis 1997).Cette période a marqué un développement de techniques efficaces pour résoudre ces problèmes. Le makespan s'est avéré être la mesure de performance la plus utilisée.

Bien qu'il existe une relation étroite entre la planification du processus et l'ordonnancement, leur intégration demeure un défi pour les chercheurs et les praticiens (Sugimura et al 2001). L'intérêt porté à ces systèmes est largement motivé par ce caractère de flexibilité donnant plus de degrés de liberté aux systèmes de production et s'approchant de plus en plus des problèmes d'ateliers réels.

L'étude présentée dans ce chapitre porte sur l'examen des techniques/méthodes de solution publiée dans la littérature pour résoudre les problèmes IPPS. Diverses approches pour l'IPPS ont été discutées avec leurs avantages et leurs inconvénients et les recherches rapportées sont classés en conséquence. De plus, il présente de la documentation sur l'optimisation de l'ordonnancement flexible de job shop.

2. Revue de littérature sur le mécanisme d'intégration de planification et ordonnancement 'IPPS'

La recherche sur l'ordonnancement s'est principalement concentrée sur la construction d'algorithmes efficaces pour résoudre différents types de problèmes d'ordonnancement, par exemple, flow shop, job shop, open shop, et ainsi de suite.

Le Job shop flexible est considéré comme un domaine de recherche important et fait donc partie de la classe de problèmes la plus publiée dans un système de production. En raison de la nature pratique de ce problème, un grand nombre de chercheurs ont proposé diverses techniques/méthodes efficaces. Un bref examen de chacune des techniques est présenté dans les sections suivantes. On constate que près de 59% des papiers ont utilisé des techniques hybrides ou les algorithmes évolutionnaires.

Toutefois, certains travaux de recherche montrent que l'intégration de la fonction de planification de processus et de la fonction d'ordonnancement permet d'acquérir des connaissances précieuses (Tan, 1998 ; Tan et Khosnevis, 1997 ; Li et McMahon, 2007).

Un autre moyen d'IPPS est d'accroître l'échange d'informations entre les fonctions de planification des processus et d'ordonnancement. Tan et Khoshnevis (2000) ont tenté dans cette direction avec un succès limité (Haddadzade et al, 2009 ; Tan et Khoshnevis, 2000).

La recherche sur IPPS a été proposée premièrement par Chryssolouris et Chan, (1985) et par la suite, il y a eu de nombreux efforts de recherche sur ce problème.

Chryssolouris et Chan, (1985) ont proposé la prise de décision en matière de production. Approche (MADEMA), la première approche disponible dans la littérature pour IPPS.

Ils ont considéré un ensemble de ressources alternatives pour l'exécution d'une tâche de production particulière. Une matrice de décision a été formée pour la sélection des alternatives, où la ligne représente l'alternative tandis que la colonne représente l'attribut et l'entrée était la valeur de l'attribut pour l'alternative correspondante. Le concept de MADEMA comportait cinq étapes consécutives :

- Déterminer des solutions de rechange,
- Déterminer les attributs,
- Déterminer les conséquences des attributs pour chaque solution de rechange,
- Appliquer les règles de décision pour choisir la meilleure solution de rechange ;
- Choisir l'option la meilleure alternative.

Les ressources alternatives ont été choisies en évaluant la contribution sur les points suivants certains critères établis pour la prise de décision, comme une combinaison linéaire d'attributs par exemple des poids ou des solutions de rechange ayant plus de chance de produire une plus grande valeur utilitaire.

Tonshoff et al (1989) ont présenté FLEXPLAN pour IPPS. Les auteurs ont créé tous les multiples plans de processus (MPP) avant le début de la production. La fonction d'ordonnancement sélectionne le plan de processus approprié, selon la disponibilité des ressources. Cette approche couvre la re-planification réactive pour permettre la réaction des perturbations survenant dans l'atelier.

Au cours des dernières décennies, diverses formes d'Intelligence Artificielle (IA) ont été mises au point pour résoudre les problèmes liés à l'IPPS. Les méthodes typiques sont : les approches basées sur les agents, les approches basées sur le réseau de Petri et les approches basées sur les algorithmes d'optimisations.

2.1. Approches basées sur les agents de l'IPPS

Une définition typique d'un agent est donnée par (Nwana et Ndumu 1997) : un agent est défini comme se référant à un composant d'un logiciel et/ou d'un matériel capable d'agir exactement afin d'accomplir des tâches au nom de son utilisateur. Lorsqu'il est appliqué à une production, un agent est un objet logiciel représentant un élément d'un système de production tel qu'un produit ou une machine (Zhao et al 2010). En particulier, un problème d'ordonnancement job shop a été résolu, où chaque agent est responsable de l'ordonnancement d'un ensemble d'ordres.

La structure interne d'un agent comprend généralement (Figure 3.1) :

- Unité de communication ;
- Unité de décision ;
- Unité opérationnelle ;
- Base de connaissances.

Asadzadeh and Zamanifar (2010), présentent une approche basée sur un système multi-agents (MAS) qui crée une population initiale et parallélise l'AG (PaGA) dans une approche basée sur les agents dans un job shop. Il est composé d'un agent de gestion (pour créer la population initiale), d'un agent d'exécution (pour ordonnancer l'opération sur les machines), d'un agent processeur (pour exécuter GA sur sa sous-population), et d'un agent de

synchronisation (pour coordonner la migration entre les sous-populations de l'agent de traitement).

Nejad et al (2011), ont proposé une architecture multi-agents d'un système IPPS pour des multi-jobs dans des systèmes de production flexibles. Un protocole de négociation a été utilisé pour générer dynamiquement et progressivement les plans de processus et les ordonnancements des machines et les jobs individuels basés sur les processus alternatifs de production.

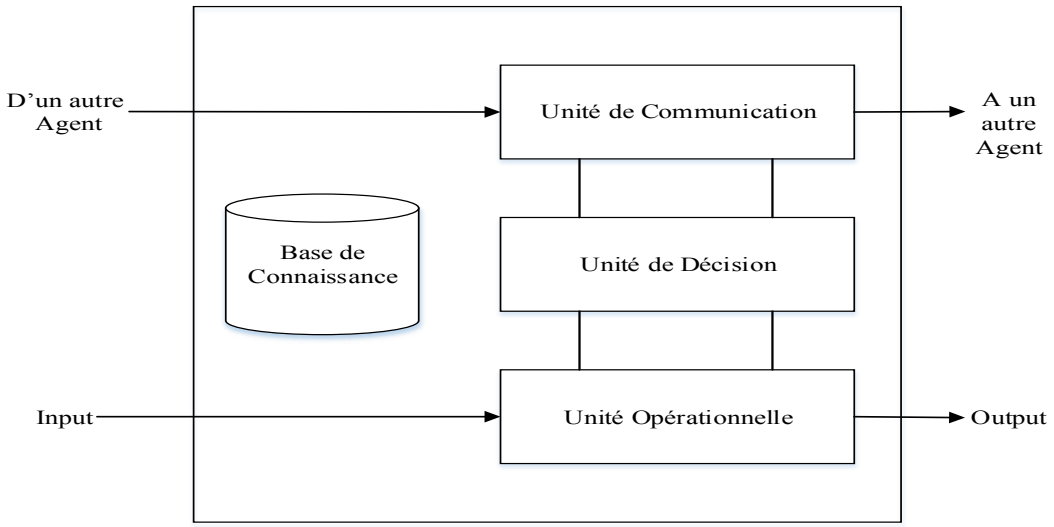


Figure 3.1 La structure d'un agent (Fang 2015)

Merzouki et al (2017), ont étudié le PJSF et proposé un système multi-agents basé sur l'optimisation des réactions chimiques CRO avec l'algorithme Greedy. Le modèle est composé d'un agent d'interface et d'agents d'ordonnancement viser à passer à la phase d'optimisation en utilisant CRO avec l'algorithme Greedy afin de minimiser le temps de traitement maximal (Makespan).

Récemment, Nouri et al, (2018) ont proposé un algorithme hybride basé sur un modèle multi-agent holonique en cluster basé sur la combinaison d'algorithme génétique et la recherche tabou GATS + HM, afin de minimiser le makespan dans un environnement JSF.

Les approches fondées sur les agents présentent plusieurs avantages potentiels pour l'ordonnancement de production.

- Ces approches utilisent le calcul parallèle par l'intermédiaire d'un grand nombre de processeurs, qui peuvent fournir des systèmes d'ordonnancement très efficaces et robustes.
- Ils peuvent faciliter l'intégration de la planification et de l'ordonnancement des processus de fabrication.
- Ils permettent à des ressources individuelles de faire des arbitrages à l'amélioration de la performance locale pour améliorer la performance globale, à la pointe de l'innovation à l'ordonnancement coopératif.
- Les agents ressources peuvent être reliés directement à des agents physiques. Les dispositifs qu'ils représentaient pour réaliser un ré-ordonnancement dynamique en temps réel, peuvent fournir un système de fabrication avec une plus grande fiabilité et une meilleure tolérance aux pannes des ressources.

2.2. Approches fondées sur des algorithmes de l'IPPS

Les étapes de base de l'approche basée sur l'algorithme sont les suivantes :

- ✓ Tout d'abord, le système de planification des processus est utilisé pour générer les plans de processus alternatifs pour tous les jobs et sélectionner les plans optimaux de nombre défini par l'utilisateur en fonction des résultats de simulation.
- ✓ Ensuite, l'algorithme du système d'ordonnancement est utilisé pour simuler les plans d'ordonnancement en fonction des plans de processus alternatifs pour tous les jobs.
- ✓ Enfin, d'après les résultats de la simulation, le plan de processus de chaque job et le plan d'ordonnancement sont déterminés.

Dans cette approche, la plupart des recherches ont concentrés sur diverses approches métaheuristiques comme l'algorithme génétique (AG), le recuit simulé (RS), l'optimisation optimisation par essaims particulaires (OEP), la recherche tabou (RT), l'optimisation par colonies de fourmis (OCF) et le système immunitaire artificiel (SIA), et des algorithmes hybrides.

Brandimarte et al (1995), ont présenté un modèle hiérarchique pour modéliser le problème d'intégration. La partie supérieure traite de la sélection des processus, et la partie inférieure, s'occupe de l'ordonnancement dans un job shop. Le problème est modélisé comme multi-objectif, en considérant à la fois le coût opérationnel et le makespan. Les deux niveaux sont représentés en programmation linéaire en nombres entiers mixtes afin de gérer les interactions entre les deux niveaux.

Palmer (1996) a proposé une approche fondée sur le recuit simulé (SA) pour IPPS.
Il contenait trois types de configuration aléatoires ;

- ❖ Inverser l'ordre de deux opérations séquentielles sur une machine,
- ❖ Inverser l'ordre de deux opérations séquentielles dans un job,
- ❖ Modifier la méthode utilisée pour effectuer une opération.

Kim et Egbelu (1999), considèrent un problème IPPS avec l'objectif makespan. L'heuristique proposée combine des techniques de branchement et de séquencement 'branch & bound' avec une approche basée sur les règles d'ordonnancement.

Lee et Kim (2001), ont proposé une méthode pour IPPS utilisant une simulation basée sur un algorithme génétique (AG). Le module de simulation calcule les mesures de performance en fonction de la combinaison de plans de processus créée par AG au lieu des alternatives de plans de processus, et édite la combinaison des plans de processus quasi optimale avant l'exécution en atelier. Les mesures des résultats ont porté sur le makespan et le retard en fonction des règles de priorité : SPT et EDD.
Ils ont conclu qu'une réduction d'environ 20 % de makespan était possible par rapport à la sélection aléatoire d'une combinaison de plans de processus. La structure de la technique considérée est illustrée dans la figure 3.2.

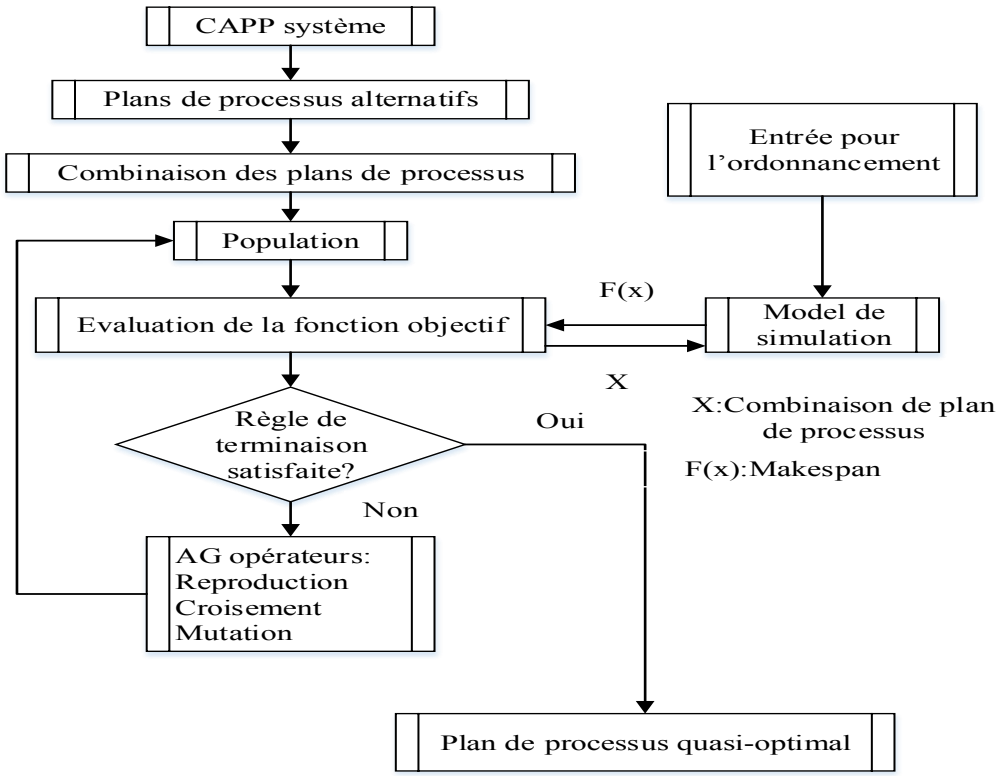


Figure 3.2 La procédure globale de l'intégration de planification des processus et l'ordonnancement selon Kim (2001)

Kim et al (2003), ont proposé une technique de recherche en Intelligence Artificielle (IA) appelée Symbiotic Evolutionary Algorithm (SEA) pour traiter simultanément la planification des processus et l'ordonnancement dans un job shop. La SEA se fondait sur le fait que la recherche parallèle de différentes solutions était plus efficace qu'une recherche unique pour l'ensemble de la solution. Ils ont tenu compte de la flexibilité de routage, de la flexibilité de séquence et de la flexibilité du traitement, tout en optimisant deux types d'objectifs : la minimisation de la charge de travail et le temps moyen d'écoulement.

Zhao et al (2004), ont proposé une approche fondée sur l'AG pour l'IPPS dans un job shop. Un système d'inférence floue (Fuzzy Inference System) a été utilisé pour sélectionner d'autres machines. Sur la base de la capacité des machines, AG a été utilisé pour équilibrer la charge de toutes les machines. L'algorithme. GT (Giffler et Thompson 1960) a été utilisé pour évaluer l'aptitude des chromosomes dans l'établissement d'ordonnancement. Les objectifs étaient de réduire le makespan, la quantité de déchets, et les coûts de traitement.

Kumar et Rajotia (2006), ont analysé l'effet de la modification d'un plan de processus flexible d'un type de pièce dans un ordre de fabrication. Ils utilisent un algorithme génétique basé sur la simulation (Simulation-based GA) afin de sélectionner la clé type de pièce (part-type key) de manière à ce qu'une performance de mesure puisse encore être améliorée. Cette approche génère des performances quasi-optimales pour le makepan.

Park et Choi (2006), proposent un AG qui exécute la planification du processus et l'ordonnancement simultanément. Un environnement job shop avec critère makepan est utilisé. En 2008 ils appliquent la planification et l'ordonnancement fondés sur la simulation pour anticiper les changements d'ordonnancement futurs en simulant différents scénarios " Que faire si " à partir de données en temps réel sur l'atelier.

Fattahi et al (2007), examinent les approches hiérarchiques et intégrées de la $FJm||C_{max}$ utilisant un modèle mathématique pour atteindre la solution optimale pour les petites instances et une approche basée sur deux heuristiques : la recherche tabou (RT) et le recuit simulé (RS). La figure 3.3 résume l'approche utilisée.

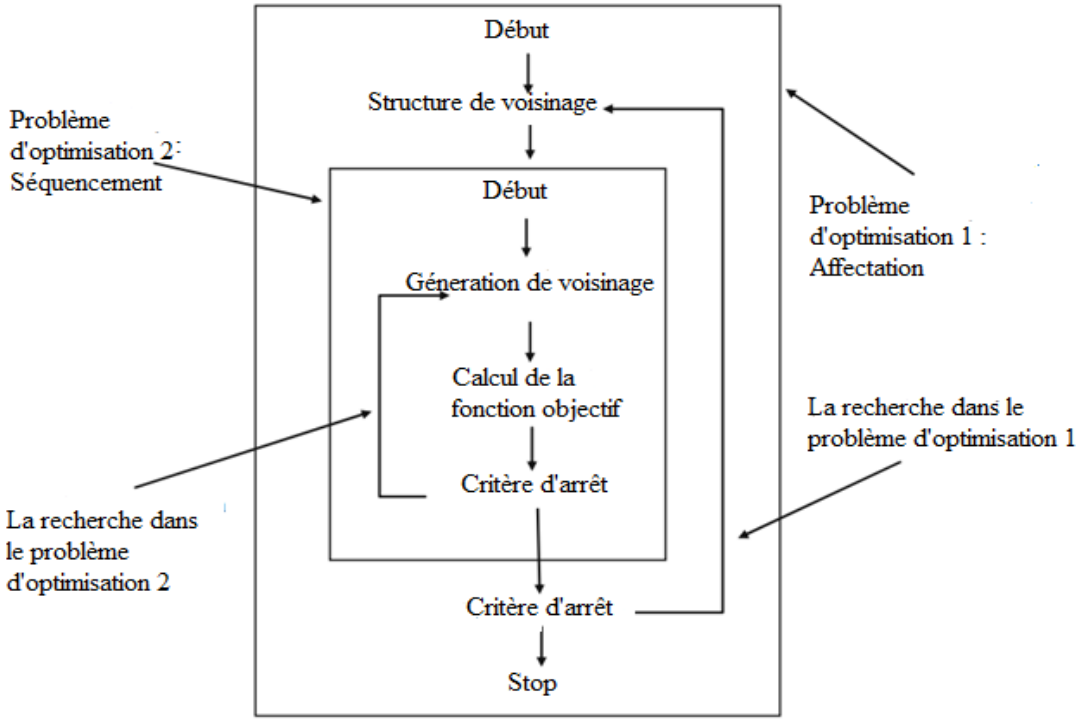


Figure 3.3. L'approche hiérarchie pour un job shop flexible (Fattahi 2007)

Li et McMahon (2007), ont proposé une approche fondée sur la RS pour l'IPPS dans un job shop. Le traitement, le séquencement des opérations et la flexibilité de l'ordonnancement ont été utilisés pour explorer l'espace de recherche de l'algorithme proposé. L'algorithme a été défini en deux ensembles de structures de données. Le premier ensemble représente les plans de processus et le second ensemble spécifie l'ordonnancement d'un groupe de pièces. Les mesures de performance ont porté sur le makespan, le niveau équilibré d'utilisation de la machine (balanced level of machine utilisation), le retard des jobs (job tardiness) et le coût de la production.

Zhao et al (2010), ont proposé un IPPS applicable au système de production holonique (Holonic manufacturing system HMS), où ils ont utilisé un algorithme hybride OEP et un algorithme d'évolution différentielle afin d'équilibrer la charge de toutes les machines.

Leung et al (2010), ont proposé une approche IPPS utilisant l'algorithme d'optimisation par colonie des formis (OCF) basée sur le système Multi-Agents (SMA) afin de minimiser le makespan dans un job shop. Ils ont envisagé de traiter la flexibilité de la séquence alternative et des machines alternatives. Les graphiques ET/OU ont été utilisés pour représenter les plans de processus multiples. Les auteurs ont conclu que l'approche ACO basée sur les agents proposée était faisable pour résoudre le problème IPPS.

Amin-Naseri et Afshari (2012), ont mis au point un AG hybride pour l'intégration de planification des processus et d'ordonnancement avec des contraintes de précédence. Ils ont conçu des opérateurs génétiques pour renforcer le pouvoir de recherche global de l'AG et d'une

procédure locale de recherche a été incorporée dans l'AG afin d'améliorer les performances de la procédure de recherche de l'algorithme.

L'AG améliorée, ainsi que le modèle mathématique développé ont été montrés dans (Qiao et Lv 2012) comme exemple de résolution du problème IPPS. Afin d'améliorer la performance d'optimisation, ils ont proposé une nouvelle méthode de sélection initiale pour les plans de processus, en plus d'une nouvelle méthode de représentation génétique pour la planification des plans et des schémas pour les opérateurs génétiques.

Lian et al (2012), ont proposé un algorithme méta-heuristique appelé algorithme compétitif impérialiste (ICA) afin d'aborder le problème IPPS avec un objectif de la minimisation du makespan. Il s'agissait d'un algorithme évolutionnaire basé sur la population qui génère simultanément un plan d'ordonnancement et un plan de processus pour chaque pièce.

Ausaf et al (2015), proposent une approche qui intègre des règles de priorités avec le concept de hiérarchisation des jobs, dans un algorithme appelé algorithme heuristique basé sur les priorités (PBHA). Le PBHA tente d'établir la priorité des jobs et des machines pour la sélection des opérations. L'affectation des priorités et un ensemble des règles de priorité sont utilisées simultanément pour générer à la fois les plans de processus et les ordonnancements pour tous les jobs et machines.

Dai et al (2015), explorent le problème de la planification et de l'ordonnancement de la production durables en tenant compte de l'énergie, ainsi qu'un nouvel indicateur de rendement de l'efficacité énergétique a été considérée comme un objectif d'optimisation. Un modèle pour le problème de la consommation d'énergie totale multi-objectifs et de l'atelier de fabrication a été développé pour décrire l'EIPPS. Un AG modifié a été adoptée pour rechercher les solutions optimales entre la consommation d'énergie et le makespan. L'organigramme de l'algorithme est donné dans la figure 3.4.

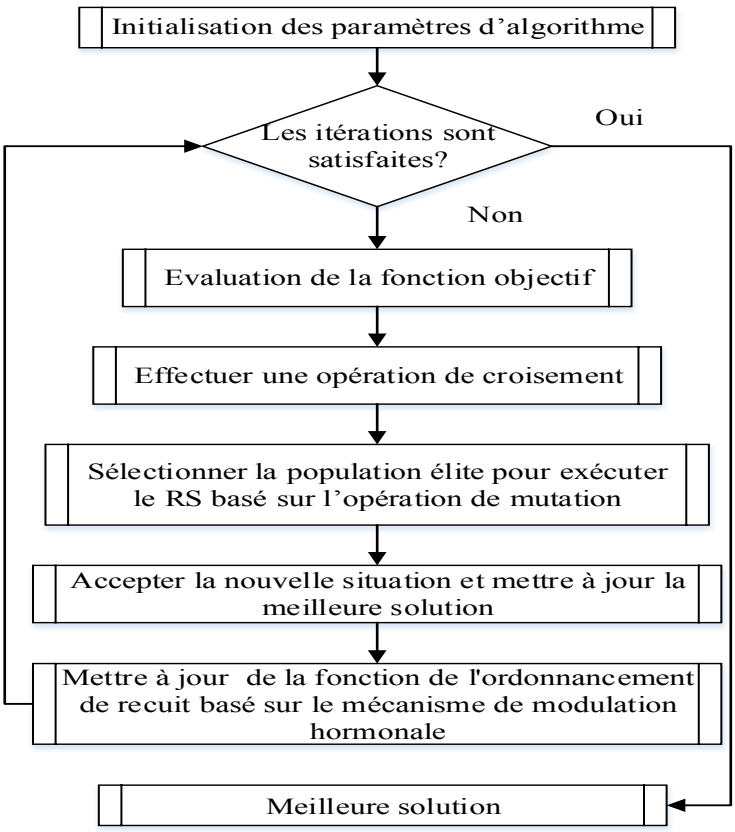


Figure 3.4. L'organigramme de l'algorithme génétique modifié avec le récuit simulé (Dai et al 2015)

Sobeyko et Mönch (2016) ont abordé un IPPS dans un job shop flexible pour les grandes instances où une structure de variable de voisinage appropriée est proposée (VNS). Un algorithme OPS chaotique est proposé pour résoudre ce problème. Dix cartes chaotiques sont mises en œuvre pour éviter une convergence prématurée vers l'optimum local.
Le taux d'utilisation de la machine et le makespan sont considérés comme fonctions objectifs. Les plans d'ordonnancement sont testés par un robot mobile dans un environnement de laboratoire.

Xia et al (2016) proposent une combinaison de l'algorithme génétique avec recherche variable de voisinage (GAVNS) et de la technologie des fenêtres roulantes (rolling windows) pour résoudre le problème dynamique IPPS (DIPPS). Deux types de perturbations sont pris en compte, qui sont pannes des machines et l'arrivée d'un nouvel job. Pour mieux comprendre cette combinaison, la figure suivante (3.5) décrit sa structure.

Dans l'article de (Chaudhry et Usman 2017), un algorithme génétique (AG) pour la planification de processus et l'ordonnancement intégrés est proposé. Dans l'approche proposée, un domaine indépendant de tableur 'spreadsheet 'est présenté pour résoudre cette catégorie de problèmes (Figure 3.6). Les relations de précédence entre les opérations des jobs sont considérées dans le modèle, sur la base duquel une représentation implicite d'un plan de processus réalisable pour chaque job peut être exécutée.

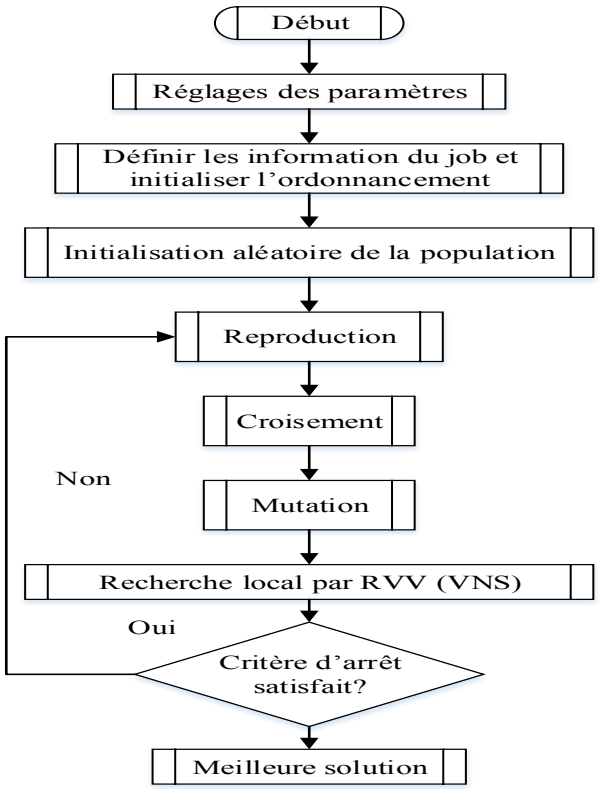


Figure 3.5. La structure de GAVNS (Xia 2016)

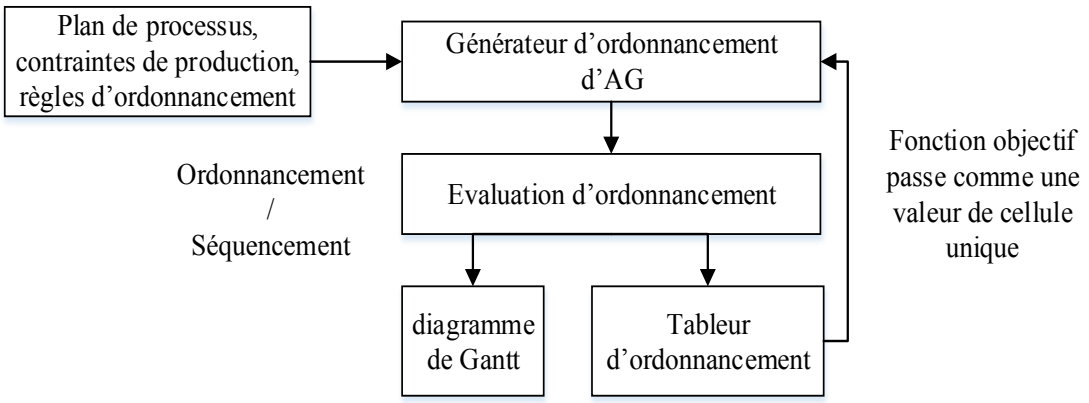


Figure 3.6. Algorithme génétique - Architecture d'interface de tableur (Chaudhry 2017)

D'autres approches IPPS dans un job shop et job shop flexible mentionnées dans la littérature qui considèrent le makespan comme fonction objectif sont résumé dans le tableau 3.1. Le type de flexibilité des plans du processus et la méthode de résolution sont donnés pour chaque référence.

2.3. Applications basés sur la génération d'une solution initiale faisable

L'un des facteurs clés des techniques de résolution est la qualité de la solution initiale. La génération d'une bonne solution de départ de manière plus intelligente permet de définir

un voisinage effectif autour de cette solution et affecte la qualité de la solution d'ordonnancement. (Garey et al 1976 ; Wang et al 2010 ; Vilcot, G., Billaut 2011).

Tan (1998) a mis au point un modèle de programmation linéaire à nombres entiers mixtes (linear mixed-integer programming model - LMIPM) pour un problème d'intégration qui met en relation une recherche tabou avec la méthode séparation et évaluation (B&B). La RT est utilisée comme heuristique rapide pour trouver une solution initiale pour la procédure de B&B dans un environnement LMIPM. La solution initiale faisable est générée à l'aide d'une règle d'ordonnancement spécialement conçue à cet effet ; la règle de priorité le plus tôt d'achèvement en premier (Earliest completion time first ECT).

Auteurs	**Flexibilité des plans de processus**	**Méthode**
Guo et al (2009)	Opération, Séquencement, Traitement (machine, outil et flexibilité TAD)	Optimisation par essaim de particules (OEP)
Ozguven et al (2010)	Opération ; Séquencement ; Traitement (flexibilité de la machine)	Programmation linéaire en nombres entiers mixtes (MILP)
Lian et al (2012)	Operation, sequencing, processing (flexibilité de la machine)	Algorithme compétitif impérialiste (ACI)
Lihong et Shengping, (2012)	Opération ; Séquencement ; Traitement (flexibilité de la machine)	Algorithme génétique amélioré (IGA)
Seker et al (2013)	Opération ; Séquencement.	Modèle neuro-flou
Zhang et Wong (2014)	Opération ; Séquencement ; Traitement (flexibilité de la machine)	Optimisation améliorée des colonies de fourmis (A-OCF)
Jin, et al (2015)	Opération ; Séquencement ; Traitement (flexibilité de la machine)	Algorithme hybride d'optimisation de l'accouplement des abeilles mellifères (HBMO)
Zhang et al (2015)	Opération ; Séquencement ; Traitement (flexibilité de la machine)	Monte Carlo Simulation

Tableau 3.1. Revue de la littérature relative à l'IPPS

Li et al (2010), considèrent un AG qui intègre différentes stratégies pour générer la population initiale et sélectionne les individus pour la reproduction et reproduire de nouveaux individus. Ensuite, la RT et l'algorithme de hill climbing algorithme (AHC) ont été réalisés pour effectuer l'affectation des machines et le séquencement des opérations, respectivement. Le voisinage dans RT a envisagé de réduire le nombre d'opérations critiques, et le voisinage

d'AHC a mis l'accent sur l'échange et l'insertion d'opérations critiques publiques afin de minimiser le makespan dans un job shop flexible.

L'AG amélioré, ainsi que le modèle mathématique développé, ont été présentés dans (Qiao et Lv 2011) comme un exemple de solution au problème de l'IPPS. Afin d'améliorer les performances d'optimisation, ils ont proposé une nouvelle méthode de sélection initiale pour les plans de processus, en plus d'une nouvelle méthode de représentation génétique pour la planification des plans et des schémas pour les opérateurs génétiques.

Yulianty et Ma'ruf (2013), ont mis au point des modèles mathématiques pour le PJSF avec temps de traitement contrôlable et temps d'arrêt prévu grâce à l'utilisation d'une approche prédictive avec minimisation des coûts de ré-échelonnement et des retards en tant que fonctions objectives. Les auteurs ont mis au point des méthodes mathématiques pour élaborer une solution initiale qui détermine l'attribution des jobs et les temps de traitement des jobs. La solution initiale est ensuite utilisée pour le ré-ordonnancement des jobs en tenant compte des temps d'arrêt prévus pour chaque machine et probabilité d'arrêt pour chaque opération.

2.4. Applications basées sur l'heuristique Shifting bottleneck SBH

En raison de sa grande efficacité pour résoudre le problème $J||C_{max}$, le concept SBH a attiré l'attention d'un grand nombre de chercheurs. Cette heuristique a été largement utilisée dans les méthodes hybrides, pour générer les solutions initiales et régénérer des solutions lorsqu'il n'y a pas d'amélioration au cours de certaines générations.

Monch et Zimmermann (2007), mènent des expériences de simulation dans un job shop afin d'évaluer la performance de SBH modifié. Il s'avère que l'utilisation de procédures de solution de sous-problèmes proches de l'optimum permet d'obtenir de meilleurs résultats que les méthodes basées sur la répartition.

La recherche effectuée par (Gen et al (2009)) a mis au point une hybridation d'un algorithme génétique avec décalage de SBH pour le PJSF. Ils ont généré la population de manière aléatoire afin de maintenir la diversité de la population d'individus.

Bulbul (2011) propose une variante SBH où le schéma classique de ré-optimisation est remplacé par une procédure de recherche tabou.

Braune et al (2012), proposent une méthode de solution exacte pour les problèmes d'ordonnancement d'une seule machine, typiquement dus à la décomposition des ateliers job shop en fonction du SBH et des retards de production pondérés. Les sous-problèmes rencontrés sont caractérisés par des contraintes de priorité retardée, de multiples échéances locales par opération et une fonction objective qui est donnée par une somme pondérée des valeurs maximales de retard. Récemment, ils ont combiné un SBH avec une méthode de recherche locale itérative pour ré-optimiser des machines déjà programmées. Les expériences de calcul montrent que cette méthode hybride améliore les résultats existants pour les instances de référence. De plus, cette combinaison du SBH avec une méthode de recherche locale itérative s'applique aux grandes instances (plus de 100 travaux et 20 machines) (Braune et Zapfel 2016).

2.5. Applications basées sur la Recherche Tabou (RT)

De nombreux chercheurs se sont penchés sur l'utilisation de la RT dans les systèmes de production. Les contributions importantes sont examinées ci-après. Autres applications de cette technique sont présentées dans le tableau 3.2.

Dauzere-Peres et al (1998) proposent un algorithme de recherche tabou 'RT' pour minimiser le makepan dans les ateliers multi-machines avec machines flexibilité. Les routages non linéaires et la représentation disjonctive des problèmes graphiques sont pris en compte. Une version améliorée de cette approche est présentée dans (Dauzere-Peres et Pavageau 2003).

L'hybridation de la recherche et du déplacement des tabous avec l'heuristique de shifting bottleneck est déjà adapté pour réduire au minimum la durée de vie du produit dans un environnement job shop, mais avec un environnement de travail différent (Pazella 2000). La figure 3.7 illustre cette hybridation.

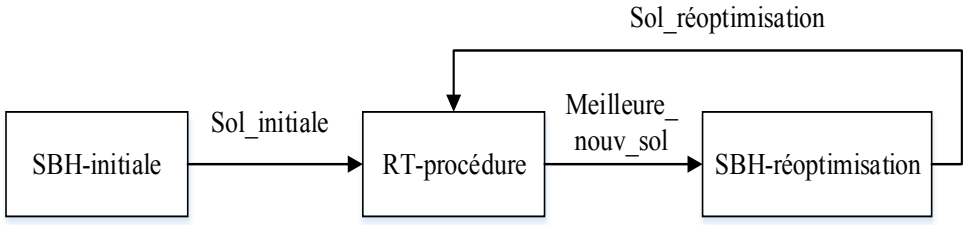


Figure 3.7 Les principaux modules de l'algorithme RTSB (Pezzella 2000)

Baykasoğlu et Özbakır, (2009) ont proposé un modèle IPPS qui comprend deux parties. La première partie était un générateur de plan de processus générique (GPP) pour générer le plan de processus final. La deuxième partie était une heuristique basée sur des règles de priorité pour générer des ordonnancements réalisables. Un algorithme MORT (multi-objectives recherche tabou/MOTS (Multiple Objective Tabu Search) a été utilisé pour trouver un ordonnancement optimal pour deux objectifs qui étaient "temps de flux" et "coût du plan du processus". Ils ont conclu que les coûts de plan de processus diminuent à mesure que la flexibilité du plan de processus augmente.

Li et al (2010a) ont suggéré une hybridation de l'algorithme recherche tabou avec un bloc critique public rapide de la structure de voisinage (appelée TSPCB) pour minimiser le makespan dans job shop flexible. Ils ont conclu que l'algorithme TSPCB proposé est supérieur à plusieurs algorithmes récemment publiés en termes de qualité de la solution, de vitesse de convergence et d'efficacité.

Li et al (2010b) ont proposé une approche hybride qui synthétise les avantages de l'AG et de la RT pour résoudre le problème IPPS. La première partie du chromosome était une chaîne de plan de processus alternatif ; la deuxième partie était une chaîne de plan de l'ordonnancement et la troisième une chaîne de machine. La troisième partie sélectionne l'ensemble de machines des opérations correspondantes de tous les jobs pour minimiser makespan. Les auteurs ont conclu que l'algorithme proposé était efficace et acceptable pour le problème IPPS.

Bozejko et al (2010) dans leur étude, ont mis en évidence une recherche tabou parallèle basée sur un nouveau voisinage de golf pour le PJSF. Cette recherche se compose de deux modules principaux : la sélection des machines exécutées séquentiellement, et l'ordonnancement des opérations.

2.6. Applications basées sur l'algorithmede kangourou (AK)

La plupart des travaux liés à l'application de l'AK, l'utilisent comme niveau d'une méthode d'hybridation. Les diverses approches utilisant l'algorithme de kangourou KA adoptées par les chercheurs sont résumées ci-dessous.

Minzu et Beldiman (2003) proposent une méthode hybride pour un problème d'ordonnancement à machine unique. Cette méthode combine un algorithme génétique qui effectue l'analyse de la phase de diversification et de l'optimisation, et un algorithme kangourou qui effectue la phase d'intensification. Chaque solution individuelle de cette population est considérée comme une solution initiale pour l'AK qui dispose d'un outil de diversification empirique.

Serbencu et al (2007), ont proposé une méthode hybride formée par colonie de fourmis et un algorithme de kangourou pour résoudre le problème d'ordonnancement d'une seule machine. Ce travail est basé sur le pouvoir collaboratif de l'ACS et la capacité d'intensification de KA. La figure suivante (3.8) décrit la technique de l'hybridation adoptée.

Marmier (2007) s'est intéressé dans sa thèse à un problème d'affectation et d'ordonnancement en prenant en compte les compétences des ressources humaines. Il a proposé une approche de résolution dynamique pour un problème mono-critère d'affectation et d'ordonnancement des activités de maintenance. Il fait appel à une méthode d'amélioration par modification partielle de l'ordonnancement, inspirée de la méthode du kangourou.

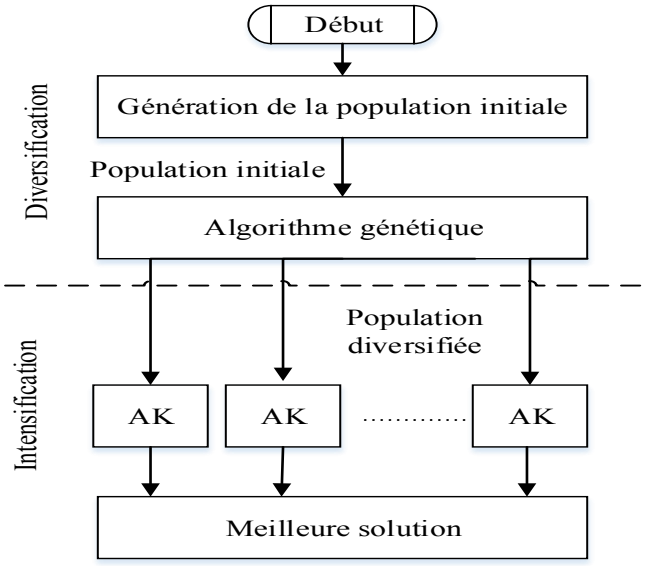


Figure 3.8 Le système parallèle hybride (Serbencu et al 2007)

Auteurs	Fonction objectif	Description de la technique utilisée
Brandimarte (1993)	Makespan Retard total pondéré	Approche hiérarchique basée sur la RT en décomposant le problème en sous-problèmes d'affectation et d'ordonnancement.
Dauzère-Pèrès et Paulli (1997)	Makespan	RT avec modèle graphique disjonctif.
Mastrolilli et Gambardella (2000)	Makespan	RT en réduisant l'ensemble des voisins possibles à un sous-ensemble qui contient toujours le voisin avec le plus faible makepan.
Scrich et al. (2004)	Retard total	RT hiérarchique et à démarrages multiples
Saidi-Mehrabad et Fattahi (2007)	Makespan	RT avec une méthode d'échange par paire adjacente pour générer les voisinages, PJSF avec temps de changement en fonction de la séquence.
Ennigrou et Ghédira (2008)	Makespan	Systèmes multi-agents à base de RT composés de trois agents classes : agents de job, agents de ressources et un agent d'interface.
Bozejko et al. (2012)	Makespan	RT parallèle implémenté sur multi-GPU.
Barzegar et al. (2012)	Makespan	RT en réduisant l'ensemble des voisins possibles à un sous-ensemble qui contient toujours le voisin avec le plus faible makepan.
Roshanaei et al. (2013)	Retard total	RT hiérarchique à démarrages multiples
Yuan et Xu (2013)	Makespan	RT parallèle avec deux modules principaux : sélection des machines exécutées séquentiellement, et l'ordonnancement des opérations module exécuté en parallèle.
Farughi et al. (2013)	Makespan	RT parallèle implémenté sur multi-GPU
Jia et Hu (2014)	-Makepan, -Charge de travail de la machine la plus chargée ; -Charge de travail totale des machines.	RT de liaison de voies avec suivi de saut en arrière

Tableau 3.2. Revue de la littérature relative à l'application de la recherche tabou

2.7. Applications basées sur les algorithmes hybrides

Pour résoudre les problèmes IPPS/FJSP, les méthodes hybrides donnent des résultats prometteurs. Les méthodes hybrides sont de plus en plus populaires pour résoudre des problèmes d'optimisation complexes. Leur succès est principalement attribuable à leur efficacité, bien qu'elles puissent prendre beaucoup de temps et être difficiles à mettre en œuvre (Talbi et al. 2007).

Chan et al (2006) ont présenté un système immunitaire artificiel incorporé au contrôleur de logique floue (appelé algorithme AIS-FLC) pour résoudre efficacement les problèmes complexes du monde réel de l'intégration de la planification des processus et de l'ordonnancement en considérant la réduction du temps de calcul ainsi que le taux de convergence.
En 2009, ils ont motivés par les inconvénients des approches fondées sur l'AG et le RS, en proposant un nouvel algorithme ESCSA (Enhanced Swift Conveging Simulated Annealing), qui résume les caractéristiques essentielles de AG, RS et de contrôleur de logique floue et surmonte les problèmes de leurs lacunes, pour résoudre le problème d'intégration de la planification et de l'ordonnancement des processus visant à minimiser le makespan.

Lei et Guo (2014), examinent la technique de recherche de variable de voisinage RVV avec deux procédures de recherche de voisinage, Dual-ressource (machines et travailleurs) dans un environnement JSF.

Un algorithme hybride discret de lucioles discrètes est présenté pour résoudre le problème d'ordonnancement multi-objectif du job shop flexible avec des contraintes limitées en ressources. La contrainte principale de ce problème est que chaque opération d'un job doit suivre une séquence de processus et chaque opération doit être traitée sur une machine affectée. Ces contraintes sont utilisées pour équilibrer la limitation des ressources et la flexibilité de la machine (Karthikeyan et al). Afin de mieux comprendre cette hybridation, la structure adaptée est donnée dans la figure 3.9 :

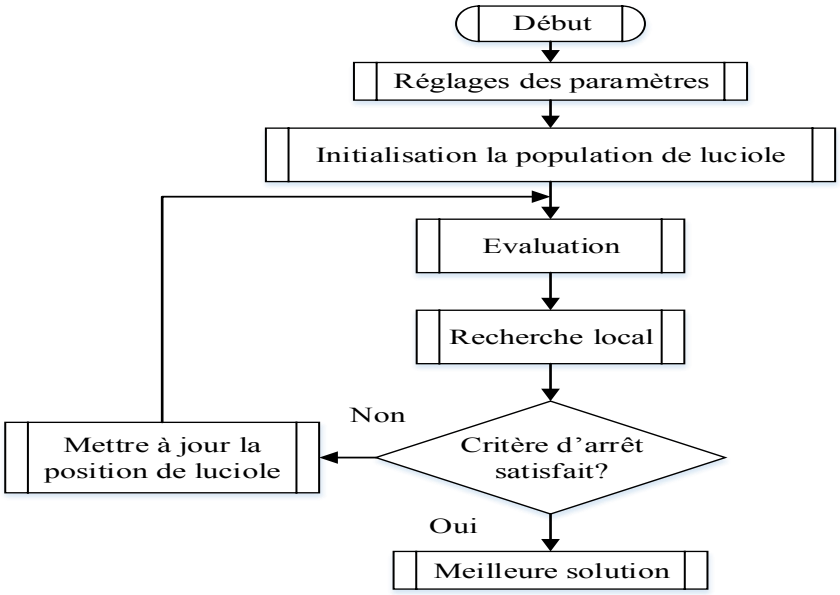


Figure 3.9. La structure de l'approche hybride proposée

Haddadzade et al (2014), ont proposé un nouvel algorithme hybride pour résoudre l'IPPS avec un temps de traitement stochastique. Cette approche combine le recuit simulé et la recherche tabou (RT/ TS).

Botsali (2016) a comparé deux heuristiques différentes basées sur des algorithmes génétiques et un algorithme de recuit simulé, pour le problème FJSP-APP, afin de réduire au minimum le makespan.

Zhang (2019) a formulé un nouveau modèle mathématique et la génération évolutive d'ensembles de règles d'ordonnancement pour l'efficacité énergétique de job shop flexible avec sélection de la machine, choix du job la prise de décision de la machine en marche-arrêt.

Plus récemment, Toshev (2019) développe une technique combinée OEP et RT pour cette classe de problèmes d'optimisation se présente naturellement. Un nouvel algorithme hybride est présenté dans la figure suivante (3.10) et sa performance est illustrée sur 12exemples de tests de référence.

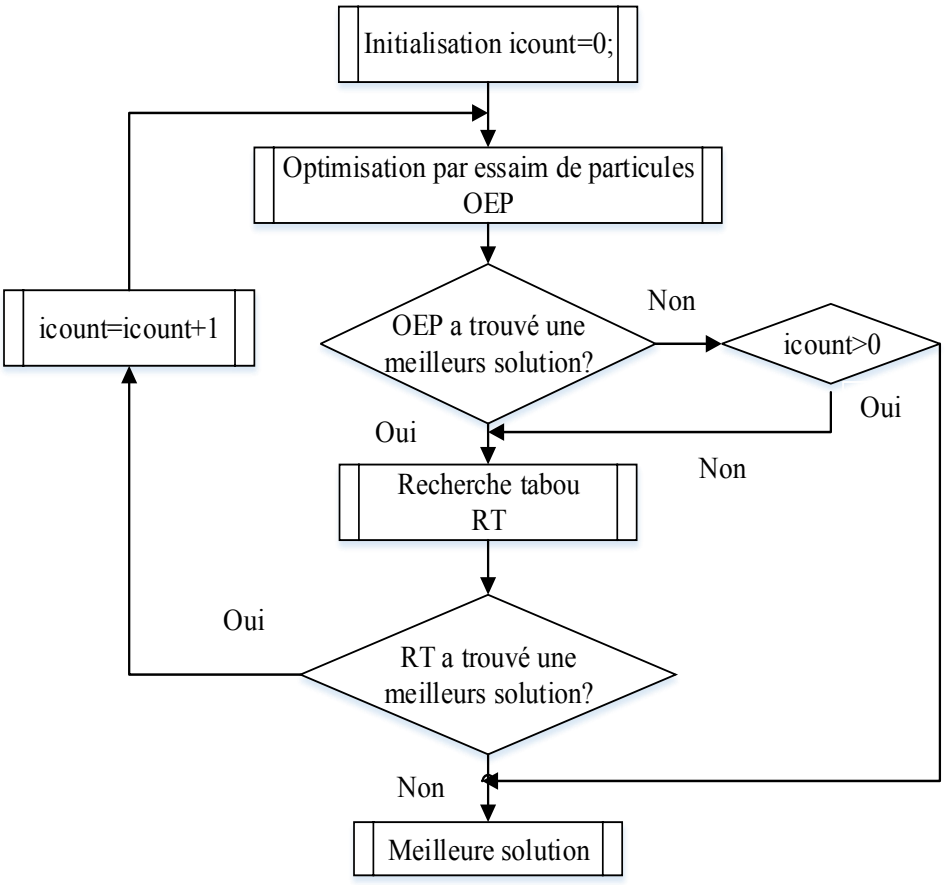


Figure 3.10. L'organigramme de l'algorithme OEP + RT

Comme ce travail traite de l'hybridation, Le tableau 3.3 présente les applications de diverses techniques hybrides employées par les chercheurs dans un IPPS et PJSF.

Auteurs	Fonction objectif	Description de la technique utilisée
Kacem et al. (2002)	-Makespan -Charge de travail de la machine la plus chargée -Charge de travail totale des machines	Recherche Approchée de localisation + AG contrôlée
Tanev et al. (2004)	-Proportion minimale de jobs en retard -Variance du temps d'écoulement -Nombre de changements de moules -Efficacité maximale des machines	AG + Règles de priorité
Liouane et al. (2007)	-Makespan	OCF+RT
Tay and Ho (2008)	-Makespan -Retard moyen -Temps d'écoulement moyen	Programmation génétique + Règles de priorités
Motaghedi-Larijani et al. (2010)	-Makespan -La charge de travail de la machine la plus chargée -Charge de travail totale des machines	AG + Hill climbing
Frutos et al. (2010)	-Makespan -Coûts de production	Algorithme mémétique basé sur le NSGAII agissant sur deux chromosomes + RS
Mahdavi et al. (2010)	-Makespan	Décision support system (DSS) basé sur la simulation.
Xing et al. (2011)	-Makespan	OCF + AG
Al-Hinai et Elmekkawy (2011)	-Makespan	AG + Recherche locale (RL)
Zhang et al. (2012)	-Makespan	AG + RT. JSF avec des contraintes de transport et des délais de traitement limités.
Dalfard et Mohammadi (2012)	-Makepan, -Temps de traitement moyen -Retard avec pénalité	GA+SA. JSF avec machines parallèles et contraintes de maintenance

Barzegar et al (2012)	-Makespan	Algorithme GS de recherche + réseaux de pétri colorés.
Srinivas, Ramachandra Raju et Rao, (2012),	-Makespan	OCF + PSO
Li et al (2012)	-Makepan, -Temps de traitement moyen -Retard avec pénalité	OCF + un système basé sur les agents
Barzegar et Motameni (2013)	Makespan	Algorithme de recherche gravitationnelle (GS) + OEP
Araghi et al (2013)	Makespan	AG+VNS avec fonction d'affinité. JSF avec temps de changement en fonction de la séquence
Haddadzade et Zarandi, (2014)	Makespan	AG+RS
Palacios et al. (2015)	Makespan	AG+RT pour le PJSF flou
Yu et al (2015)	Makespan	AG+OEP
Kundakcı et Kulak (2016)	Makespan	AG hybride + RT avec différents voisinages appliquée pour la recherche locale pour le JS
Nouri et al (2018)	Makespan	GATS + HM qui un algorithme hybride basé sur un modèle multi-agent holonique en cluster

Tableau 3.3. Revue de la littérature relative à l'application des méthodes hybrides

3. Approches de l'IPPS

Trois approches pour les IPPS sont différenciées en fonction de la façon dont l'information est prise en compte lors de la prise de décision, à savoir la planification non linéaire du processus (NLPP), le processus en boucle fermée. (CLPP) et la planification des processus distribués (DPP). La description détaillée de ces trois mécanismes d'intégration est la suivante.

3.1. Planification non linéaire des processus NLPP

NLPP peut aussi être appelé comme planification flexible des processus (Saygin et Kilic, 1999 ; Zhang et al. 2003 ; Gan et Lee, 2002 ; Kim et al. 1997), de planification multiprocessus (Li et al. 2010) ou de planification alternative (Yang et al. 2001 ; Kim et Egbelu, 1999 ; Usher,

2003 ; Kis, 2003 ; Nasr et Elsayed, 1990). La figure 3.11 présente l'organigramme de base du NLPP.

Dans le cadre de cette approche : (Rakesh et al 2013) :

- Les plans de processus multiples (MPP) pour chaque pièce avant son entrée dans l'atelier sont d'abord créé par la prise en compte de la flexibilité de l'opération (possibilité d'effectuer une opération sur plus d'une machine), flexibilité de séquencement (possibilité d'inter-changer les l'ordre dans lequel les opérations de fabrication requises sont effectuées) et la flexibilité de traitement (possibilité de produire la même caractéristique de production avec les opérations alternatives ou séquence d'opérations).
- Tous ces plans de processus possibles sont classés en fonction des critères de planification des processus. (tels que le temps total d'usinage et le temps total de production) et stockés dans un processus de données de planification.
- Le plan de première priorité est toujours prêt à être soumis au moment où le job est requis alors l'ordonnancement prend la décision en temps réel.
- Si le premier plan prioritaire ne s'intègre pas bien dans l'état actuel de l'atelier, le deuxième plan prioritaire sera mis en œuvre. Le plan des priorités est fourni à l'ordonnancement.
- Cette procédure est répétée jusqu'à ce qu'un plan approprié soit identifié à partir d'un plan déjà généré.

.

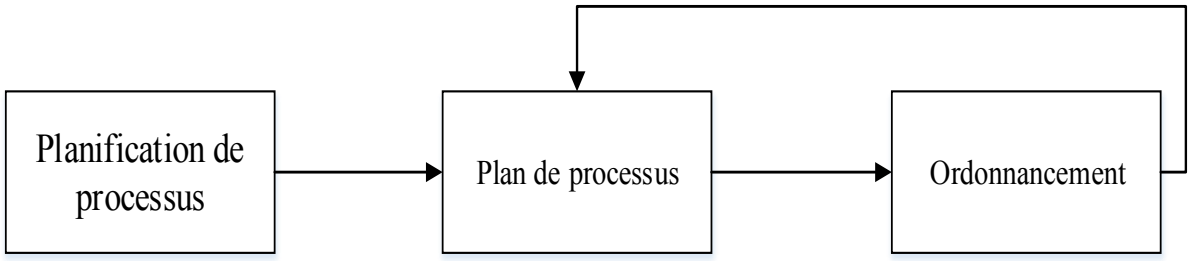


Figure 3.11 Approche non linéaire NLPP

3.2. Planification en boucle fermée CLPP

CLPP est également appelé planification en ligne des processus (Baker and Maropoulos, 2000 ;Kumar et Rajotia 2006), la planification des processus en temps réel (Phanden et al 2013) ou la planification dynamique des processus (Chang et Chen, 2002 ; Lim et Zhang, 2000).

Contrairement au NLPP, CLPP génère les plans de processus d'un type de pièce au moyen d'un retour d'information dynamique à partir de l'ordonnancement de la production et les ressources disponibles. Chaque fois qu'une opération est effectuée sur l'atelier, une description de la pièce à usiner basée sur les caractéristiques est étudiée afin de déterminer l'opération suivante et allouer les ressources. Cette approche prend en compte le comportement dynamique de la fabrication en considération. Ainsi, le statut en temps réel est crucial pour CLPP (Zhang et Merchant, 1993). La Figure 3.12 montre l'organigramme de base de la CLPP.

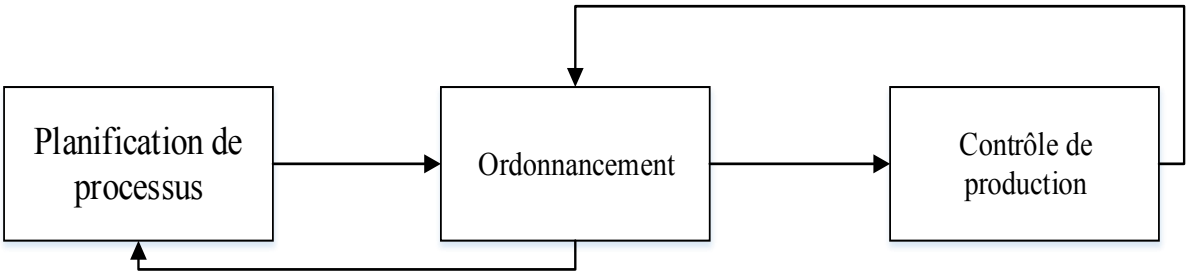


Figure 3.12 Approche de boucle fermée CLPP

3.3. Planification des processus distribuée DPP

DPP est également appelée approche juste-à-temps ou approche progressive. Cette approche exécute simultanément les fonctions de planification des processus et d'ordonnancement en les divisant en deux phases (Huang et al. 1995).

- La première phase est la pré-planification. Dans cette phase, la fonction de planification de la production en processus analyse le job en fonction des données du produit.
- La deuxième phase est la planification finale, qui fait correspondre les opérations de travail requises aux capacités d'exploitation de la ressource de production disponible.

L'intégration aura lieu au moment où les ressources sont disponibles et les jobs sont requis. Il en résulte une planification dynamique des processus et un ordonnancement de la production soumis aux contraintes en temps réel.

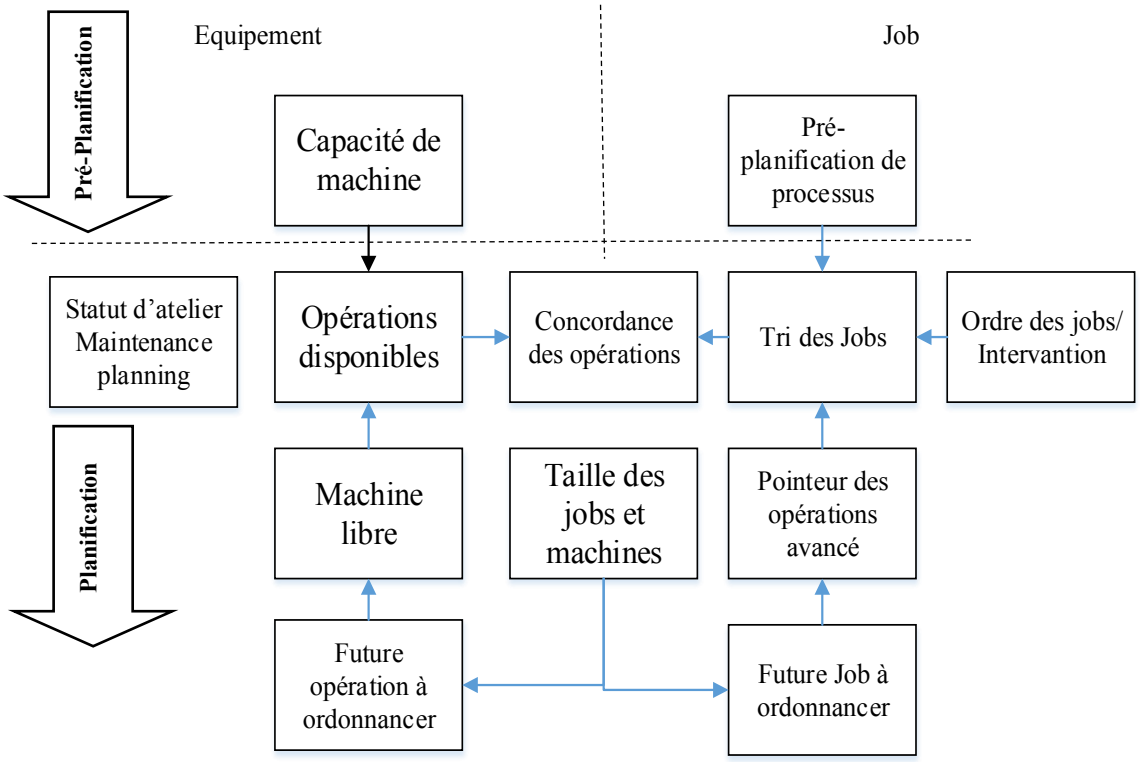


Figure 3.13 Approche de la planification de processus distribué DPP

La plupart des approches récentes qui appartiennent à cette classe sont basées sur les systèmes multi-agents car les agents logiciels peuvent être implémentés de manière distribuée. La figure 3 13 présente le diagramme de base du DPP.

4. Schéma itératif pour IPPS

Selon Brandimarte et Calderini (1995), le problème IPPS peut être décomposé en deux sous-problèmes :

1. Déterminer les plans de processus en sélectionnant les séquences pour les produits/jobs.
2. Planifier les opérations des sous-parties pour les plans de processus choisis.

Selon cette approche, le planificateur de processus reçoit un retour d'information de l'ordonnanceur qui, à son tour, peut analyser le retour d'information des machines.
Le planificateur de processus est en mesure de sélectionner et de comparer les plans de processus en fonction des commentaires d'ordonnanceur. Nous sommes intéressés par la sélection d'un plan de processus qui mène à la plus petite valeur de makepan pour les commandes données.

La procédure commence par la sélection d'un plan de processus initial. La qualité de ce plan est évaluée par l'ordonnanceur. Basé sur la valeur du makepan, le planificateur sélectionne ensuite itérativement les nouveaux plans de processus qui sont évalués par l'ordonnanceur.
La décomposition du problème en deux sous-problèmes permet d'appliquer des algorithmes sophistiqués tant au niveau de la planification des processus qu'au niveau de l'ordonnancement. Le schéma IPPS global est décrit dans la figure 3.14.

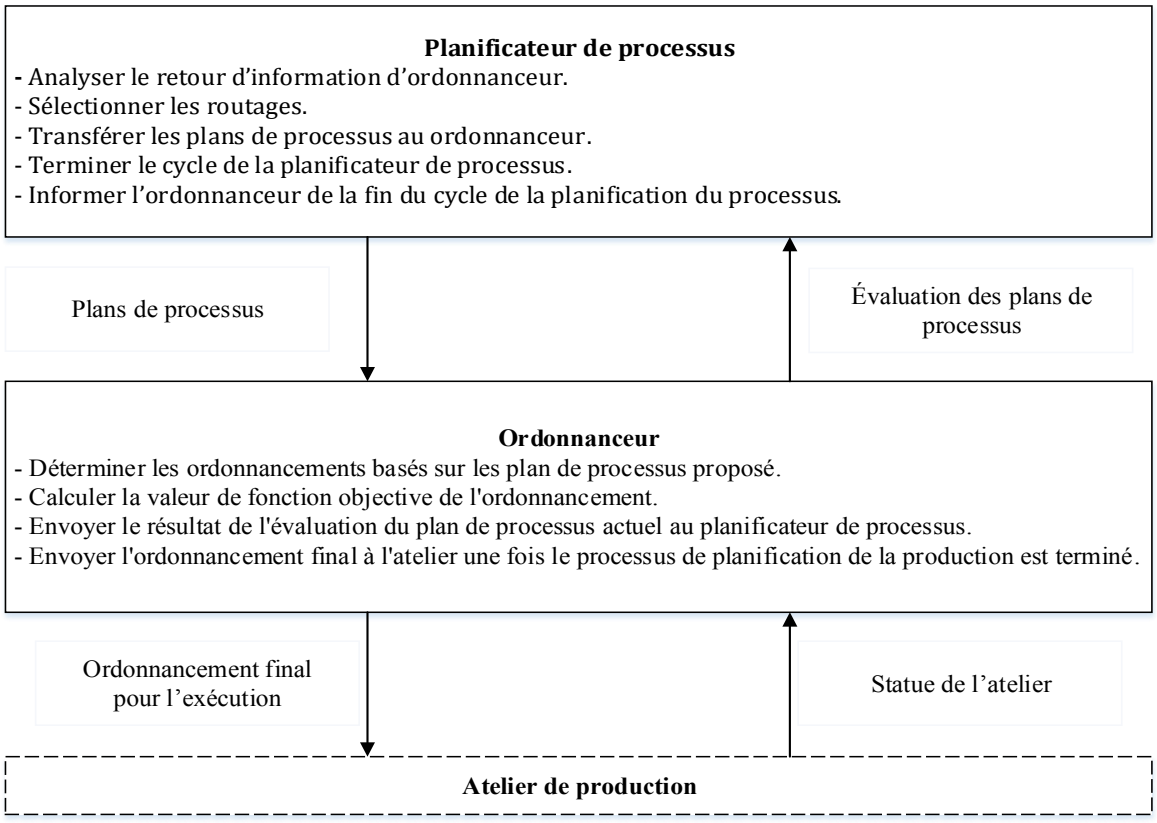


Figure 3.14. Schéma de décomposition itératif pour IPPS

5. Analyse de la revue de la littérature

L'examen de la littérature révèle les points suivants.

- ✓ L'environnement de job shop flexible est un type important de configuration d'atelier. L'ordonnancement d'un job shop est typique et NP - difficile. De plus, l'analyse de l'ordonnancement fournit un aperçu de qualité de la solution rencontrée dans des systèmes plus réalistes et plus complexes.

- ✓ Auparavant, les chercheurs s'intéressaient principalement soit à la planification des processus, soit aux fonctions d'ordonnancement, car ces fonctions sont considérées séparément dans un système de fabrication conventionnel. Le fait de considérer séparément la planification et l'ordonnancement des processus entraîne plusieurs problèmes et le comportement imprévisible du système de production. Ainsi, un besoin est estimé qu'il fallait intégrer ces deux fonctions afin de faire face à la nature dynamique du système de production.

- ✓ La plupart des chercheurs ont considéré un seul objectif pour résoudre les problèmes IPPS/JSF tels que le makespan, le temps d'écoulement moyen (mean flow time), le nombre de pièces en retard et le retard moyen. La réduction du makespan avait été la performance la plus largement utilisée.

- ✓ Les techniques/méthodes hybrides ont été les méthodes les plus populaires en s'appuyant sur les forces de chacune des méthodes afin de trouver de meilleures solutions.

- ✓ Comme l'espace de recherche du problème IPPS est de plus en plus grand, le temps de calcul augmente de façon considérable. Par conséquent, une approche d'optimisation hybride peut trouver une solution optimale rapidement. Dans ce sens, une combinaison d'heuristique et de méta-heuristique peut être utile.

- ✓ Il est nécessaire d'officialiser une approche IPPS qui peut être mise en œuvre dans un atelier avec les départements de planification des processus et d'ordonnancement existants et qui peut intégrer rapidement la planification et l'ordonnancement. Cela peut se faire en combinant diverses approches de sorte que l'approche formalisée est en mesure de tirer les avantages de la combinaison des approches. Il peut en résulter une amélioration du niveau d'échange d'information entre les fonctions de planification des processus et d'ordonnancement.

- ✓ L'AG est une technique d'optimisation prometteuse qui peut être utilisée pour optimiser l'ordonnancement des jobs dans les ateliers JSF/IPPS.

- ✓ Il existe plusieurs règles de priorité utilisées pour la planification des commandes en atelier, telles que Premier arrivé, premier servi (FIFO), Temps de traitement le plus court (SPT), Temps de traitement le plus long (LPT), date d'échéance au plus tôt (EDD), La plupart des travaux restants (MWKR), le moins de travaux restant (LWKR).

6. Conclusion

La planification des processus et l'ordonnancement intégrés (IPPS) devient un sujet de recherche actuel et attire de plus en plus l'attention des chercheurs et des ingénieurs. Les avantages d'IPPS sont d'augmenter la faisabilité et l'optimalité de la production en combinant à la fois les fonctions de planification des processus et l'ordonnancement.

Ce chapitre donne un aperçu de l'IPPS et le PJSF. L'intérêt porté à la planification, à l'ordonnancement et à l'intégration de ces deux fonctions a donné lieu à l'élaboration de nombreuses approches, comme nous avons discuté dans ce chapitre. Le besoin d'IPPS a été identifié. En outre, les contributions de différents chercheurs ont été brièvement présentées.

Avec les progrès de la puissance de calcul, les techniques/méthodes de résolution sont devenues de plus en plus efficaces et puissantes. La métaheuristique a été utilisée plus largement que d'autres méthodes d'analyse. Les techniques les plus populaires ont été les méthodes hybrides, car ces techniques tirent parti des avantages de chacune de ces méthodes pour trouver de meilleures solutions.

En raison de leur capacité, Le chapitre suivant sera consacré à la présentation de l'approche développée basée sur l'hybridation.

CHAPITRE 4

Une approche hybride pour la résolution du problème job shop flexible

Ce chapitre est consacré à l'adaptation de notre approche hybride aux problèmes de type Job Shop Flexible. Une comparaison avec des méthodes connues par leur performance sur les problèmes de Job Shop Flexible montre la qualité de notre méthode.

1. Introduction

Le scénario du marché global mondial exige un ordonnancement flexible de production pour rester compétitif dans l'économie actuelle. Plusieurs problèmes d'ordonnancement réels peuvent être formulés en problèmes d'ordonnancement d'atelier job shop flexible, en raison de la flexibilité de production. Ce type de système conduit souvent à une utilisation efficace des ressources.

Les systèmes de fabrication flexibles (Flexible Manufacturing System FMS) sont des systèmes de production hautement intégrés. Leur introduction dans l'industrie a été une nouvelle étape importante dans le développement du système de production entièrement automatisé. Il est composé de machines à commande numérique et de dispositifs automatiques de manutention des matériaux (p. ex. robots, véhicules guidés automatisés, etc.) (Nouri et al 2016). Les interrelations entre ses composants sont très complexes dans un réseau contrôlé par ordinateur (Brown et al 1984).

Parmi les diverses configurations de FMS, le job shop flexible est présenté. Le problème de JSF est l'un des problèmes les plus importants de l'industrie et aussi une extension de l'application du problème classique de job shop (JSP), qui offre une approximation plus proche des problèmes réels (Gao et al 2007). L'ordonnancement est devenu un facteur critique pour les applications industrielles réelles en particulier pour un job shop flexible.

Nous prolongeons le problème classique de job shop pour permettre à la possibilité qu'une machine puisse être capable d'exécuter plus d'un type d'opération.
Le problème de job shop flexible se réfère à une très forte complexité de problème de décision. Il est NP-hard depuis qu'il est considéré comme une extension du problème classique job shop dont il a été démontré NP-hard (Garey et al 1976). De plus, il est peu probable qu'il trouve une solution optimale dans un temps de calcul raisonnable, en particulier pour les cas de problèmes de grande taille.

Les problèmes liés au job shop flexible se divisent en deux sous-problèmes :

- La première est d'affecter chaque opération à une sortie machine d'un ensemble de machines alternatives (sous-problème de routage).
- La seconde est d'ordonner les opérations assignées sur les machines pour atteindre l'objectif souhaité (sous-problème d'ordonnancement).

Les algorithmes comme le recuit simulé, la recherche de tabou, la recherche local, sont des méthodes itératives d'amélioration de la solution, ce qui signifie que seule la solution est améliorée par une procédure itérative. Ce type de méthodes a la capacité d'intensifier la recherche locale et de détecter les minima locaux. Au cours des dernières années, des métaheuristiques hybrides ont été développées, donnant des résultats très intéressants.

La contribution principale de cette recherche est de proposer une méthode hybride efficace avec un effort de calcul acceptable, permettant d'évaluer et d'explorer intelligemment l'espace de recherche dans le but de minimiser le makepan. Dans la méthode proposée, une hybridation séquentielle est considérée, est basée sur trois étapes :

1) La première étape est la génération d'une solution initiale à l'aide de la fonction shifting bottleneck heuristique (SBH). À mesure que le SBH est l'une des heuristiques les plus réussies pour le problème J||Cmax **(**Pinedo, 2008**),** elle est d'un grand intérêt pour l'utiliser comme solution initiale pour la RT.

2) La deuxième étape est basée sur la RT qui a une bonne capacité de recherche locale est appliquée pour effectuer l'exploitation. Cette technique est apparue comme l'une des stratégies de recherche locale les plus efficaces pour résoudre les problèmes d'ordonnancement de jobs shop/job shop flexible. La solution trouvée par la procédure SBH devient la solution initiale de RT. Une initialisation efficace de la solution est un aspect essentiel d'une solution en termes de qualité de solution et de temps de calcul.

3) La troisième étape est basée sur l'algorithme de kangourou. AK possède une capacité de recherche globale et peut très bien équilibrer l'intensification et la diversification. l'AK proposée a une capacité de l'exploration de l'espace des solutions à l'aide de la recherche tabou (RT). La stratégie de l'exploration consiste à stocker les propriétés des zones visitées et la mise en place d'un mécanisme pour s'éloigner de ces zones.

2. Formulation du problème

Un atelier job shop flexible se compose de n job indépendants J, chaque job J_i ($1 \leq i \leq n$) se compose d'une séquence de n_i opérations dans un ordre prédéfini. Chaque opération O_{ij} ($i=1,2,...,n$; $j=1,2,...,n_i$) de job J_i peut être traitée par un ensemble de m machines alternatives M_k, ($1 \leq k \leq m$), et sa durée de traitement dépend de la machine utilisée. P_{ijk} indique le temps d'exécution de l'opération O_{ij} sur la machine k.

2.1. Modèle mathématique

Le problème de Job Shop Flexible que nous considérons est soumis aux hypothèses suivantes :

- Tous les jobs et les machines sont disponibles au temps zéro.
- Les machines sont disjonctives et non-reliées, c'est-à-dire que les durées opératoires dépendent de la ressource choisie.
- Chaque machine ne peut traiter qu'une seule opération à la fois.
- L'ordre des opérations pour chaque job est prédéfini et ne peut pas être modifié.
- Les opérations appartenant à des jobs différents peuvent être traitées en parallèle.
- Les opérations sont non interruptibles.
- Les durées des opérations sont entières.
- Nous n'imposons aucune limite sur le nombre d'opérations qui peuvent attendre entre deux machines, les stocks placés à ces endroits sont considérés en quantité infinie.
- Les temps de transport et/ou de maintenance sont considérés comme nuls, ou éventuellement dans certains cas, ils sont contenus dans la durée d'exécution des jobs.
- Il n'y a pas de préemption de jobs, c.-à-d., une opération / une fois démarré, le job ne peut pas être arrêté.
- L'exécution de chaque opération nécessite une machine d'un ensemble de machines alternatives.
- Pas de pannes sur les machines.

Les indices, les paramètres et les variables de décision du modèle mathématique sont exposés comme suit :

- Notations et paramètres

i	Indices pour les jobs
j	Indices pour les opérations
k	Indices pour les machines
$O_{i,j}$	$J^{iéme}$ opération du job i
N	Ensemble des opérations
M	Ensemble des machines
$M_{i,j}$	Ensemble des machines disponible pour l'opération $O_{i,j}$
$P_{i,j,k}$	Temps de traitement de l'opération $O_{i,j}$ sur la machine k
A	Nombre positif de valeur très grande

- Variable De décision

C_{max}	Makespan
$C_{i,j}$	Variable continue pour le temps de réalisation de la $j^{ième}$ opération du job i
C_i	Le temps de réalisation du job i
$S_{i,j}$	La date de début au plus tôt de l'opération $O_{i,j}$
$S_{k,r}^{m}$	La date de début au plus tôt de l'opération effectuée sur la machine k en priorité r
q_r	Nombre d'opérations affectées à la machine k
$X_{i,j,k}$	$= \begin{cases} 1, & \text{si la machine k est selectionnée pour l'opération } O_{i,j} \\ 0, & sinon \end{cases}$
$Z_{i,j,k,r}$	$= \begin{cases} 1, & \text{si } O_{i,l,j} \text{ est traité sur la machine k avec la priorité r} \\ 0, & sinon \end{cases}$

- Objective 'Minimisation du makespan'

L'objectif consistant à réduire au minimum le makespan, qui est le temps maximum d'exécution de tous les jobs, il est défini comme suit :

$$\min C_{\max} = \max_{1 \le i \le n}\{C_i\}$$

- Garantissent que la relation de précédence entre deux opérations sur la même machine :

$$S_{i,j} + \sum_{k \in M_{i,j}} (P_{i,j,k} \cdot X_{i,j,k}) \le S_{i,j+1} \qquad \text{For j = 1,…, } n_{i-1} \text{ ; i = 1,…, n ;} \qquad (1)$$

- Chaque opération est affectée à une seule machine parmi les machines disponibles :

$$\sum_{k \in M_{i,j}} X_{i,j,k} = 1 \qquad \text{For j = 1,…, } n_i \text{ ; i = 1,…, n ; k = 1,…, m ;} \qquad (2)$$

- Chaque opération doit être traitée avec une priorité sur une machine :

$$\sum_r Z_{i,j,k,r} = X_{i,j,k} \qquad \text{For j = 1,…, } n_i \text{ ; i = 1,…, n ;} \qquad (3)$$

k = 1,…, m ; r = 1,…, q_k;

- Chaque machine ne peut traiter qu'une seule opération à la fois :

$S_{k,r}^m + P_{i,j,k} \cdot Z_{i,j,k,r} \leq S_{k,r+1}^m$ For j = 1,…, n_i ; i = 1,…, n ; k = 1,…, m ; r = 1,…, $q_{k-1;}$ (4)

- Chaque opération doit être lancée une fois que la machine qui lui est affectée est à l'arrêt et que l'opération précédente est terminée :

$S_{k,r}^m + (1 - Z_{i,j,k,r}) \cdot L \geq S_{i,j}$ For j = 1,…, n_i ; i = 1,…, n ; k = 1,…, m ; r = 1,…, q_k; (5)

$S_{k,r}^m \leq S_{i,j} + (1 - Z_{i,j,k,r}) \cdot L$ For j = 1,…, n_i ; i = 1,…, n ; k = 1,…, m ; r = 1,…, q_k; (6)

$S_{i,j} \geq 0$ For j = 1,…, n_i ; i = 1,…, n ;

$P_{i,j,k} \geq 0$ For j = 1,…, n_i ; i = 1,…, n ; k = 1,…, m ;

$S_{k,r}^m \geq 0$ For k = 1,…, m ; r = 1,…, q_k;

$X_{i,j,k} \in \{0,1\}$ For j = 1,…, n_i ; i = 1,…, n ; k = 1,…, m ;

$Z_{i,j,k,r} \in \{0,1\}$ For j = 1,…, n_i ; i = 1,…, n ; k = 1,…, m ; r = 1,…, q_k;

2.2. Schéma de l'approche hybride SBH-TS-KA pour JSF

La question centrale de toute approche approximative est de savoir comment orienter la recherche depuis la solution initiale jusqu'à la solution la plus efficace et les domaines prometteurs.

La structure globale de l'approche hybride est illustrée par le diagramme schématique de la figure 4.1.

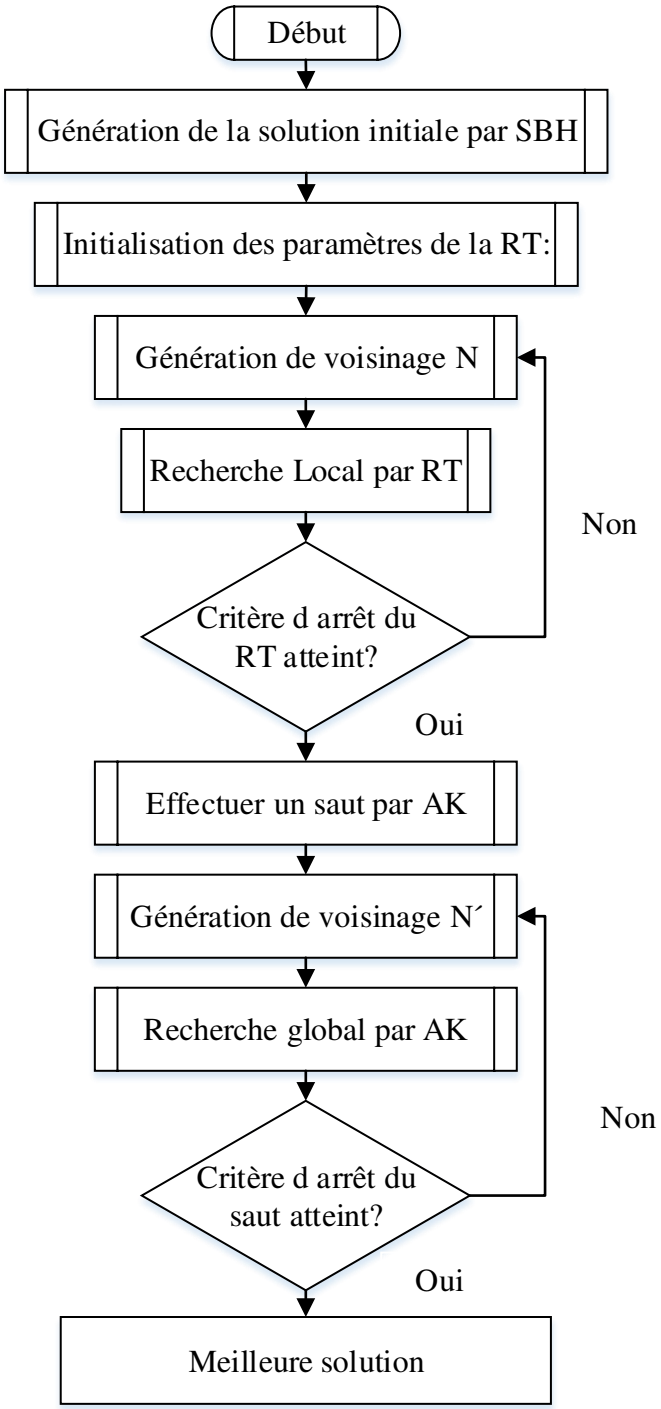


Figure 4.1 Le diagramme schématique de l'approche proposée

2.3. Fonctionnement de l'approche proposée dans un JSF

La procédure étape-par étape de l'approche adaptée illustrée dans la figure précédente est la suivante :

Étape 1 : Application de l'heuristique shifting bottleneck pour générer une solution initiale efficace.
Étape 2 : Initialiser les paramètres de la recherche tabou :
La liste tabou (TL) = Ø

Définir la solution obtenue par SBH comme solution initiale de RT ;
Étape 3 : Générer les nouvelles solutions de voisinage par la fonction de voisinage N, et Évaluer ces solutions ;
Étape 4 : Pour chaque solution, si son déplacement existe dans TL alors activez l'option tabou sur la solution et aller à l'étape 5
Étape 5 : Le critère d'aspiration est-il satisfait ?
Oui, passez à l'étape 6 ;
Sinon, passez à l'étape 7 ;
Étape 6 : Choisir la meilleure solution qui n'est pas une solution taboue dans les solutions de voisinage et le définir comme solution actuelle, puis passer à l'étape 7 ;
Étape 7 : Définir la solution qui satisfait le critère d'aspiration comme étant la solution actuelle et la meilleure et passer à l'étape 8 ;
Étape 8 : Mettre à jour la liste des tabous et passer à l'étape 3 ;
Etape 9 : Sortir la meilleure solution.
Étape 10 : Si le critère d'arrêt de RT est satisfait, aller à l'étape 11 sinon retourner à l'étape 3.
Étape 11 Effectuer un saut basé sur la mutation globale de l'algorithme de kangourou.
Étape 12 : Basé sur la solution trouvée par RT, appliquer l'algorithme de kangourou
Étape 13 : Générer les nouvelles solutions de voisinage par la fonction de voisinage N', et Evaluer ces solutions ;
Étape 14 : Trouver la meilleure solution voisine dans le voisinage N(s) de la solution actuelle
Étape 15 : Si le critère d'arrêt d'AK est satisfait, aller à l'étape 16 sinon retourner à l'étape 3.
Étape 16 : Sortir la meilleure solution.

2.3.1. La génération de la solution initiale par SBH

L'efficacité des algorithmes méta-heuristiques qui utilisent les solutions initiales comme point de départ peut être améliorée en générant des solutions appropriées à des points prometteurs sur l'espace-solution. Dans notre approche la SBH est adaptée afin de générer une solution initiale faisable.

SBH est une méthode heuristique efficace pour le problème d'ordonnancement de job shop/job shop flexible. Elle repose sur une idée empirique : la performance du système pour les applications industrielles dépend de l'utilisation correcte de ressources rares. Dans le cadre du problème d'ordonnancement de l'atelier, la ressource rare est représentée par une machine à goulot d'étranglement (bottleneck), c.-à-d., la machine qui affecte le plus la valeur du makespan.

L'ordre dans lequel les machines sont traitées dépend du critère SBH. Une approche courante consiste à ordonnancer d'abord la machine qui provoque la plus forte augmentation de makespan. Cependant, de légères variations dans la séquence des machines traitées peuvent améliorer significativement la qualité de solution.

Nous construisons d'abord une solution réalisable en appliquant la SBH à partir de la représentation de la solution (la procédure commence par un ensemble des machines dont les séquences sont fixes).La méthode séquence les machines d'opérations puis la machine est identifiée comme une machine bottleneck 'goulot d'étranglement'.

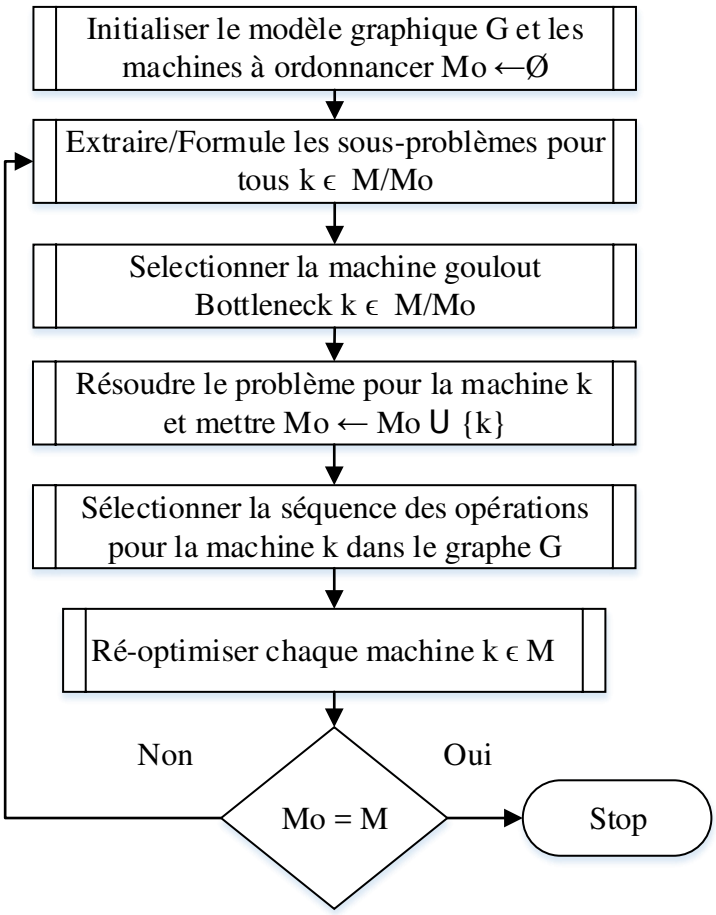


Figure 4.2 La structure de l'heuristique shifting bottleneck

2.3.2. La recherche locale par la Recherche Tabou

Cette méthode a montré son efficacité pour la résolution de plusieurs problèmes difficiles, parmi lesquels figure le problème de job shop/job shop flexible (Wat et al 2006). La recherche tabou a de puissante méthode d'optimisation interne d'exploitation. Le plan général d'exploitation est illustré dans la figure 4.3.

Tout d'abord, nous présentons la mise en œuvre des différents paramètres nécessaires à l'implémentation de cette méthode :

- La génération de la solution initiale,
- Structure de voisinage et déplacements,
- La mémoire tabou,
- Critère d'aspiration,
- Critère d'arrêt

2.3.2.1 La génération de la solution initiale

Il a été démontré que l'efficacité des méthodes basées sur le principe de la recherche locale dépend étroitement de la qualité de la solution initiale. Dans notre approche, la RT reçoit la solution générée par SBH comme une solution initiale.

2.3.2.2. Structure de voisinage et déplacements

Une structure de voisinage est un mécanisme qui permet d'obtenir un nouvel ensemble de voisins, en appliquant une simple modification à une solution donnée. La structure de voisinage est directement efficace sur l'environnement de RT.

La RT procède de manière itérative à partir d'un voisin à l'autre dans l'espace de résolution des problèmes. Elle garde une trace non seulement des informations locales (la valeur actuelle de la fonction objectif) mais aussi de certaines informations relatives au processus d'exploration, afin d'améliorer l'efficacité de ce processus. Cette utilisation systématique des informations stockées est la caractéristique essentielle de RT.

- **Le mouvement**

Le voisinage d'une solution courante est constitué de toutes les solutions obtenues en effectuant un mouvement élémentaire sur cette solution. Le processus de recherche consiste, à chaque itération, à choisir la meilleure solution qui n'est pas tabou dans le voisinage de la solution courante, ou qui ne satisfait pas le critère d'aspiration donné est sélectionné comme nouvelle solution même si cette solution n'entraine pas une amélioration. Le "meilleur" voisin est celui dont Cmax, est minimale. Si tous les voisins sont tabous ou aucun voisin ne satisfait au critère d'aspiration alors le voisin le plus ancien, entrant dans la liste des tabous au début, est sélectionné comme nouvelle solution.

Un mouvement tabou est le mouvement inverse d'un mouvement déjà mémorisé dans la liste des tabous.

- **La taille de voisinage**

La taille de l'ensemble des solutions voisines représente le nombre de solutions voisines recherchées. Ce paramètre influe sur le nombre de choix qui s'offrent à la RT pour passer de la solution courante à la voisine.

2.3.2.3. La mémoire tabou

Afin d'éviter le piège des optima locaux dans lequel le processus de recherche peut être facilement attrapé et s'épuisé beaucoup plus longtemps, la recherche tabou utilise la mémoire des tabou.

- **Liste tabou**

La structure de la mémoire des tabous se réalise à l'aide d'une liste circulaire de longueur *l (TL=l)* qui peut être constante ou variable. Glover et Laguna (1997) affirment que l'effet des deux types de mémoires peut être considéré comme modifiant le voisinage d'une solution donnée.

La gestion de cette liste suit la stratégie FIFO, l'ordre de dégagement des éléments est celui de leur insertion. La mise à jour de la liste se fait à chaque itération de la recherche. A chaque itération, un nouvel élément est introduit et un autre plus ancien est dégagé de la liste.

La liste tabou dépend de la taille du voisinage de la solution, dans notre application, chaque tabou liste contient :

- L'opération déplacée O_{ij},

- L'ancienne machine sélectionnée pour O_{ij},
- Le vecteur des machines alternatives, qui est obtenu par la ré-affectation des opérations sur le chemin critique vers les machines alternatives disponibles.

La procédure proposée utilise une liste dynamique. S'il n'y a pas d'amélioration dans la solution courante, la longueur l de la liste change à mesure que l'itération augmente. Cette procédure est modifiée selon (Li et al 2010) et (Fatahi 2007).

Le comportement de la longueur est décrit par les cinq étapes suivantes :

1. Pour les premières itérations K, la liste a une longueur constante égale à l
2. De K à 2K itérations, la longueur de la liste diminue linéairement jusqu'à l min=1/2 l
3. De 2K à 3 K itérations, la longueur de la liste reste constante,
4. De 3K à 4 K itérations, la longueur de la liste augmente linéairement de l_{min} à l_{max}=2 l
5. De 4K à 5K itérations, la longueur de la liste reste constante.

Avec

l : La racine carrée arrondie de n× m pour un problème de n job et m machine.

K : Une constante fixée à genmax/5 (genmax est le nombre de génération).

2.3.2.4. Critère d'aspiration

Le critère d'aspiration est une condition nécessaire et satisfaisante pour qu'un mouvement tabou soit accepté malgré son statut tabou (Schi 2001).

Le critère d'aspiration appliqué ici consiste à révoquer le statut tabou d'un mouvement si ce dernier conduira à un meilleur makepan que celui de la meilleure solution actuelle.

2.3.2.5 Critère d'arrêt

L'approche s'arrête lorsque la solution est suffisamment proche de la limite inférieure de la valeur de la fonction objectif. Sinon, elle s'arrête lorsque la meilleure solution n'est pas améliorée pour les itérations itermax.

- **Itermax**

Ce paramètre représentant le nombre maximum d'itérations sans amélioration de la meilleure solution pendant la recherche locale. Le nombre d'itérations tabou autorisées sans l'amélioration de la meilleure solution qui était fixé au nombre des opérations de chaque instance.

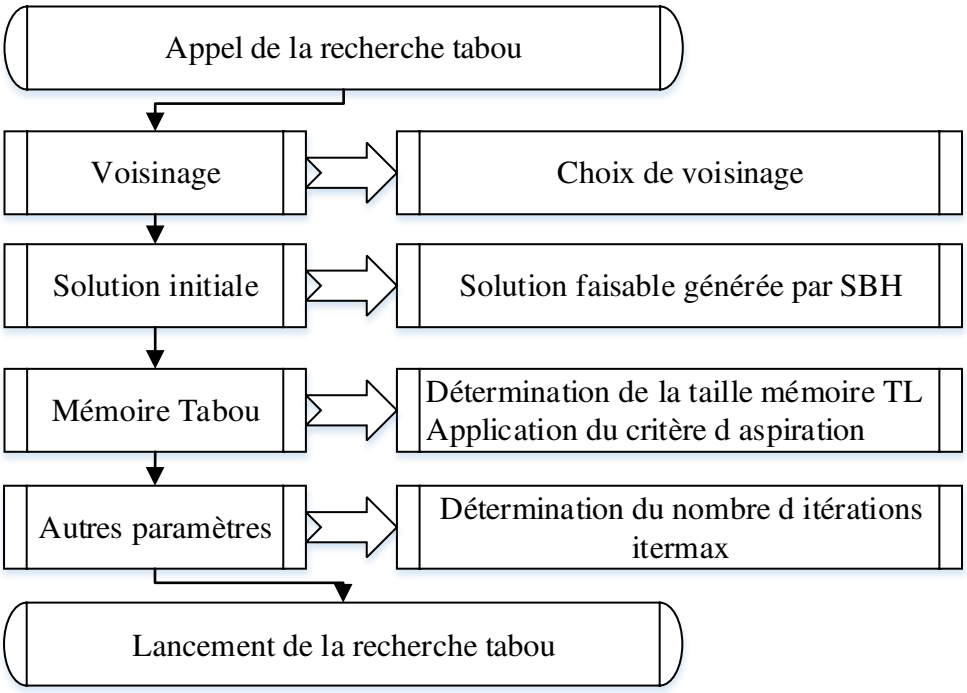


Figure.4.3. Plan de l'utilisation de la recherche tabou

2.3.3. La recherche globale par l'Algorithme de Kangourou

L'AK est une combinaison de recherche locale et globale. L'algorithme tente d'améliorer la solution actuelle en explorant son voisinage à l'aide d'une procédure itérative de descente stochastique. Contrairement à la RT, cet algorithme n'a besoin que d'une seule évaluation du critère de performance à chaque itération, ce qui est intéressant du point de vue temps de calcul.

- La génération de la solution initiale,
- Structure de voisinage et déplacements,
- Le compteur de stationnement,
- La procédure de saut,
- Le critère d'arrêt / Le nombre de sauts maximal.

Tout comme la recherche tabou, la mise en œuvre des différents paramètres nécessaires à l'implémentation de cet algorithme sont exactement ceux présentés pour le cas de RT. Dans notre approche, l'AK reçoit la solution générée par RT comme une solution initiale. Dans ce cas, l'algorithme trouve plus rapidement une bonne ou une solution optimale. Les mêmes structures de voisinage sont réutilisés ici ainsi le critère d'arrêt et le nombre d'itérations maximum. Le plan général de l'AK est illustré dans la figure 4.4.

2.3.3.1.Le compteur de stationnement

Le point principal, dans une mise en œuvre réelle de l'AK, est le choix de nombre d'itération A sans amélioration de la fonction objectif. Le paramètre A est le nombre maximum d'itérations sans amélioration de la solution actuelle.

2.3.3.2.La procédure de saut

La zone de recherche est étendue par cette procédure de recherche globale, qui améliore la solution actuelle en effectuant un saut élargissant la zone de recherche et évite tout enfermement dans un optimum local : Ainsi, la chance d'atteindre un minimum global augmente de façon asymptotique.

Si une nouvelle amélioration n'est plus possible, une procédure de "saut" est effectuée, afin d'échapper à l'attraction d'un minimum local, l'algorithme trouve un autre voisinage N' plus large par un saut. Après un nombre d'itérations un minimum local est trouvé, ce minimum est plus ou moins proche du minimum global. Dans le cas idéal le minimum local u*est le même avec le minimum global. Le fait d'effectuer un saut permet à l'algorithme, de sortir d'une vallée c'est-à-dire d'un minimum local en sautant les barrières de potentiel.

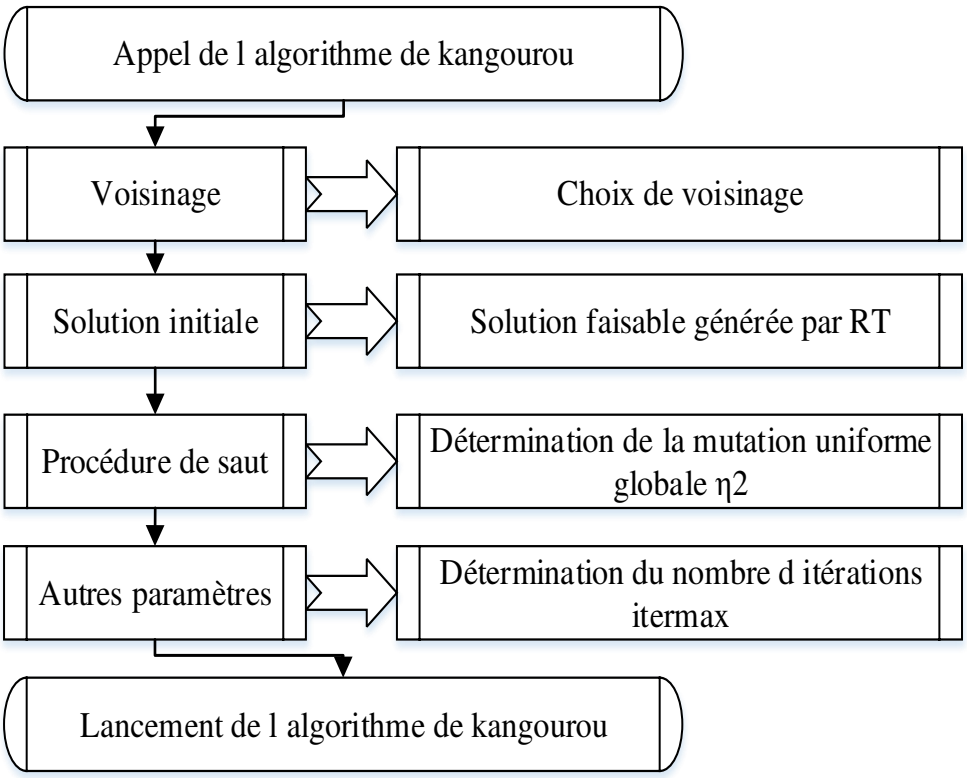


Figure 4.4.Le plan général de l'utilisation de l'algorithme de kangourou

3. Etude de cas

Dans cette recherche, nous effectuons des expériences sur un nombre d'instances de problèmes de différentes tailles. Pour expliquer le PJSF, nous considérons l'exemple présenté dans Tableau 4.1 : (3 jobs et 3 machines).

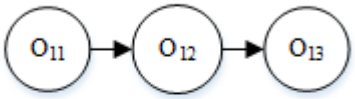
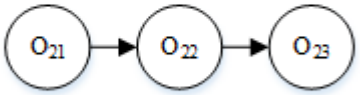
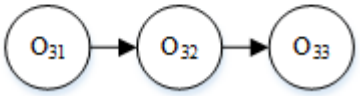
J_i	O_{ij}	Machines Alternatives			Plan de processus
		M_1	M_2	M_3	
1	O_{11}	1	3	2	
	O_{12}	3	4	2	
	O_{13}	3	1	6	
2	O_{21}	3	1	3	
	O_{22}	3	4	4	
	O_{23}	4	4	5	
3	O_{31}	2	1	1	
	O_{32}	2	4	4	
	O_{33}	5	6	4	

Tableau 4.1. Instance de job shop flexible

3.1.Codage de la solution

Le problème du JSF est une combinaison de décisions d'affectation de machines et d'ordonnancement d'opérations. Une solution peut être décrite par l'affectation des opérations aux machines et la séquence de traitement des opérations sur les machines.

Dans cette étude, nous représentons la solution de notre problème en tant qu'une chaîne de deux vecteurs. (Figure 4.5)

1) Le vecteur de séquence d'opérations (V1).
2) Le vecteur d'affectation des machines (V2).

3.1.1. Vecteur d'ordonnancement des opérations

Pour le vecteur des opérations, nous utilisons la représentation de (Gen et al 2009). L'idée principale de cette représentation est de nommer toutes les opérations d'un job avec le même symbole et de les interpréter en fonction de l'ordre d'apparition dans la séquence d'une solution donnée. Pour un problème $n \times m$, la solution inclut la valeur $n \times m$, chaque job apparaît n fois. La désignation de ce symbole est construite en fonction de son occurrence dans la liste, .ex. le k^{th} occurrence du job signifie son k^{th} opération.

Avec le vecteur d'opération de la figure 4.5, la solution correspondante peut être interprétée comme suit : $O_{31}>O_{21}>O_{22}>O_{23}>O_{32}>O_{11}>O_{12}>O_{13}>O_{33}$.

3.1.2. Vecteur d'affectation des machines

Ce vecteur symbolise le numéro de la machine alternative sélectionnée correspondante pour l'opération indiquée en position *i*. Il a la même taille que le vecteur d'opération.
S'il n'y a pas de machine alternative correspondante au nombre aléatoire pour choisir une machine, la première machine doit être attribuée.

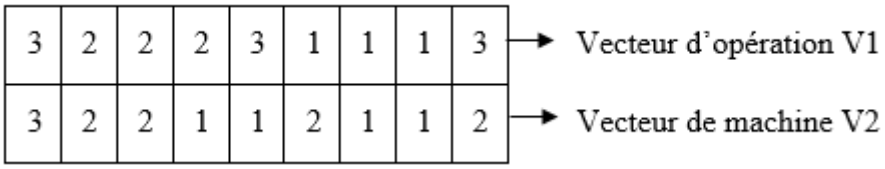


Figure 4.5. Codage des vecteurs de la solution

3.2. Fonction de voisinage utilisée

Le domaine de solution du problème job shop flexible s'élargit avec le nombre de jobs, le nombre de machines alternatives pour chaque job et le nombre d'opérations par job. Pour définir les zones accessibles de notre solution, nous devons présenter la fonction de voisinage. Il s'agit d'un ensemble des solutions accessibles en un seul mouvement élémentaire de la solution actuelle. La complexité d'une résolution d'une approche de recherche locale dépend fortement de l'approche de l'évaluation de la méthode de voisinage.

Par conséquent, le choix d'un voisinage riche avec un grand nombre de candidats, augmenteront la probabilité de trouver de bonnes solutions. Il augmentera également les calculs requis du temps.
De nombreuses analyses liées à l'ordonnancement du job shop/job shop flexible sont considérées. Brandimarte (1993) a proposé les structures de voisinage suivantes pour ce type d'atelier :

- **Voisinage N1** où deux opérations successives sont inversées sur le chemin critique.
- **Voisinage N2** où un job est sélectionné au hasard et il est échangé sur chaque machine avec les jobs adjacents dans chaque séquence de machine.
- **Voisinage N3** où une machine est choisie au hasard et où l'ensemble des jobs sur cette machine est pris en compte.

La deuxième et la troisième structure de voisinage limite l'espace de recherche en une disposition spécifique, tandis que la première donne un certain degré d'exploration aléatoire.
A titre d'information, la première structure de voisinage réussie pour le PJS/PJSF a été présentée par Van Laarhoven (1987), qui est souvent désigné par N1.

Plusieurs chercheurs ont vérifié que lorsque le makepan est considéré comme fonction objective, le voisinage N1 est très utile puisque les opérations ne se trouvant pas sur le chemin critique n'affectent pas le makepan. Dans le présent chapitre ce type de voisinage est considéré.

3.2.1. Voisinage pour la composante d'ordonnancement des opérations

Pour trouver les solutions voisines de l'opération nous utilisons le concept du chemin critique. Le makespan d'une solution est égale à la longueur de son chemin critique. En d'autres termes, le makepan ne peut pas être réduit tout en maintenant les chemins critiques actuels.

3.2.2. Voisinage pour la composante d'affectation des machines

Dans la phase de recherche locale, trouver une machine plus adaptée pour une opération produira une solution quasi-optimale. Une règle de répartition efficace est utilisée afin d'introduire de nouvelles permutations dans le vecteur V2.
Cette règle se repose sur le principe d'assigner une machine ayant une charge de travail relativement moindre parmi les machines alternatives disponibles et programmer l'opération sur cette machine.
La structure de voisinage adaptée est présentée comme suite :

(1) Sélectionner l'opération O_{ij} critique à partir du vecteur V1, dont la machine passe le maximum de temps ;
(2) Détecter la machine M_k qui est actuellement sélectionnée pour traiter O_{ij}
(3) Détecter l'ensemble de machines M qui peuvent traiter O_{ij} ;
(4) Sélectionner la machine qui a la charge de travail (W_M) la plus petite pour traiter O_{ij}

4. Conception expérimentale et résultats

4.1.Génération de la solution initiale

Les bottlenecks (M_1, M_2 et M_3) sont extraits et résolus séparément. Nous obtenons les voisins de la solution obtenue en échangeant les arcs disjonctifs (disjonctifs).
Selon la nature du job shop flexible (ensemble des machines alternatives), il peut y avoir plusieurs sous-systèmes pour la représentation du graphe disjonctif. Les sections ombrées du tableau 4.2 sont établies par :

4.1.1. Composant d'affectation des machines

Pour produire le vecteur d'affectation machine V2, la machine avec le temps de traitement minimum (SPT) parmi les machines candidates Mi sera sélectionné et placé à la position i dans V2, à condition que la machine ne soit pas utilisée par les opérations précédentes du même job.

La sélection de cette règle de priorité est due à sa meilleure performance parmi les règles de répartition communes pour les problèmes de makespan (Zhou 2001). En conséquence, nous pouvons garantir que chaque opération sera usinée dans un temps minimal, ce qui peut s'avérer être utile pour obtenir une fonction objectif optimale (makespan).
Si la machine choisie est déjà choisie la machine suivante sera sélectionnée et ainsi de suite.

L'objectif est d'éviter qu'une solution n'assigne trop de job à une seule machine et de maintenir l'équilibre de charge sur cette machine ;

- ✓ Mise à jour du taux d'utilisation de machine de chaque machine.
- ✓ Réaffectation des opérations aux machines moins utilisées.

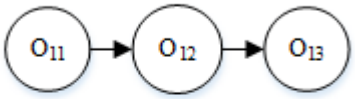
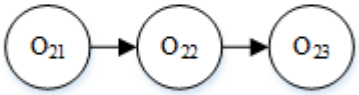
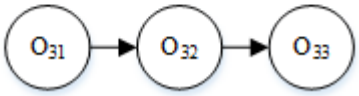
J_i	O_{ii}	Machines Alternatives			Plan de processus
		M_1	M_2	M_3	
1	O_{11}	1	3	2	
	O_{12}	3	4	2	
	O_{13}	3	1	6	
2	O_{21}	3	1	3	
	O_{22}	4	4	4	
	O_{23}	4	4	5	
3	O_{31}	2	1	1	
	O_{32}	2	4	4	
	O_{33}	5	6	4	

Tableau 4.2. Les opérations et les machines sélectionnées pour la génération de la solution initiale

La sélection des machines aux opérations est illustrée dans la figure suivante (4.6) :

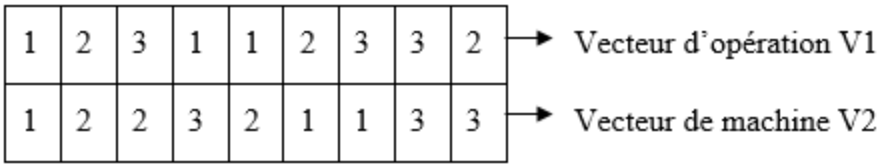


Figure 4.6. Le codage de la sélection initiale

Nous pouvons clairement observer qu'il s'agit simplement d'une transformation en un problème classique de JS, en sélectionnant différents ensembles de vecteurs. Différents cas de PJS sera produit. Le résultat de l'analyse du vecteur sélectionné est :

Job 1: M_1 — M_3 — M_2
Job 2: M_2 — M_1 — M_3
Job 3: M_2 — M_1 — M_3

4.1.2. Mise en œuvre de SBH

Lors de la première itération, nous recherchons la machine bottleneck entre M_1, M_2 et M_3 en résolvant une série de trois problèmes de machine unique.
Les calculs ont montré que la machine bottleneck est M_3. Lors de l'itération suivante, nous recherchons la nouvelle bottleneck machine parmi M_1, M_2 de la même manière pour deux problèmes de machine unique.

Les calculs ont montré qu'il n'y a pas de machines bottlenecks. Les gammes obtenues pour chaque machine sont présentées dans le graphe suivant où la longueur maximale (makespan) de la trajectoire de 'S' à 'P' est de 12. Le chemin critique est mis en évidence par des arcs à brosse large dans la Figure. 4.7. Les opérations sur le chemin critique (cercles ombrés) sont appelées opérations critiques. La figure suivante présente le graphe disjonctif selon le tableau 4.2.

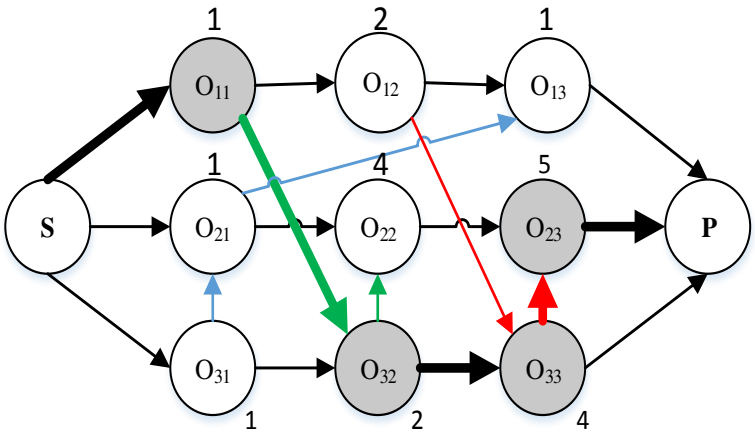


Figure 4.7 : Le graphe disjonctif de la solution SBH

Le résultat du SBH (la solution initiale) donne les opérations critiques suivantes : O_{11}, O_{32}, O_{33}, O_{23}. Le diagramme de gantt du SBH est illustré dans la figure 4.8.

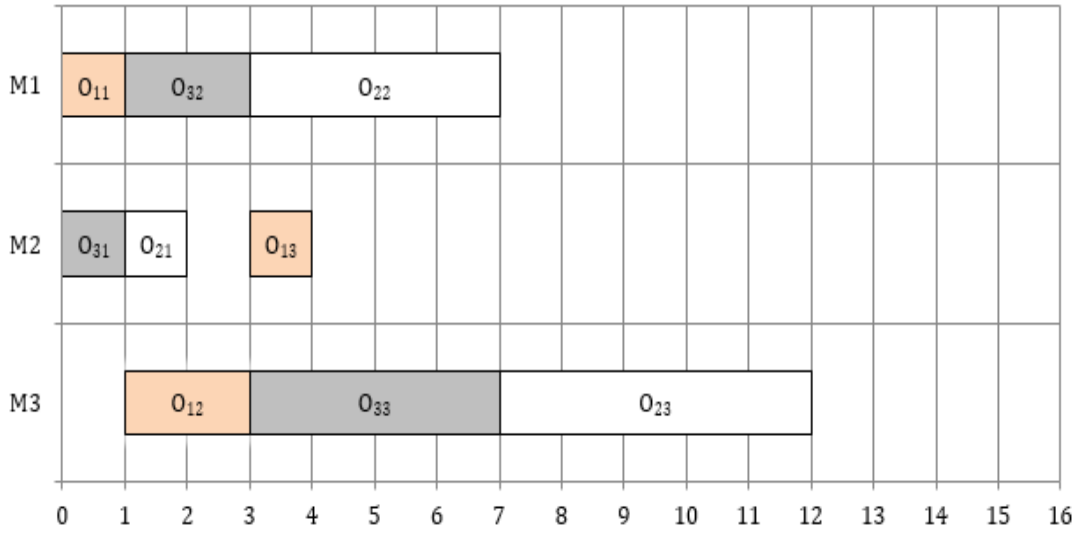


Figure 4.8. Diagramme de Gantt de la solution SBH

Comme il est indiqué dans la figure 4.8., le C_{max} obtenue par SBH est 12. Le SBH fonctionne avec les sélections indiquées dans le tableau 2. Le concept appliqué, peut être assuré que toutes les machines sont équilibrées et ne seront utilisées qu'une seule fois pour un seul job. Comme le SBH ne traite qu'une seule instance JSP, la solution est obtenue immédiatement dans le cas de petite instance. Dans ce qui suit, nous présentons l'étape suivante de notre méthode, sur la même étude de cas.

4.2. Mise en œuvre de la Recherche Tabou

Dans la présente étude, la recherche tabou a été adopté comme stratégie de recherche locale dans l'approche proposée dans un environnement JSF, et la structure de voisinage N1 a été adoptée ici.

- La solution initiale pour la RT est présentée dans la figure 4.9, et les opérations ombrées présentent les opérations critiques :

1	3	2	3	2	1	1	3	2	→ Vecteur d'opération V1
1	2	2	1	1	3	2	3	3	→ Vecteur de machine V2

Figure 4.9. La représentation de la solution initiale générée par SBH

- La recherche locale

Une fois que la solution initiale est déterminée, la flexibilité des routages est introduite Dans cette phase, nous devons considérer à la fois le choix de la machine alternative et le séquencement des opérations sur chaque machine, c'est-à-dire déterminer la date de début de chaque opération.

Le chemin critique dans la solution initiale est composé d'opérations O_{11}, O_{32}, O_{33} *et* O_{23} avec une taille égale à 12 : C_{max}=12.
Les opérations critiques O_{11}, O_{32} sont concernées par le voisinage de la solution actuelle de M_1, et O_{33}, O_{23} par le voisinage de la solution actuelle de M_3. Le voisinage est alors composé par les déplacements suivants :

- Permutation de O_{11}, O_{32} sur la machine M_1
- Permutation de O_{33}, O_{23} sur la machine M_3
- Remplacement de O_{11} de la ressource M_1 par une autre machine alternative (M_2, M_3).
- Remplacement de O_{32} de la ressource M_1 par une autre machine alternative (M_2, M_3).
- Remplacement de O_{33} de la ressource M_3 par une autre machine alternative (M_1, M_2).
- Remplacement de O_{23} de la ressource M_3 par une autre machine alternative (M_1, M_2).

Le nombre d'opérations prises en compte pour d'éventuels re-calculs diminue de plus en plus lorsque les opérations à échanger sont placées à la fin d'ordonnancement.

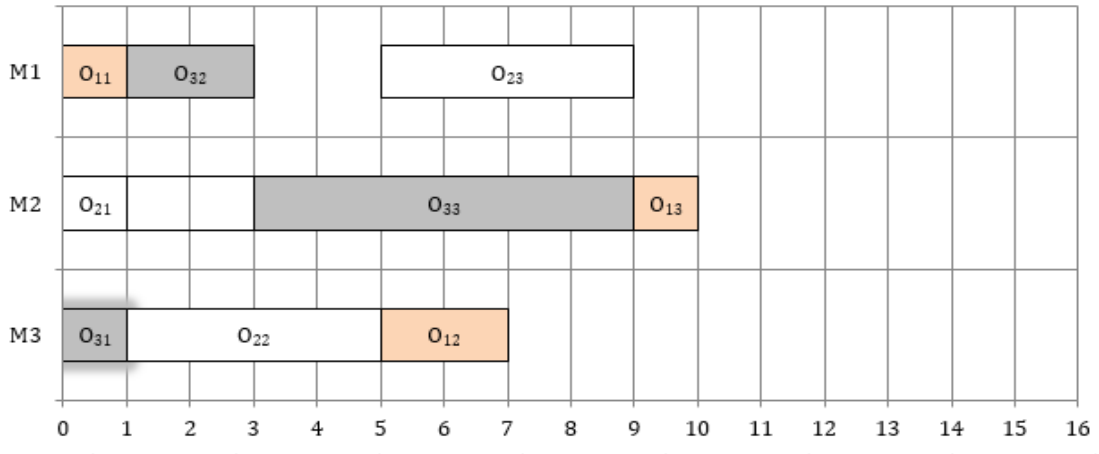


Figure 4.10. Diagramme de gant pour la solution SBH-RT

La figure ci-dessus (4.10) illustre le résultat obtenu après l'introduction de la procédure RT. Dans cette deuxième étape, la flexibilité de routage, qui concerne la possibilité d'effectuer une opération sur des machines alternatives, est incorporée dans la solution. L'avantage du

système est la disponibilité de plusieurs machines, ce qui introduit plus de flexibilité dans le système. Alors qu'une machine fixe peut entraîner une charge de machines importante.

Dans cette phase, nous observons un échange de machine sur les opérations critiques, O_{33} et O_{23}, la solution finale de RT a réassigné ces opérations aux machines alternatives M_2 et M_1 respectivement. Ce changement a généré les nouvelles opérations critiques suivantes : O_{11}, O_{32}, O_{33} *et* O_{13} avec une taille égale à 10 : C_{max}=10, selon le graphe disjonctif suivant (figure 4.11) :

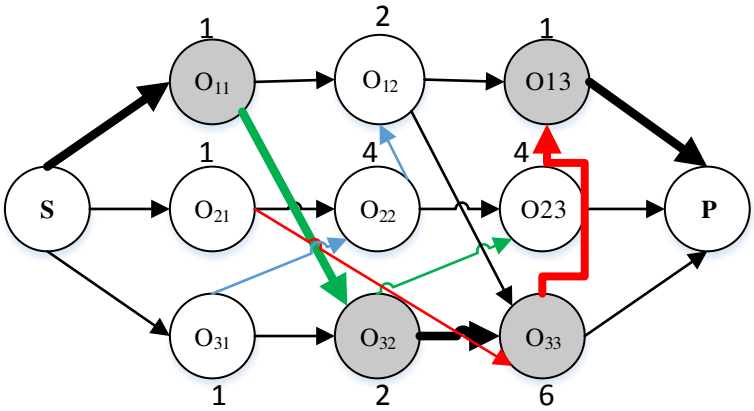


Figure 4.11 : Le graphe disjonctif de la solution SBH-RT

4.3. Mise en œuvre de l'Algorithme de kangourou

Dans cette phase, l'algorithme de kangourou a été adopté comme stratégie de recherche globale dans l'approche proposée dans un environnement JSF.

- La solution initiale pour l'AK est présentée dans la figure 4.12, et les opérations ombrées présentent les opérations critiques :

1	3	2	3	2	1	1	3	2	→	Vecteur d'opération V1
1	3	2	1	3	3	2	2	1	→	Vecteur de machine V2

Figure 4.12. La représentation de la solution générée par SBH-RT

- La recherche globale

Une fois que la solution finale de SBH-RT est déterminée, l'algorithme de kangourou effectue un saut vers un nouveau voisinage tout en considérant la flexibilité des machines de chaque opération. Le chemin critique dans la solution initiale est composé d'opérations O_{11}, O_{32}, O_{33} *et* O_{13} avec une taille égale à 10 : C_{max}=10.

Les opérations critiques O_{11}, O_{32} sont concernées par le voisinage de la solution actuelle de M_1, et O_{33}, O_{13} par le voisinage de la solution actuelle de M_2. Le nouveau voisinage est réalisé par quatre permutations du saut et il se compose par les déplacements suivants :

- Permutation de O_{11}, O_{32} sur la machine M_1 ; et remplacement de O_{11}, O_{32} par une autre machine alternative (M_2, M_3).

- Permutation de O_{33}, O_{13} sur la machine M_3, et remplacement de O_{33}, O_{13} par une autre machine alternative (M_1, M_2).

Les permutations et les remplacements sont faites, au même temps, les autres opérations sont automatiquement mis à jour avec les nouveaux changements dans l'ordonnancement et les opérations modifiées sont illustrées dans la le diagramme de gant de la solution finale (fig. 4.13).

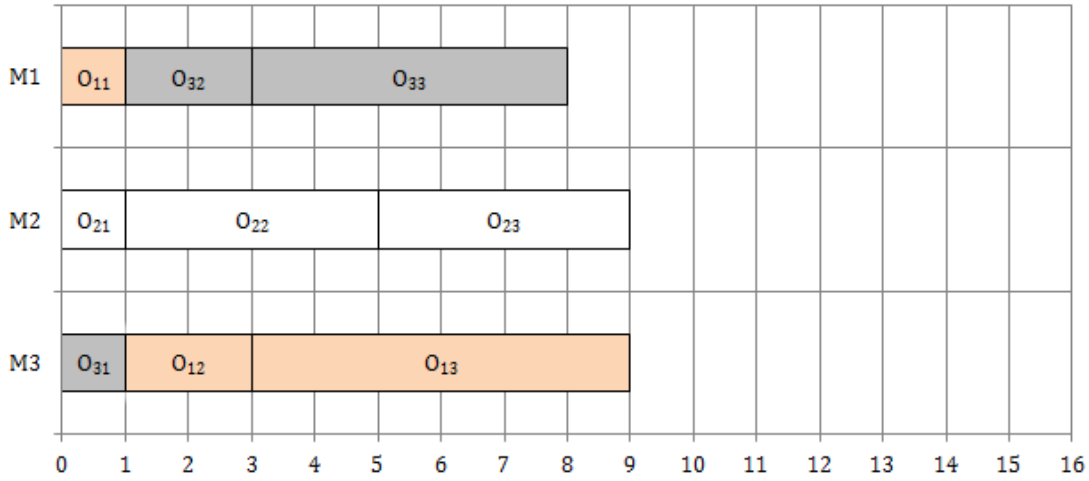


Figure 4.13. Diagramme de gant pour la solution SBH-RT-AK

Dans cette phase, toutes les opérations de J_2 sont affectées à la machine M_2, ce qui donne une charge de travail de 9 pour cette machine. Alors que plus d'une opération d'un même job est affectée à la même machine (M_1 et M_3).Les machines alternatives sont utilisées pour tous les jobs et permettent d'améliorer la solution.

Bien que la RT ait déjà bien exploré l'espace de recherche et ait donné une solution de haute qualité, l'AK parvient à améliorer cette solution. La solution finale d'AK a réaffecté les opérations O_{33} *et* O_{13} aux machines alternatives M_1et M_3 respectivement.
Par conséquent, l'espace de recherche utilisé par le SBH est restreint par rapport au RT et AK. Globalement, la solution finale obtenue est supérieure à celle obtenue par SBH et SBH-RT.

5. Expérimentations et résultats

Les PJSF sont classés en deux sous-problèmes connus sous le nom de problèmes total job shop flexibles (T-PJSF) et des problèmes partiel job shop flexibles (P-PJSF). Dans T-PJSF, chaque opération peut être traitée sur n'importe quelle machine de M. Par contre, dans P-PJSF, chaque opération peut être traitée sur au moins une machine d'un sous-ensemble de M. Par conséquent, plusieurs ensembles de cas problématiques ont été considérés.

Nous comparer la performance de notre approche avec celles obtenu par d'autres auteurs avec la considération du makespan comme objectif. Plusieurs algorithmes sont utilisés pour les évaluer avec l'approche proposée :

- Le 'TS' (Brandimarte 1993), le 'AL+CGA' (Kacem et al 2002), '*ITS'* et '*ISA'* (Fatahi 2007), sont considérées comme les premiers résultats dans la littérature.

- La '*LEGA'* de (Ho et al 2007), la '*BBO'* de (Rahmati et Zandieh 2012), et '*l'Heuristique de Ziaee'* (2014) sont des méthodes heuristiques et métaheuristiques standard.
- Le '*IAI'* (Bagheri 2010), '*MOPSO+LS* '(Moslehi et Mahnam 2011), '*HA (GA+TS)'* (Li et al 2016) et Le '*GATS+HM'* par (nouri 2018) sont deux des algorithmes hybrides récents.
- Le '*MATSLO'* (Ennigrou et Ghedira 2008) et le *'MATSPSO'* de (Henchiri et Ennigrou 2013), '*MACRO*' (Merzouki 2016) sont des nouveaux algorithmes métaheuristiques hybrides distribués dans un modèle multi agent.

En raison de la nature non déterministe de l'algorithme proposé, nous simulons cinq fois indépendamment pour chacun des trois expériences afin d'obtenir des résultats significatifs.
Les résultats des calculs sont présentés par trois mesures telles que le makepan (Cmax), la moyenne du temps CPU en secondes, et le taux d'amélioration du makepan (TA%), qui est calculé par l'équation suivante :

$$TA = [(C_{maxc} - C_{max0})/C_{maxc}] \times 100$$

Avec C_{max0} est le makepan obtenu par notre approche et C_{maxc} est le makepan d'un algorithme de référence (Ref) que nous avons choisi pour comparer.

Tous les algorithmes sont implémentés en utilisant le langage de programmation JAVA, et les expériences de calcul sont effectuées sur un ordinateur personnel avec un processeur Intel Core i5 2,66 GHz et 4 Go de RAM. Les paramètres utilisés pour notre algorithme sont ajustés expérimentalement et présentés comme suit :
- Nombre maximal d'itérations sans amélioration et le compteur de stationnement égalent au nombre des opérations de chaque instance.
- Nombre de génération est fixé à 100.

5.1. Expérience 1

L'ensemble des problèmes se divise en quatre instances. Les différents résultats comparatifs sont affichés dans les tableaux 4.3 et 4.4, où la première colonne donne la taille de chaque instance, la deuxième colonne désigne la méthode adaptée et les colonnes restantes détaillent les résultats expérimentaux. Les valeurs en gras dans les tableaux indiquent les meilleurs résultats obtenus et le N/A signifie que le résultat n'est pas disponible.

- Instance 1(Problème 4x5) : Il s'agit d'une application de petite taille, 4 jobs, 5 machines et 20 opérations.
- Instance 2 (Problème 8x8) : Il s'agit de l'application à partielle flexibilité, contenant 8 jobs avec 27 opérations qui ont à réaliser sur 8 machines.
- Instance 3 (Problème 10x10) : Il s'agit de l'application d'un problème de taille intermédiaire. Il y a 10 jobs avec 30 opérations à être traitées sur 10 machines.
- Instance 4 (Problème 15x10) : Pour cette application, nous avons un problème de plus grande taille qui a une flexibilité totale. Il s'agit composé de 15 jobs avec 56 opérations qui peuvent être réalisé sur 10 machines avec une flexibilité totale.

Problème	SBH-RT-AK			AL+CGA		LEGA	
	C_{max}	TA%	CPU	C_{max}	TA%	C_{max}	TA%
4×5	12	-9	1	16	-54.4	**11**	0
8×8	**14**	0	4.68	15	7	N/A	N/A
10×10	**7**	0	1.62	**7**	0	**7**	0
10×15	**11**	0	9.81	23	N/A	12	0

Tableau 4.3. Résultats des instances Kacem (partie 1)

MOPSO+LS		BBO		HZiaee		GATS+HM	
C_{max}	TA%	C_{max}	TA%	C_{max}	TA%	C_{max}	TA%
16	-54.4	**11**	Ref	**11**	0	**11**	0
14	0	**14**	Ref	15	-7	14.2	-1.42
7	0	**7**	Ref	**7**	0	7.6	0.6
11	0	12	Ref	12	8.33	11.6	-0.4

Tableau 4.4. Résultats des instances Kacem (partie 2)

En examinant les résultats, nous observons que l'approche hybride surpasse l'algorithme AL+CGA dans presque tous les cas et la différence moyenne entre eux est significative, principalement dans le problème du 15×10. Alors que pour le 10x10 la valeur C_{max} la plus faible peut être produite par toutes les méthodes.
La comparaison de la méthode proposée avec d'autres méthodes montre que la valeur C_{max} la plus faible a été trouvée par notre méthode hybride sur la plupart des cas testés. En effet il n'existe pas de différences significatives entre les résultats. Cependant, le pire makespan (C_{max}) s'est produit dans le cas d'AL+CGA. Dans l'ensemble, notre système hybride produit la solution optimale ou presque optimale pour tous les cas avec d'autres méthodes.

5.2. Expérience 2

Nous testons également notre approche sur des instances de benchmark de (Fatahi 2007). L'ensemble des problèmes se divise en vingt instances. Les différents résultats comparatifs sont affichés dans le tableau 4.5, et le symbole (*) indique que la solution optimale a été trouvée.

En résolvant ce deuxième ensemble de données, notre approche atteint les mêmes résultats obtenus par certaines approches pour les dix premières instances. Les expériences montrent que notre modèle atteint l'optimum pour 13 instances de 20 avec un pourcentage égal à 65% des cas. Les résultats montrent que notre approche donne les mêmes résultats dans 70% des cas.

Pour la comparaison avec la HA et ITS qui sont basées sur la recherche tabou, les résultats montrent également que notre approche surpasse ITS dans 70 % des cas et dépasse l'approche HA pour une seule instance MFJS1. Le résultat fournit par ITS et ISA et nettement classé dans la dernière instance.

Problème	n×m	SBH-RT-AK	MACRO	AIA	ITS	ISA	HA
SFJS1	2×2	66*	66	66	66	66	66
SFJS2	2×2	107*	107	107	107	107	107
SFJS3	3×2	221*	221	221	221	221	221
SFJS4	3×2	355*	355	355	355	355	355
SFJS5	3×2	119*	119	119	137	119	119
SFJS6	3×3	320*	320	320	320	320	320
SFJS7	3×5	397*	397	397	397	397	397
SFJS8	3×4	253*	253	253	253	253	253
SFJS9	3×3	210*	210	210	215	215	210
SFJS10	4×5	516*	516	516	617	516	516
MFJS1	5×6	469	477	468	458*	488	486
MFJS2	5×7	458	464	448*	457	478	446
MFJS3	6×7	564	578	468	606	599	466
MFJS4	7×7	554*	745	554	870	703	554
MFJS5	7×7	696	708	527	729	674	514*
MFJS6	8×7	634*	836	635	816	856	634
MFJS7	8×7	1454	1465	879*	1048	1066	879*
MFJS8	9×8	1034	1934	884	1220	1328	884
MFJS9	11×8	1153	2965	1088	1124	1148	1055
MFJS10	12×8	1196*	5223	1267	1737	1546	1196

Tableau 4.5. Résultats des instances de Fatahi

5.3. Expérience 3

Les données de l'expérience 3 sont tirées de Hurink et al. (1994). Le tableau 4.6 présente les résultats de l'expérience et les comparaisons avec les autres algorithmes. Le GATS+HM est considéré comme algorithme de référence.
D'après ce tableau, les résultats obtenus montrent que note SBH-RT-AK obtient cinq des dix meilleurs résultats pour le programme Hurink edata instances (la01-la05) et (la16-la20).

En effet, notre SBH-RT-AK obtient le meilleur makespan pour les instances la03 et la16, mais le résultat est légèrement pire pour l'instance la17, et la19. De plus, le GATS+HM a obtenu le meilleur résultat pour quatre instances.
La SBH-RT-AK surpasse la MATSLO+ dans quatre cas sur dix. Les résultats montrent également que notre modèle et MACRO et MATSLO+ fournissent les mêmes résultats dans 30% des cas. Pour réduire clairement la comparaison de toutes les méthodes proposées, voir la figure 4.14.

Problème n × m		SBH-RT-AK		MACRO	MATSLO+	GATS+HM
		C_{max}	TA%	C_{max}	C_{max}	C_{max}
la01	10×5	609*	0	609	609	609
la02	10×5	655*	0	655	655	655
la03	10×5	554*	0	573	575	567
la04	10×5	578	-1.7	579	579	568*
la05	10×5	503*	0	503	503	503
la16	10×10	875*	1.9	896	896	892
la17	10×10	765	-8.2	707*	708	707
la18	10×10	845	-.23	867	845	843*
la19	10×10	878	-9.2	806	813	804*
la20	10×10	860	-.35	863	863	857*

Tableau 4.6 Résultats des instances de Hurink

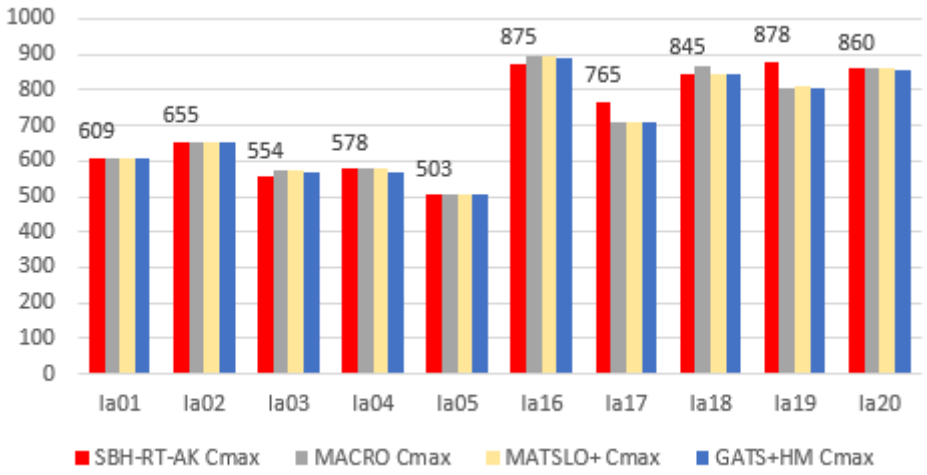


Figure 4.14. Illustration des techniques pour les instances de Hurink

6. Résumé des expériences

Pour de nombreux problèmes, les solutions de SBH-RT-AK sont préférables aux résultats d'autres algorithmes. Par conséquent, les résultats expérimentaux ci-dessus indiquent que notre approche hybride peut résoudre efficacement le PJSF et obtenir de nombreux résultats positifs et utiles.

Nous résumons nos principales conclusions des expériences pour obtenir des lignes directrices sur la façon d'appliquer les diverses heuristiques :

- ✓ Le nombre de machines disponibles pour chaque opération influence fortement l'amélioration C_{max} obtenue. La flexibilité élevée, assurée par de nombreuses machines alternatives, permet de réduire les coûts d'exploitation. l'avantage d'une heuristique plus sophistiquée.
- ✓ Cet algorithme a une structure simple et une grande flexibilité, est facile à mettre en œuvre et ne nécessite que très peu d'intervention ce qui le rend préférable à d'autres

approches plus complexes et plus longues, même si ses résultats pour les instances de référence sont si faiblement dominés dans la littérature.

- ✓ Il a été démontré que RT peut être utilisée pour optimiser les ordonnancements avec des itinéraires/machines alternatifs.
- ✓ Le principal effet positif de l'hybridation de RT avec la recherche globale est l'amélioration de la vitesse de convergence à l'optimum local. D'autre part, le principal effet négatif est l'augmentation du temps de calcul par génération. Ainsi, le nombre de générations est diminué lorsque le temps de calcul disponible est limité.
- ✓ L'approche proposée a une très bonne capacité de recherche avec peu de temps de calcul et peut maintenir un bon équilibre entre intensification et diversification.
- ✓ Nous concluons que la SBH, hybridée avec les approches RT, est capable de surpasser les heuristiques qui sont purement basées sur RT.

7. Conclusion

Dans ce chapitre, nous avons développé un algorithme hybride dans un job shop flexible afin de réduire au minimum le makespan C_{max}. Cette approche consiste à hybrider trois techniques et combine ces avantages en se basant sur les caractéristiques du JSF.

Notre stratégie consiste à commencer à partir d'une solution initiale faisable créée par la SBH. Cette solution devient une phase initiale pour la recherche tabou. Ensuite, RT améliore solution en intégrant la flexibilité opérationnelle (machine alternative). Finalement, l'intégration d'AK vise à utiliser largement les informations qualitatives pendant la recherche globale.

Le traitement des résultats montre que l'approche hybridé conduit à des améliorations significatives en surpassant certaines des approches existantes dans la littérature. Par ailleurs, la solution initiale joue un rôle essentiel dans la détermination de la qualité de l'approvisionnement en solution finale.

L'efficacité de l'approche développée peut être notée aussi par les solutions intermédiaires réalisables qui convergent dans un temps raisonnable, ce qui est tout à fait satisfaisable dans un atelier de production.

Il est intéressant de noter que les taux d'amélioration sont considérablement plus grands lorsque la flexibilité des machines est considérée. Notez que pour l'environnement Rm où les machines d'un groupe (étage) ne sont pas identiques, une affectation appropriée des jobs aux machines est crucial et les décisions d'ordonnancement sont plus importantes que dans le cas du Pm.

Sur la base de la qualité des résultats fournis dans ce chapitre, nous continuons d'adapter cette approche au problème de l'IPPS dans le chapitre suivant (5).

CHAPITRE 5

Une approche hybride pour la planification des processus et l'ordonnancement intégré dans un job shop flexible

Dans le chapitre précédent, nous avons considéré une approche hybride pour un problème job shop flexible où les plans de processus sont déjà sélectionnés et fixés (uniques) pour chaque job, dans le but de minimiser le makespan (C_{max}). Cependant, dans ce chapitre nous adaptons cette approche hybride dans le deuxième volet de notre étude qui considère également le niveau de planification du processus.

1. Introduction

La planification des processus, et l'ordonnancement sont étroitement liés. Toutefois, de façon conventionnelle, ces deux fonctions sont exécutées séparément et séquentiellement.

La séparation de la planification et l'ordonnancement entraînent de nombreux problèmes en raison d'objectifs conflictuels, de l'incapacité de communiquer les caractéristiques dynamiques d'un atelier (par exemple, la charge de machine, le trafic sur le système de manutention) ou des situations inhabituelles (p. ex. panne d'une machine, etc.) et les goulots d'étranglement). Par conséquent, ces deux fonctions doivent être prises en compte de manière intégrée afin d'obtenir un ordonnancement réaliste et efficace de la production.

Dans ce chapitre, notre étude s'est élargi en considérant un environnement intégré de la planification des processus et l'ordonnancement dans un job shop flexible.
Notre approche hybride développée dans le chapitre précédent, est considérée sur la base duquel une représentation implicite d'un plan de processus réalisable pour chaque job peut être faite.
Une étude comparative s'est abordée afin de valider la performance de l'approche sur les problèmes d'intégration de planification des processus et l'ordonnancement, nous sommes intéressés à trouver un ensemble de plans de processus qui conduit à la plus petite valeur du makespan.

Les sections suivantes présentent les détails du problème formulé pour l'IPPS dont l'objectif de la minimisation du makespan est considéré.

2. Formulation du problème

Dans le présent travail, le job shop flexible est choisi comme cible car il s'agit d'une stratégie de production adaptée à l'environnement de production actuel intégré. De plus, l'analyse de l'ordonnancement dans un job shop flexible fournit un aperçu de qualité de la solution d'ordonnancement rencontrée dans des systèmes plus réalistes et complexes (Kutanoglu et Sabuncuoglu, 1999).

Le problème IPPS peut être généralement défini comme (Kim et al 2003) : Étant donné un ensemble de *N* jobs qui doivent être traités sur *M* machines avec la flexibilité d'opération et la flexibilité de machine, trouver une séquence d'opérations et une séquence de machines-outils correspondantes pour chaque job et un plan d'ordonnancement dans lequel les opérations sur les machines sont traitées de manière à satisfaire les contraintes de précédence et la fonction objectif. Il comprend également trois dimensions de décision :

1- La décision pour l'affectation des plans de processus.
2- La décision pour l'affectation des machines.
3- La décision pour l'ordonnancement.

2.1. Modèle mathématique

Le problème à l'étude repose sur les même hypothèses présentées dans le chapitre précèdent, puisqu'il s'agit du même environnement de production (JSF). A l'exception de l'hypothèse suivante qui est relative à la fonction de planification des processus :

- Tous les jobs ont les mêmes priorités.

Le modèle mathématique du problème IPPS est donné comme suit (Jin et al. 2015).

- Notations et paramètres

i,i'	Indices pour les jobs
j,j'	Indices pour les opérations
k,k'	Indices pour les machines
l,l'	Indices pour les plans de processus
$O_{i,l,j}$	$J^{iéme}$ opération du job i sur le $l^{iéme}$ plan de processus
N	Ensemble des opérations
M	Ensemble des machines
T_i	Ensemble des plans de processus du job i
$NO_{i,l}$	Ensemble des opérations du $l^{iéme}$ plan de processus du job i
$R_{i,l,j}$	Ensemble des machines disponible pour l'opération $O_{i,l,j}$
$P_{i,j,k,l}$	Temps de traitement de l'opération $O_{i,l,j}$ sur la machine k
A	Nombre positif de valeur très grande

- Variable De décision

C_{max}	Makespan
$C_{i,j}$	Variable continue pour le temps de réalisation de la $j^{ième}$ opération du job i
$X_{i,l} =$	$\begin{cases}1, & \text{si le } l^{\text{ème}} \text{ plan de process du job i est sélectionné}\\ 0, & sinon\end{cases}$
$Z_{i,j,k,l} =$	$\begin{cases}1, & \text{si } O_{i,l,j} \text{ est traité sur la machine k}\\ 0, & sinon\end{cases}$
$Y_{i,j,l,i',j',l'} =$	$\begin{cases}1, & \text{si } O_{i,l,j}\ précède\ O_{i',l',j'} \text{ sur la machine k}\\ 0, & sinon\end{cases}$

- Objective 'Minimisation du makespan'

L'objectif consistant à réduire au minimum le makespan, qui est le temps maximum d'exécution de tous les jobs, il est défini comme suit : min C_{max} . L'objectif est soumis aux contraintes suivantes :

$$f_{c_{max}} = \min \max_{j \in J} C_{i,j} \,.\, X_{i,l} \,.\, Z_{i,j,k,l}$$

- Chaque job ne peut sélectionner qu'un seul plan de processus alternatif :

$$\sum_{l \in T_i} X_{i,l} = 1 \qquad \forall\, i \epsilon N \qquad (1)$$

- Attribuer les opérations à leur machines correspondantes selon le plan de processus sélectionné pour ce job :

$$\sum_{k \in R_{i,l,j}} Z_{i,j,k,l} + (1 - X_{il}) = 1 \qquad \forall\, i \epsilon N, l \epsilon T_i, j \epsilon NO_{i,l} \qquad (2)$$

- Garantir que deux opérations d'un même job suivent une séquence correcte en déterminant le temps de réalisation des opérations pour chaque job :

$$C_{i,0} = 0 \qquad \forall\, i \epsilon N \qquad (3)$$

$$C_{i,j} \geq C_{i,j-1} + \sum_{k \in R_{i,l,j}} P_{i,j,k,l}\, Z_{i,j,k,l} \qquad \forall\, i \epsilon N, l \epsilon T_i, j \epsilon NO_{i,l}, j \geq 1 \qquad (4)$$

- Garantissent que la relation de précédence entre deux opérations sur la même machine :

$$C_{i,j} \geq C_{i',j'} + P_{i,j,k,l} - A\,(3 - Y_{i,j,l,i',j',l'} - Z_{i,j,k,l} - Z_{i',j',k',l'}) \qquad \forall\, i \epsilon N, i < |n|, i' > i, l \epsilon T_i, l' \epsilon T_i, j \epsilon NO_{i,l}, j' \epsilon NO_{i',l'}, k \epsilon R_{i,l,j} \cap R_{i',l',j'} \qquad (5)$$

$$C_{i',j'} \geq C_{i,j} + P_{i',j',k',l'} - A(2 + Y_{i,j,l,i',j',l'} - Z_{i,j,k,l} - Z_{i',j',k',l'}) \qquad \forall\, i \epsilon N, i < |n|, i' > i, l \epsilon T_i, l' \epsilon T_i, j \epsilon NO_{i,l}, j' \epsilon NO_{i',l'}, k \epsilon R_{i,l,j} \cap R_{i',l',j'} \qquad (6)$$

- Calcule du makespan :

$$C_{max} \geq C_{i,j} \qquad \forall\, i \epsilon N, l \epsilon T_i, j \epsilon NO_{i,l} \qquad (7)$$

- Définir respectivement les variables continues et binaires :

$$C_{i,j} \geq 0 \qquad \forall\, i \epsilon N, l \epsilon T_i, j \epsilon NO_{i,l} \qquad (8)$$

$$X_{i,l}, Z_{i,j,k,l}, Y_{i,j,l,i',j',l'} \in \{0,1\} \qquad \forall\, i \epsilon N, l \epsilon T_i, j \epsilon NO_{i,l} \qquad (9)$$

2.2. Cadre général de l'approche hybride pour IPPS

Le cadre de cette approche adaptée pour un problème d'intégration de planification des processus et ordonnancement est montré dans la figure 5.1. L'idée est d'obtenir une solution de meilleure qualité aux étapes successives en employant des algorithmes plus rigoureux.

Trois problèmes sont conçus sur la base de planification de processus et l'ordonnancement ; un problème JS avec un plan de processus fixé, un problème JSF avec des machines alternatives et un problème IPPS avec une grande flexibilité du processus.

La structure de notre approche est basée sur trois étapes :

1) La première étape est la génération d'une solution initiale à l'aide de la fonction shifting bottleneck heuristique. la SBH fonctionne avec une instance d'un JS dérivée du modèle étudié.
2) La deuxième étape est basée sur la recherche tabou qui considère l'instance job shop flexible en tenant compte des machines alternatives de chaque opération.
3) La troisième étape dont les plans de processus sont considérés par l'algorithme de kangourou.

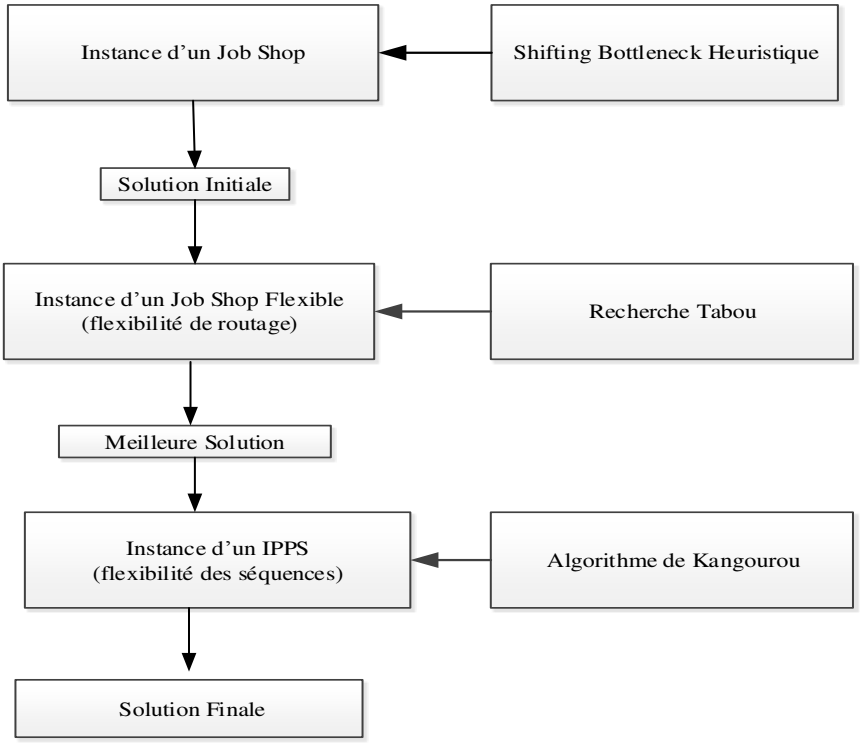


Figure 5.1. Cadre général de l'approche proposée

2.3. Fonctionnement de l'approche proposée dans un IPPS

2.3.1. Adaptation de l'heuristique shifting bottleneck (SBH)

La SBH décompose le problème d'ordonnancement envisagé en une série de sous-problèmes d'ordonnancement plus petits (Scheduling Sub-Problem) 'SPP').

Nous examinons un ensemble de tous les groupes de machines disponibles qui représentent les ressources de production : $M= \{M_1, \ldots M_k\}$.
On note M^s un ensemble de groupes de machines qui sont ordonnancées. Cela signifie qu'une décision d'ordonnancement pour chaque opération nécessitant l'utilisation d'une machine de M^S est déjà faite. L'algorithme 5.1. décrit la structure de base du SBH.

Algorithme 5.1 : La structure de SBH

1. Début :
2. Poser $M^s=\emptyset$;
3. Repeter
4. Selectionner le groupe de machines critique $M_c \in M \setminus M^s$;
5. Résoudre le SPP de M_c ;
6. Introduire la solution correspondante pour M_c dans la solution du graphe G ;
7. Ré-ordonnancer les opérations du groupe de machines de M^s ;
8. Poser $M^s \leftarrow M^s \cup \{M_c\}$;
9. Jusqu'à $M^s = M$;
10. Fin.

L'idée principale du SBH est de décomposer le problème d'ordonnancement en fonction des groupes de machines et de les ordonnancer un par un. Les groupes de machines les plus critiques (bottlenecks) sont programmés au début. En outre, nous donnons une description plus détaillée des différentes étapes de l'algorithme.

2.3.1.1. Criticité des sous-problèmes

Le concept de machines critiques ou de groupes de machines est de central intérêt du SBH. Différentes approches pour définir et estimer la criticité cr(M_i) de certains SPP pour le groupe de machines M_i au cours de l'algorithme peuvent être trouvées dans la littérature telle que (Monch et Zimmermann 2007).

Dans le cadre de l'approche conventionnelle, la criticité d'un sous-problème du SBH est mesurée dynamiquement, c'est-à-dire que la criticité d'un sous-problème fixe peut changer en fonction de l'itération actuelle de l'algorithme. L'approche la plus courante pour mesurer la criticité consiste à résoudre le SSP correspondant.

Après avoir résolu les sous-problèmes pour tout $M_i \in M \setminus M^s$ et la mesure de leurs valeurs de criticité cr (M_i), on peut découvrir le groupe de machines le plus critique M_{cr} avec :

$$cr(M_{cr}) = \max_{Mi \in M \setminus Ms}\{cr(Mi)\}$$

L'ordonnancement correspondant à M_{cr} doit être reflété dans le graphique de la solution G.

2.3.1.2. Ré-optimisation

Réaliser la phase de ré-optimisation au sein du SBH est une procédure importante qui permet d'améliorer significativement la qualité des solutions trouvées. Si cette étape est omise, la performance du SBH peut se dégrader considérablement (Bulbul 2011), et la procédure SBH peut même être surperformée par des règles de priorité.

Fondamentalement, il y a deux approches principales pour exécuter la phase de re-optimisation au sein du SBH. La procédure conventionnelle de ré-optimisation de l'application de SBH est décrite par l'algorithme suivant (Pinedo 2008).

Algorithme 5.2 : La phase de ré-optimisation de SBH

1. Pour $M_i \in M^s \setminus \{M_{cr}\}$ Faire
2. Retirer la sélection A_S^{Mi} correspond à M_i de G
3. Formuler et résoudre un nouveau SPP de M_i
4. Introduire la nouvelle sélection A_S^{Mi} au graphe G
5. Fin Pour

La procédure de ré-optimisation se poursuit jusqu'à ce qu'il n'y ait plus d'amélioration de la qualité de la solution. Habituellement, le nombre de cycles de ré-optimisation est limité afin d'éviter des temps de calcul élevés (Demirkol et al. 1997).
Dans notre application, nous appliquons la procédure itérative ou le C_{max} de l'ordonnancement partiel correspondant aux groupes de machines déjà programmés est prise en compte.

2.3.1.3. Méthodes de solution des sous-problèmes

La performance du SBH peut se manifester de manière significative dans les domaines selon la qualité de ses solutions et de sous-problèmes. Il est également important de formuler les sous-problèmes de manière appropriée en fonction du critère d'optimisation.

Dans notre étude, nous considérons les environnements machines indépendantes. La phase initiale commence par la sélection du premier plan de processus de chaque job et les routages correspondants en fonction de la règle SPT dont le temps opératoire est utilisé comme un paramètre de priorité.

2.3.2. La recherche locale

Une fois la solution initiale reçue par SBH, la phase d'optimisation locale commence. La RT détermine alors le voisinage de la solution actuelle et l'évalue afin de choisir le meilleur voisin non-tabu ou le meilleur qui satisfait le critère d'aspiration. Par la suite, le mouvement sera mémorisé dans la liste des tabous et la nouvelle solution actuelle sera obtenue après avoir effectué le mouvement sélectionné et après avoir satisfait toutes les contraintes du problème. Dans le cas où la nouvelle solution actuelle améliore la meilleure solution rencontré jusqu'à présent, la RT l'envoie à l'AK.

Dans cette phase, plusieurs problèmes SJF doivent être résolus, bien que la RT soit rapide, l'exécution d'un grand nombre d'itérations dans l'approche hybride pourrait encore conduire à des temps de calcul qui ne sont pas pratiques. En raison de l'ampleur des recherches dans le cas de nombreux jobs, un grand nombre d'itérations ont souvent à exécuter avant de trouver une meilleure sélection des machines pour chaque opération.

La RT suit une stratégie consistant à décider comment le problème des SJF doit être résolu à l'étape actuelle du processus de recherche. Autrement dit, Les décisions de routages des opérations sont prises à l'aide de la recherche tabou

2.3.3. La recherche globale

Dès que le SBH-RT atteindra une région prometteuse de l'espace de recherche, nous devrons déterminer les plans de processus de haute qualité. Cette étape est basée sur l'approche AK. L'algorithme commence à partir d'une solution faisable dont la valeur Cmax la plus faible est fournie par SBH-RT, et poursuit la recherche globalement en évaluant la qualité des plans de processus et détermine les ordonnancements. Chaque nouveau plan de processus sélectionné soit évalué en résolvant l'instance de planification des JSF correspondante.

Cette approche est avantageuse parce qu'elle permet de considérer la sélection et l'ordon-nancement des plans de processus. D'une part la RT détermine les machines alternatives qui mènent à des solutions avec des valeurs Cmax quasi-optimales au début du processus de recherche, d'autre part l'AK dépense moins d'efforts pour trouver des solutions de haute qualité dans les premières itérations de l'approche.

2.3.3.1.Job critique

Dans l'ordonnancement, il y a toujours des jobs qui influenceront la fonction d'objectif plus que d'autres. Puisque makespan a été choisi comme fonction objective pour cette étude, les jobs qui ont un temps d'usinage plus long sont plus critiques.
Si le temps de réalisation de ces jobs peut être maintenu à un minimum, il y a plus de chances de trouver un meilleur Cmax.

Le temps d'usinage minimum (Tm) requis pour chaque job est calculé et le job avec le Tm le plus grand est le job critique (Jc) et ce Tm est le temps critique Tc. Un job avec plus d'opérations non affectées auront plus de chance d'être sélectionnées par rapport aux jobs où il y a moins d'opérations non assignées. Une fois un job a été sélectionné, le plan de processus est utilisé pour sélectionner la première opération qui doit être affectée.

2.3.4 Intensification et diversification

Afin d'éviter qu'une grande partie de l'espace de recherche reste complètement inexplorée, il est important de diversifier la recherche pendant le processus d'optimisation. Pour cette raison, nous ajoutons de nouvelles techniques de diversification à la solution actuelle.

La force de notre nouvelle approche est, en fait, la coopération d'un ensemble de techniques de diversification par le biais de la procédure de saut. La diversification de la liste tabou est considérée localement comme une technique de diversification mais le saut est considéré globalement comme une technique de diversification pour l'ensemble du système.

L'intensification et la diversification jouent donc un rôle complémentaire. Comme pour la plupart des méthodes heuristiques, il n'existe pas de résultats théoriques garantissant la convergence d'une procédure tabou vers un optimum global. La raison principale de cet état tient à la nature même de la méthode. Celle-ci étant hautement adaptative et modulable, son analyse par les outils mathématiques est rendue plus difficile.

2.3.4.1. Les techniques d'intensification

La phase d'intensification est l'une des phases les plus importantes pour mieux exploiter l'espace de recherche en exécutant la fonction "On Wall" et "Inter Ineffective Collision". L'idée de cette fonction est de trouver une nouvelle solution à partir de la solution initiale.

2.3.4.2. Les techniques de diversification

Comme leur nom l'indique, sont conçues pour orienter la recherche vers de nouvelles régions. L'idée de base est de s'éloigner du minimum pseudo-local actuel en modifiant l'affectation actuelle de certains éléments pour favoriser des mouvements n'ayant pas été effectués ou à pénaliser les mouvements ayant été souvent répétés.

La diversification a un objectif inverse de l'intensification : elle cherche à diriger la recherche vers des zones inexplorées. Pendant la phase de diversification, des transitions sur les plans du processus des opérations sélectionnées au hasard sont effectuées. Si une meilleure

solution est trouvée, la diversification prend fin. Le processus de diversification commence dès que la solution n'est pas améliorée dans les dernières itérations.

2.4. La structure de voisinage adaptée pour l'IPPS

Au cours de l'étude exploratoire du RT, deux types de structures de voisinage sont ajoutés pour prendre en compte les deux décisions distinctes du problème IPPS, la sélection des plans de processus et l'affectation des machines. Le mouvement de cette méthode sera selon les opérateurs suivants :

- Pour une machine, permutez l'ordre de deux opérations séquentielles (échange d'opération) lorsque ces deux opérations sont sur le chemin critique de la solution.
- Pour une opération critique, affectez une autre machine (échange de machine).
- Pour un job avec au moins une opération critique, modifiez le plan de process (échange de séquence).

Ces trois opérateurs de mouvement seront également utilisés par l'algorithme de kangourou

RT essaie d'effectuer une légère perturbation dans la chaîne d'ordonnancement en utilisant une structure de voisinage basée sur le vecteur machine en continu jusqu'à un nombre d'itération définie. Puis, l'AK saute au deuxième voisinage en utilisant une structure de voisinage basée sur le vecteur du plan de processus en continu jusqu'à un nombre d'itération définie.

3. Étude de cas

Afin d'expliquer le problème d'intégration, un modèle de production déjà utilisé par Park et Choi (2006) est présenté dans le tableau 5.1. Park et Choi (2006) ont utilisé un algorithme génétique pour résoudre ce problème et leurs résultats sont présentés dans la section suivante.

Ce modèle comprend 3 jobs dont chacun comporte 3 opérations. Lors du traitement de chacune des opérations, il existe un ensemble de machines alternatives pour chaque opération. La dernière colonne représente les plans de processus possibles pour chaque job.

Ce modèle présente trois difficultés. :

- La première consiste à choisir l'ordre dans lequel les opérations de chaque job sont séquencées (flexibilité de séquencement). Par exemple, le job J_2 a trois plans de processus ;
- La seconde consiste à affecter chaque opération à une machine sélectionnée dans l'ensemble des machines alternatives (flexibilité de routage). En outre, l'opération O_{32} peut être traitée par M_3, M_1 ou M_2, avec un temps de traitement égale à 5, 5 et 6, respectivement, mais elle doit être traitée après l'opération O_{31} ;
- La troisième consiste à ordonnancer les opérations devant chaque machine.

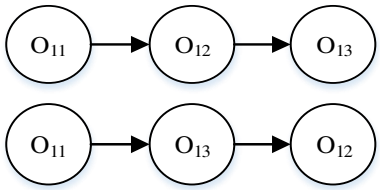
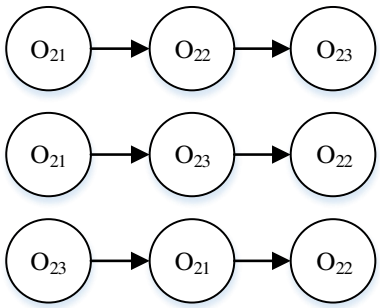
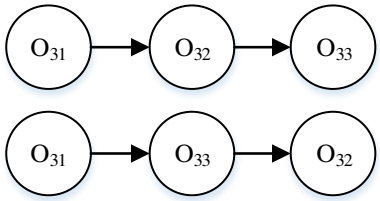
J_i	O_{ij}	Machines Alternatives			Plan de processus
		M_1	M_2	M_3	
1	O_{11}	6	6	-	
	O_{12}	6	5	6	
	O_{13}	-	-	4	
2	O_{21}	3	-	4	
	O_{22}	-	7	-	
	O_{23}	6	7	6	
3	O_{31}	7	-	8	
	O_{32}	5	6	5	
	O_{33}	-	4	-	

Tableau 5.1. Instance d'IPPS

3.1. Codage de la solution

Le codage utilisé considéré en cas de JSF est similaire en l'IPPS pour les deux vecteurs (opération et machines). Cependant, l'intégration des plans de processus génère un vecteur supplémentaire appelé vecteur de plan de processus. La figure 5.2 montre la chaine de ces trois vecteurs :

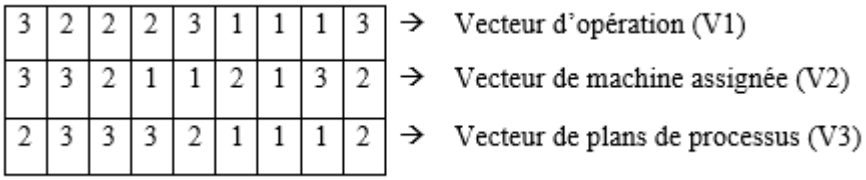


Figure 5.2. Le codage de la solution

3.1.1. Vecteur d'affectation des plans de processus

Chaque solution se compose de trois vecteurs - opération, machine et plan de processus. Sur la base des informations de la fig. 5.2, l'opération O_{11} est associée au premier plan de processus du job1 pendant que l'opération O_{22} est associée au troisième plan de processus du job2. Toutes les opérations d'un même job obtiennent le même numéro de plan de processus.

Par exemple, comme le job J_2 (voir tableau 5.1) a trois séquences d'opérations alternatives, toutes ses opérations sont associées au même numéro du plan de processus. L'ordre d'apparition des opérations dans le vecteur V1 représente l'ordre dans lequel les opérations sont planifiées (contrainte de précédence).

Les plans de processus sélectionnés ainsi que les machines sont présentés dans le tableau 5.2 et l'ordonnancement correspondant est illustré dans la figure 5.3.

Job_i	J_1			J_2			J_3		
Num du plan de processus	1			3			2		
O_{ij}	O_{11}	O_{12}	O_{13}	O_{23}	O_{21}	O_{22}	O_{31}	O_{33}	O_{32}
M_k	M_2	M_1	M_3	M_1	M_3	M_2	M_3	M_2	M_1
P_{ijk}	6	6	4	5	4	7	8	4	5

Tableau 5.2. Instance de PJS dérivée des vecteurs sélectionnés de la figure 5.2

Avec cette représentation, le problème IPPS est transformé en JSP. Plusieurs instances du PJS peuvent se produire en fonction de la sélection d'un plan de processus de chaque job et une machine pour chaque opération.

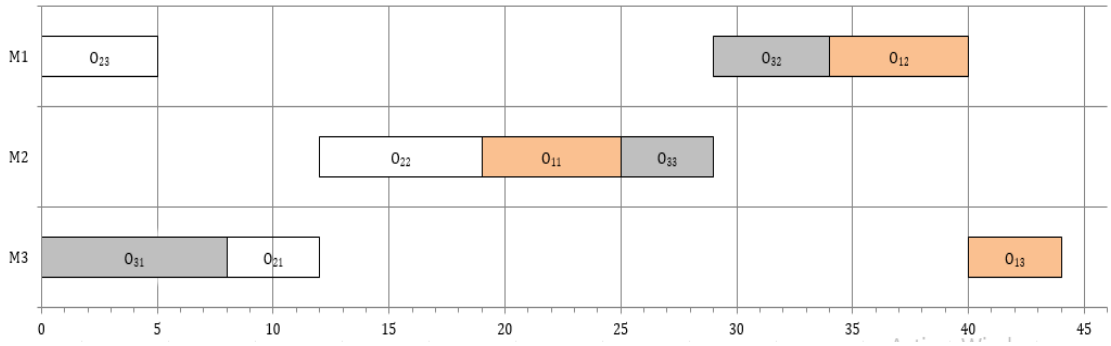


Figure 5.3. Le diagramme de gant de la solution dérivée des vecteurs de la figure 5.2

On peut remarquer que la représentation de cette solution donne un ordonnancement semi-actif, et pas nécessairement un ordonnancement actif. Par exemple, l'exemple de la figure 5.3 n'est pas un ordonnancement actif. (Un semi-actif est un ordonnancement dans lequel une opération ne peut pas commencer plus tôt sans changer la séquence des opérations).

4. Études expérimentales et discussion

Dans notre étude computationnelle, nous examinons plusieurs ensembles de cas problématiques qui sont décrits dans la documentation connexe. Une variété d'exemples de problèmes de référence joue un rôle important dans la comparaison des différentes approches de solution pour les problèmes énoncés.

Pour évaluer la performance de la méthode proposée, plusieurs algorithmes sont utilisés pour les évaluer avec l'approche proposée Pour tenir compte des effets stochastiques, notre algorithme a été exécuté 10 fois pour chaque problème.

4.1. Expérience 1

Dans cette section, le problème présenté dans (Park et Choi, 2006) est examiné. Nous comparons les résultats obtenus par notre approche avec ceux obtenus par l'algorithme génétique développé par Park et Choi (2006).
Avant de présenter nos résultats, la solution obtenue par Park et Choi (2006) est présentée dans la figure 5.4.

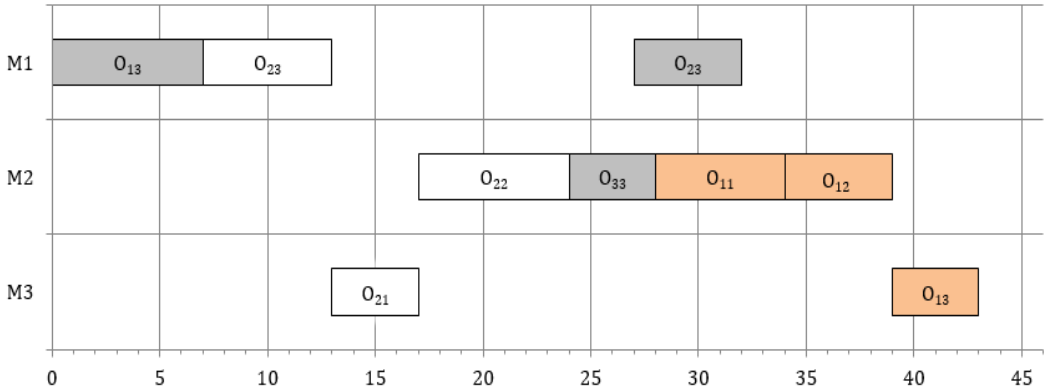


Figure 5.4. Le diagramme de gantt de la solution propose par Park and Choi (2006)

La solution proposée par Park et Choi (2006) n'est pas un ordonnancement actif. Par exemple, les opérations O_{11}, O_{12}, O_{13}, O_{33} et O_{32} pourraient commencer plus tôt, sur les mêmes machines, sans retarder aucune autre opération, ce qui donne un ordonnancement actif. Cependant, leur solution donne un Cmax=43. Dans ce qui suit, nous présentons les différentes solutions, sur une même étude de cas, obtenues à chaque étape de notre méthode.

4.1.1. Mise en œuvre de SBH

Comme il est mentionné plus haut la phase initiale commence par la sélection du premier plan de processus de chaque job. Pour les routages correspondants, la procédure adaptée dans le chapitre précédent est appliquée dont le temps de traitement est utilisé comme un paramètre de priorité. La SBH fonctionne avec les sections ombrées indiquées dans le tableau 5.3

Autrement dit, dans cette phase le problème étudié est un Job Shop (absence de machines alternatives et plan de processus alternatifs).

Le diagramme de Gantt de la solution initiale générée par SBH est illustré à la figure 5.6. Cette solution est un ordonnancement actif.

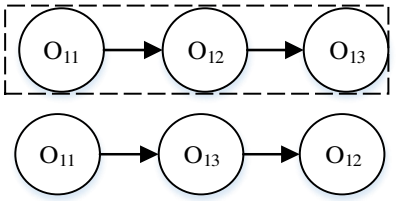
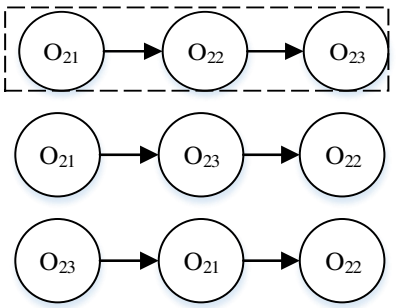
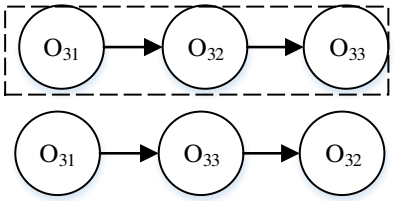
J_i	O_{ij}	Machines Alternatives			Plan de processus
		M_1	M_2	M_3	
1	O_{11}	**6**	6	-	
	O_{12}	6	**5**	6	
	O_{13}	-	-	**4**	
2	O_{21}	**3**	-	4	
	O_{22}	-	**7**	-	
	O_{23}	6	7	**6**	
3	O_{31}	**7**	-	8	
	O_{32}	5	6	**5**	
	O_{33}	-	**4**	-	

Tableau 5.3 Les opérations et les machines sélectionnées pour la génération de la solution initiale

La sélection des machines aux opérations est illustrée dans la figure suivante (5.5) :

3	2	2	2	3	1	1	1	3	→ Vecteur d'opération (V1)
1	1	2	3	3	1	2	3	2	→ Vecteur de machine assignée (V2)
1	1	1	1	1	1	1	1	1	→ Vecteur de plans de processus (V3)

Figure 5.5. Le codage de la sélection initiale

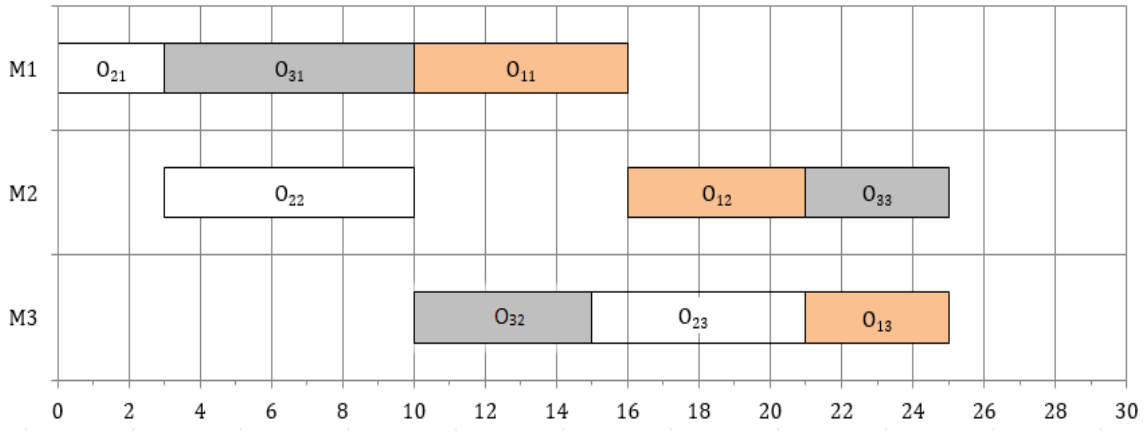


Figure 5.6. Diagramme de Gantt de la solution SBH

Le résultat de la SBH donne les opérations critiques suivantes : O_{21}, O_{31}, O_{11}, O_{12} et O_{33}.

3	2	2	2	3	1	1	1	3	→ Vecteur d'opération (V1)
1	1	2	3	3	1	2	3	2	→ Vecteur de machine assignée (V2)
1	1	1	1	1	1	1	1	1	→ Vecteur de plans de processus (V3)

Figure 5.7 Le codage de la solution de SBH

4.1.2. Mise en œuvre de RT

La figure suivante 5.8 montre les résultats obtenus après l'application de la recherche tabou RT. Dans cette phase, les machines alternatives sont incorporées. Les changements des machines pour les opérations critiques deviennent comme suit :
O_{21} : {M_1 → M_3} ; O_{11} : {M_1 → M_2} ; O_{12} : {M_2 → M_3}
Dans cette étape le problème traité est un job shop flexible (flexibilité de routage).

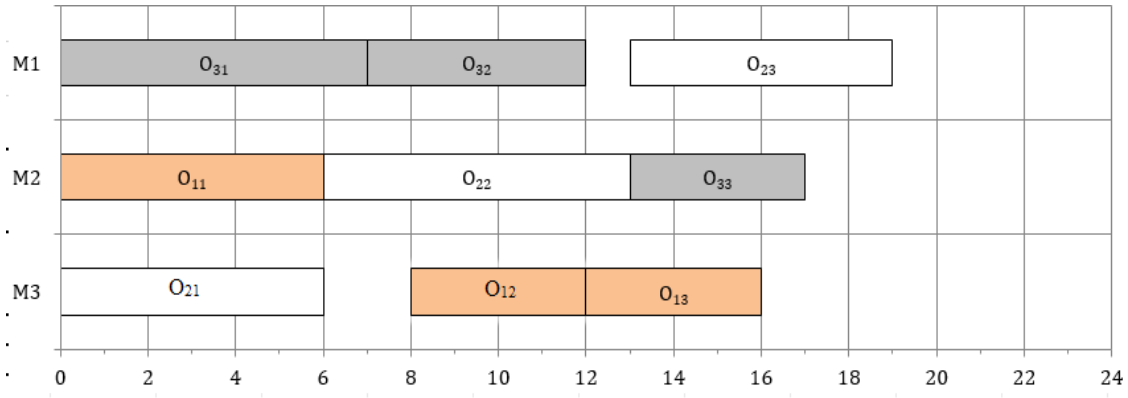


Figure 5.8. Diagramme de Gantt de la solution SBH-RT

Le nouveau Cmax=19 avec les opérations critiques suivantes : O_{11}, O_{22} et O_{23}.

3	2	2	2	3	1	1	1	3	→ Vecteur d'opération (V1)
1	3	2	1	1	2	3	3	2	→ Vecteur de machine assignée (V2)
1	1	1	1	1	1	1	1	1	→ Vecteur de plans de processus (V3)

Figure 5.9 Le codage de la solution de SBH-RT

4.1.3. Mise en œuvre d'AK

La figure suivante 5.10 montre les résultats obtenus après l'application de l'algorithme de kangourou AK. Dans cette phase, les plans du processus sont introduits (flexibilité des séquences). Dans cette étape le problème traité est un problème d'intégration de planifications processus et ordonnancement dans un job shop flexible (flexibilité de séquence-flexibilité de routage).

Deux types de changement sont considérés dans ce cas ; plans alternatif et machine alternative.

- Les changements des plans pour les opérations critiques deviennent comme suit :
  Le job 2 utilise le deuxième plan de processus alternatif avec la séquence d'opérations { O_{21}- O_{23} -O_{22}} et séquence de machine {M_3-M3-M2}.
- Les changements des machines alternatives pour les opérations critiques deviennent comme suit : O_{23} : {M_2 → M_3}

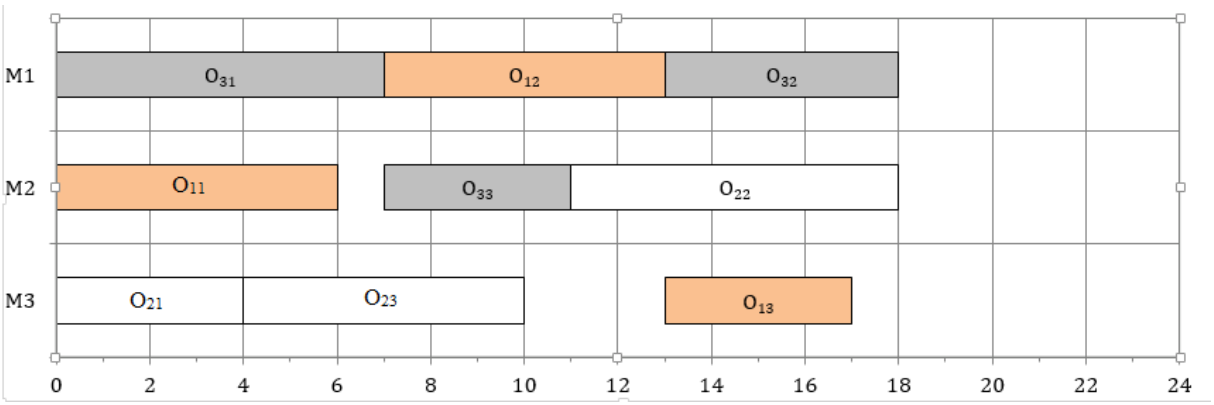


Figure 5.10. Diagramme de Gantt de la solution SBH-RT-AK

Le nouveau Cmax=18 avec le nouveau codage de la solution finale illustrée dans la figure 5.11

3	2	2	2	3	1	1	1	3	→ Vecteur d'opération (V1)
1	3	3	2	2	2	1	3	1	→ Vecteur de machine assignée (V2)
2	3	3	3	2	1	1	1	2	→ Vecteur de plans de processus (V3)

Figure 5.11 Le codage de la solution de SBH-RT-AK

Le tableau suivant 5.4 donne le temps de calcul et la valeur Cmax obtenus à chaque étape de notre approche hybride.

Methode	**CPU (sec)**	**C_{max}**	**Taux d'amélioration**
GA	Non fourni	44	
SBH	<1	25	43%
SBH+TS	1.45	19	21%
SBH+TS+KA	1.8	18	5%

Tableau 5.4 Résultats de calcul entre GA et SBH-TS-KA

4.1.4 Analyse de la première expérience

Dans cette expérience, la solution fournie par Park et choi (2006) n'est pas bien équilibrée (voir Figure 5.4). Parmi les neuf opérations, quatre d'entre elles sont affectées à la machine M2, ce qui donne une charge de travail de 22 pour cette machine, alors que la machine M_3, avec seulement deux opérations, a une charge de 8.
Deux opérations successives du job J_1 :O_{11} et O_{12} sont effectuées sur la même machine M_2.
Il y a un autre cycle, avec le job J_3 sur la machine M_1. Ces cycles sont l'une des causes du mauvais fonctionnement de la procédure d'AG proposée par Park et Choi (2006).

La solution fournie par le SBH est la solution la plus équilibrée de toutes les solutions obtenues (une charge de travail de 16, 16 et 15 pour les machines M_1, M_2 et M_3 respectivement). Comme on peut le voir dans le tableau 5.4, le makespan est significativement améliorée par SBH à la solution obtenue par Park et Choi (2006). Ceci montre l'importance de la solution initiale lors de l'utilisation de la métaheuristique.

De plus, comme la SBH ne traite qu'avec un seul JSP la solution est obtenue immédiatement. Grâce à la haute qualité de la solution initiale du RT, l'exploration de la RT permet d'améliorer à nouveau, et largement, le makespan (21%).

A l'intérieur du système hybride, les méthodes SBH et RT sont utilisées pour obtenir des résultats autour d'une exploitation locale. Tandis que l'AK est utilisé pour réaliser l'exploration globale. Bien que la RT ait déjà bien exploré l'espace de recherche et ait donné une solution de haute qualité, l'AK parvient à améliorer cette solution.

Grâce à sa procédure de saut, l'AK peut explorer globalement l'espace de recherche et enfin la flexibilité du plan de processus est utilisée pour deux jobs et permet d'améliorer la solution. Il peut on notera qu'à la fin de l'approche hybride (SBH-TS-KA), la solution finale s'est améliorée de 24% par rapport à la solution obtenue avec le seul SBH, et surpasse de 24% la solution obtenue avec le SBH-RT, et l'algorithme génétique par un facteur 2.3.

4.2. Expérience 2

Le même problème est abordé par Li et al (2010b). Il comprend 6 jobs pour être traitées sur 5 machines. Il y a 18 opérations à effectuer. Multiple machines peuvent être utilisées pour traiter chaque opération. L'objectif était de réduire au minimum le makespan en utilisant une AG hybride pour un problème IPPS.

La figure 5.12 illustre le diagramme de gantt de la solution initiale générée par SBH. L'objectif de la présentation de cette solution est de la comparer avec la solution du modèle sans intégration (No-I-M) présenté dans (Li et al. 2010b). Ils ont comparé le modèle d'intégration (I-M) avec le No-I-M, ce qui signifie que chaque job sélectionne le premier plan de processus comme étant son seul plan. De même, la SBH s'inscrit dans notre démarche : chaque job est affecté au premier plan.

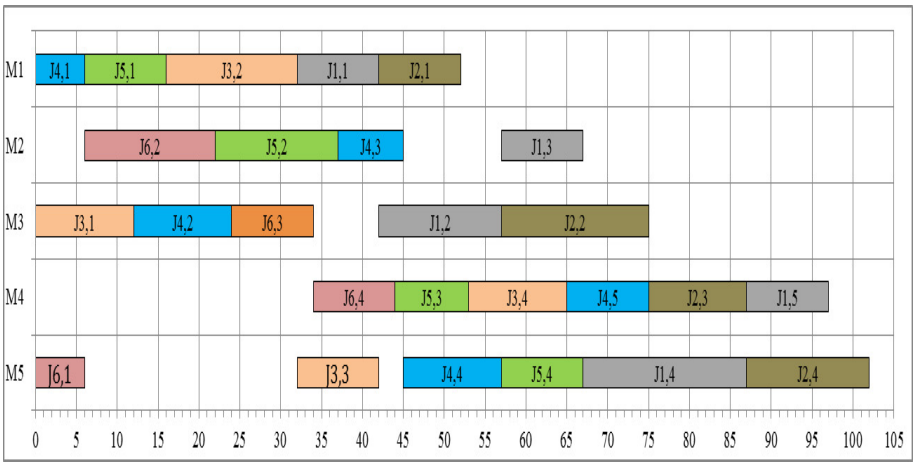


Figure 5.12. Diagramme de Gantt de la solution SBH (J2,1 signifie première opération de job 2)

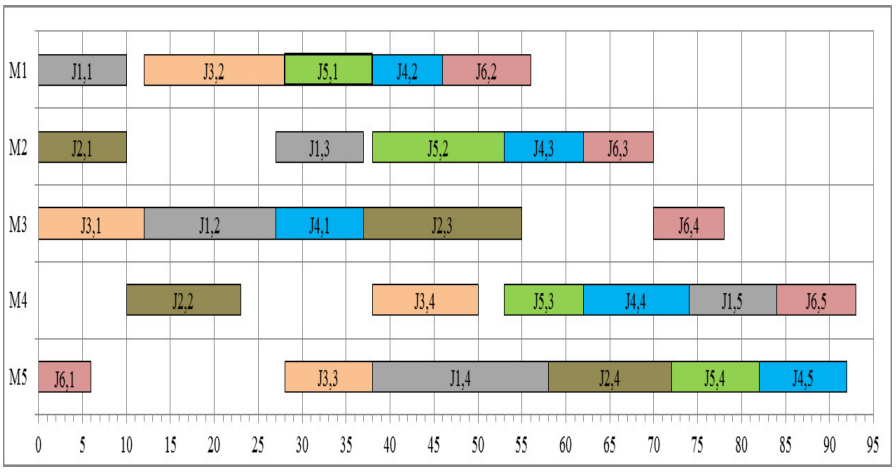


Figure 5.13 Diagramme de Gantt de la solution SBH+RT+AK

En comparant les résultats des figures 5.12 et 5.13 les plans de processus sélectionnés des différents jobs sont différents. Comme indiqué, avec la méthode SBH (fig. 5.12), le

processus sélectionné est le premier pour tous les jobs. Cependant, dans la figure 5.13 les plans de processus sélectionnés sont différents pour certains jobs. Par exemple, le plan de processus sélectionné pour le job 2 est le troisième plan. Il en va de même pour les jobs 4 et 6, où le plan de processus sélectionné devient le deuxième et le troisième plan respectivement. Par l'extension des plans de processus, l'espace de recherche devient beaucoup plus grande que la première étape, c.-à-d. SBH seulement. De plus, la capacité de notre approche peut encore être améliorée par l'intégration des machines alternatives. Nous rappelons que dans (Li et al. 2010), le fait que les opérations peuvent être effectuées sur des machines alternatives n'est pas pris en compte.

Job	**Plans de processus alternatifs**	
	Intégration	**No-Intégration**
1	1(10)- 3(10)- 2(15)- 5(10)- 4(20)	1(10)- 3(10)- 2(15)- 5(10)- 4(20)
2	2(10)- 4(13)- 3(18)- 5(14)	1(10)- 3(18)- 4(12)- 5(15)
3	3(12)- 1(16)- 5(10)- 4(12)	3(12)- 1(16)- 5(10)- 4(12)
4	3(10)- 1(8)- 2(9)- 4(12)- 5(10)	1(6)- 3(12)- 2(8)- 5(12)- 4(10)
5	1(10)- 2(15)- 4(9)- 5(10)	1(10)- 2(15)- 4(9)- 5(10)
6	5(6)- 1(10)- 2(8)- 3(8)-4(9)	5(6)- 2(16)- 3(10)- 4(10)
Cmax=93		Cmax=102

Tableau 5.5 La comparaison entre le modèle d'intégration et no-intégration

4.3. Expérience 3

Le problème de cette expérience est traité par Amin-Naseri et al (2012), il comporte 6 jobs et 5 machines. Des machines alternatives sont envisagées pour les opérations.
Dans notre approche, la SBH sélectionne la machine qui a le temps de traitement le plus court pour chacun opération. La figure 5.14 illustre le diagramme de Gantt de la solution obtenue par notre approche hybride pour ce problème.

Comme l'indique, la solution obtenue est de 27 par rapport à la valeur de 33 obtenue par SBH et No-I-M. L'avantage de ce système est la disponibilité de plusieurs machines et plan de processus, ce qui donne plus de flexibilité à l'infrastructure du système. Bien qu'un plan de processus fixe puisse guider l'ordonnancement à des goulots d'étranglement ou charges des machines déséquilibrées.

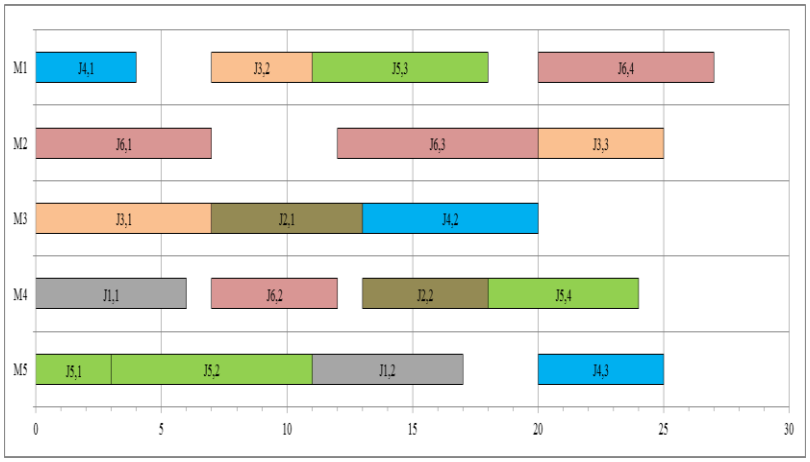


Figure 5.14 Diagramme de Gantt de la solution SBH-RT-AK

4.4. Analyse de la deuxième et troisième expérience

Dans les deux problèmes étudiés, nous pouvons distinguer la performance de l'approche hybride à un stade précoce. La performance de notre approche s'explique par l'incorporation des machines possibles pour chaque opération et le plan de processus sélectionné pour chaque job. Les spécifications de chaque problème et les résultats numériques sont résumés dans les tableaux suivants :

Paramètre	**No-Intégration**		**Intégration**	
	EA	**SBH-RT-AK**	**EA**	**SBH-RT-AK**
Cmax	102	102	92	93
TA%	100	100	100	91
CPU (s)	3.31	1.5	3.23	2.28

Tableau 5.6 Résultats comparatifs de l'expérience 1

D'après le tableau 5.6, on tente d'évaluer l'efficacité de la solution initiale générée par la SBH. La qualité de la solution n'a fait que progresser de la solution initiale à la solution finale. La SBH obtient des résultats similaires en beaucoup moins de temps que le No-I-M. Cependant, il existe encore un potentiel d'amélioration de l'efficacité et de l'optimalité du calcul si de nouvelles caractéristiques sont introduites pour la SBH.

Paramètre	**HGA**	**EA**	**SBH-RT-AK**
Cmax	27	33	27
TA%	0	-22	0
CPU (s)	-	3.20	3.45

Tableau 5.7 Résultats comparatifs de l'expérience 2

Dans la deuxième expérience, le makespan est diminué de 33 à 27, soit un taux d'amélioration de -22%. De plus, il a été en mesure de trouver une solution presque optimale

par rapport à celles de Li et al. (2010b). Pour cette expérience, les plans de processus sélectionnés dans l'approche proposée ne sont pas tous optimaux. Les résultats révèlent que la modification des plans de processus aide à réduire le Cmax. Alors que la meilleure valeur est trouvée en considérant simultanément les machines alternatives et les plans de processus alternatif.

4.5. Expérience 4 :

Cette expérience regroupe quatre instances adoptées dans la littérature de l'IPPS. Les spécifications de chaque problème et les résultats des solutions sont résumés dans le tableau 5.7. Les problèmes de référence sont choisis de manière à ce que leur fonction objective et leurs hypothèses soient les mêmes que celles examinées dans le présent travail.

- *Problème P1 (Sundaram et Fu 1988)* : Le problème est d'assigner 5 jobs à 3 machines; chaque job comporte 4 opérations. L'article original a trouvé un Cmax= 38. Ce problème a été résolu par de nombreux chercheurs et les meilleurs L'espérance de vie atteinte pour ce problème est de 33 ans selon Lihong et Shengping (2012).

- *Problème P2 (Li et al 2010)* : Ce problème est déjà traité dans l'expérience 2.

- *Problème P3 (Shao et al 2009)* : Ce problème est une version modifiée du problème original. présenté dans (Moon et Seo 2005). Le problème est d'affecter 5 jobs avec un total de 21 opérations à 6 machines. Shao et al (2009) ont réussi à obtenir un Cmax=28 pour ce problème. Naseri et Ahmed (2012) a amélioré le makepan à 27. La solution exacte à ce problème, est basée sur un algorithme génétique hybride.

- *Problème P4 (Moon et al 2008)* : Ce problème consiste à attribuer 13 opérations de 5 jobs sur 5 machines différentes. Le même problème a été résolu par Shao et al (2009) et Lihong et Shengping (2012) entre autres 5 jobs et 5 machines, et le nombre d'opérations dans chaque job varie de 2 à 4. Le meilleur makespan égale à 14 est obtenu par l'application d'une méthode basée sur l'AG.

Problème	n×m	SBH-RT-AK	Exact	Hybrid GA	IGA
P1	5×5	33	33	33	33
P2	6×5	27	27	27	27
P3	5×6	28	27	27	28
P4	5×5	14	14	14	16

Tableau 5.8. Les résultats numériques pour les petites instances d'IPPS

4.6. Analyse de la quatrième expérience

Pour chaque problème, la solution de l'algorithme proposé est comparée avec la solution exacte. Les résultats montrent que notre approche est capable de résoudre ces problèmes. La SBH-RT-AK s'est avérée efficace et acceptable pour optimiser le problème IPPS en ce qui

concerne la flexibilité de la machine, et de la séquence en comparant avec d'autres algorithmes.

Les résultats expérimentaux montrent que cette approche peut obtenir un résultat d'optimisation satisfaisant pour P3 et une solution optimale pour les trois autres instances. Pour le P4 l'approche proposée s'est révélé plus performant que l'algorithme d'IGA.

5. Résumé des expériences

- ✓ Le nombre de machines disponibles pour chaque opération a une influence significative sur l'amélioration du makespan, ainsi que le nombre de plan de processus.
- ✓ L'importance de spécifier des machines alternatives pour une opération donnée a été soulignée. Cette approche vise non seulement à minimiser le makespan, mais aussi à équilibrer les charges pour les machines.
- ✓ L'AK dispose d'une puissante capacité de recherche globale et la RT dispose d'une fonction de recherche locale précieuse. Cette méthode combine les avantages de RT et AK. Par conséquent, elle a un potentiel de recherche et peut très bien équilibrer l'intensification et la diversification.
- ✓ Le principe d'AK est d'éviter la convergence vers un optimum local et de déterminer efficacement l'optimum global d'une fonction objective dans un espace de solution complexe.
- ✓ L'approche proposée permet de trouver un compromis entre la qualité de la solution et le temps de calcul.
- ✓ La meilleure solution est obtenue quand les plans de processus sont intégrés.
- ✓ En comparant les résultats entre le modèle d'intégration et le modèle no-intégration, on constate que les nouveaux plans sélectionnés par les jobs sont différents et que le makespan du modèle no-intégration est inférieur que celle du modèle d'intégration.
- ✓ Les solutions correspondantes obtenues par le SBH sont plausibles dans de nombreux cas, tandis que la solution finale obtenue par l'approche hybride est améliorée. Ceci montre l'importance de la solution initiale lors de l'utilisation des métaheuristiques. Une explication possible peut être l'extraction et l'utilisation des informations utiles contenues dans la solution initiale pendant le processus d'évolution.

6. Conclusion

Une variété de ressources de production alternatives (machines-outils, outils de coupe, directions d'accès aux outils, etc.) fait que le problème de planification et d'ordonnancement des processus intégré (IPPS) est fortement NP-hard (non polynôme déterministe) en termes d'optimisation combinatoire. Par conséquent, une solution optimale au problème est cherchée dans un vaste espace de recherche. Afin d'explorer l'espace de recherche de façon exhaustive et d'éviter d'être pigé par un optimum local, nous avons élaboré une approche hybride qui intègre simultanément la fonction de planification et d'ordonnancement dans un job shop flexible basée sur la combinaison de la recherche locale et globale. Cette approche a été adaptée dans un système job shop flexible autrement dit sans intégration des plans de processus, et elle a fournis des résultats prometteurs dans la plupart des instances.

La même stratégie a été adoptée : A partir d'une solution initiale réalisable créée par l'heuristique shifting botteleneck qui dépend essentiellement du sous-problème employé, la recherche tabou reçoit cette solution comme un point de départ pour la recherche locale. Cette dernière améliore la solution obtenue à partir du SBH par l'intégration de la flexibilité des machines. Encore une fois, la solution obtenue devient la solution initiale pour la prochaine phase. Dans cette phase, la zone de recherche est étendue par l'intégration de la flexibilité de séquence à l'aide de l'algorithme de kangourou, qui améliore la solution actuelle en effectuant un saut élargissant la zone de recherche.

Afin d'améliorer la performance d'optimisation de l'approche proposée, des essais expérimentaux ont été réalisés sur un certain nombre de problèmes prononcés dans la littérature sont présentés afin d'optimiser le makespan. Dans l'ensemble, les tests indiquent que l'approche proposée est plus acceptable et prometteuse pour le problème IPPS, elle atteint des solutions optimales ou quasi-optimales.

Dans ce chapitre, nous proposons une approche en trois étapes. L'efficacité de SBH-RT-AK peut être notée par les solutions de faisabilité intermédiaires qui convergent en un délai raisonnable, ce qui est tout à fait acceptable dans un cas réel des systèmes de production. Avec une telle approche, il peut être d'une grande aide pour les utilisateurs qui sont intéressés à obtenir des informations sur des solutions acceptables dans un court terme de temps.

Conclusion générale

Dans les systèmes de production traditionnels, tant le plan de processus que l'ordonnancement sont préparés pour un environnement de production statique avec les ressources de fabrication disponibles. Dans les approches traditionnelles, la planification des processus et l'ordonnancement se faisaient de façon séquentielle et présentaient donc les problèmes suivants :

- Traditionnellement, la planification des processus était donc effectuée séparément pour un job dans un système de fabrication. Cela signifie que les machines manufacturières pour un job ont été utilisées sans tenir compte des autres jobs.
- Il en est résulté des machines favorables pour chaque job, ce qui a entraîné une surcharge de certaines machines (par exemple les machines-outils), tandis que les autre machines sont restées inactifs.
- Les plans d'ordonnancement ont souvent été générés après la phase de planification du processus. Par conséquent, les plans d'ordonnancement peut présenter de graves déséquilibres dans l'utilisation des machines et des goulots d'étranglement en raison de l'absence d'un équilibre entre l'utilisation des machines et les goulots d'étranglement déterminés précédemment par des plans de processus fixes.
- En production réelle, l'environnement de fabrication qui change fréquemment et implique des incertitudes et des perturbations telles que des pannes de machines, des arrivées de commandes urgentes et des annulations de commandes. En conséquence, le plan du processus et l'ordonnancement préparé à l'étape de la planification peut devenir moins efficace, voire irréalisable, en raison de ces changements.

Pour faire face aux perturbations inattendues, il est nécessaire de tenir compte du plan du processus et de l'échéancier dynamiquement. Par conséquent, l'IPPS dynamique qui peut mieux modéliser l'environnement de production pratique devient de plus en plus important. Cependant, les tâches de planification et d'ordonnancement des processus sont très compliquées et prennent beaucoup de temps, si elles sont appliquées à des changements dynamiques. L'activité d'ordonnancement, avec un plan de processus fixe, doit souvent résoudre des conflits entre les ressources disponibles, en raison de l'évolution de l'environnement. Fréquemment le plan de processus d'origine doit être modifié pour s'adapter aux changements dans l'atelier.

Étant donné que la flexibilité a été reconnue comme un outil permettant d'améliorer le rendement du système et de gérer les incertitudes, une approche faisant appel à la flexibilité est plus appropriée dans le scénario actuel de production. Ainsi, dans le présent travail, on a tenté d'officialiser une méthodologie qui peut intégrer rapidement les fonctions de planification et d'ordonnancement et qui peut être mise en œuvre dans un atelier avec les services existants de planification et d'ordonnancement des processus. Le Makespan est la fonction objective d'intérêt dans notre étude dont il est considéré comme l'objectif la plus répandue dans les ateliers de production.

Dans ce travail, nous avons traité la problématique d'application des Métaheuristiques hybrides pour un job shop flexible sans et avec intégration des plans de processus. Les méthodes hybridées sont : l'heuristique shifting bottleneck, la recherche tabou et l'algorithme de kangourou. Des tests sont réalisés, afin d'évaluer et de comparer la performance de notre approche avec les différentes approches implémentées. Les expériences de calcul fournissent plusieurs aperçus intéressants sur le comportement de l'approche intégrée. L'intuition la plus importante est que cette approche peut potentiellement améliorer la performance de l'environnement de production de manière significative en termes de Cmax. Les résultats présentés sont obtenus pour différents scénarios de simulation avec de multiples. Il s'avère que la performance de l'approche intégrée est plus sensible à la qualité des plannings générés.

Au cours des trois dernières décennies, de nombreux chercheurs ont mené des études approfondies et approfondies sur les problèmes IPPS et job shop flexible ont obtenu de bons résultats. Sur la base de l'analyse complète des résultats et des lacunes de la recherche existante, cette thèse a mené une recherche détaillée et approfondie dans les aspects suivants :

- **Étude de l'état de l'art**

Un état de l'art sur les problèmes liés à l'IPPS a été donné sur la base des travaux actuels ainsi qu'une analyse documentaire des problèmes étroitement liés. Les concepts et les définitions de la planification des processus, de l'ordonnancement des jobs ont été présentés. Basé sur l'analyse de la relation entre ses deux fonctions, la nécessité de L'intégration des deux a été illustrée. Aussi, les approches de mise en œuvre de l'optimisation de l'IPPS dans la documentation ont été résumées en présentant ces techniques clés.

La bibliographie de l'ordonnancement des jobs dans un job shop/ job shop flexible et l'IPPS révèle que plusieurs mesures de performance telles que makespan, le retard moyen, le délai de flux, le retard total et le retard maximum sont utilisés par les chercheurs, le makespan et le retard sont les plus fréquemment utilisés pour évaluer le problème du IPPS flexible et le temps moyen de flux.

- **Une approche hybride pour la minimisation du makespan dans un job shop flexible**

Un nouveau modèle hybride a été mis en place pour le problème de job shop flexible. Cette approche consiste à hybrider trois métaheuristiques et combine ces avantages en se basant sur les caractéristiques du JSF. Les métaheuristiques, en plus de leur adaptabilité aux différents problèmes combinatoires, ont l'avantage de fournir des solutions acceptables dans une durée minimale. L'approche expérimentale que nous avons appliquée, nous a permis de déduire plusieurs résultats concernant l'application de chacune des métaheuristiques.

Notre stratégie consiste à commencer à partir d'une solution initiale faisable créée par la SBH. Cette solution devient une phase initiale pour la recherche tabou. Ensuite, RT améliore la solution en intégrant la flexibilité de routage (machine alternative). Finalement, l'intégration d'AK vise à utiliser largement les informations qualitatives pendant la recherche globale.

Une étude de cas a été conçue et effectuée pour démontrer la faisabilité de l'approche proposée. Les taux d'amélioration sont considérablement plus grands lorsque la flexibilité des

machines est considérée. Cette flexibilité évite les conflits et une utilisation déséquilibrée de machines, assurant la stabilité et l'efficacité de la production dans un job shop flexible.

- **L'adaptation de l'approche hybride dans un job shop flexible avec des plans de processus multiples**

Basé sur le concept de l'approche hybride développée, un environnement intégré est considéré. La planification et l'ordonnancement sont réalisés par le biais des trois techniques. De plus de la flexibilité de routage introduite au niveau du job shop flexible, des plans de processus alternatifs presque optimaux sont sélectionnés pour être intégré à l'ordonnancement, ce qui améliore les performances de la production. Les tests indiquent que l'approche proposée est plus acceptable et prometteuse pour le problème IPPS, elle atteint des solutions optimales ou quasi-optimales.

Dans le cas du non intégration des plans de processus, il est possible de choisir au hasard un des plans de processus possibles. Les relations de précédence dans la planification du processus sélectionnée ont été considérées comme input dans l'approche proposé. Dans l'ampleur du problème dont la complexité est faible, l'existence de l'intégration montre sa performance.

Perspectives

Les travaux actuels peuvent être prolongés de plusieurs façons :

- Il sera très intéressant d'évaluer l'efficacité (en termes de qualité et de rapidité) de notre approche avec des problèmes réels qui dépassent 500 opérations avec des multiples plans. Etendre cette approche de résolution à des instances plus volumineuses est un objectif primordial, surtout du point de vue pratique.

- L'état d'un environnement de fabrication réel peut changer entre le moment où les procédures d'ordonnancement commencent et le moment où l'exécution des ordres de fabrication s'effectuent, ce qui le rend légèrement désuet à un horizon court. Il est intéressant de mettre en place une logique capable de gérer de telles situations en temps réel et d'effectuer de légères adaptations du planning, si nécessaire.

- Des comparaisons de cette approche avec différents algorithmes d'optimisation, comme les méthodes exactes ou d'autres métaheuristiques, devrait également être très intéressante.

- Un autre travail à venir est de proposer un cadre multi-objectif pour résoudre le problème IPPS afin de minimiser simultanément le makespan et le retard maximum. L'emploi d'une stratégie multi-objectifs semble être plus bénéfique.

- L'extension de la planification du processus du cadre de simulation permet de prendre en compte le retour d'expérience du niveau d'exécution lorsque les instances IPPS sont résolues face aux arrivées dynamiques d'ordres et aux pannes de machines. Cela conduit à des hybrides d'approches de la NLPP et de la CLPP.

- Il peut être intéressant, d'envisager un raffinement des méthodes pour améliorer davantage les résultats obtenus. Il est envisagé aussi, d'implanter des concepts d'apprentissage afin d'apporter une supplémentaire pour l'ajustement des nombreux paramètres rencontrés.

Références bibliographiques

A

Adekanmbi O., Green P. (2015) :Conceptual Comparison of Population Based Metaheuristics for Engineering Problems'. Scientific World Journal. 2015; 2015: 936106 .

Ahmad N., Haque A.F.M.A., & Hasin A.A. (2001):Current trend in computer aided process planning. In Proceedings of the 7th Annual Paper Meet and 2nd International Conference, volume 25, page 27, 2001.

Al-Hinai, N. & Elmekkawy, T.Y., (2011): An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem.Flexible Services and Manufacturing Journal23, 64–85.

Alting L. & H. Zhang (1989): Computer aided process planning: the state-of-the-art survey International Journal of Production Research, 27(4):553{585, 1989.

Amin-Naseri M. R. & Afshari J (2012): A hybrid genetic algorithm for integrated process planning and scheduling problem with precedence constraints. The International Journal of Advanced Manufacturing Technology, 59, 273–287.http://dx.doi.org/10.1007/s00170-011-3488-y

Araghi, M.E.T., Jolai, F.& Rabiee, M., (2013). Incorporating learning effect and deterioration for solving a SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach. International Journal of Computer Integrated Manufacturing27, 733–746

Artigues C, Billaut J-C, & Esswein C (2005): Maximization of solution flexibility for robust shop scheduling. European Journal of Operational Research, 165(2) :314–328, September 2005.

Asadzadeh, L. & Zamanifar, K., (2010): An agent-based parallel approach for job shop scheduling problem with genetic algorithms. Mathematical and Computer Modelling, 52(11 - 12), 1957-1965.

Ausaf M.F, Gao L, Li X, & Al Aqel G. (2015): A priority-based heuristic algorithm (PBHA) for optimizing integrated process planning and scheduling problem Cogent Engineering 2: 1070494.

B

Bagheri, A., Zandieh, M., Mahdavi,I. & Yazdani, M. (2010):. An artificial immune algorithm for the flexible job-shop scheduling problem. Future Generation Computer Systems 2010;26(4):533-541.

Bagchi Tapan P (1999) Pareto-optimal solutions for multi-objective production scheduling problems. In: Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001),

Baker K.R., (1974). Introduction to Sequencing and Scheduling. Wiley, New York.

Baker, R. P., & Maropoulos, P. G. (2000). An architecture for the vertical integration of tooling considerations from design to process planning. Robotics and Computer-Integrated Manufacturing, 16(2),

Baptise P (1998) : une étude théorique et expérimentale de la propagation des contraintes de ressources. Thèse de doctorat, université de technologies de compiègne,1998.

Barzegar, B., & Motameni, H., (2013): Solving flexible job-shop scheduling problem using hybrid algorithm based on gravitational search algorithm and particle swarm optimization.Journal of Advances in Computer Research Quarterly4, 69–81.

Barzegar, B., Motameni, H. & Bozorgi, H., (2012): Solving flexible job-shop scheduling problem using gravitational search algorithm and colored petri net. Journal of Applied Mathematics2012, 1–20.

Baykaso˘ glu, A., & Ozbakır, L., 2009. Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system. International Journal of Production Economics124, 369–381.

Benjaafar S, & Ramakrishnan R. (1996): Modelling, measurement and evaluation of sequencing 0exibility in manufacturing systems. International Journal of Production Research 1996;34:1195–220. 121-131.

BELLMAN R (1957) Metaheuristics in combinatorial optimization :Overview and conceptual comparison, ACM computing surveys Journal of Mathematics and Mechanics Vol. 6, No. 5 (1957), pp. 679-684Blum C & Andrea R (2003):, vol 35 n°3 september 2003 pp 268-308.

BELKADI K.(2006) « Les méta-heuristiques » Cours, Usto ; 2006

Blumi, C., Roli A., & Alba E. (2005) : an introduction to metaheuristic techniques. Parallel Metaheuristics: A New Class of Algorithms,47:1, 2005.

Botsalı A (2016): “Comparaison of simulated annealing and genetic algorithm approaches on integrated process routing and scheduling IJISAE, pp. 101-104, Dec. 2016.

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

Bozejko, W., Uchronski, M., & Wodecki, M., (2012): Flexible job shop problem – parallel tabu search algorithm for multi-GPU. Archives of Control Sciences22, 389–397.

Bozejko, W., Uchronski, M., & Wodecki, M., (2010). The new golf neighborhood for the flexible job shop problem. In Proceedings of the International Conference on Computational Science, pages 289–296.

Brandimarte P., & Calderini M., (1995): A hierarchical bicriterion approach to integrated process plan selection and job shop scheduling. International Journal of Production Research, 33(1):161{181, 1995

Brandimarte, P. (1993) :Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 1993;4 (1-4):157-183.

Braune M., & Zapfel G. (2016): Shifting bottleneck scheduling for total weighted tardiness minimization|a computational evaluation of subproblem and re-optimization heuristics. Computers & Operations Research, 66:130{140, 2016.

Braune R., Zäpfel G., & Affenzeller M. (2012): An exact approach for single machine subproblems in shifting bottleneck procedures for job shops with total weighted tardiness objective European Journal of Operational Research 218 (2012) 76–85

Brucker P. (2007) 'The job shop problem', Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: The-ory and Applications (MISTA 2007), Paris, France, pp. 15{22}

Brucker P, (1998). Scheduling Algorithms. Springer-Verlag, Berlin Heidelberg.

Brucker P., Jurisch B, & Sievers B (1994):. A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Ma-thematics, 49(1-3):107–127, 1994.

Brucker C., & Schlie R., (1991) Job-shop scheduling with multi-purpose machines. Computing, 45(4):369{375, January 1991.

Bulbul K. (2011): A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem. Computers & Operations Research, 38(6):967{983, 2011

C

Carlier J., & Pinson E., (1989). An algorithm for solving the job-shop problem. Management Science, 35(2) :164–176, 1989.

Carlier J., & Pinson E., (1994): Adjustment of heads and tails for the job-shop problem. European Journal of Operational Research, 78(2) :142–147, October 1994

Chambers, J., & Barnes, J., (1996): Classical and Flexible Job Shop Scheduling by Tabu Search. PhD thesis University of Texas, Austin, USA 1996.

Chandrasekharan R, & Ziegler H., (2004): Ant Colony Algorithms For permutation Flow Shop Scheduling to minimize makespan/total flow time of jobs.‖ European Journal of Operational Research 155 pp. 426-438.

Chaudhry I.A., & Usman M., (2017): INTEGRATED PROCESS PLANNING AND SCHEDULING USING GENETIC ALGORITHMS Tehnički vjesnik 24, 5(2017), 1401-1409 ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20151121212910.

Chen, M. S. Lee, P. S. Pulat, &S. A. Moses (2006): The shifting bottleneck procedure for job-shops with parallel machines. International Journal of Industrial and Systems Engineering, 1(1):244{262, 2006

Chryssolouris, G., Chan, S., & Cobb, W. (1984). Decision making on the factory floor: an integrated approach to process planning and scheduling. Robotics and Computer-Integrated Manufacturing, 1(3), 315- 319.

Chryssolouris. G, (2005) :manufacturing system: theory and practice", United States of America, Springer, Mechanical Engineering Series,2005, 623p.

Colorni, A., Dorigo, M., Maniezzo, V., & Trubian, M., (1994): Ant system for job-shop scheduling.Belgian Journal of Operations Research, Statistics and Computer Science34, 39–53.

Conway R.W., Maxwell W.L. & Miller L.W., (1967). Theory of Scheduling. Addison Wesley, Reading, Mass., USA.

D

Dai M., Tang D, Xu Y., & Li W., (2015) : Energy-aware integrated process planning and scheduling for job shops J Engineering Manufacture 2015, Vol. 229(S1) 13–26 DOI: 10.1177/0954405414553069

Dalfard,V.M., & Mohammadi, G., (2012):.Two metaheuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel machine and maintenance constraints. Computers & Mathematics with Applications64, 2111–2117.

Dauzere P., Roux W., & Lasserre JB. (1998): Multi-resource shop scheduling with resource flexibility. European Journal of Operational Research, 107(2):289{305, 1998.

Dell' Amico, M., & Trubian, M. (1993). Applying Tabu search to the job-shop scheduling problem. Annals of Operations Research, 41, 231–252.

Demirkol E., Mehta S., & Uzsoy R (1997): A computational study of shifting bottleneck procedures for shop scheduling problems. Journal of Heuristics, 3(2):111{137, 1997.

Demir, Y., & Kursat Isleyen, S., (2013): Evaluation of mathematical models for flexible job-shop scheduling problems. Applied Mathematical Modelling37, 977–988

Dréo, J., Pétrowski, A., Siarry, P., & Thaillard, E. (2003). Métaheuristiques pour l'optimisation difficile. Eyrolles, 2003.

E

Elshokouhi I., (2018) Integrated multi-objective process planning and flexible job shop scheduling considering precedence constraints. Production & Manufacturing research, 2018 VoL. 6, no. 1, 61–89 doi.org/10.1080/21693277.2017.1415173.

Ennigrou, M., Ghedira, K., (2008): New local diversification techniques for flexible job shop scheduling problem with a multi-agent approach.Autonomous Agents and Multi-Agent Systems17, 270–287.

F

France. P., Esquirol, L., (1999) : Lordonnancement. Economica, Paris, France.

Fang L., (2015) Contribution to Key Techologies of Integrated Process Planning and Scheduling in Job Shops Thèse de doctorat sous le label de L'Université Nantes Angers Le Mans

Farughi, H., Yegane, B.Y., & Fathian, M., (2013) : A new critical path method and a memetic algorithm for flexible job shop scheduling with overlapping operations.Simulation89, 264–277.

Fattahi, P., Jolai, F., & Arkat, J., (2009): Flexible job shop scheduling with overlapping in operations.Applied Mathematical Modelling33, 3076–3087.

Fattahi P., Mehrabad MS., & Jolai F., (2007): Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 18:331{342, 2007.

Fisher ML., Lageweg BL., Lenstra JK., & Rinnoy Kan AHG (1983): Surrogate duality relaxation for job shop scheduling. Discrete Appl. Math., 5, 65, 75

Fleury, G. (1995) : Applications des méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement", Automatique, Productique, Informatique Industrielle – APII, Vol. 29, No. 4-5, 1995, pp. 445-470.

French S., (1982). Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. Horwood, Chichester.

Frutos, M., Olivera, A.C., & Tohm e, F., (2010) : A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem.Annals of Operations Research181, 745–765.

G

Gao, K. Z., Suganthan, P. N., Chua, T. J., Chong, C. S., Cai, T. X., & Pan, Q. K.,(2015) : A two-stage artificial bee colony algorithm scheduling flexible job shop scheduling problem with new job insertion. Expert Systems with Application, 42, 7652-7663.

Gao, J., Sun, L., & Gen, M., (2008) : A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operationals Research 2008; 35(9):2892-2907.

Gao, J., Gen, M., Sun, L., & Zhao, X., (2007): A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems", Computers & Industrial Engineering 53 (2007) 149–162.

Garey, M.R., Johnson, D.S., & Sethi, R., (1976): The complexity of flow shop and job-shop scheduling. Mathematics of Operations Research", 1(2), 117–129.

Gen M., Gao J., & Lin L. (2009) : Multistage-Based Genetic Algorithm for Flexible Job shop Scheduling Problem', Intelligent and Evolutionary Systems, Volume 187 of the series Studies in Computational Intelligence pp 183-196.

Gholami, M., & Zandieh, M., (2009) : Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop.Journal of Intelligent Manufacturing20, 481–498.

Giard V. (1988) : Gestion de la Production, 2ème édition, Economica, Paris, 1988. Gourgand, M. N., & Norre S., (2003) : Problemes D'ordonnancement Dans Les Systèmes De Production DeType Flow-Shop Hybride En contexte Déterministe » ; J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes ; EDP Sciences, 2003.

Giffler, B., & Thompson, G.L., (1960) :Algorithms for solving production-scheduling problems.Operations Research8,487–503.

Glover F., (1977) : Tabu search-Part I. ORSA 7. Computing, 1989, 1(3): 190-296.

Goldberg, D.E., (1989). Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley.

Grabowski J., Nowicki E., & Zdrzalka S., (1986) :A block approach for single machine scheduling with release dates and due dates, European Journal of Operational Research 26 (1986)278±285

Grefenstette J.J., (1987) : Incorporating problem specific knowledge into genetic algorithms, L.Davis (Ed.) Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, p.42-60, 1987.

Gomes, M.C., Barbosa-Povoa, A.P., & Novais, A.Q., (2005) : Optimal scheduling for flexible job shop operation.International Journal of Production Research43, 2323–2353.

Günther R., & Raidl K., (2006), A Unified View on Hybrid Metaheuristics, Institute of Computer Graphics and Algorithms Vienna University of Technology, Vienna, Austria, 2006.

Guo, Y. W., Li, W. D., Mileham, A. R., & Owen, G. W. (2009) : Applications of particle swarm optimisation in integrated process planning and scheduling. Robotics and Computer-Integrated Manufacturing, 25(2), 280-288.

H

Haddadzade M., & Zarandi M, (2014): Integration of process planning and job shop scheduling with stochastic processing time. The International Journal of Advanced Manufacturing Technology, 71(1-4), 241-252.

Haddadzade M, Razfar MR, & Farahnakian M (2009) Integrating pro-cess planning and scheduling for prismatic parts regard to due date.World Acad Sci Eng Technol 51:64–67

Ho YC., & Moodie CL. (1996): Solving cellformation problems in a manufacturing environment with 0exible processing and routing capabilities. International Journal of Production Research 1996; 34:2901–23.

Holland JH., (1975) : Adaptation in Natural and artificial systems, MIT Press, Cambridge , Mass.

Huang S. H., Zhang, H.-C. & Smith M. L., (1995). A progressive approach for the integration of process planning and scheduling. IIE Transactions, 27(4):456{464, 1995.

Hurink, J., Jurisch, B., Thole, M.. Tabu search forthe job-shop scheduling problem with multi-purpose machines. Operational Research Spektr 1994;15 (4):205-215.

Hutchinson A.G., & Pughoeft K.A., (1994): Flexible process plans: their value in flexible automation systems. International Journal of Production Research, 32(3):707{719,1994

J

Jain A, Jain PK., & Singh IP (2006): An integrated scheme for process planning and scheduling in FMS. Int J AdvManuf Tech 30(11-12): 1111–1118. doi:10.1007/s00170-005-0142-12.

Jain N.K., & Jain, V.K., (2001) : Computer Aided Process Planning in Agile Manufacturing Environment. In Chapter 27 of Agile Manufacturing: 21st Century Competitive Strategy, (Editor: A. Gunasekaran), Elsevier Science Publications, 515-534

Jain A.S., & Meeran, S., (1999): A State-of-the-Art Review of Job-Shop Scheduling Techniques. European Journal of Operations Research, 113,390-434 Jacques Carlier. The one-machine sequencing problem. European Journal of Operational Research, 11(1):42–47.

Javel L., (2004), Organisation et Gestion de la Production, édition, D U NOD, Paris 2004.

Jia S., & Hu, Z.-H., (2014): Path-relinking tabu search for themulti-objective flexible job shop scheduling problem.Computers & Operations Research47, 11–26.

Jin L., Zhang B., & Shao Z., (2015): An effective hybrid honey bee mating optimization algorithm for integrated process planning and scheduling problems. The International Journal of Advanced Manufacturing Technology, 1-12.

Jourdan, L., Basseur M., & Talbi E.-G : (1993). Hybridizing exact methods and meta-heuristics: A taxonomy. European Journal of Operational Research, 199(3):620–629, 2009.

K

Kacem, I., Hammadi, S., & Borne, P., (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics-Part C, 32(1), 1–13.

Karthikeyan S., Asokan P., & Nickolas S (2014) : A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints Int J Adv Manuf Technol (2014) 72:1567–1579 DOI 10.1007/s00170-014-5753-3

Khoshnevis B., & Chen Q. M (1991) : Integration of process planning and scheduling functions.Journal of Intelligent Manufacturing, 2(3):165{175, 1991.

Kim YK, Park K & Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30(8):1151–1171. doi: 10.1016/s0305-0548(02) 00063-1

Kim KH., & Egbelu PJ (1999) Scheduling in a production environment with multiple process plans per job. Int J Prod Res 37:2725–2753. doi:10.1080/002075499190491],ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)

Kirkpatrick S., Gelatt C. D., & Vecchi M. P., (1983): Optimization by simulated annealing.Science, 220:671{680, 1983.

Kis, D., Kiritsis, P., Xirouchakis C., & Neuendorf K-P., (2000): A Petri net model for integrated process and job shop production planning. Journal of Intelligent Manufacturing, 11(2):191{207, 2000.

Kumar M., & Rajotia S., (2006) Integration of process planning and scheduling in a job shop environment. Int J Adv Manuf Tech 28(1-2):109–116. doi:10.1007/s00170-004-2317-y

Kutanoglu E., & Sabuncuoglu I., (1999) : An analysis of heuristics in a dynamic job shop with weighted tardiness objectives. International Journal of Production Research, 37(1), 165-187

Kundakcı, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem.Computers & Industrial Engineering, 96, 31–51.

L

Lasi H., Fettke P., Kemper, H-G., Feld T., & Ho M., (2014): Industry 4.0. Business & Information Systems Engineering, 6(4):239{242, 2014

Lawler E. L., (1973): A pseudo polynomial" algorithm for sequencing jobs to minimize total tardiness. Annals of Discrete Mathematics, 1:331{342, 1973

Lee, H., & Kim, S. S. (2001): Integration of process planning and scheduling using simulation based genetic algorithms. The International Journalof Advanced Manufacturing Technology 18(8), 586-590.

Lei, D., Guo, X., (2014): Variable neighbourhood search for dual-resource constrained flexible job shop scheduling.International Journal of Production Research52, 2519–2529.

Lei DM (2008): Multi-objective production scheduling: a survey. Int J Adv Manuf Technol 43(9–10):926–938

Leung CW., Wong TN., Mak KL., & Fung RYK., (2010) :Integrated process planning and scheduling by an Agent based ant colony optimization. Computers & Industrial Engineering, 59(1), 166-180

Lian KL, Zhang CY, Shao XY., & Gao L (2012): Optimization of process planning with various flexibilities using an imperialist competitive algorithm. Int J AdvManuf Tech 59(5-8):815–828. doi:10.1007/s00170-011-3527-8

Li X, Gao L., & Wen X (2013): Application of an efficient modified particle swarm optimization algorithm for process planning. Int JAdv Manuf Technol 67(5–8):1355–1369.

Li XY, Gao L., & Li WD (2012): Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. Expert Syst Appl 39(1):288–297. doi:10.1016/j.eswa.2011.07.019

Li, J.-Q., Pan, Q.-K., Suganthan, P. N., & Chua, T. J., (2011): A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible jobshop scheduling problem. International Journal of Advanced Manufacturing Technology 2011; 52(5-8):683-697.

Li J.Q., Pan Q.K., Suganthan P.N., & Chua T.J. (2010a): A hybrid Tabu Search algorithm with an efficient neighborhood structure for the flexible job shop-scheduling problem', International Journal of Advanced Manufacturing Technology 52(5–8):683–697 19

Li X, Gao L, Shao X, Zhang C., & Wang C (2010b): Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling. Comput Oper Res 37:656–667.doi:10.1016/j.cor.2009.06.008].

Li W. D., & McMahon C. A., (2007): A simulated annealing-based optimization approach for integrated process planning and scheduling. International Journal of Computer Integrated Manufacturing, 20(1):80{95, 2007

Lihong Q., & Shengping L., (2013): An improved genetic algorithm for integrated process planning and scheduling. International Journal of Advanced Manufacturing Technology, 58(5):727{740, 2013

Lihong Q., & Shengping L., (2012): An improved genetic algorithm for integrated process planning and scheduling. International Journal of Advanced Manufacturing Technology, 58(5):727{740, 2012

Lim, M., & Zhang, Z. (2000). APPSS–An agent-based dynamic process planning and scheduling system. In Proc. Volume from the IFAC Workshop(pp. 51-56).

Lin YJ., & Solberg JJ. (1991): Effectiveness of flexible routing control. The International Journal of Flexible Manufacturing Systems 1991;3:189–211.

Liouane, N., Saad, I., Hammadi, S., & Borne, P., (2007):. Ant systems & local search optimization for flexible job shop scheduling production.International Journal of Computers Communications & Control2, 174–184

Lopez F., & Roubellat, A (2001): Ordonnancement de la Production. Hermes Science,France.

Lopez P., & Esquirol P0., (1999) : L'ordonnancement , Economica, Paris 1999.
Lv S., & Qiao L., (2014a): Research on the key technology of integrated process planning and scheduling. PHD thesis, Beihang University.

Lv, S.,& Qiao, L. (2014b). Process planning and scheduling integration with optimal rescheduling strategies. International Journal of Computer Integrated Manufacturing, 27(7), 638-655.

M

Mahdavi, I., Shirazi, B., & Solimanpur,M., (2010) .Development of a simulation-based decision support system for controlling stochastic flexible job shop manufacturing systems.Simulation Modelling Practice and Theory18, 768–786.

Marzouki B., Belkahla Driss O., & Ghedira k., (2017): Multi Agent model based on Chemical Reaction Optimization with Greedy algorithm for Flexible Job shop Scheduling Problem Procedia Computer Science 112 (2017)81–90

Marzouki, B., & Belkahla Driss, O., (2015): Multi Agent model based on Chemical Reaction Optimization for Flexible Job Shop Problem. International Conference on Computational Collective Intelligence 2015; 9329 (1):29-38

Mastrolilli, M., & Gambardella, L.M., (2000) :. Effective neighbourhood functions for the flexible job shop problem.Journal of Scheduling3, 3–20

Mati Y., Dauzere-Peres M., & Lahlou C., (2011) :. A general approach for optimizing regular criteria in the job-shop scheduling problem. European Journal of Operational Research, 212(1):33{42, 2011.

Matsuo, H., Suh, C., & Sullivan, R. (1988) :.A controlled search simulated annealing method for the general jobshop scheduling problem. Working paper 03-44-88, Graduate School of Busi-ness, University of Texas.

Mehrabad S., & Fattahi P., (2007): Flexible job shop scheduling with v1search algorithms.International Journal of Advanced Manufacturing Technology32, 563–570.

Mitchell M. (2002): An introduction to Genetic Algorithms. PHI, New Delhi.

Minzu V., & Beldiman L., (2003): A parallel hybrid metaheuristic for the single machine scheduling problem," Proceedings of the IEEE International Symposium onAssembly and Task Planning, 2003., Besancon, France, 2003, pp. 134-139. doi: 10.1109/ ISATP.2003. Mladenovi N., & Hansen P., (1997): Variable neighborhood search.Computers & Operations Research24, 1097–1100.

Mohammadi G, Karampourhaghghi A, & Samaei F (2012): A multi-objective optimisation model to integrating flexible process plan-ning and scheduling based on hybrid multi-objective simulated an-nealing. Int J Prod Res 50(18):5063–5076. doi:10.1080/00207543.2011.631602

Monch, L., & Zimmermann, J. (2011): A computational study of a shifting bottleneck heuristic for multi-product complex job shops. Production Planning & Control, 22(1), 25–40.

Monch L& Zimmermann J., (2007):.Simulation-based assessment of machine criticality measures for a shifting bottleneck scheduling approach in complex manufacturing systems. Computers in Industry, 58(7):644{655, 2007.

Morad N, & Zalzala A (1999): Genetic algorithms in integrated process planning and scheduling. J Intell Manuf 10:169–179. doi:10.1023/A:1008976720878.

Motaghedi-Larijani, A., Sabri-Laghaie, K., & Heydari, M., (2010) :. Solving flexible job shop scheduling with multiobjective approach.International Journal of Industrial Engineering & Production Research21, 197–209.

1217200.

M. François (2007) : Contribution à l'ordonnancement des activités de maintenance sous contrainte de compétence: une approche dynamique, proactive et multi-critère.. Automatique / Robotique. Université de Franche-Comté, 2007. Français. fftel-00212750f.

Mühlenbein, M. Gorges-Schleuter & O. Krämer (1988):Evolution algorithms in combinatorial optimization. Parallel Computing 7 : 65-85, 1988.

N

Nasr, N., & Elsayed, E. A. (1990). Job shop scheduling with alternative machines. The international journal of production research, 28(9), 1595-1609.

Nejad, H. T. N., Sugimura, N., Iwamura, K., & Tanimizu, Y. (2010) :. Multi agent architecture for dynamic incremental process planning in the flexible manufacturing system. Journal of Intelligent Manufacturing, 21(4), 487-499.

Nouri H.E., Belkahla, Driss, O., & Ghedira, K., (2018): "Solving the flexible job shop problem by hybrid metaheuristics- multiagent model". J Ind Eng Int (2018) 14:1–14 doi.org/10.1007/s40092-017-0204-z.

Nouri, H.E., Belkahla Driss, O., & Ghédira, K., (2016): "Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model", Computers & Industrial Engineering 102, 488-501.

Nouri, H.E., Belkahla Driss, O., & Ghedira, K., (2015): Hybrid Metaheuristics within a Holonic Multiagent Model for the Flexible Job Shop Problem. Procedia Computer Science 2015; (60):83-92

Nowicki E., & Smutnicki C., (2005): An advanced tabu search algorithm for the job shop problem, Journal of Scheduling 8 (2) (2005) 145–159.

Nwana H.S., & Ndumu D.T. (1997): An introduction to agent technology. In: Nwana H.S., Azarmi N. (eds) Software Agents and Soft Computing Towards Enhancing Machine Intelligence. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1198. Springer, Berlin, Heidelberg

O

Ozguven C., & Ozbakır, L., & Yavuz, Y., (2010) : Mathematical models for job-shop scheduling problems with routing and process plan flexibility.Applied Mathematical Modelling34, 1539–1548.

P

Phanden, R. K., Jain, A., & Verma, R. (2013):. An approach for integration of process planning and scheduling. International Journal of Computer Integrated Manufacturing, 26(4), 284-302.

Pacino D., & Van Hentenryck P (2011): Large Neighborhood Search and Adaptive Randomized Decompositions for Flexible Jobshop Scheduling Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence

Palacios, J. J., Gonzalez, M. A., Vela, C. R., Rodriguez, I. G., & Puente, J., (2015) :. Genetic tabu search for the fuzzy flexible job shop problem. Computers & Operations Research, 54, 74-89.

Palmer, J., (1996): A Simulated Annealing Approach to Integrated Production Scheduling. Journal of Intelligent Manufacturing, 7, 163-176.

Park B. J., &. Choi H. R., (2006): A genetic algorithm for integration of process planning and scheduling in a job shop. In AI 2006: Advances in Articial Intelligence, pages 647{657. Springer, 2006.

Petrovi M., Vukovi N., (2016): Integration of process planning and scheduling using chaotic particle swarm optimization algorithm, Expert Systems with Applications (2016), doi:10.1016/j.eswa.2016.08.019

Pezzella, F., Morganti, G., & Ciaschetti, G.,(2008) :. A genetic algorithm for the flexible job-shop scheduling problem.Computers & Operations Research35, 3202–3212.

Phanden, R. K., Jain, A., & Verma, R. (2011): Integration of process planning and scheduling: a state-of-the-art review.International Journal of ComputerIntegrated Manufacturing, 24(6), 517-534.).

Pinedo, ML., (2012): Scheduling : Theory, Algorithms, and Systems. Springer-Verlag New York, 4rd édition, 2012. 8

Pinedo, M. L. (2008). Scheduling: Theory,Algorithms, and Systems.

Q

Qiao & Lv (2012): An improved genetic algorithm for integrated process planning and scheduling, the international journal of advanced manufacturing technology, 58(5-8), 727-740.

Qiu X, & Lau HK (2014) : An AIS-based hybrid algorithm for static problème d'emploi du temps dans l'atelier. J Intell Manuf 25(3):489-503. doi /10.1007/s10845-012-0701-236.

R

Rinnooy K, (1976): Machine Scheduling Problems: Classification, Complexity and Computations. Martinus Nijhoff, The Haque.

Ronald L. Graham, Eugene L. Lawler, & Lenstra A. G. H. & Rinnooy Kan (1979): Optimization and approximation in deter-ministic sequencing and scheduling : a survey. Annals of Discrete Mathematics, 5 :287–326,1979

Roshanaei, V., Azab, A., & Elmaraghy, H., (2013): Mathematical modelling and a meta-heuristic for flexible job shop scheduling.International Journal of Production Research51, 6247–6274.

Rossi, A., & Dini, G., (2007): Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method.Robotics and Computer-Integrated Manufacturing23, 503–516.

S

Sadrzadeh, A., (2013):. Development of both theAIS and PSO for solving the flexible job shop scheduling problem.Arabian Journal for Science and Engineering38, 3593–3604.

Saygin C, & Kilic SE (1999): Integrating flexible process plans with scheduling in flexible manufacturing systems. Int J Adv Manuf Tech 15(4):268–280. doi:10.1007/s001700050066

Schmidit K., (2001): Using tabu search to solve job shopscheduling problem with sequence dependent step times, communication 2001

Scrich, C.R., Armentano, V.A., & Laguna, M., (2004):. Tardiness minimization in a flexible job shop: a tabu search approach. Journal of Intelligent Manufacturing15, 103–115

Seker, Erol, & Botsali, (2013) A neuro-fuzzy model for a new hybrid integrated Process Planning and Scheduling system. Expert Systems withApplications, 40(13), 5341-5351.

SERBENCU A, & MINZU V., (2007): An ant colony system based metaheuristic for solving single machine scheduling problem" ; the annals of "dunarea de jos" university of galati fascicle III, 2007 P19-24

Shao, X., Li, X., Gao, L., & Zhang, C. (2009): Integration of process planning and scheduling—A modified genetic algorithm-based approach. Computers & Operations Research, 36, 2082–2096.http://dx.doi.org/10.1016/j.cor.2008.07.006

Shen, W., Wang, L., & Hao, Q. (2006): Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey.Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 36(4), 563-577.

Sobayko O., (2018): Integrated Process Planning and Scheduling in Flexible Job Shops Department of Mathematics and Computer Science, University of Hagen, Hagen, Germany I thesis

Sobayko O., & monch L., (2016): Integrated process planning and scheduling for large-scale flexible job shops using Metaheuristics International Journal of Production Research, DOI: 10.1080/00207543.2016.1182227

Srinivas P. S., Ramachandra Raju V., & Rao C.S.P., (2012) : Optimization of Process Planning and Scheduling using ACO and PSO Algorithms International Journal of Emerging Technology and Advanced Engineering

Sundaram, R.M. & Fu, S.S., (1988): Process planning and scheduling. Computer and Industrial Engineering, 15, 296–307.

T

Talbi E.L., (2004): Sélection et réglage de paramètres pour l'optimisation de logiciels d'ordonnancement industriel » ; Institut National Polytechnique de Toulouse Ecole Doctorale Systèmes ; Spécialité : Informatique Industrielle Soutenu le 12 novembre 2004

Tan, Y., Hildebrandt, T., & Scholz-Reiter, B. (2016): Configuration and the advantages of the shifting bottleneck procedure for optimizing the job shop total weighted tardiness scheduling problem. Journal of Scheduling, 19(4), 429–452. http://doi.org/10.1007/s10951-015-0441-1.

Tan, W., & Khoshnevis, B. (2000): Integration of process planning and scheduling – a review'. Journal of Intelligent Manufacturing 11(1):51–63 (2000)

Tan W. (1998): Integration of process planning and scheduling. A mathematical programming approach'. Ph.D. Dissertation, University of Southern California.

Tan W., & Khoshnevis B., (1997): Integration of process planning and scheduling|a review.Journal of Intelligent Manufacturing, 11(1):51{63, 1997

Tanev, I.T., Uozumi, T., & Morotome,Y., (2004): Hybridevolutionaryalgorithm-based real-worldflexible job shop scheduling problem: application service provider approach.Applied Soft Computing5, 87–100.

Tanaev V.S., V.S. Gordon, & Y.M. Shafransky, (1994): Scheduling theory. Single-stage systems. Kluwer Academic Publishers. Dordrecht / Boston / London.

Tay, J.C., Ho, & N.B., (2008): Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems.Computers & Industrial Engineering54, 453–473

Tonshoff, H.K., Beckendorff, U. & Andres, N., (1989). FLEXPLAN: A concept for intelligent process planning and scheduling. CIRP International Workshop, September 1989, Hannover, Germany, 21-22.

Toshev A (2019): Particle Swarm Optimization and Tabu Search Hybrid Algorithm for Flexible Job Shop Scheduling Problem – Analysis of Test Results CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume 19, No 4 Sofia • 2019 Print ISSN: 1311-9702; Online ISSN: 1314-4081 DOI: 10.2478/cait-2019-0034

U

Usher, J. M. (2003): Evaluating the impact of alternative plans on manufacturing performance. Computers & Industrial Engineering, 45(4), 585-596.

Usher JM, & Fernandes KJ. (1996): Dynamic process planning-the static phase. Journal of Materials Processing Technology 1996;61:53–8.

V

Van Laarhoven, P. J. M., & Aarts, E. H. L. (1987).Simulated annealing theory and applications. Dordrecht: Reidel

Vilcot, G., & Billaut, J.C., (2011): A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem", International Journalof Production Research, 49:23, 6963-6980, DOI: 10.1080/00207543.2010.526016.

W

Wang JF., Fan XL., Zhang CW., & Wan ST. (2014): 'ALi WD, McMahon C A. (2007) 'A simulated annealing-based optimization approach forintegrated process planning and scheduling'. International Journal of Computer Integrated Manufacturing 20(1)

Wan S. Y., Wong T. N., Zhang S., & Zhang L. (2013).Integrated process planning and scheduling with setup time consideration by ant colony optimization. Proceedings of the 41st International Conference on Computers & Industrial Engineering, 998-1003 .

Wang, S., & Yu, J., (2010). An effective heuristic for flexible job-shop scheduling problem with maintenance activities. Computers & Industrial Engineering59, 436–447.

Wang, X., Gao, L., Zhang, C., & Shao X., (2010):" A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem", Int J Adv Manuf Technol (2010) 51: 757. doi:10.1007/s00170-010-2642-2.

Weintraub, A., Cormier, D., & Hodgson, T. (1999): Scheduling with alternatives: a link between process planning and scheduling. IIE Transactions 31, 1093–1102 (1999) doi:10.1023/A:1007683710427

Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y.K. (2006). An agent-based negotiation approach to integrate process planning and scheduling.International journal of production research, 44(7), 1331-1351.

Wu, S. H., Fuh, J. Y. H., & Nee, A. Y. C. (2002). Concurrent process planning and scheduling in distributed virtual manufacturing. IIE Transactions, 34(1), 77-89.).

X

Xia,W., & Wu,Z., (2005): An effective hybrid optimization approach formulti-objective flexible job-shop scheduling problems. Computers & Industrial Engineering48, 409–425.

Xia H, Xinyu Li, & Liang Gao (2016): A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling Computers & Industrial Engineering 102 (2016) 99–112

Xing, L.-N., Chen, Y.-W., & Yang, K.-W., (2011): Multi-population interactive coevolutionary algorithm for flexible job shop scheduling problems.Computational Optimization and Applications48, 139–155.

Xing, L.N, Chen, Y.W, Wang, P., Zhao, Q.S., Xiong, J., (2010): A knowledge-based ant colony optimization for flexible job shop scheduling problems". Appl Soft Comput 10(3):888–89.

X-y W, X-y L, Gao L, & H-y S (2014): Honey bees matingoptimiza-tion algorithm for process planning problem. J Intell Manuf 25(3): 459–472. doi:10.1007/s10845-012-0696-8

Y

Yang, Y. N., Parsaei, H. R., & Leep, H. R. (2001) : A prototype of a feature-based multiple-alternative process planning system with scheduling verification. Computers & industrial engineering, 39(1), 109-124.

Yazdani, M., Gholami, M., Zandieh, M., & Mousakhani, M., (2009): A simulated annealing algorithm for flexible job shop scheduling problem.Journal of Applied Sciences9, 662–670

Yu, M., Zhang, Y., Chen, K., & Zhang, D. (2015): Integration of process planning and scheduling using a hybrid GA/PSO algorithm.International Journal of Advanced Manufacturing Technology, 78(1–4), 583–592.

Yuan, Y., & Xu, H., (2013):. Flexible job shop scheduling using hybrid differential evolution algorithms.Computers & Industrial Engineering65, 246–260.

Yulianty, A., & Ma'ruf, A., (2013): Predictive approach on flexible job shop scheduling problem considering controllable processing times. International Journal of Innovation, Management and Technology4, 565–569.

Yusof, R., Khalid, M., Hui, G.T, Yusof, S.M. & Othman, M.F., (2011): Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm. Applied Soft Computing, 11(8), 5782-5792

Z

Zhang, L., Tang, Q., Wu, Z., Wang, F. (2017): "Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops" Energy. 138. 10.1016/j.energy.2017.07.005.

Zhang R, Ong S. K., &. Nee. A. Y. C., (2015): A simulation-based genetic algorithm approach for remanufacturing process planning and scheduling. Applied Soft Computing, 37: 521{532, 2015

Zhang R., & Wong L, (2014): Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. Journal of Intelligent Manufacturing, 1-17

Zhang, C., Gu, P., & Jiang, P., (2014): "Low-carbon scheduling and estimating for a flexible job shop based on carbon footprint and carbon efficiency of multi-job processing". Journal of Engineering Manufacture, 39(32):1–15.

Zhang, Q., Manier, H., & Manier, M.A., (2012) : A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times.Computers & Operations Research39,1713–1723.

Zhang, G.H., Shao, X.Y., Li, P.G., & Gao, L., (2009): "An effective hybrid swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Computers and Industrial Engineering" 56(4), 1309–1318 (2009).

Zhang H.-C. &.Merchant M. E (1993):. IPPM{a prototype to integrate process planning and job shop scheduling functions. CIRP Annals-Manufacturing Technology, 42(1): 513{518, 1993.

Zhao, F. (2006). Integration of process planning and production scheduling based on a hybrid PSO and SA algorithm. In IEEE International Conference on Mechanical and Automation(pp. 2290–2295). Luoyang.

Zhao, F. (2004): A genetic algorithm based approach for integration of process planning and production scheduling. In International Conference on Intelligent Mechatronics and Automation (pp. 483–488). Chengdu

Zhao, L. & Kops, L. (1987): An integrated CAPP and scheduling system. NAMRC Proceedings of Society of Manufacturing Engineers, 552–557.

Zhou H, Feng Y, Han L (2001) The hybrid heuristic genetic algorithm for job shop scheduling. Comput Ind Eng 40:191–200. doi:10.1016/S0360-8352(01)00017-1

Ziaee, M., (2014): "A heuristic algorithm for solving flexible job shop scheduling problem" International Journal of Advanced Manufacturing Technology, 71, 519-528.

Zimmermann A (2007): Simulation-based assessment of machine criticality measures for a shifting bottleneck scheduling approach in complex manufacturing systems. Computers in Industry, 58(7):644{655, 2007.