République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida 1

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

ABBAD Djamila

En vue d'obtenir le diplôme de Master

Filière : MI

Spécialité : Informatique

Option : Génie des systèmes informatiques

Sujet :

« Hybridation Méta-heuristique-Simulation évènement Discret pour le pilotage des systèmes de production automatisés »

Organisme d'accueil : Centre de Développement des Technologies Avancées

Soutenue le : 23/05/2015

Devant le Jury composé de :

Mme Toubaline	Président
Mme Guessoum	Examinatrice
Mr Kamèche	Examinateur

Promotrice : Mme Aroussi Sana.

Encadreur : Mr. Gaham Mehdi.

DEDICACES

A mes très chers parents.

A mon frère et mes sœurs.

A mes amies.

REMERCIEMENTS

Ce n'est pas par tradition que cette page figure au préambule de ce mémoire, mais c'est plutôt un devoir moral et une reconnaissance sincère qui me poussent à la faire. Je serais en effet ingrate si je n'exprime pas ma reconnaissance et ma gratitude à tous ceux qui ont, de près ou de loin, facilité ma tâche et m'ont permis de mener à bien ce travail.

Tout d'abord, je tiens à remercier le bon dieu tout puissant de m'avoir donné la volante, le courage et la force d'entreprendre ce modeste travail.

C'est un agréable devoir pour moi d'exprimer ma très vive reconnaissance à mon encadreur Monsieur Gaham Mehdi pour m'avoir proposé ce sujet et pour m'avoir guidé durant toute l'élaboration de ce mémoire avec la compétence qui les caractérise.

Je tiens à remercier chaleureusement ma promotrice madame Aroussi Sana pour ses efforts fournis pour m'aider, ses corrections, ses conseils prodigués, sa patience, sa compréhension et sa gentillesse durant tout le long de mon mémoire.

Je tiens tout particulièrement à exprimer toute ma gratitude et mes vifs remerciements à Mme Rashedi Esmat. Qu'elle me permet de l'exprimer l'assurance de ma gratitude et de mon profond respect.

Je remercie tous les membres de laboratoire Robotique Productique de CDTA pour leur accueil et leurs conseils qui m'ont été très précieux et indispensables.

Mes remerciements vont également à tous les membres de jury pour avoir accepté d'évaluer ce mémoire.

Je ne saurais oublier de remercier vivement mes chers parents, qu'ils voient dans l'aboutissement de mes études et de ce mémoire, la réussite de leur mission.

ملخص

مشاكل الجدولة هي لنظم الإنتاج واحدة من أكبر الصعوبات في نظام القيادة. من بين هذه المشاكل، هناك مشكل المحل الوظيفي المرن وهو مشكل واقعي ومقيد جدا.

نقترح في هذا العمل وضع نهج المحاكاة-التحسين على أساس قواعد الجدولة من أجل حل المشكلة المحل الوظيفي المرن. وسيكون الهدف الرئيسي من هذا العمل أن يكون تنفيذ خوارزمية ما وراء الكشف جديدة تسمى خوارزمية البحث بالجاذبية حيث سيتم تنفيذ مهمة التقييم باستخدام برنامج المحاكاة منفصلة الحدث FLEXSIM.

النتائج التي تم الحصول عليها عن طريق تنفيذ هذا النهج هي نتائج مرضية، ولكن كأسلوب تحسين جديد، فإنه يستحسن القيام بعدة محاولات للحصول على نتائج مثالية.

كلمات مفتاحية: جدولة، محل الوظيفي المرن ، خوارزمية ما وراء الكشف ، خوارزمية البحث بالجاذبية، برنامج المحاكاة منفصلة الحدث FLEXSIM .

Abstract

The scheduling problems are for production systems One of the major difficulties of their steering system. Among these problems, there is the Flexible Job Shop which is a realistic and very constrained problem.

We suggest in this work to develop a simulation-optimization approach based on scheduling rules for solving the Flexible Job Shop problem. The main objective of this work is the implementation of a very recent metaheuristic called gravitational search algorithm where the evaluation function will be performed using the simulation software discrete events **FLEXSIM**.

The obtained results by the implementation of this approach are satisfactory, but as the optimization method is new, it takes several attempts to perfections.

Keywords: Scheduling, Job Shop Flexible, metaheuristic, gravitational search algorithm, simulation software discrete events **FLEXSIM**.

Résumé

Les problèmes d'ordonnancement constituent pour les systèmes de production une des difficultés majeures de leur système de pilotage. Parmis ces problèmes, on trouve le Job Shop Flexible qui est un problème réel et très contraints.

Nous proposons dans ce travail le développement d'une approche par simulation-optimisation basée sur les règles d'ordonnancement pour la résolution du problème Job Shop Flexible. Le principal objectif du travail consistera en l'implémentation d'une très recente métaheuristique

nommée Algorithme de recherche gravitationnelle où la fonction d'évaluation sera réalisée en utilisant le logiciel de simulation à évènements discrets **FLEXSIM**.

Les résultats obtenus par la mise en oeuvre de cette approche sont satisfaisants, mais comme la méthode d'optimisation est nouvelle, elle demande plusieurs tests de perfections.

Mots Clés : Ordonnancement, Job Shop Flexible, métaheuristique, Algorithme de recherche gravitationnelle, logiciel de simulation à évènements discrets **FLEXSIM**.

LISTE DES FIGURES

LISTE DES TABLEAUX

SOMMAIRE

INTRODUCTION GÉNÉRALE

L'excellence par la qualité est la stratégie adoptée par toutes les entreprises performantes qui connaissent le succès commercial, économique et technologique. La clé de ce succès repose sur le même principe, qu'il s'agisse d'une entreprise manufacturière ou de services et consiste à dépasser les attentes des clients en offrant des produits et/ou des services de qualité et à bas prix, tout en respectant les délais.

Maintenir ce succès devient de plus en plus une tâche difficile, vu que les entreprises manufacturières ont connu de grands bouleversements. En effet, l'objectif majeur de toute entreprise actuelle n'est plus seulement la productivité mais aussi et surtout la compétitivité. Cette dernière passe forcément par la diversification des produits, un outil de production de plus en plus flexible et un pilotage fiable des systèmes de production.

L'amélioration de pilotage des systèmes de production est alors, à la base de toute tentative de changement. Cette fonction vise en effet à organiser le fonctionnement du système de production, et a mieux gérer ses différentes opérations.

Pour ce faire, il faut opter pour un bon ordonnancement d'ateliers. Ce dernier se défini par un ensemble de ressources, qu'on doit affecter à un ensemble de tâches de façon à satisfaire un ou plusieurs objectifs, et en prenant en compte un certain nombre de contraintes temporelles (délais, contraintes d'enchainement) et de contraintes portant sur la disponibilité des ressources requises.

Ainsi, les problèmes d'ordonnancement d'ateliers constituent pour les entreprises une des difficultés importantes de leurs systèmes de gestion et de pilotage de la production.

De nombreuses méthodes de résolution de ce type de problèmes ont été proposées dans la littérature. Puisque la performance de méthodes de résolution de problèmes est mesurée en termes de deux notions souvent antagonistes: la rapidité de la recherche et la qualité des solutions fournies, les méthodes de résolution de problèmes ont été classées en deux catégories: les méthodes exactes et les méthodes approchées. Les méthodes exactes sont connues par le fait qu'elles garantissent l'optimalité de la solution mais elles demandent des coûts de recherche (temps de calcul et espace mémoire) souvent prohibitifs qui augmentent avec la taille de l'instance du problème traité. Tandis que, les méthodes approchées sont connues par le fait qu'elles ne garantissent pas l'optimalité de la solution (elles fournissent

des solutions de bonne qualité et des fois optimales) mais elles nécessitent des coûts de recherche raisonnables. Par conséquent, les méthodes exactes sont plus pratiques pour la résolution de problèmes faciles. En revanche, les méthodes approchées sont plus pratiques dans le cas où on cherche une solution de bonne qualité dans un bref délai.

Les méthodes approchées sont classées en deux catégories: les heuristiques et les métaheuristiques. Une heuristique est une méthode approchée spécifique à un problème donné. Par ailleurs, les métaheuristiques sont des méthodes approchées polyvalentes, elles peuvent être appliquées sur de nombreux problèmes. De leur part, les métaheuristiques peuvent être classées en deux catégories: les métaheuristiques à base de solution unique et les métaheuristiques à base de population de solutions. Une métaheuristique à base de solution unique manipule une seule solution durant la procédure de recherche. Elle lance la recherche avec une seule solution et essaye d'améliorer sa qualité au cours des itérations. Cependant, une métaheuristique à base de population de solutions manipule un ensemble de solutions parallèlement. Elle lance la recherche avec une population de solutions et essaye d'améliorer leurs qualités au cours des itérations dans le but de fournir la ou les meilleures solutions trouvées. La majorité des métaheuristiques sont inspirées des systèmes naturels.

La plus part des problèmes réels d'ordonnancement sont NP-difficiles et la taille de leurs instances est souvent très grande, ce qui implique l'utilisation des méthodes approchées pour leur résolution.

Dans le cadre de ce mémoire, on se focalise sur l'ordonnancement d'atelier de type Job Shop Flexible qui est une généralisation de Job Shop classique. Et pour sa résolution, nous nous intéressons à l'algorithme de recherche gravitationnelle qui est une métaheuristique à base de population de solutions, basée sur les lois de gravitation et de mouvement.

Cette méthode récemment proposée, a prouvé son efficacité dans le domaine de la résolution des problèmes combinatoires complexes. Mais, très peu appliquée sur les problèmes d'ordonnancement.

Ce travail s'inscrit dans le cadre du projet « Simulation et Interaction Avancées dans les Systèmes de Production Cyber-Physiques (SIA-SPCP) » initié au sein de l'équipe SRP (Systèmes Robotisés de Production) du CDTA. Le problème qui nous a été posé consiste en la réalisation d'une application permettant d'optimiser les règles pouvant être utilisées par les

produits et par les machines en vue d'atteindre un objectif donné (minimisation du temps de production). Ces règles sont généralement des règles d'ordonnancement (allocation et priorité) qu'utilisent les produit et ressources.

On explore donc dans notre travail l'hybridation de l'algorithme d'optimisation et la simulation à évènements discrets. Cette dernière, est utilisée comme fonction d'évaluation au sein de l'algorithme d'optimisation. L'objectif principal de cette partie étant de prouver la faisabilité de cette approche proposée. En effet, la simulation-optimisation est très intéressante puisqu'elle permet le calcul en ligne d'un jeu de règles d'ordonnancement utilisable par les produit ainsi que par les machine.

Le mémoire présenté est structuré en quatre chapitres qui permettent un cadrage progressif du sujet.

- ✓ **Chapitre 1**

Ce chapitre introduit les notions de bases relatives au pilotage des systèmes industriels, à la théorie de l'ordonnancement et au problème Job Shop flexible en particulier.

- ✓ **Chapitre 2**

Dans ce chapitre on aborde la complexité des problèmes d'ordonnancement, en présentant une classification des différentes méthodes de résolution proposées dans la littérature.

- ✓ **Chapitre 3**

On présentera dans ce chapitre le problème Job Shop flexible en détails, la méthode de résolution proposée ainsi que notre cas d'études.

- ✓ **Chapitre 4**

Dans ce quatrième chapitre, on présente les éléments de réalisation de notre travail et également une synthèse des résultats obtenus.

Ce mémoire se termine par une conclusion générale où on résume l'apport essentiel de ce travail.

CHAPITRE 1 : PILOTAGE DES SYSTÈMES DE PRODUCTION

1.1 INTRODUCTION

Quel que soit le domaine de production, face à une demande évolutive de produits variés, les entreprises se trouvent obligées de survivre et d'être compétitives. Elles doivent donc optimiser le trio qualité-coût-délai, cherchant ainsi à produire mieux, moins cher et/ou plus vite que leurs homologues. Si l'entreprise ne parvient pas à ces améliorations, dans un environnement de concurrence acharnée, elle le paiera cher, au prix de la perte de ses clients, et au profit de ses concurrents !

L'entreprise ne peut plus se permettre de vivre au gré de ces aléas. Elle doit être capable de gérer ces derniers de la façon la moins perturbante (la plus rapide) quand elle les subit : c'est sa capacité de réactivité. Cette amélioration passe forcément par l'amélioration du processus de pilotage. « Amélioration », puisque le pilotage d'atelier a toujours existé. On pilote sans le savoir... à chaque fois qu'on prend une décision suite à un événement, une perturbation, et qu'une action est mise en place : on pilote ! Mais ceci ne suffit plus. Il est désormais indispensable de structurer le pilotage industriel, afin de le maîtriser et de l'améliorer.

Dans ce chapitre introductif, on cherche à positionner le concept de l'ordonnancement dans le pilotage des systèmes de production. Le chapitre commence par une présentation concise des systèmes de production et de pilotage de ces systèmes où apparait le rôle primordial de l'ordonnancement. La suite du chapitre est concernée au cadrage de la problématique générale de l'ordonnancement par présentation des points essentiels : définition, éléments de base, objectifs, classification et organisations des systèmes concernés.

1.2 LES SYSTÈMES DE PRODUCTION

La production est le processus conduisant à la création de produits par l'utilisation et la transformation de ressources (Giard, 1988). Ce processus de production est constitué d'un ensemble d'opérations qui sont les activités conduisant à la création de bien et de services (Fontanili, 1999).

1.2.1 Description des systèmes de production

Le système de production est l'ensemble de ressources réalisant une activité de production. C'est un ensemble de moyens divers : humains, matériels, informationnels et d'autres, constituant un tout, dont l'objectif est la réalisation de biens et de services (Fontanili, 1999).

C'est en réalisant des opérations que le système produit, les opérations peuvent être de différents types : opérations de transformation, d'assemblage, de stockage ou de transport. Les ressources sont ce que le système utilise pour produire, elles peuvent être des machines, des moyens de transport, des palettes,... Il est difficile d'énumérer tous les types d'opérations et toutes les ressources possibles puisque elles varient selon le domaine du système de production. Il est tout aussi difficile de donner une définition générale utilisant une terminologie générale car le nombre de domaines comportant des systèmes de production est élevés et leurs terminologies sont diverses et variées.

Dans ce travail on s'intéresse aux systèmes de production industriels. Ces systèmes sont de plus en plus diversifiés et compliqués ces dernières années. En effet, ils peuvent être décomposés en plusieurs sous-systèmes, qui s'intègrent en vue d'assurer la pérennité et la compétitive de l'entreprise.

1.2.2 Décomposition des systèmes de production

Les systèmes de production sont en générale des systèmes très complexes et difficiles à gérer. Classiquement, un système de production peut être décomposé en trois sous-systèmes (**Figure 1**) : le système physique de production, le système de décision et le système d'information (Javel, 2004 ; Fontanili, 1999 ; Letouzey, 2001).

Cette décomposition est structurée en fonction de la nature des flux qui traversent chaque système *i.e.* flux de décisions, flux d'informations et flux physique.

- Le système physique de production (système opérant) : qui englobe toutes les ressources humaines et physiques nécessaires pour la transformation des matières premières en produits finis.
- Le système de décision : qui contrôle le système physique de production à travers l'organisation des différentes activités en prenant des décisions basées sur les données transmises par le système informationnel.
- Le système d'information : qui joue d'une part un rôle d'interface entre le système de décision et le système physique de production et d'autre part, il intervient à l'intérieur du système de décision, pour la gestion des informations utilisées lors de la prise de décision. Son rôle est de collecter, stocker et transmettre des informations de différents types.

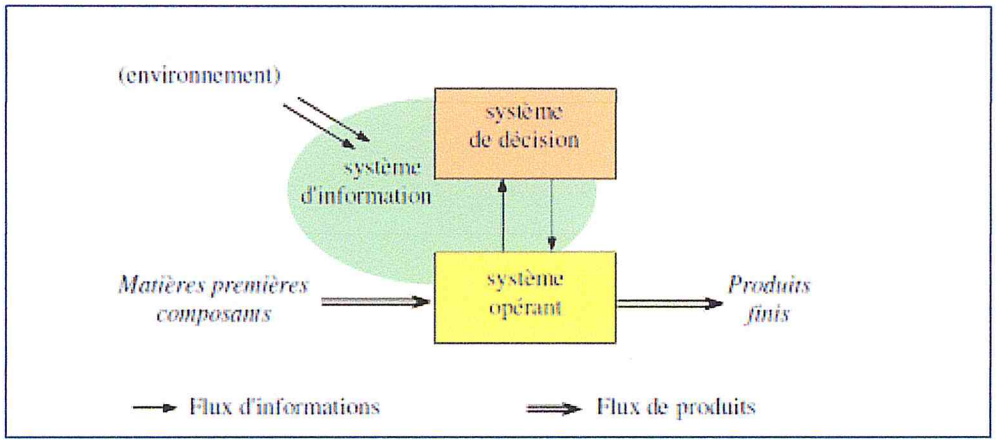

Figure 1 : Décomposition générale du système de production selon *(Fontanili, 1999)*.

1.2.3 Evolution des systèmes de production

Depuis le début du XXe siècle, la production industrielle n'a cessé de se diversifier et de devenir plus complexe en terme (Mirmadi, 2009):

- De contraintes externes : concurrence, prix du marché, diversité des produits, cycle de vie produit réduit, etc.
- De contraintes internes : inertie des processus de fabrication, aléas de fonctionnement, etc.

Les contraintes externes imposent aux industries de réduire les cycles de conception-fabrication des produits. Cette réduction nécessite la flexibilité du système de production afin de faire face au développement d'une diversité de produits dont la durée de vie est sans cesse

réduite. La problématique de la production est globalisée et concerne désormais non seulement le cycle de vie produit mais aussi l'évolution de la structure du système de production.

Par ailleurs, les contraintes internes réduisent cette capacité à suivre les évolutions du marché. L'inertie du processus de production détermine fortement les potentiels de vente d'un produit. En outre, la présence de perturbations de natures diverses altère considérablement le fonctionnement et l'efficacité du système de production.

Il a fallu attendre les années 1980 pour répondre au manque de flexibilité, avec l'arrivée des systèmes de production flexibles. Les systèmes flexibles de production sont définis par MacCarthy et Liu (1993) comme : « un système flexible de production est un système de production capable de produire différents types de pièces, composé de machines à commande numérique ou à contrôle numérique, et des systèmes de transport et de stockage automatisés, connectés par un système automatisé de manutention. Le fonctionnement du système entier est sous le contrôle et le pilotage d'un système informatique ». Ces systèmes ont pour objectifs en particulier la fabrication tout en procurant un meilleur compromis entre productivité et flexibilité par un degré d'automatisation poussé.

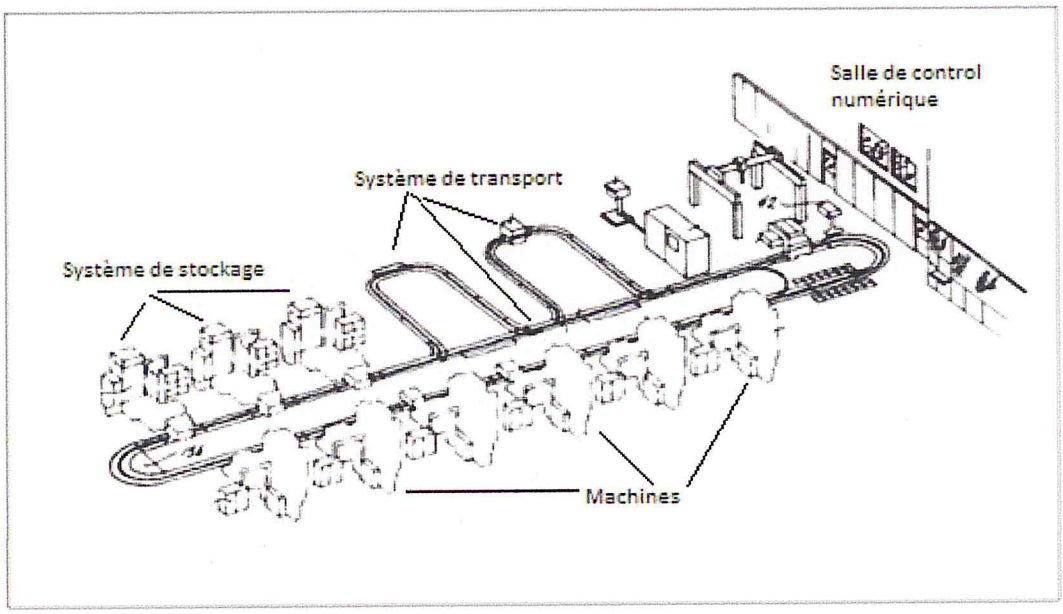

Figure 2 : Exemple d'un système de production flexible *(Chové, 2010)*.

Cette nouvelle organisation des ateliers de production permet de répondre à beaucoup de contraintes liées aux systèmes de production :

- Chaque produit entrant dans le système peut avoir une gamme de fabrication propre pour répondre à la demande de personnalisation grandissante des produits. Ce type de production correspond, dans la littérature, à l'appellation de Job Shop.
- Un même système de production peut facilement prendre en compte de nouvelles références sans que celle-ci aient explicitement été prévues au moment de l'implantation du système.
- Une machine peut être ajoutée ou enlevée du système pour mieux répondre à la charge de production.
- Plusieurs machines peuvent être capables de produire les mêmes produits pour sécuriser le système par la redondance des machines.

1.2.4 Problèmes posés par les systèmes de production

Tout au long du cycle de vie d'un système de production, il faut résoudre des problèmes de planification de production. Dans (Looveren et al., 1986), les auteurs classifient ces problèmes selon trois niveaux temporels lors de gestion de production (**Figure 3**) :

• le niveau stratégique qui correspond aux problèmes de conception d'un système de production relativement aux objectifs retenus. Ce niveau induit clairement des décisions à long terme concernant notamment les composants du système de production.

• le niveau tactique qui comprend, entre autres, le choix des produits à usiner simultanément et conformément aux demandes de production. Plus globalement, ce niveau inclut les décisions à prendre à la suite du niveau stratégique avant qu'une exploitation ne puisse débuter. Dans ce contexte, il s'agit de décisions à moyen terme.

• le niveau opérationnel qui concerne les décisions liées au pilotage d'un système de production. Il s'agit donc d'un niveau décisionnel à court terme.

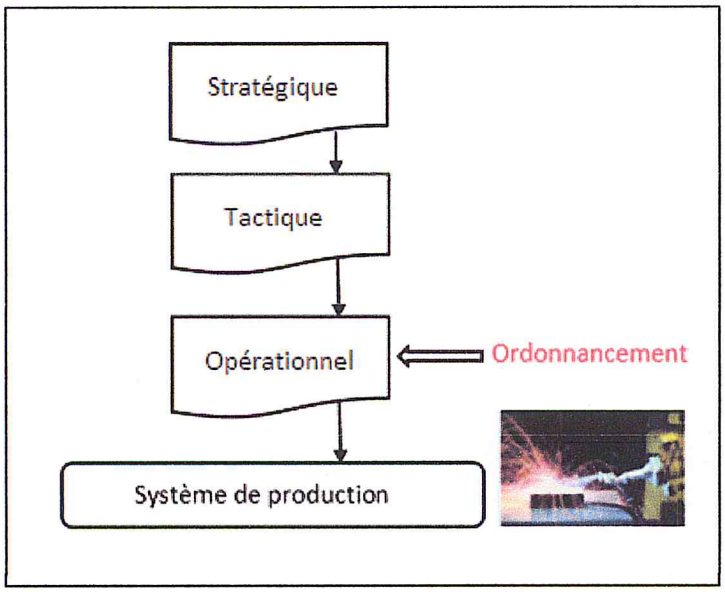

Figure 3 : Organisation hiérarchique de la gestion de production (Ferrah et al., 2010).

Dans ce travail, on s'intéresse plus particulièrement à résoudre un des problèmes de décisions opérationnels, celui lié au pilotage des systèmes de production. La décision à ce niveau met en application le plan d'actions prévu par le niveau tactique. Ce sont les décisions qui concernent les actions à court terme.

1.3 LE PILOTAGE DES SYSTÈMES DE PRODUCTION

Le pilotage des systèmes de production a fait l'objet de nombreux travaux et de publications depuis deux ou trois décennies (Lopez et al., 1999 ; Letouzey, 2001 ; Mirmadi, 2009), que ce soit dans le milieu industriel ou dans le milieu académique. Force est de constater qu'il n'y a pas d'accord complet sur la définition de ce vocable, ni même sur ce qu'il recouvre, selon que l'on a à faire à des automaticiens, des gestionnaires, des chercheurs en génie industriel ou des éditeurs de logiciel. L'objectif de cette section est de délimiter le champ de notre étude et de préciser les termes employés.

1.3.1 Définition du pilotage des systèmes de production

Pour les dictionnaires, piloter c'est tout à la fois conduire, diriger, gouverner. Implicitement, le pilotage fait référence à un système complexe. Ceci se retrouve dans le langage commun : on conduit sa voiture, mais on pilote une formule 1 ou un avion. Piloter un engin revient en premier lieu à : fixer la cible à atteindre, déterminer la meilleure trajectoire de référence pour atteindre cette cible.

D'après (Ferrah et al., 2010), le pilotage consiste en un ensemble d'actions effectuées pour diriger et guider de manière à assurer la pertinence et la cohérence du système en présence de perturbations afin de réaliser un ensemble d'opérations sur un flux de produit. Deux types de perturbations peuvent être distingués: les perturbations externes et les perturbations internes.

- Les perturbations externes sont définies, dans le cas général, comme étant les perturbations affectant l'entrée d'un système ou d'un procédé. Il s'agit des changements de gamme, des changements d'objectifs pour une gamme, de la variation des instants d'arrivée des produits.
- Les perturbations internes sont celles qui affectent la constitution du procédé ou son fonctionnement: les pannes des machines, les changements des durées opératoires, la suppression de ressources pour maintenance préventive…

L'entité du système de pilotage comme le montre la **figure 4** est identifiée par un système de décision qui permet, de préparer les actions ou exécuter les ordres, qui pilotent localement chaque entité sur la base d'un objectif déduit de l'objectif global.

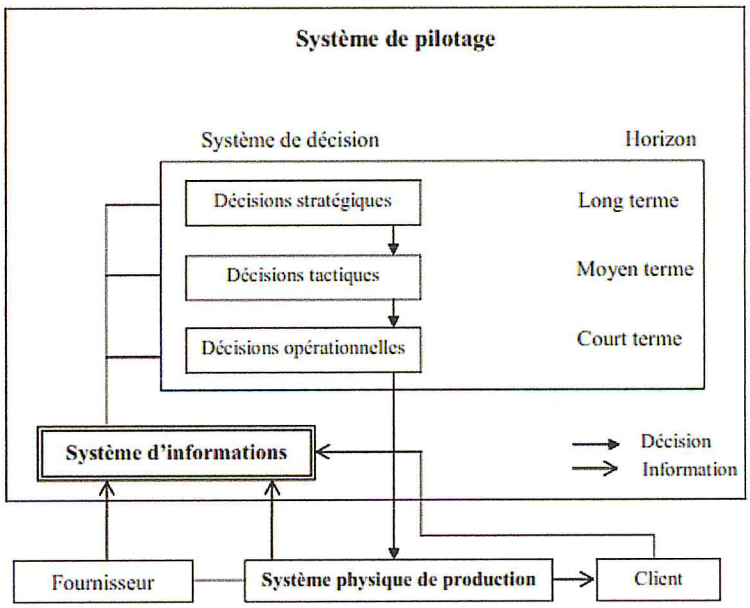

Figure 4 : Le pilotage dans les systèmes de production (Fontanili, 1999) (Letouzey, 2001).

1.3.2 Fonctions de pilotage

Pour atteindre ces objectifs, l'organisation de la production repose en général sur la mise en œuvre d'un certain nombre de fonctions (la planification, la programmation, l'ordonnancement, la conduite et la commande) dont la fonction ordonnancement joue un rôle essentiel (**Figure 5**).

1.3.2.1 Planification

La planification de la production se décline sur les trois niveaux de décision. Elle part d'une vision à long terme sur les évolutions du marché, en prenant en considération les risques et opportunités de l'environnement (concurrence, fournisseurs, évolution des législations et des technologies…) (Boukef, 2011).

1.3.2.2 Programmation

Elle est chargée, à partir du plan directeur de production, d'établir un programme prévisionnel de production atelier par atelier, à capacité infinie ou suivant une charge globale admissible par l'atelier. Ce programme prend en compte les besoins dépendants et indépendants et calcule des besoins nets en fonction des stocks, des tailles des lots de fabrication, des taux de rebut… La programmation consiste essentiellement à décliner les objectifs de la planification en ordre de fabrication (OF) sur les différents ateliers et postes de travail et reste dans une logique de définition du « quoi produire ». (Mirmadi, 2009).

1.3.2.3 Ordonnancement ou séquencement

Ordonnancer, c'est programmer dans le temps l'exécution des tâches à réaliser sur une ressource particulière (une machine par exemple) ou sur un groupe de ressources (comme un atelier). L'ordonnancement fournit donc un calendrier d'organisation du travail pour l'atelier fixant les dates de début et de fin de chaque tâche. Séquencer consiste plus simplement à déterminer l'ordre dans lequel les différentes tâches ou les différents produits passeront sur une ressource ou un groupe de ressources. Les dates de début et de fin de chaque tâche ne sont donc pas explicitement fixées. Dans les deux cas, l'objectif est de réaliser les ordres de fabrication (OF) du programme prévisionnel (délais, quantité, qualité, etc.) tout en cherchant à optimiser l'utilisation des moyens de production en termes de charge, d'en-cours etc (Mirmadi, 2009).

L'ordonnancement doit respecter les contraintes de succession dues aux gammes de fabrication et les contraintes technologiques. Il est basé sur des calculs de charge, de temps

moyen de fabrication et de respect du délai. Un ordonnancement, aussi bon soit-il, ne peut atteindre tous les objectifs, généralement contradictoires. Généralement, il retient comme critère essentiel le respect des délais (minimisation des retards ou des retards pondérés). On se rapproche ici de la définition du « comment produire ». (Mirmadi, 2009).

Dans ce travail, on s'intéresse plus particulièrement à cette fonction du pilotage. La prochaine section sera consacrée pour bien expliciter ce point.

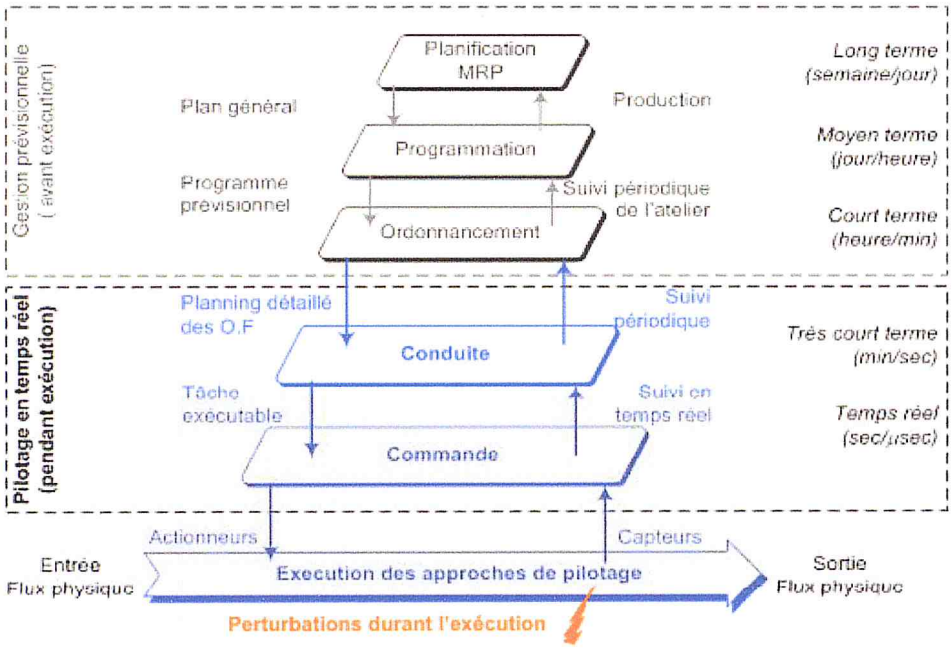

Figure 5 : Fonctions du pilotage dans la production (Mirmadi, 2009).

1.3.2.4 La conduite

La phase de conduite est chargée de réaliser la production prévue par l'ordonnancement. Ce dernier fournit une solution réalisable à son niveau, solution qui n'est pas toujours une solution « réalisable » au niveau de la conduite. Une des raisons de cet écart réside dans le caractère rigide des données et contraintes prises en compte pour établir l'ordonnancement prévisionnel. La conduite peut être décomposée en sous-fonctions dont les principales sont (Trentesaux, 1996):

- Le contrôle : cette fonction assure la cohérence globale du système de pilotage, et gère les interactions entre les différents centres de décision, etc.
- La définition et la gestion de l'environnement : gestion de la logistique technique, des configurations, etc.

A ces deux fonctions est associée la fonction de suivi qui collecte les données nécessaires au suivi de fonctionnement du système, de ses performances, du respect des objectifs, etc.

1.3.2.5 La commande

Ce niveau, directement en relation avec le système de production, a un rôle d'interface et d'interpréteur. Sa tâche essentielle est de traduire un ordre en une séquence d'instructions compréhensibles par la partie opérative. Il est concrétisé soit par un opérateur pilotant une machine et assurant le suivi de réalisation, soit par un automatisme capable d'interpréter un ordre et de renseigner la conduite sur l'état d'avancement de celui-ci. C'est le plus bas niveau, ou le niveau physique (Mirmadi, 2009).

1.4 ORDONNANCEMENT DES SYSTÈMES DE PRODUCTION

L'ordonnancement est une branche de la recherche opérationnelle et également de la gestion de la production, qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activités de l'économie, depuis l'industrie manufacturière (Pinedo, 1955) jusqu'à l'informatique (Blazewicz, et al., 1996).

Ordonnancer le fonctionnement d'un système industriel de production consiste à gérer l'allocation des ressources au cours du temps, tout en optimisant au mieux un ensemble de critères (Rodammer, et al., 1999). C'est aussi programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution (Carlier , et al., 1988).

Les problèmes d'allocation des ressources, d'organisation des tâches, de respect des délais et de prise de décision en temps requis constituent autant de difficultés qu'il est nécessaire de surmonter dans la gestion des systèmes de production en milieu industriel. (Boukef, 2011).

Pour mieux comprendre ce problème, le reste de cette section est organisé comme suit : nous commençons par donner une définition générale du problème d'ordonnancement. Ensuite, nous présentons ses éléments de base.

1.4.1 Définition du problème d'ordonnancement

Les problèmes d'ordonnancement apparaissent dans tous les domaines : informatique, industrie, construction, administration, etc. (Carlier , et al., 1988).

Une définition générale d'un ordonnancement est donnée par Parunak (Parunak, 1991): "Un ordonnancement est un sous-ensemble du produit cartésien Quoi*Où*Quand pour un sous-ensemble de l'ensemble des tâches à réaliser".

Les différentes données d'un problème d'ordonnancement sont les tâches, les ressources, les contraintes et les objectifs. Ainsi, étant donnés un ensemble de tâches et un ensemble de ressources, l'ordonnancement s'agit de programmer les tâches et les affecter aux ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), tout en respectant un ensemble de contraintes.

1.4.2 Eléments du problème d'ordonnancement

1.4.2.1 Les tâches

Une tâche[1] (Job) est un travail mobilisant des ressources et réalise un progrès significatif dans l'état d'avancement du projet. Une tâche « i » est une entité élémentaire de travail localisée dans le temps par une date de début « t_i » ou de fin « c_i », dont la réalisation est caractérisée par une durée « p_i » (on a $c_i = t_i + p_i$). En outre, la tâche « i » utilise une (ou plusieurs) ressource(s) « k »avec une intensité « a_{iK} » souvent supposée constante pendant l'exécution de la tâche. (Ferrah , et al., 2010).

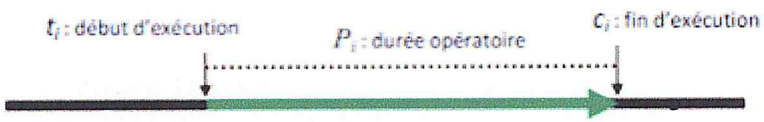

Figure 6 : Caractéristiques d'une tâche i.

Dans certains problèmes, les tâches peuvent être exécutées par morceaux ; l'entrelacement des différents morceaux permet de laisser le moins possible les ressources inactives. Dans d'autres, au contraire, on ne peut pas interrompre une tâche une fois commencée. On parle alors respectivement de problèmes préemptifs et non préemptifs.

Une séquence d'opérations ou tâches, est appelée gamme opératoires. Elle détermine l'ordre de passage des opérations permettant la réalisation d'un produit.

[1] Dans ce mémoire les termes tâche, produit et job ont une même signification.

1.4.2.2 Les ressources

Une ressource est un moyen technique ou humain requis pour la réalisation d'une tâche et disponible en quantité limitée. On distingue plusieurs types de ressources :

a. Ressources renouvelables vs. Ressources consommables

Une ressource est dite renouvelable si après avoir été utilisée par une ou plusieurs tâches, elle est à nouveau disponible en même quantité comme : les ouvriers, les machines, l'espace, … etc. ; La quantité de ressources utilisée à chaque instant est limitée. Par contre une ressource est dite consommable, lorsque elle devient non disponible après avoir été utilisé comme par exemple : la matière première, l'énergie, le budget, … etc. La consommation globale (ou cumul) au cours du temps est limitée. (Lopez, et al., 1999).

b. Ressources disjonctives vs. Ressources cumulatives

On distingue par ailleurs les ressources disjonctives (ou non partageables) qui ne peuvent être affecté qu'à une tâche à la fois (machine-outil, robot manipulateur). Dans le cas contraire les ressources cumulatives (ou partageables) peuvent être utilisées par plusieurs tâches simultanément (équipes d'ouvriers, poste de travail) (Kebabla, 2008).

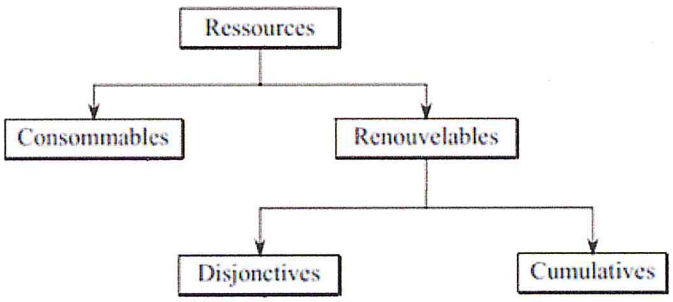

Figure 7: Classification des types de ressources.

1.4.2.3 Les contraintes

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décision (Lopez, et al., 1999) .Donc les contraintes représentent les limites imposées par l'environnement, tandis que l'objectif est le critère d'optimisation.

Plusieurs types de contraintes doivent être respectés. On distingue notamment (Baptise, 1998) (Lopez, et al., 1999) (Multicreteria scheduling Theory,Models and Algorithms, 2006):

- Les contraintes temporelles intègrent en général :
  - Les contraintes de temps alloué qui sont issues généralement d'impératifs de gestion et relatives aux dates limites des tâches (délai de livraison par exemple) ou à la durée totale d'un projet ;
  - Les contraintes d'antériorité et plus généralement les contraintes de cohérence technologique, qui décrivent le positionnement relatif de certaines tâches par rapport à d'autres (ex. contraintes de gammes dans le cas des problèmes d'ateliers) ;
  - Les contraintes de calendrier liées au respect d'horaires de travail, etc.
- Les contraintes de ressources : représentent le fait que les activités utilisent une certaine quantité de ressource, tout au long de leur exécution. Ces contraintes sont essentiellement soit :
  - Des contraintes d'utilisation des ressources qui expriment la nature, la quantité et les caractéristiques d'utilisation de ces ressources.
  - Des contraintes de disponibilités des ressources qui déterminent les quantités des ressources disponibles au cours du temps.

1.4.2.4 Les objectifs

Tout ordonnancement est guidé par un ou plusieurs objectifs qu'il doit chercher leur optimisation. Les objectifs que doit satisfaire un ordonnancement sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement donné (Lopez, et al., 1999) :

- Les objectifs liés au temps : par exemple, minimiser le temps total d'exécution (le makespan), les durées totales de réglage ou les retards par rapport aux dates de livraison.
- Les objectifs liés aux ressources : par exemple, maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches.
- Les objectifs liés au coût : ces objectifs ont généralement pour but de minimiser les coûts par exemple, minimiser le coût de production, de stockage, de transport, etc.
- Les objectifs liés à l'énergie ou au débit : par exemple, minimiser l'énergie consommée et augmenter le débit de production.

Ces objectifs peuvent correspondre à des exigences quantitatives (valeurs à atteindre ou à ne pas dépasser) et se présentent sous forme de contraintes à respecter, ou bien à des exigences qualitatives s'exprimant sous forme de critères à optimiser.

Dans le cadre de réalisation de notre travail, le critère qu'on cherche à optimiser (minimisation) est celui de la durée totale de l'exécution de l'ordonnancement (makespan).

- **La Durée totale d'exécution (Makespan ou Cmax)**

C'est la durée totale nécessaire à la fabrication de tous les produits de l'instance considérée selon l'ordre de fabrication imposé par un ordonnancement x. La minimisation de cette durée est le critère le plus souvent rencontré puisque ça conduit inévitablement à une utilisation efficace des ressources.

Autrement dit, le $Cmax = Max\ C_i$: est les plus grands délais d'achèvement, ou bien la durée totale, le temps nécessaire pour finir toutes les tâches (i), en terme anglo-saxon on l'appel 'Makespan'.

Afin de calculer cette durée, on peut utiliser l'un de ces outils :

a. Le diagramme de Gantt

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

La **figure 8** montre un exemple d'ordonnancement de 3 produits sur 3 machines, tel que le premier et le deuxième produit (barres bleu et orange respectivement) disposent de 3 opérations chacun, alors que le troisième produit en dispose de 2 (barres vertes).

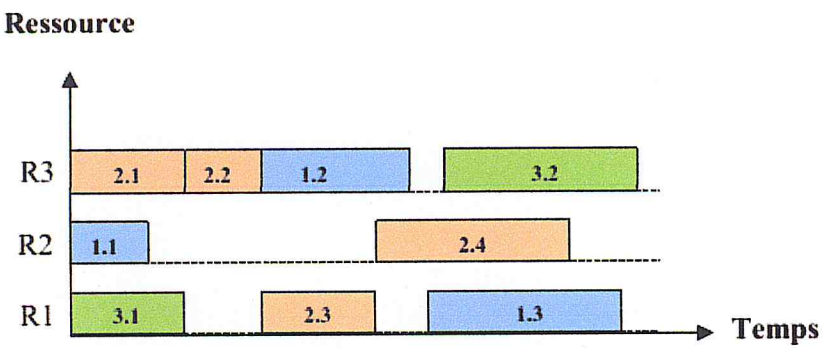

Figure 8 : Exemple d'un diagramme de Gantt.

b. Le graphe conjonctif

Une solution est représentée par le graphe conjonctif classique $G = (V, E)$. L'ensemble des sommets est $V = \{O_{i,j} / 1 \leq i \leq n, 1 \leq j \leq n_i\} \cup \{s, t\}$ avec « s » le sommet source et « t »le sommet puits. On a un arc entre deux sommets s'il existe une contrainte de précédence entre les opérations correspondantes. Il y a un arc entre O_{i,n_i} et « t » de longueur $P_{i,n_i,k}$ ou R_k est la ressource sur laquelle O_{i,n_i} s'exécute. Il y a un arc entre « s » et $O_{i,j}$ de longueur r_i . Pour les autres arcs, la longueur est égale à la durée opératoire de l'opération sur la ressource exécutée à l'origine de l'arc. Ce graphe ne doit pas contenir de circuit, sinon l'ordonnancement est infaisable (non admissible) et tous les arcs ont une longueur positive ou nulle. La ***figure 9*** donne un exemple d'un graphe conjonctif correspondant au diagramme de Gantt de la ***figure 8*** dont les arcs pleins représentent les contrainte de gammes (contrainte de précédence entre les opérations d'un même job)et les arcs pointillés les contraintes de ressources.

En faisant une recherche de plus long chemin entre set t dans ce graphe on obtient la valeur du makespan.

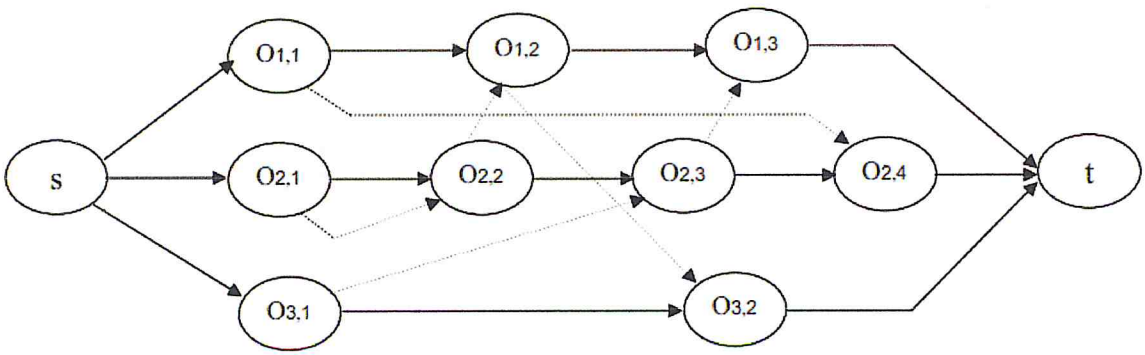

Figure 9 : Exemple d'un graphe conjonctif.

1.5 CLASSIFICATION DES PROBLÈMES D'ORDONNANCEMENT

Classifier les problèmes d'ordonnancement revient à étudier les différents types de machines utilisées ainsi que la manière avec laquelle l'atelier est organisé.

Les ateliers[2] dans les systèmes de production sont nombreux, ils se diffèrent par les nombres de machines, la nature des jobs, l'unicité ou la diversité du routage des jobs au niveau des machines, ou par la flexibilité des ressources (c'est-à-dire la possibilité d'avoir un sous-ensemble de machines candidates dans lesquelles une telle tâche ou opération peut être traitée), dans le cas d'une flexibilité partielle, sinon (cas d'une flexibilité totale) n'importe quelle tâche peut être exécutée sur n'importe quelle ressource. Selon ces paramètres, et comme le montre la **figure 10**, nous distinguons les trois types d'ateliers suivants : flow-shop, job-shop et open-shop, avec des extensions possibles pour chacun d'eux (Hentous, 1999 ; T'Kind, 2006).

1.5.1 Atelier à cheminement unique : Flow shop

Dans ce type d'atelier, la ligne de fabrication est constituée de plusieurs machines en série, de telle sorte que toutes les opérations de tous les jobs passent par toutes les machines en respectant le même ordre. C'est pour cela ces ateliers sont appelés les ateliers à cheminement unique. Ce type se rencontre fréquemment en pratique : montage, chaine de traitement, etc. Si on trouve plusieurs exemplaires identiques et parallèles de la même machine, l'atelier est dit alors Flow-Shop Hybrides. La figure suivante montre des exemples de ce type d'ateliers.

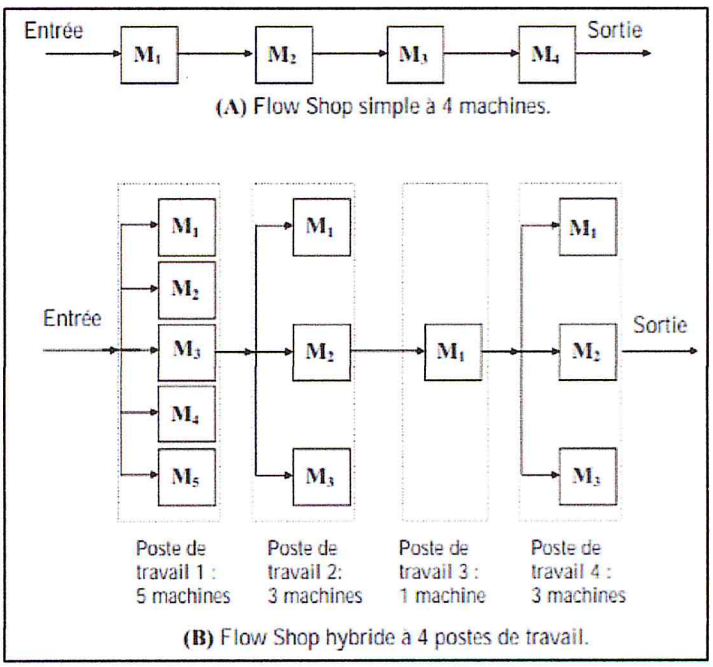

Figure 10 : Exemples de flow shop simple et hybride (Kebabla, 2008).

[2] Un atelier est un local ou un espace consacré à la fabrication, dans une usine.

1.5.2 Atelier à cheminement multiple : Job Shop et Job Shop flexible

Contrairement au type d'atelier précédent, l'atelier Job Shop se caractérise par un cheminement multiple, puisque les opérations de chaque job peuvent emprunter divers chemins (routage des opérations).

L'atelier Job Shop Flexible est une extension du modèle Job Shop classique ; sa particularité réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations Un exemple de ce type d'atelier est présenté dans la **figure 11**.

Dans ce travail, on s'intéresse exclusivement à l'ordonnancement Job Shop flexible.

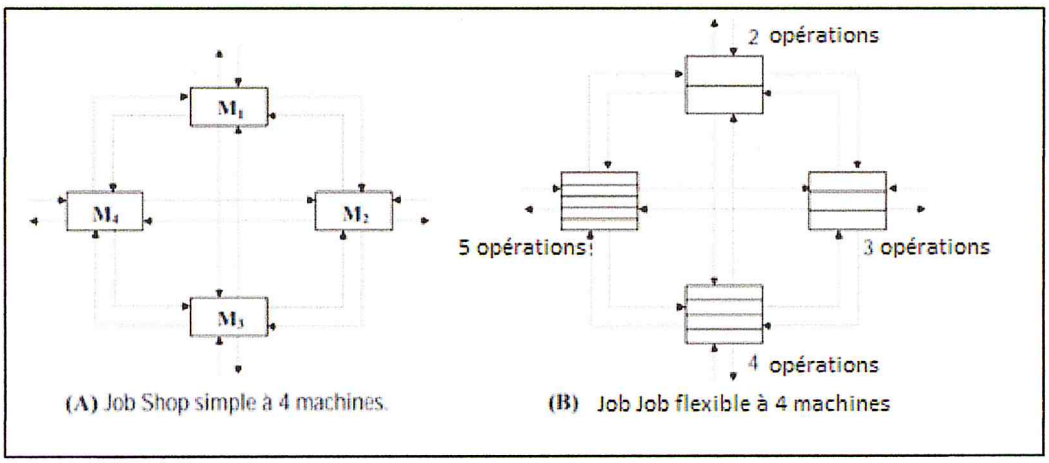

Figure 11 : Exemples de Job Shop simple et Job Shop flexible (Kebabla, 2008).

1.5.3 Atelier à cheminement libre : Open shop

C'est un modèle d'atelier moins contraint que le job-shop et le flow-shop. Dans les ateliers de type Open Shop, les contraintes de précédence sont relâchées (Duvivier, 2000). Autrement dit, les opérations nécessaires à la réalisation de chaque tâche peuvent être effectuées dans n'importe quel ordre.

1.6 LE PROBLÈME D'ORDONNANCEMENT JOB SHOP FLEXIBLE

Un job shop flexible est une extension du problème à cheminements multiples classique (Job Shop simple). Dans un tel problème d'ordonnancement, ***n*** jobs doivent être effectués par ***m*** machines, sachant que chaque machine ne peut travailler que sur un seul job à la fois.

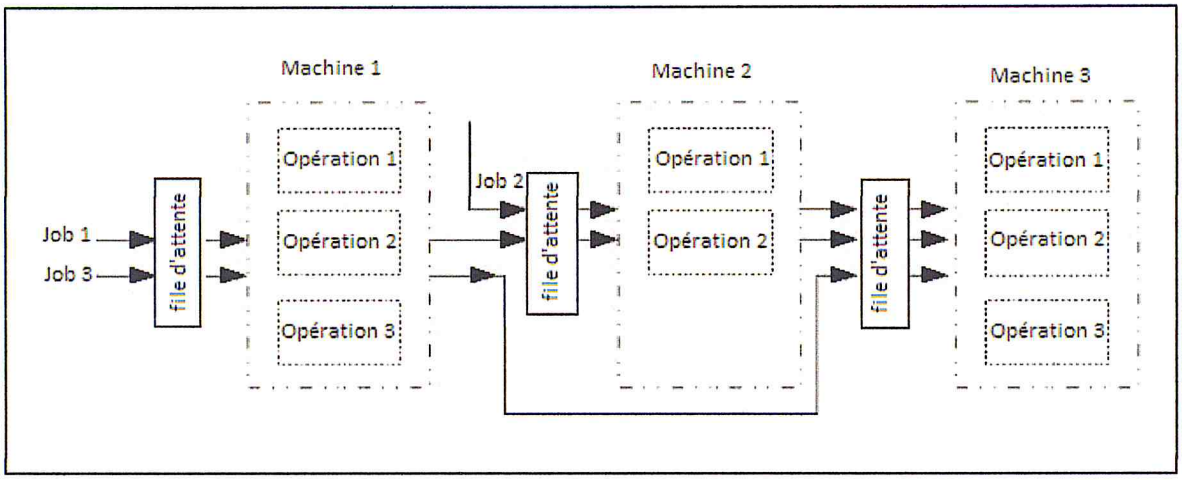

Figure 12 : Représentation d'un système de type Job-shop flexible.

Ce type de problème peut se décomposer en deux phases :

- La première phase est décrite par des contraintes potentielles qui regroupent les contraintes de précédence entre tâches, prenant en compte la succession des opérations de la gamme opératoire et les contraintes d'une tâche dans le temps. En fait, cette phase tente d'affecter les différents produits aux machines disponibles selon des règles d'affectation.
- La deuxième phase est décrite par les contraintes disjonctives qui expriment le fait que deux produits utilisant une même ressource, ne peuvent avoir d'intervalle de temps d'exécution commun. De plus, les ressources utilisent des règles de priorité pour choisir quel produit exécuter dans sa liste d'attente.

Notons ici que ces dites règles d'ordonnancement (d'affectation et de priorité) seront détaillées dans le chapitre 3.

D'une part, la résolution de ce problème Job-shop flexible présente une difficulté dans le fait de trouver la meilleure combinaison des règles d'ordonnancement qui donne un temps total de production le plus court. Afin d'atteindre cet objectif, les méthodes d'optimisation combinatoire sont les plus utilisés dans la littérature. C'est dans ce contexte que s'inscrit notre problématique. En effet, dans notre travail, la méthode de résolution que nous avons adoptée est l'algorithme de recherche gravitationnelle. Cette méthode et d'autres seront décrites dans le chapitre suivant.

D'autre part, dès que le problème d'ordonnancement devient plus complexe et plus difficile à modéliser graphiquement, les représentations décrites précédemment (Section 1.4.2.4.1) ne sont plus pratiques. Le recours à la simulation des ateliers de production devient primordial.

La simulation est un outil largement répandu dans le monde industriel, et dans le monde de la recherche académique. Depuis 30 ans, la simulation s'impose, avec l'objectif de pouvoir travailler sur un système de production virtuel, dont le comportement peut être très proche du système réel, à moindre coût et sans risques. Elle est conçue pour représenter l'évolution d'un système dans le temps. Elle est définie comme une manipulation numérique d'un modèle, dont les simulations faites sur ce modèle a pour but de comprendre le comportement du système et/ou d'évaluer différentes stratégies pour différentes opérations du système et calculer leurs temps d'exécution, et de mesurer l'impact relatif de chacune de ces composantes sur la performance globale du système de production.

Ce travail porte sur la simulation des systèmes de production à évènements discrets pour lesquelles les changements d'état surviennent lors d'événements tels que, le début ou à la fin d'une opération, la mise en attente d'une pièce dans un stock, la libération d'une ressource... La simulation des systèmes à évènements discrets intègre à la fois la construction d'un modèle et l'utilisation expérimentale de ce modèle pour étudier un problème. Cette approche est plus détaillée dans le troisième chapitre.

L'optimisation des solutions au problème de Job Shop Flexible est donc basée sur l'hybridation simulation-optimisation. Selon cette hybridation, la simulation est utilisée comme mécanisme d'évaluation au sein de l'algorithme d'optimisation.

1.7 CONCLUSION

Nous avons présenté dans ce chapitre les concepts relatifs à notre étude. Dans un premier temps nous avons présenté succinctement les systèmes de production, puis nous avons abordé le concept de pilotage dans ces systèmes en mettant l'accent sur le rôle de l'ordonnancement au sein de ces derniers. Le troisième point exposé dans ce chapitre, est la présentation du problème d'ordonnancement en présentant sa définition et en précisant notamment les différents éléments qui le déterminent : les tâches, les ressources, les contraintes et les objectifs ; ainsi que les approches de classification qui s'appuient sur un ensemble de paramètres portant sur différentes caractéristiques du problème dont l'organisation d'ateliers décrites par la suite, constitue un champ important de la classification.

Parmi les problèmes d'ordonnancement présentés, nous sommes particulièrement intéressés au problème de Job Shop flexible, car il généralise la pluparts des modèles d'ateliers. La dernière section du chapitre est consacrée pour bien expliquer ce problème.

Dans la littérture, plusieurs approches de résolution du problème d'ordonnancement ont été proposées. Elles sont génaralement classées en deux catégories : approches exactes et appoches approximatives. Le chapitre suivant permettra d'expliciter ces approches.

CHAPITRE 2 : LES MÉTHODES DE RÉSOLUTION DU PROBLÈME D'ORDONNANCEMENT

2.1 INTRODUCTION

La résolution des problèmes de production, constitue une des principales préoccupations des industriels. En effet, comment réussir à produire dans les meilleurs délais avec un meilleur coût tout en tenant compte des différentes contraintes ?

Des techniques d'ordonnancement dans ce cadre ont pour objectif de répondre au mieux à ces besoins. La résolution optimale d'un problème d'ordonnancement consiste en la détermination d'une séquence d'exécution des tâches sur les machines de sorte qu'un certain critère d'optimisation atteigne sa valeur optimale.

Généralement, les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoire dont la complexité est établie NP-difficile pour la plupart d'entre eux.

Il existe des méthodes déterministes permettant de résoudre certains problèmes d'optimisation en un temps fini. On peut citer comme exemple de ces méthodes : la méthode Branch & Bound et la programmation dynamique,… Ces méthodes garantissent en général l'optimalité de la solution fournie. Mais vu la complexité des problèmes d'ordonnancement ces méthodes deviennent très vite inadaptées lorsque la taille du problème croit, d'où la nécessité d'utiliser des méthodes de résolution approchées, généralement efficaces pour les problèmes NP- difficiles.

Dans ce chapitre, une description des principales méthodes d'optimisation utilisées dans la littérature est réalisée après avoir abordé la complexité des problèmes d'ordonnancement. Ensuite, un état de l'art des principales méthodes d'optimisation utilisées dans la littérature pour la résolution des problèmes d'ordonnancement est présenté. Ces méthodes sont synthétisées dans la dernière section du chapitre.

2.2 COMPLEXITÉ DES PROBLÈMES D'ORDONNANCEMENT

Dans les problèmes d'ordonnancement, la complexité est liée à la complexité des méthodes de résolution et des algorithmes utilisés. (Garey, et al., 1979) (Carlier , et al., 1988). La complexité de quelques types de problèmes d'ordonnancement dotés d'une taille relativement grande devient importante. Notamment le problème d'ordonnancement de type Job Shop flexible (Tangour, 2007).

Selon (Meziane, 2011), la complexité d'ordonnancement peut être divisée en deux grandes catégories : algorithmique et problématique.

2.2.1 La complexité algorithmique

L'objectif de la théorie de complexité est d'analyser les coûts de résolutions surtout en termes de temps de calcul. Elle vise aussi à classifier les problèmes en plusieurs niveaux de difficulté. Une étude a prouvé que les problèmes d'ordonnancement sont des problèmes difficiles (Lopez, et al., 2001).

En général la complexité algorithmique se mesure par rapport à deux paramètres :

- Le temps alloué pour l'exécution de l'algorithme : il est relatif au nombre d'instructions à exécuter ainsi qu'à la taille des données manipulées.
- Espace mémoire requis : associé à la taille d'instance d'un problème donné. (Paschos, 2005).

2.2.2 La complexité problématique

La complexité problématique est relative au problème à résoudre ainsi que la méthode de résolution adoptée pour élaborer la solution optimale par rapport aux critères retenus.

- Un problème de décision comprend deux parties : une partie contenant les données du problème et une deuxième, le processus binaire ayant « oui » ou « non » comme réponses possibles.
- Un problème de recherche est un problème constitué d'un ensemble de données dont chacun représente un ensemble de solutions. Donc la résolution d'un problème de recherche consiste à trouver pour chaque ensemble de données **D** des solutions **S** associées. Un problème d'optimisation est un problème de recherche en associant à chaque solution une valeur qualitative. A chaque problème d'optimisation, on peut associer un problème de décision, donc l'étude de la complexité du problème de

décision peut donner des indications au problème d'optimisation associé. ("Ordonnancement dynamique dans les industries agroalimentaires", 2007).

La théorie de complexité permet de classer les problèmes en deux classes *P* et *NP*. La classe *P* regroupe les problèmes qui peuvent être résolus par des algorithmes polynomiaux. Un algorithme est dit polynomial, lorsque son temps d'exécution est borné par *O (P(x))* où *P* est un polynôme et *x* est la longueur d'entrée d'une instance du problème. Les algorithmes dont la complexité ne peut pas être bornée polynomialement sont qualifiés d'exponentiels et correspondent à la classe *NP*. (Kebabla, 2008).

Un problème de décision est dit *NP-Complet* s'il appartient à la classe *NP* et il est résolu, au mieux, en un temps exponentiel.

Un problème d'optimisation est dit *NP-Difficile,* si le problème de décision associé est *NP-Complet.*

2.3 LES APPROCHES DE RÉSOLUTION DES PROBLÈMES D'ORDONNANCEMENT

La difficulté de résolution des problèmes d'optimisation a poussé les chercheurs à proposer plusieurs méthodes en recherche opérationnelle (RO) et en intelligence artificielle (IA). Ces méthodes peuvent être classées sommairement en deux grandes catégories : Les méthodes exactes et les méthodes approchées.

2.3.1 Les méthodes exactes

Les méthodes exactes permettent de fournir une solution optimale en parcourant la totalité de l'ensemble de l'espace de recherche de manière à assurer l'obtention de toutes les solutions ayant le potentiel d'être meilleures que la solution optimale trouvée au cours de la recherche. Ainsi, ces méthodes exactes nécessitent un coût de recherche souvent prohibitif en termes de ressources requises. Du coup, l'utilisation de ce type de méthodes s'avère particulièrement intéressante dans les cas des problèmes de petites tailles (Boukef, 2009).

2.3.2 Les méthodes approchées

La nécessité de disposer d'une solution de bonne qualité (semi optimale) avec un coût de recherche raisonnable a incité les chercheurs à proposer un autre type de méthodes de résolution de problèmes, communément connues par les méthodes approchées. Ces dernières constituent une alternative aux méthodes exactes. En fait, elles

permettent de fournir des solutions de bonne qualité en un temps de calcul raisonnable. De nombreuses méthodes approchées ont été proposées. Elles sont pratiques pour la résolution de problèmes difficiles ou de problèmes dont on cherche des solutions en un bref délai. Ces méthodes sont souvent classées en deux catégories : des méthodes heuristiques et des méthodes métaheuristiques (**Figure 13**).

2.3.2.1 Les heuristiques

Les méthodes dites heuristiques sont des méthodes spécifiques à un problème particulier. Elles nécessitent des connaissances du domaine du problème traité. En fait, elles sont basées sur des méthodes empiriques, elles utilisent des règles simples pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décisions pour construire une solution proche de l'optimal tout en essayant d'avoir un temps de calcul acceptable (Kacem, 2003).

2.3.2.2 Les métaheuristiques

On a vu précédemment que les heuristiques se caractérisent par leur simplicité, ce qui les rend rapides en termes de temps d'exécution, mais une méthode trop simplifiée peut conduire à fournir une mauvaise décision.

Contrairement aux heuristiques, les méthodes dites métaheuristiques sont des méthodes générales, elles sont des heuristiques polyvalentes applicables sur une grande gamme de problèmes. Elles peuvent construire une alternative aux méthodes heuristiques lorsqu'on ne connaît pas l'heuristique spécifique à un problème donné. En général, les métaheuristiques fournissent une solution quasi-optimale afin de garantir la performance (offrir une solution qui s'approche de l'optimum, tout en réduisant au maximum les coûts). La majorité des métaheuristiques sont bio inspirées.

2.3.3 Comparaison entre les méthodes exactes et les méthodes approchées

On présente dans le tableau ci-dessous quelques différences entres les méthodes exactes et les méthodes approchées.

Tableau 1: Tableau de comparaison entre les méthodes exactes et les méthodes approchées (Kebabla, 2008).

Méthodes approchées	Méthodes exactes
Efficacité pour les problèmes de grande taille	Efficacité pour des cas particuliers des problèmes de taille raisonnable
Espace mémoire raisonnable	Espace mémoire considérable
Durée de résolution très petite par rapport aux méthodes exactes	Durée de temps de résolution est généralement considérable pour les problèmes de grande taille
Aucune garantie d'optimalité	Garantie d'optimalité

2.4 PRINCIPAUX SOLUTIONS POUR RÉSOUDRE LE PROBLÈME D'ORDONNANCEMENT

Traditionnellement, les problèmes d'optimisation sont résolus à travers *les méthodes exactes*. Mais lorsque la taille du problème devient importante, la recherche d'une solution optimale dans un espace important devient un processus difficile à achever dans un temps raisonnable, car l'exploration complète de l'espace de solutions prend un temps très important. C'est pour cette raison que les chercheurs ont opté pour l'utilisation des méthodes dites *approchées* ou *approximatives*. Ces dernières sont basées sur des techniques d'exploration de l'espace de solutions plus rapides que celles utilisées dans les méthodes exactes et qui peuvent fournir des solutions quasi optimales ou optimales dans un temps acceptable.

La figure suivante (**Figure 13**) énumère quelques principales méthodes exactes et approchées utilisées pour résoudre le problème d'optimisation Job Shop (Kebabla, 2008) (Tongour Toumi , 2007):

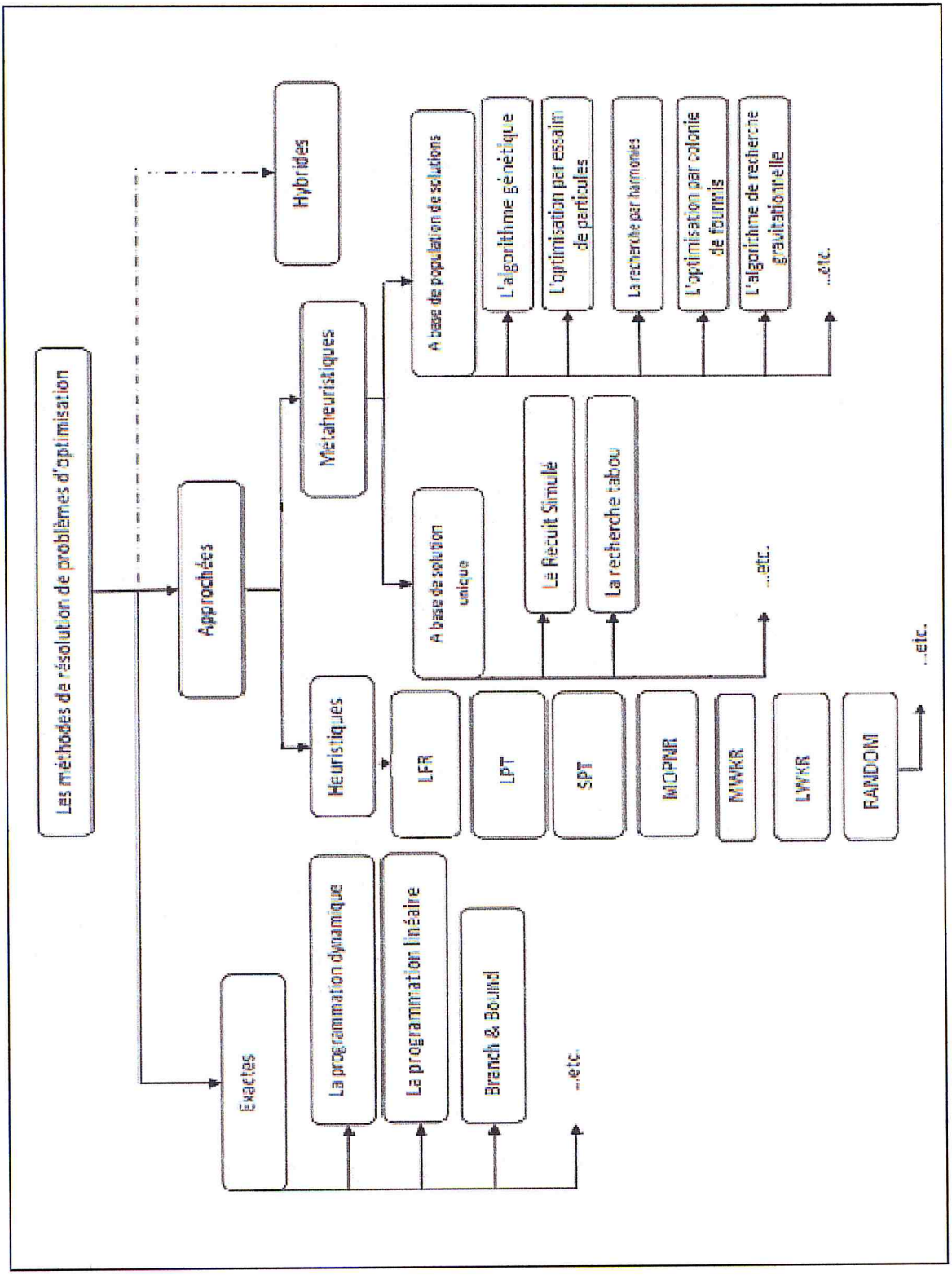

Figure 13: Les principales méthodes de résolution des problèmes d'optimisation.

2.4.1 Les méthodes exactes

Il existe de nombreux algorithmes exacts pour résoudre les problèmes d'optimisation Job Shop flexible, on peut citer : la méthode de Séparation et d'Evaluation Progressive ou « Branch and Bound », la programmation linéaire et la programmation dynamique.

2.4.1.1 La méthode Branch & Bound

L'algorithme Branch and Bound consiste à placer progressivement les tâches sur les ressources en explorant un arbre de recherche décrivant toutes les combinaisons possibles. Il s'agit de trouver la meilleure configuration donnée de manière à élaguer les branches de l'arbre qui conduisent à de mauvaises solutions.

L'algorithme branch and bound effectue une recherche complète de l'espace des solutions d'un problème donné, pour trouver la meilleure solution.

La démarche de l'algorithme Branch and Bound consiste à (Collette, et al., 2002) :

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

2.4.1.2 La programmation dynamique

Dans cette technique, toute solution optimale s'appuie elle-même sur des sous problèmes résolus localement de façon optimale. Concrètement, cela signifie que l'on va pouvoir déduire la solution optimale d'un problème en combinant des solutions optimales d'une série de sous problèmes. Les solutions des problèmes sont étudiées « de bas en haut », c'est-à-dire qu'on calcul les solutions des sous problèmes les plus petits pour ensuite déduire petit à petit les solutions de l'ensemble (Meziane, 2011).

Par exemple pour optimiser la production de 30 produits à budget donné, on optimise la gestion de 2 produits pour tout budget inferieur ou égale, puis on considère l'ensemble comme un produit unique et on ajoute les produits suivants un par un.

2.4.2 Les heuristiques

Comme décrit précédemment (section 2.3.2.1), ce sont des méthodes très simples à mettre en œuvre et qui dépendent du domaine d'utilisation. Elles sont souvent utilisées

pour avoir rapidement des solutions admissibles (réalisables) de qualité relativement bonne, bien que, en générale leur performance ne puisse être garantie.

Plusieurs heuristiques ont été proposées, y compris :

- **LFR** (Last Release Time First) : Cet algorithme choisit parmi les opérations exécutables celle dont le temps écoulé depuis son apparition est le plus grand. Si aucune tâche n'est disponible, alors un temps libre est généré.
- **SPT** (Shortest Processing Time) : La prochaine opération ordonnancée est celle dont la durée opératoire est inférieure à celles des autres opérations.
- **LPT** (Longest Processing Time) : La prochaine opération ordonnancée est celle dont la durée opératoire est supérieure à celles des autres opérations.
- **MOPNR** (Most Operations Remaining or Most Task Remaining) : La prochaine opération ordonnancée est la première opération non encore ordonnancée du Job contenant le plus grand nombre d'opérations non encore achevées.
- **MWKR** (Most WorK Remaining or LRT Most Longest Remaining Processing Time) : La prochaine opération ordonnancée est la première opération non encore ordonnancée du Job dont la somme des durées d'opérations non encore ordonnancée est la plus grande.
- **LWKR** (Least WorK Remaining or SRT Most Shortest Remaining Processing Time) : La prochaine opération ordonnancée est la première opération non encore ordonnancée du Job dont la somme des durées d'opérations non encore ordonnancée est la plus courte.
- **RANDOM** : La prochaine opération ordonnancée est choisie aléatoirement parmi les opérations non encore ordonnancées.

2.4.3 Les métaheuristiques

Dès l'avènement des métaheuristiques comme approche de résolution importante des problèmes difficiles offrant un bon compromis entre le coût de résolution et la qualité des solutions, le problème d'ordonnancement Job Shop constitue l'un des domaines d'application de ces méthodes intensivement investigué. Bien qu'il existe une multitude de métaheuristiques, quelques-uns ont en trouvé une place importante (Kebabla, 2008).

Généralement, on distingue deux grandes classes de métaheuristiques : Les méthodes à base d'une solution unique, et les méthodes à base de population de solutions. Cette classification est basée sur le nombre de solutions manipulées à chaque itération.

2.4.3.1 Les méthodes à base de solution unique

Elles s'appelent encore métaheuristiques à tragectoire ou de voisinage ou même de recherche locale. Cette métaheurisique peut être résumée comme étant une procédure de recherche itérative qui, à partir d'une première solution réalisable, l'améliore progressivement en appliquant une série de modifications (ou mouvements) locales, comme montré dans la **figure 14**. Il faut, pour cela introduire une structure de voisinage qui consiste à spécifier un voisinage pour chaque solution. Ainsi, à chaque itération, la recherche s'oriente vers une nouvelle solution réalisable qui diffère légèrement de la solution courante en remplaçant celle-ci par une meilleure située dans son voisinage. La recherche se termine si un optimum local est rencontré. L'inconvénient important de cette méthode est qu'à moins d'être extrêmement chanceux, cet optimum local est souvent une solution assez médiocre. Dans la recherche locale, la qualité des solutions obtenues dépend fortement de la richesse de l'ensemble des transformations (mouvements) considérées à chaque itération.

Pour faire face à cette limitation, des méthodes de recherche locale plus sophistiquées ont été développées au cours de ces vingt dernières années. Ces méthodes acceptent des solutions voisines moins bonnes que la solution courante afin d'échapper aux minima locaux. En règle générale, seule une portion du voisinage courant est explorée à chaque étape (« Les Métaheuristiques : Des outils performants pour les problèmes industriels », 2001).

Les méthodes les plus connues sont le recuit simulé (Kirkpatrick, et al., 1983), et la recherche tabou (« Tabu search, part I »., 1989).

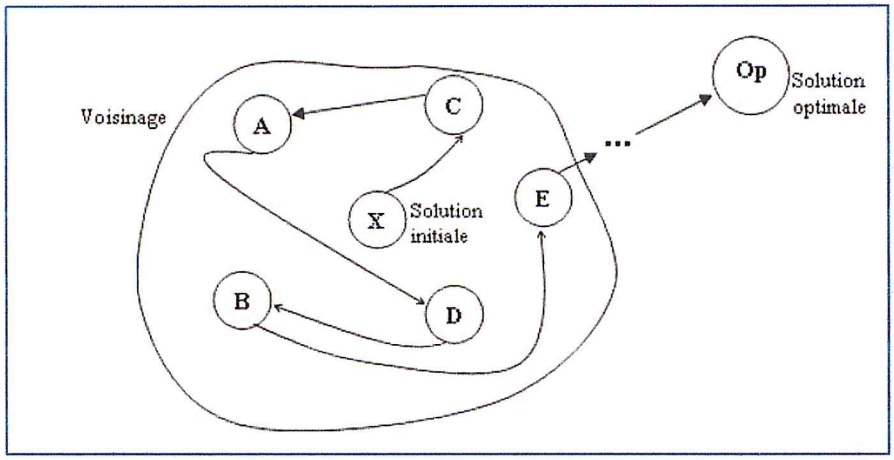

Figure 14: Exploration de l'espace de recherche dans la méthode de recherche locale (Boukef, 2009).

2.4.3.1.1 Le recuit simulé

Le recuit simulé (simulated annealing) est souvent présenté comme la plus ancienne des métaheuristiques, en tout cas, la première à mettre spécifiquement en œuvre une stratégie d'évitement des minima locaux (Dréo, et al., 2005).

Elle s'inspire d'une procédure utilisée depuis longtemps par les métallurgistes qui, pour obtenir un alliage sans défaut, chauffent d'abord à blanc leur morceau de métal, avant de laisser l'alliage se refroidir très lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement (technique du recuit).

Dans (Kirkpatrick, et al., 1983), les auteurs se sont inspirés d'une telle technique pour résoudre des problèmes d'optimisation combinatoire. Le voisinage *N(s)* d'une solution, s'apparente à l'ensemble des états atteignables depuis l'état courant, en faisant subir des déplacements aux atomes du système physique.

A chaque itération, une seule solution voisine est générée. Celle-ci est acceptée si elle est meilleure que la solution courante. Dans le cas contraire, la nouvelle solution est acceptée avec une certaine probabilité qui dépend de l'importance de la détérioration et du paramètre T correspondant à la température. En règle générale, la température est diminuée par paliers, à chaque fois qu'un certain nombre d'itérations est effectué. La meilleure solution trouvée est mémorisée. L'algorithme est interrompu lorsqu'aucune solution voisine n'a été acceptée pendant un cycle complet d'itérations à température constante (Widmer, 2001).

L'algorithme du recuit simulé est présenté dans la **figure 15** :

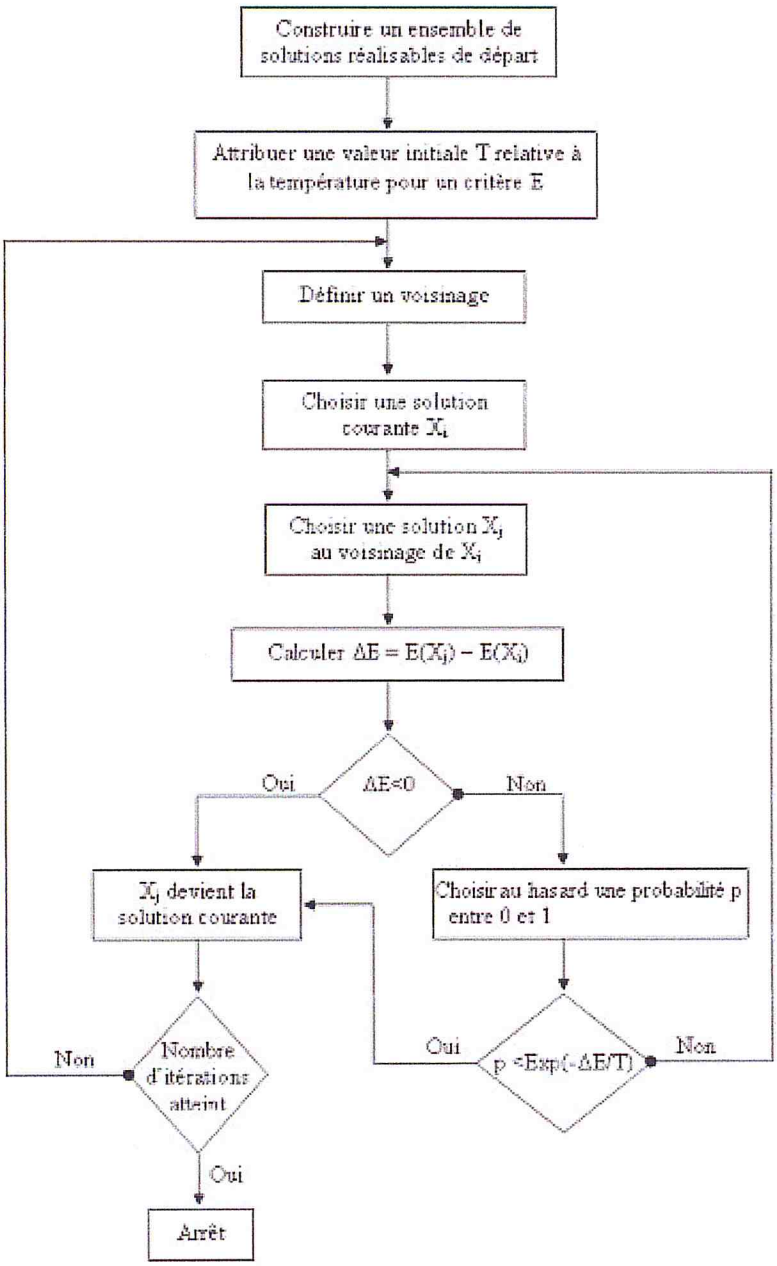

Figure 15: Algorithme relatif au fonctionnement général du recuit simulé.

2.4.3.1.2 La recherche tabou

Bien que son origine remonte à 1977, la recherche Tabou n'est proposée qu'au milieu des années 80 par Fred Glover (Glover, 1989). Cette méthode, développée pour résoudre des problèmes combinatoires, la plupart NP-difficiles.

Elle représente une variante du paradigme de la recherche locale. La composante principale de la recherche Tabou est la fonction de voisinage qui détermine l'efficacité de la recherche.

Son principe consiste à améliorer à chaque étape la valeur de la fonction objectif. Une mémoire qui garde les informations sur les solutions déjà visitées est utilisée. Le contenu de cette mémoire représente la liste Tabou qui sert à empêcher l'accès aux dernières solutions visitées. Lorsqu'un optimum local est atteint, il y'a interdiction de revenir sur le même chemin (Glover, 1989).

L'organigramme de la recherche tabou est présenté dans la **figure 16** :

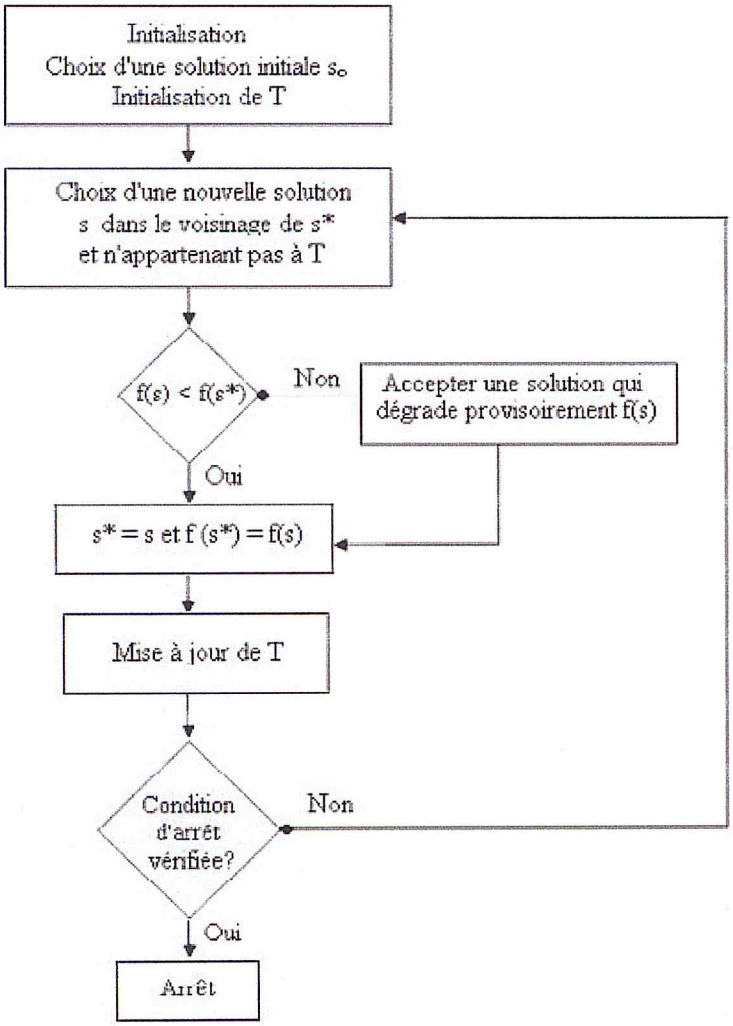

Figure 16: Organigramme relatif au fonctionnement de la méthode de recherche tabou.

2.4.3.2 Les méthodes à base de population

Travaillent sur un ensemble de points de l'espace de recherche en commençant avec une population de solution initiale puis de l'améliorer au fur et à mesure des itérations. L'intérêt de ces méthodes est d'explorer un très vaste espace de recherche et d'utiliser la population comme facteur «de diversité» de plus elle sont très adaptées et très largement utilisées pour la résolution des problèmes d'ordonnancement.

2.4.3.2.1 Algorithmes génétiques

C'est l'une des métaheuristiques les plus connues. Elle est inspirée du concept de sélection naturelle élaboré par Darwin. Ainsi, le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique.

On parle d'individus, pour parler de solutions. L'ensemble des individus formera une population, qu'on fait évoluer pendant une certaine succession d'itérations appelées générations, jusqu'à ce qu'un critère d'arrêt soit vérifié. Pour passer d'une génération à une autre, on soumet la population à des opérateurs de sélection, de croisement et de mutation qui permettront de transformer la population, de façon à favoriser l'émergence de meilleurs individus.

- **La sélection** : qui permet de favoriser les individus qui ont un meilleur fitness pour nous le fitness sera le plus sauvent la valeur de la fonction objectif de la solution associée à l'individu.
- **Le croissement** : qui combine deux solutions parents pour former un ou deux enfants en essayant de conserver les bonnes caractéristiques des solutions parents.
- **La mutation** : qui permet d'ajouter de la diversité à la population en mutant certaines caractéristiques (gènes) d'une solution.

La représentation des solutions (le codage) est un point critique de la réussite d'un algorithme génétique. Il faut bien sûr qu'il s'adapte le mieux possible au problème et à l'évaluation d'une solution (Ferrah, et al., 2010).

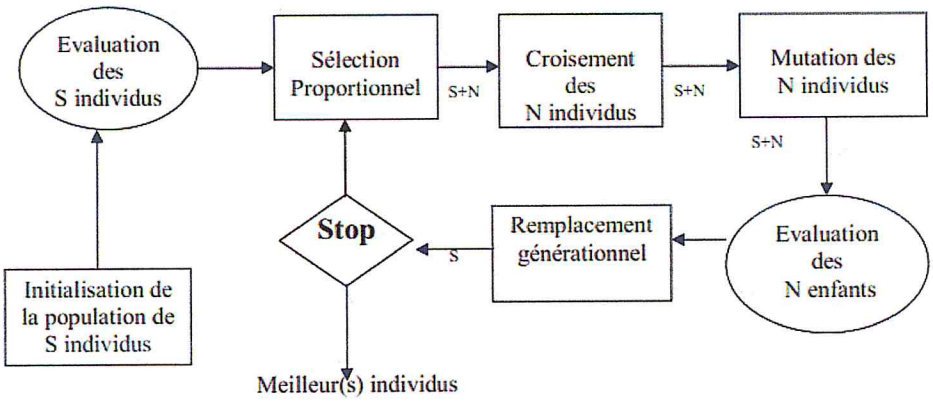

Figure 17: Organigramme d'un algorithme génétique (Dréo, 2004).

Les algorithmes génétiques sont considérés par plusieurs chercheurs comme une méthode bien adaptée au problème de Job Shop, même si elle ne peut pas arriver à l'optimum dans certains cas difficiles (Fang et al., 1993).

2.4.3.2.2 Essaims particulaires

La méthode d'Optimisation par Essaim Particulaire (OEP) a été proposée en 1995 par James Kennedy et Russel Eberhart (Kennedy et al., 1995), qui cherchaient à simuler la capacité des oiseaux à voler de façon synchrone et leur aptitude à changer brusquement de direction, tout en restant en formation optimale.

Le fonctionnement de l'OEP fait qu'elle peut être classée parmi les méthodes itératives (approche progressive de la solution) et stochastiques (faisant appel au hasard) dans le but d'améliorer la situation existante en se déplaçant partiellement au hasard et partiellement selon des règles prédéfinies, en vue d'atteindre la solution globale souhaitée (Clerc et al., 2009).

La méthode d'optimisation par essaim particulaire, est une procédure de recherche basée sur une population d'individus, appelés particules, qui changent leur position (état) avec le temps.

Dans un système d'OEP, les particules se déplacent à l'intérieur d'un espace de recherche. Pendant le déplacement, chaque particule ajuste sa position (X_i) selon sa propre expérience, et selon l'expérience des particules voisines, se servant de sa meilleure position produite ($Pbest_i$) et de celle de ses voisines ($Gbest$). Ce comportement est semblable à celui du comportement humain consistant à prendre des décisions où les individus considèrent leur expérience antérieure et celle des personnes qui les entourent (Kennedy et al., 1995). L'OEP peut ainsi combiner des méthodes de recherche locale avec des méthodes de recherche globale (métaheuristiques).

Même si beaucoup de similitudes existent entre les méthodes évolutionnaires et l'OEP, cette dernière se distingue par le fait qu'elle n'utilise pas l'opérateur de sélection qui choisit les individus gardés dans la prochaine génération.

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

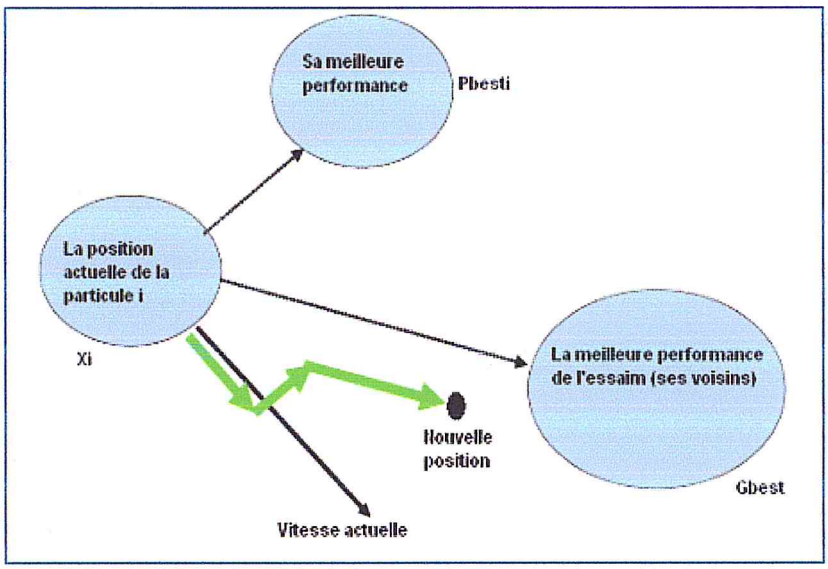

Figure 18: Exemple expliquant les paramètres de la méthode OEP.

2.4.3.2.3 Colonie de fourmis

Les algorithmes à base de colonies de fourmis ont été introduits par Dorigo (Dorigo et al, 1996). Une des applications principales de la méthode originale était le problème du voyageur de commerce et depuis elle a considérablement évolué.

Cette nouvelle métaheuristique imite le comportement de fourmis cherchant de la nourriture. A chaque fois qu'une fourmi se déplace, elle laisse sur la trace de son passage une odeur (la phéromone). Avec plusieurs de ses congénères, elle explore une région en quête de nourriture. Face à un obstacle, le groupe des fourmis explore les deux côtés de l'obstacle et se retrouvent, puis elles reviennent au nid avec de la nourriture. Les autres fourmis qui veulent obtenir de la nourriture elles aussi vont emprunter le même chemin. Si celui-ci se sépare face à l'obstacle, les fourmis vont alors emprunter préférablement le chemin sur lequel la phéromone sera la plus forte. Mais la phéromone étant une odeur elle s'évapore. Si peu de fourmis empruntent une trace, il est possible que ce chemin ne soit plus valable au bout d'un moment, il en est de même si des fourmis exploratrices empruntent un chemin plus long. Par contre, si le chemin est fortement emprunté, chaque nouvelle fourmi qui passe redépose un peu de phéromone et renforce ainsi la trace, donnant alors à ce chemin une plus grande probabilité d'être emprunté.

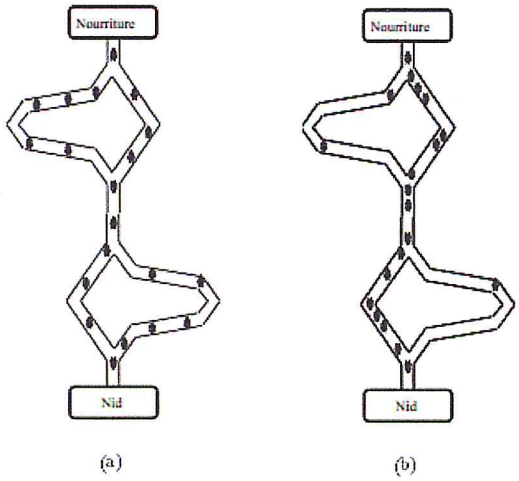

Figure 19: (a) Des fourmis explorant les chemins possibles. (b) Des fourmis suivant le chemin le plus court.

2.4.3.2.4 Recherche par Harmonies

La recherche par harmonies (HS : Harmony Search) est une très récente métaheuristique. Elle a été proposée par Geem et ses collègues (Geem et al, 2001) (Geem et Choi, 2007). A l'opposé des autres métaheuristiques qui s'inspirent des phénomènes naturels, la recherche par harmonies est basée sur le processus de performance musical qui consiste à trouver l'harmonie parfaite dans un orchestre musical où chaque musicien joue une note (improvise) pour trouver une meilleure harmonie.

Les étapes du processus de recherche de l'algorithme de recherche par harmonies sont résumées dans l'algorithme (1) et expliquées dans ce qui suit.

L'algorithme HS commence par une étape d'initialisation des paramètres nécessaires et de la mémoire d'harmonies (population de solution) composée d'un ensemble de 1 à HMS harmonies (i.e. solutions) aléatoires (**Figure 20**) et des paramètres nécessaires pour le fonctionnement du HS qui sont:

- La taille de la mémoire d'harmonies (i.e. la population), notée par HMS (de l'anglais : Harmony Memory Size).
- Le taux de considération de la mémoire harmonique, noté par HMCR (de l'anglais: Harmony Memory Considering Rate), dont le rôle est de décider si la mémoire HM sera utilisée ou non.
- Le paramètre PAR (de l'anglais: Pitch Adjusting Rate), représentant la probabilité d'apporter quelques modifications à un élément de la HM.

- Le critère d'arrêt (généralement un nombre maximum d'itérations).

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_N^1 \\ x_1^2 & x_2^2 & \dots & x_N^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & \dots & x_N^{HMS} \end{bmatrix}$$

Figure 20: Structure d'une population d'harmonies.

Ensuite, l'algorithme passe à l'étape d'amélioration des harmonies. Cette étape consiste à améliorer une solution en se basant sur trois règles: la considération de la mémoire HM, l'ajustement des valeurs des variables de la solution et la sélection aléatoire.

Après génération du nouveau vecteur (nouvelle solution), les composantes obtenues par considération de la mémoire HM sont examinées pour décider s'ils devront être ajustées ou non. Suit à l'étape d'amélioration de solutions, le HS passe à l'étape de mise à jour de la mémoire HM. Cette étape consiste à remplacer la mauvaise solution de la matrice HM par la nouvelle solution trouvée si cette dernière est de meilleure qualité (comparée avec la mauvaise solution). Les étapes d'amélioration et de mise à jour seront répétées jusqu'à la satisfaction du critère d'arrêt.

Algorithme 1 : L'algorithme général de la recherche par harmonies

Début
Initialiser les paramètres nécessaires ;
Initialiser la mémoire HM (une population d'harmonies) ;
Tant que (La condition d'arrêt n'est pas satisfaite) **faire**
Produire une nouvelle solution en améliorant la solution x_i ;
Mettre à jour la mémoire HM ;
Fin tant que
Retourner la meilleure solution ;
Fin

2.4.3.2.5 L'algorithme de recherche gravitationnelle (GSA)

En 2009, Esmat Rashedi et Hossein Nezamabadi (Rashedi, et al., 2009) ont proposé une nouvelle métaheuristique stochastique à base de population, appelée algorithme de recherche gravitationnelle (GSA : Gravitational Search Algorithm). Elle est motivée par les lois Newtoniennes de *gravitation* et de *mouvement.* Comme la plupart des métaheuristiques, elle a un mécanisme souple et bien équilibré pour améliorer les capacités de l'exploration (l'intensification) et d'exploitation (la diversification).

Le principe de base de GSA classique est très simple. Dans l'algorithme proposé, une solution est considérée comme un objet et sa performance est mesurée par sa masse. Tous les objets s'attirent entre eux avec une force gravitationnelle, cette force cause un mouvement global de tous les objets vers l'objet ayant la plus grande masse. Les objets ayant les plus grandes masses correspondent aux meilleures solutions, ces derniers se déplacent plus lentement que les plus légers, ce qui garantit l'étape de l'exploitation de l'algorithme.

Dans GSA, chaque objet a deux caractéristiques :

- Sa position qui représente une solution du problème.
- Sa masse qui correspond à la performance de la solution (calculée à partir de la fonction de fitness).

Loi de Gravitation : Deux corps ponctuels de masses respectives M_A et M_B s'attirent avec des forces de mêmes valeurs (mais vectoriellement opposées), proportionnelles à chacune des masses, et inversement proportionnelle au carré de la distance qui les sépare.

$$F_{A/B} = F_{B/A} = G\frac{M_A M_B}{d^2}$$

Loi de mouvement : La vitesse d'un objet est gale à la somme de sa vitesse précédente et de l'accélération. L'accélération est définie par la variation de la vitesse par rapport au temps (quand la vitesse est constante, l'accélération est nulle).

$$A = \frac{F}{M}$$

À titre d'exemple, considérons un espace bidimensionnel contant les objets O_1,O_2, O_3, et O_4. Comme on le voit dans la **figure 21**, F_{1j} (j {2,3,4}) est la force agissant sur O_1 par O_j (j

{2,3,4}), et F_1 est la force globale qui agissent sur O_1 par tous d'autres objets et génèrent l'accélération a_1 basé sur la seconde loi de Newton.

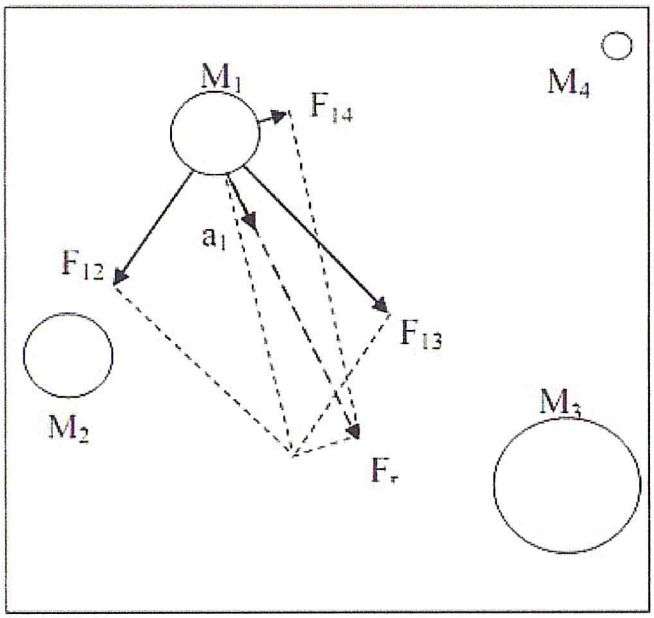

Figure 21: Chaque objet accélère vers la direction de la force résultante des autres objets qui agissent sur lui.

Le pseudocode de GSA original est indiqué dans l'algorithme (2).

Algorithme 2 : pseudocode de GSA original

Début
Génération de la population initiale ;
Evaluation de chaque agent ;
Tant que (La condition d'arrêt n'est pas vérifiée) **faire**
Mise à jour de G(t), Best(t), Worst(t) et M_i(t) tel que i = 1, 2, …, N;
Calcul de la force totale dans des directions différentes ;
Calcul de l'accélération et de la vitesse ;
Mise à jour de la Position des agents ;
Fin tant que
Retourner la meilleure solution ;
Fin

2.5 UN ÉTAT DE L'ART SUR LES MÉTHODES D'OPTIMISATIONS COMBINATOIRES APPLIQUÉES POUR RÉSOUDRE LE PROBLÈME D'ORDONNANCEMENT D'ATELIERS JOB SHOP

Les problèmes d'ordonnancement d'ateliers ont été très étudiés dans la littérature depuis plus de 50 ans. Ces problèmes sont de type NP-Difficile. Ainsi, plusieurs recherches ont été poussées pour la résolution de ce type de problème. Un état de l'art de quelque unes de ces recherches est présenté dans ce qui suit.

Cet état de l'art permet néanmoins de faire un tour d'horizon assez large des approches de résolution développées et permet de dégager les méthodes actuellement les plus performantes pour la résolution des problèmes d'ordonnancement en général et des problèmes d'ateliers de type Job Shop Flexible en particulier.

Plusieurs méthodes exactes ont été développées pour résoudre ce type de problème mono- et/ou multicritères. Ces approches ont été proposées pour la résolution des instances de petite taille ou pour des cas particuliers. A titre d'exemple, nous citons :

- Le travail de (Jurisch , 1992) qui a utilisé une procédure par séparation et évaluation (Branch & Bound) ainsi que des heuristiques pour les problèmes d'ordonnancement d'atelier avec machines multi-usages (plus connue sous le nom « shop scheduling problems with multi-purpose machine »). Les ressources considérées ont une durée identique quelle que soit la ressource. L'auteur a également proposé des bornes inférieures pour le problème.

- Le travail du Szwarc en 1960 (Szwarc, 1960), qui a présenté une méthode de programmation dynamique pour déterminer un ordonnancement optimal. Des études théoriques sur la base de cette approche graphique ont été également réalisées en 1963 par HARDGRAVE & NEMHAUSER pour le problème général.

Par ailleurs, vu le temps important que prennent ces dernières pour résoudre les problèmes difficiles, des méthodes approchées ont été proposées pour remplacer les méthodes exactes. Parmi ces méthodes nous citons :

- Dans (Dauzère-Pérès , et al., 1994), les auteurs se sont intéressés au problème de job shop flexible avec minimisation du makespan et ont proposé des conditions suffisantes permettant d'assurer que le déplacement des opérations n'augmente pas la valeur du makespan. Basée sur ces conditions, une méthode Tabou a été développée pour résoudre le problème du job shop

une fois l'affectation des machines aux opérations fixée. Les machines considérées sont identiques. Les résultats trouvés par leur méthode sont comparables à ceux présentés dans (Hurink , et al., 1994).

- Une autre méthode de recherche Tabou a été proposée dans (Barnes , et al., 1996). Les auteurs considèrent deux types de voisinages. Le premier consiste à permuter deux opérations critiques adjacentes et le deuxième consiste à réaffecter une opération critique sur toutes les machines pouvant l'exécuter. Ils ont considéré trois exemples de problème de type job shop classique FT10 de (Fisher, et al., 1963), LA24 et LA40 de (Lawrence, 1984) et ils ont fait une extension de ces derniers afin d'avoir des problèmes flexibles. Les machines considérées sont identiques.

- Dans (Mastrolilli et al., 2000), les auteurs proposent une autre approche intégrée basée sur une recherche Tabou. Les machines considérées sont identiques. Cette approche utilise le graphe disjonctif pour la représentation des solutions, ainsi que deux types de voisinages. Ces voisinages sont basés sur le déplacement d'une opération dans le graphe disjonctif. Tabou est parmi celles fournissant actuellement les meilleurs résultats sur les instances connues de job shop flexible.

- Un algorithme génétique a été également proposé pour résoudre le problème de job shop flexible dans (Kacem I, et al., 2002). Cet algorithme a été testé pour la résolution du problème mono et multicritère. L'objectif de cette méthode est d'optimiser conjointement deux critères classiques : le makespan et la charge des ressources. La première étape de cette approche permet de résoudre le problème d'affectation de ressources et de construire un schéma d'affectation. La deuxième étape est une approche évolutionnaire contrôlée par le modèle des affectations généré dans la première étape. Dans cette approche, les auteurs appliquent des manipulations génétiques avancées pour améliorer la qualité des solutions.

- Dans leur travail, (Guohui , et al., 2008) ont proposé un algorithme génétique pour la résolution du job shop flexible multicritère. L'objectif est de minimiser le makespan, la charge maximale et la somme des charges de ressources. La méthode a été évaluée via les instances utilisées dans (Kacem I, et al., 2002)et a donné de meilleures solutions.

- Fattahi et Fallahi en 2009, ont proposé un modèle mathématique pour le problème de job shop flexible et une hybridation de deux méthodes : une méthode de recherche tabou (TS) et un recuit simulé (SA). Sur la base de ces deux méthodes, les auteurs ont développé six algorithmes différents basés sur la combinaison des deux méthodes et des deux approches : intégrée et

hiérarchique. Les expérimentations ont permis de comparer les algorithmes entre eux. Il en résulte qu'en général, l'approche hiérarchique donne les meilleures solutions et ce en appliquant la méthode de recherche tabou pour le problème d'affectation et le recuit simulé pour le problème de séquencement.

- Le travail de (Gao, et al., 2007) s'intéresse au problème de job shop flexible multicritère avec trois objectifs : minimisation du makespan, minimisation de la charge maximale des machines et minimisation de la charge de travail totale. Les auteurs proposent ainsi une amélioration de l'algorithme génétique développé dans (Gao, et al., 2007). En effet, pour restreindre l'espace de recherche, les individus donnés par l'algorithme génétique sont améliorés par la méthode de recherche du voisinage par descente (VND), qui implique deux types de mouvements, le déplacement d'une seule opération ou de deux opérations. Déplacer une opération doit supprimer l'opération, lui trouver un autre intervalle de temps, et l'affecter pendant cette période de temps. Pour ce faire, ils ont développé une méthode efficace pour trouver ces intervalles. Afin d'éviter les optima locaux donnés par la première méthode (Gao, et al., 2007), ils ont proposé de faire le déplacement de deux opérations simultanément. Une étude expérimentale sur des benchmarks de la littérature montre l'efficacité de leur approche. En effet, l'Algorithme Génétique proposé est connu pour être un des meilleurs algorithmes évolutionnaire pour la résolution de ce type de problème (Job Shop Flexible) et comme recherche local en peut dire que la recherche tabou à prouver son efficacité dans le domaine de la résolution des problèmes combinatoire.

- En 2004, Boryczka a développé un algorithme sur la base des colonies de fourmis, l'efficacité de cet algorithme a été testée en le combinant avec quelques règles de priorités et le comparant avec d'autres métaheuristiques, cet algorithme a donné les meilleurs résultats concernant le Makespan. Kang et al (Kang, et al., 2007) ont proposé un système multi agents pour l'ordonnancement dynamique d'un FMS où les programmes d'ordonnancement sont générés à la base des colonies de fourmis. Geet al (Ge, et al., 2006), ont proposé un algorithme à base d'essaims de particules pour résoudre ce type de problème, dans cet algorithme un nouveau concept de distance et de vitesse a été proposé. En 2006, Xia et Wu ont développé un nouvel algorithme en combinant les essaims de particules avec le recuit simulé pour minimiser le makespan. Lei (Lei, 2008) a adapté cette métaheuristique pour la résolution des problèmes multi objectifs (minimisation du makespan et du retard total des jobs).

- Dans (Barzegar, Motamni & Bozorgi, 2011), les auteurs se sont intéressés au problème de job shop flexible avec minimisation du makespan. Ces derniers sont les premiers à utiliser l'algorithme de recherché gravitationnelle pour résoudre le problème job Shop Flexible. Les travaux (Barzegar, Motamni & Bozorgi, 2012) (Barzegar & Motamni ,2013) ont continué le travail commencé précédement.

En conclusion, le tableau suivant résumé les différents travaux décrits ci-dessus.

Tableau 2 : Principales méthodes de résolution de problème Job Shop.

Travaux	Problème d'ordonnacement		Méthode de resolution	
	Type de problème	Critère à optimiser	Type de la méthode	Nom de l'algorithme
(Jurisch , 1992)	Job shop	minimisation du makespan	Exacte	Branch & Bound
(Szwarc,1963)	Job Shop classique générale	Minimisation de makespan	Excate	Programmation dynamique
(Dauzère-Pérès & Paulli, 1994)	general job-shop	minimisation du makespan	Approchée	Recherche Tabou
(Hurink , Jurisch, & Thole , 1994)	general job-shop	minimisation du makspan	Approchée	Recherche Tabou
(Barnes & Chambers , 1996) (Fisher & Thompson, 1963) (Lawrence, 1984)	job shop classique	minimisation du makspan	Approchée	Recherche Tabou
(Mastrolilli & Gambardella , 2000)	job shop flexible	minimisation du makspan	Approchée	Recherche Tabou
(Kacem I, Hammadi, & Borne , 2002)	job shop flexible	le makespan et la charge des ressources	Approchée	Algorithme génétique
(Guohui , Yang, & Liang , 2008)	job shop flexible	minimiser le makespan	Approchée	Algorithme génétique
Fattahi & Fallahi, 2009	job shop flexible	problème de séquencement	Approchée	Recuit simulé et recherché Tabou
(Gao, Gen, Sun, & Zhao, 2007)	job shop flexible	minimisation du makespan, minimisation de la charge maximale des machines et minimisation de la charge de travail totale	Approchée	Algorithme génétique
(Boryczka, 2004)	general job-shop	minimisation du makespan	Approchée	Colonie de fourmie

(Kang, et al., 2007)	Système de production pharmaceutique	minimisation du makespan	Approchée	Colonie de fourmie
(Ge, et al, 2006)	Système de production pharmaceutique	minimisation du makespan	Approchée	Essaims de particules
(Xia et al., 2006) (Lei, 2008)	Job Shop flexible générale	minimisation du makespan	Approchée	Essaim particulaire et recuit simulé
(Berzegar et al., 2011 ; Berzegar et al, 2012 ; Berzegar et al.,2013)	Job Shob flexible générale	Minimisation de makespan	Approchée	Algorithme de recherche gravitationnelle

2.6 QUELLE MÉTHODE UTILISER POUR RÉSOUDRE LE PROBLÈME DE JOB SHOP FLEXIBLE ?

Comme constaté dans le tableau 2, beaucoup de travaux concernant l'ordonnancement d'un système de production flexibles ont été proposés pendant les trois dernières décennies. Bien que plusieurs approches analytiques, telles que la programmation mathématique, aient été utilisées pour résoudre des problèmes d'ordonnancement, leur incapacité de manipuler beaucoup de dispositifs réalistes et dynamiques d'un système de production flexible a toujours été un obstacle pour les applications industrielles. En outre, ces modèles ont besoin d'une certaine durée de calcul pour trouver une solution, les rendant peu convenables pour des problèmes multi critères.

Les métaheuristiques forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile, pour lesquels on ne connaît pas de méthode classique plus efficace. Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. Les problèmes d'ordonnancement dans les systèmes de production sont généralement de type NP difficile, à cause de leur complexité et l'absence de méthodes de résolution optimale, plusieurs ont pensés à utiliser ces techniques pour les résoudre et surmonter ces obstacles.

Les métaheuristiques constituent ainsi une classe de méthodes approchées adaptables à un grand nombre de problèmes de Job Shop Flexible. Et malgré leur grande efficacité, il existe très peu de résultats permettant de comprendre la raison de cette efficacité, et aucune méthode

particulière ne peut garantir qu'une métaheuristique sera plus efficace qu'une autre sur ce type de problème.

En effet, le premier obstacle pratique qui se pose à un développeur confronté à ce problème est d'effectuer un choix parmi les différentes métaheuristiques disponibles. Ce choix est d'autant plus difficile qu'il n'existe pas de comparaison systématique et fiable des différentes métaheuristiques. Cependant, il est possible de caractériser les métaheuristiques selon quelques critères généraux, de manière à faciliter ce choix.

Concrètement, certaines métaheuristiques présentent l'avantage d'être simples à mettre en œuvre, d'autres sont plutôt bien adaptées à la résolution de certaines classes de problème, très contraints, comme l'algorithme de colonies de fourmis.

La qualité des solutions trouvées par les métaheuristiques dépend de leur paramétrage (il faut éviter que les algorithmes ne convergent trop rapidement vers un optimum local), et de l'équilibre à trouver entre un balayage de tout l'espace des solutions (diversification) et une exploration locale poussée (l'intensification). Le choix d'une bonne représentation, d'un bon voisinage, sont également, des facteurs influençant grandement l'efficacité de la méthode choisie, qu'elle que soit.

Tout l'art du concepteur d'une heuristique résidera dans l'assemblage judicieux des différents principes que nous avons vus (intensification, diversification, hybridation…). La connaissance d'éléments de théorie connus en algorithmique, ou dans le domaine des méthodes dites exactes, permet d'améliorer les performances, selon les caractéristiques du problème à résoudre.

La métaheuristique que nous avons adoptée pour la résolution de notre problème d'ordonnancement Job Shop Flexible est l'algorithme de recherche gravitationnelle. Ce choix n'est nullement le fruit du hasard mais très bien réfléchi. Car c'est un algorithme très récent et qu'il existe très peu de travaux utilisant ce dernier pour résoudre les problèmes de type Job Shop flexible. Donc, il est nécessaire d'éffectuer plus de tests sur cet algorithme pour étudier son efficacité.

2.7 CONCLUSION

On a entamé ce deuxième chapitre par la présentation de la complexité algorithmique des problèmes d'ordonnancement. Ensuite, nous avons introduit les différentes méthodes d'optimisation combinatoires, commençant par les méthodes exactes jusqu'aux méthodes approchées, qui sont couramment utilisées pour la résolution des problèmes d'ordonnancement d'ateliers.

Les méthodes exactes permettent d'aboutir à la solution optimale, mais elles sont trop gourmandes en termes de temps de calcul et d'espace mémoire requis. Cependant, les méthodes approchées demandent des coûts de recherche raisonnables. Mais, elles ne garantissent pas l'optimalité de la solution. Elles peuvent être également partagées en deux classes : des méthodes heuristiques et des méthodes métaheuristiques. Une méthode heuristique est applicable sur un problème donné. Tandis qu'une méthode métaheuristique est plus générique et elle peut être appliquée sur une panoplie de problèmes d'optimisation.

En outre, les méthodes métaheuristiques peuvent être partagées en deux sous classes : des méthodes à base d'une solution unique et des méthodes à base de population de solutions. Les méthodes de la première sous classe (i.e. les méthodes à base d'une solution unique) se basent sur la recherche locale pour trouver la solution du problème à traiter. Elles sont souvent piégées par l'optimum local d'un voisinage donné. Par contre, les méthodes de la deuxième classe (i.e. les méthodes à base de population de solutions) se basent sur une recherche globale ce qui leur permet d'échapper au problème de convergence vers l'optimum local et augmente leur possibilité de fournir des solutions de bonnes qualités.

La troisième section du chapitre décrit brièvement les principales méthodes de résolution utilisées dans la littérature pour la résolution des problèmes d'ordonnancement de type Job Shop Flexible. Ensuite, un état de l'art concis de ces méthodes est réalisé dans la quatrième section. En fin, une synthèse de toutes méthodes vues au court de ce chapitre est effectuée.

Afin de résoudre notre problème Job Shop flexible, on a choisi l'algorithme de recherche gravitationnelle GSA qui fait partie des méthodes approchées. Après avoir présenté le principe général de cet algorithme au cours de ce chapitre, le chapitre suivant explicite en détails l'adaptation de ce dernier pour le problème d'ordonnancement d'ateliers de type Job Shop flexible.

CHAPITRE 3 : GSA POUR LA RÉSOLUTION DU PROBLÈME JOB SHOP FLEXIBLE

3.1 INTRODUCTION

Les problèmes d'ordonnancement des systèmes flexibles de production, sont parmi les problèmes d'optimisation les plus étudiés. Améliorer le rendement des ressources et minimiser les coûts de production sont devenus les leitmotivs que ce soit dans le milieu industriel ou dans le milieu académique. Chercher le meilleur moyen de maximiser son profit est aujourd'hui l'un des objectifs principaux de toute entreprise de production.

C'est dans ce contexte que s'inscrit ce travail de recherche. Il concerne la résolution de problèmes d'ordonnancement de type Job Shop flexible en utilisant l'algorithme de recherche gravitationnelle. Notre cas d'étude est une vraie cellule de montage : la cellule AIP-PRIMECA en simulation (basé sur une cellule existante à l'Université de Valenciennes en France). Du point de vue recherche opérationnelle, ce système peut être considéré comme Job Shop flexible. L'ordonnancement représente l'une des fonctions les plus importantes pour assurer la performance globale de ce système.

Dans un premier temps, une présentation du modèle générique des problèmes d'ordonnancement de type Job-Shop Flexible ainsi que sa formulation sont présentées. Après avoir présenté l'algorithme de base de GSA, on entame la présentation de notre solution GSA-FJSP-FlexSim pour l'ordonnancement d'ateliers de type Job Shop flexible. En fin, une instanciation statique du modèle générique de ce dit problème est proposée selon les paramètres de la cellule flexible de production AIP-PRIMECA.

Le but du travail présenté est la mise en œuvre effective de l'hybridation simulation/optimisation, en utilisant l'algorithme d'optimisation par recherche gravitationnelle pour résoudre le problème JSF dans la cellule AIP-PRIMECA.

3.2 MODÈLE GÉNÉRIQUE DU PROBLÈME D'ORDONNANCEMENT DE TYPE JOB SHOP FLEXIBLE

3.2.1 Présentation du modèle

Dans la littérature de la recherche opérationnelle, le problème Job Shop flexible (FJSP en anglais) est considéré comme une généralisation du problème de job shop classique (JSP). Les termes soulignés sont importants pour comprendre la formalisation de ce problème :

Soient *n* produits à traiter sur *m* machines différentes. Chaque produit *j* a sa propre séquence de production composée de quelques opérations de fabrication élémentaires. Ces opérations peuvent être exécutées sur une ou plusieurs machines.

La principale différence entre FJSP et JSP réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations dans les ateliers Job Shop flexible et ce n'est pas le cas dans les ateliers Job Shop classique.

L'affectation des opérations aux machines n'est pas fixée a priori comme dans le problème de job shop classique. Pour cette raison, de nombreux travaux ont utilisé des méthodes à deux phases pour faire face à la FJSP. Le problème d'affectation est résolu dans la première étape, tandis que la seconde étape a pour but de résoudre le séquencement des produits sur des machines affectées.

La plupart des problèmes Job Shop flexibles sont avérés être NP-difficile (Conway et al. 1967). La flexibilité augmente la complexité du problème, car elle nécessite un niveau supplémentaire de décisions (par exemple, la sélection des machines sur lesquelles le Job doit être procédé) (Brandimarte, 1993).

C'est pourquoi ce problème est généralement résolu en relaxant des contraintes (comme la capacité des ressources qui est considérée illimitée), en négligeant les temps de transport entre ressources, en ne prenant pas en compte les perturbations ou encore, la résolution est envisagée en cherchant une solution approchée plutôt que la solution optimale. Le travail de ce mémoire s'inscrit dans un contexte où le modèle à résoudre se doit d'être le plus proche possible du système physique afin d'obtenir des résultats de référence fortement pertinents. C'est pourquoi le choix a été fait de considérer un maximum de contraintes malgré le risque d'accroître la complexité du modèle résultant (nous montrerons cependant que notre approche permet de limiter cet accroissement).

3.2.2 Formalisation du problème

Dans cette partie, on donne une formalisation linéaire du problème définit ci-dessus.

On suppose qu'on dispose, au début de l'ordonnancement, de toutes les ressources et en quantités suffisantes, que tous les jobs sont disponibles, que les files d'attente des machines sont de capacités limités et les temps de transport sont pris en compte.

Les problèmes d'ateliers job-shop flexibles peuvent être formulés comme suit (Saad et al., 2007) :

- On considère un ensemble de produits (jobs). Ces produits doivent êtres ordonnancés sur des ressources (machines) disjonctives.
- Chaque machine ne peut réaliser qu'une seule opération à la fois.
- Chaque produit est composé de plusieurs opérations liées entre elles par des contraintes de précédence.
- Les gammes d'opérations (autrement dit séquences d'opérations) sont linéaires c'est-à-dire que chaque opération de la gamme a un seul successeur et que chaque produit n'a qu'une seul opération de fin.
- Chaque opération peut être exécutée sur une ou plusieurs machines.
- L'assignation d'une opération à une machine entraîne l'occupation de cette machine durant tout le temps d'exécution de l'opération.
- Une opération en cours d'exécution ne peut pas être interrompue (La préemption n'est pas autorisée).

Le FJSP présente deux difficultés principales :

- La première est relative à l'assignation de chaque opération à une machine,
- La seconde correspond au calcul des temps de début et des temps de fin des opérations.

Afin de bien formuler le FJSP, nous introduisons quelques paramètres, variables et contraintes.

3.2.2.1 Notations de paramètres

- P : ensemble des produits P = {1,2,…, n}.
- M : ensemble de machines M = {1,2,…, m}.
- R : ensemble de règles d'ordonnancement.
- R_{pj} : ensemble de règles d'ordonnancement sur le produit j.

- R_{mr} : ensemble de règles d'ordonnancement sur la machine r.
- I_j : ensemble d'opérations sur le produit j $I_j = \{1,2,\ldots, |I_j|\}$, j ∈ P.
- O_{ij} : opération i du produit j.
- M_{ij} : ensemble de machines pouvant exécuter l'opération O_{ij}, $M_{ij} \in M$.
- P_{ij} : temps de traitement de l'opération i du produit j, i∈ I_j.
- T_{r1r2} : temps de transport d'une machine r1 à une machine r2.
- MP : nombre maximale de produits pouvant être exécutés à la fois dans le système.
- F_r : capacité de la file d'attente d'une machine r.
- d_j : la date d'échéance du produit j, j = 1, . . ., n

➢ **Les règles d'ordonnancement**

Les règles d'ordonnancement correspondent aux mécanismes de décision utilisés par les produits, pour choisir les machines sur lesquelles ils vont être exécutés, et par les machines, pour ordonner l'exécution des produits. On trouvera alors de règles liées aux produits, et des règles liées aux machines.

Les règles d'affectation utilisées par les produits sont :

1- La machine la moins chargée en nombre d'opérations.
2- La machine la moins chargée en temps : basée sur le calcul des temps de transport ainsi les temps d'exécution sur les machines.
3- La machine la plus proche : basée sur le calcul des temps de transport entre les machines.
4- SPT (Smallest Processing Time) : basée sur le temps d'exécution le plus court sur les machines.
5- LPT (longest Processing Time) : basée sur le temps d'exécution le plus long sur les machines.
6- Affectation aléatoire (Random).

Les règles de priorités utilisées par les machines sont :

1- SPT (Smallest Processing Time).
2- LPT (longest Processing Time).
3- EDD (date de livraison la plus proche).
4- La pièce la moins chargée en nombre d'opérations restantes.
5- Random.

6- FIFO (First In First Out).

7- LIFO (Last In First Out).

3.2.2.2 Notations des variables

- t_{ij} : temps d'achèvement de l'opération O_{ij} (i$\in I_j$), $t_{ij} \in N$.
- μ_{ijr} : une variable binaire mise à 1 si l'opération O_{ij} est effectuée sur la machine r ; 0, autrement.
- b_{ijkl} : une variable binaire mise à 1 si l'opération O_{ij} est effectuée avant l'opération O_{kl}; 0, autrement.
- tr_{ijr1r2} : une variable binaire mise à 1 si le produit j est transporté à la machine r2 après avoir effectué l'opération O_{ij}; 0, autrement.
- w_{ijr} : temps d'attente de l'opération O_{ij} dans la file d'attente de la machine r.
- wv_{ijklr} : une variable binaire mise à 1 si l'opération O_{ij} est en attente pour l'opération O_{kl} dans la file d'attente de la machine r; 0, autrement.
- z_{lj} : une variable binaire mise à 1 si le produit l et le produit j s'exécutent en même temps dans le système ; 0, autrement.

Toutes les variables sont supposées être positives ou nulles. Cette formalisation génère un grand nombre de variables binaires qui rendent difficile la résolution de ce problème. Cependant, les bornes inférieures obtenues en limitant le temps de calcul sont très utiles pour juger de la qualité des méthodes approchées proposées pour la résolution de ce problème dans la suite du chapitre.

3.2.2.3 Les contraintes

On recense quatre ensembles de contraintes :

- **Contraintes disjonctives** : Une machine peut traiter une opération à la fois, et une opération est effectuée par une seule machine.
- **Contraintes de précédence** : Ces contraintes assurent le séquencement de production d'un produit. La date d'achèvement de l'opération suivante considère le temps d'achèvement de la précédente, le temps d'attente et le temps de transport si les deux opérations ne sont pas effectuées dans la même machine.

- **Capacité de la file d'attente de machines et la politique FIFO** : Chaque machine a une capacité de file d'attente limitée. Pas d'opérations plus que cette capacité peuvent attendre dans la file d'attente. La premier produit arrivé à la file est le premier traité (FIFO).
- **Limitation de nombre de produits pouvant être exécutés à la fois sur le système** : Le nombre de navettes qui transportent les produits est limité et donc, à chaque instant, un nombre (à fixer) maximum de produits peut être en cours d'assemblage sur le système.

3.2.2.4 Critères d'optimisation

Différents critères sont utilisés pour l'optimisation des systèmes flexibles de production. Ces critères sont cités, en fonction des variables et des paramètres présentés ci-dessus.

- **Le makespan** : Le critère à minimiser est la date de fin de la dernière opération du dernier produit dans le système. En général, le makespan est noté C_{max}. Sa valeur est calculée par cette formule :

$$C_{max} = max_{\forall i \epsilon I_j \forall j \epsilon P}\, t_{ij} \quad (1)$$

- **Temps d'écoulement** : c'est le temps passe le produit dans l'atelier, égale à la somme du temps de traitement sur chaque machine, y compris le temps de transport et le temps d'attente dans les files d'attente. Soit C_j le temps de l'achèvement de la dernière opération du produit j. Le temps d'écoulement du produit j est donc C_j. Dans ce cas, l'objectif à minimiser est le temps de l'achèvement total :

$$\sum_{\forall j \epsilon P} C_j \quad (2)$$

- **Précocité et retards des produits** : c'est la comparaison du temps de réalisation effective de l'emploi avec le temps de réalisation désiré.
- **Utilisation des machines** : dépend de l'atelier plutôt que les produits. Il est une fraction de la capacité de la machine utilisée dans la production. L'utilisation moyenne de m machines et de n produits n est proportionnelle à la durée d'écoulement maximale, telle qu'exprime cette formule :

$$U = \frac{\sum_{\forall i \epsilon I_j \forall j \epsilon P} P_{ij}}{m.C_{max}} \quad (3)$$

Ce critère est plutôt un indicateur de la performance comportementale qu'un indicateur de performance de la production, mais il est utilisé lorsque l'analyse des goulots d'étranglement est effectuée.

- **les travaux en cours** : c'est le temps que prennent les produits dans la file d'attente d'une machine. L'objectif est de minimiser le temps total d'attente des entrées dans la machine W.

$$w = \sum_{\forall r \in M} \sum_{\forall j \in P} \sum_{\forall i \in I_j} w_{ijr} \quad (4)$$

- **Optimisation multi-objectifs** : Les différents critères cités ci-dessus peuvent être métissés pour optimiser plus d'un objectif. Dans la littérature, de nombreux documents utilisés optimisation multi-objectifs pour FJSP (Taboun et Ulger, 1992 ; Ho et Tay, 2008 ; Azardoost et Imanipour, 2011), mais à notre connaissance, personne n'a pris en compte des contraintes supplémentaires présentées dans ce document.

3.3 PRÉSENTATION DE L'ALGORITHME GSA ORIGINALE

La méthode d'optimisation par recherche gravitationnelle fait partie des méthodes de recherche globale (à base de population) dont le système est initialisé avec une population, composée de solutions aléatoirement générées, et pour laquelle la recherche de la solution optimale se fait par une mise à jour de cette population.

Comme déjà mentionné dans le chapitre précédent, l'algorithme GSA est basé sur les lois de gravitation et de mouvement. Chaque solution potentielle, appelée encore objet ou agent, possède une vitesse ajustée dynamiquement, relativement à la somme des forces appliquées sur cette même solution par les autres objets. Grâce à cette vitesse, l'objet change de position et donc on obtient une nouvelle solution.

L'algorithme GSA a été initialement introduit pour traiter des problèmes d'optimisation dans le domaine continu (Rashedi, et al., 2009), alors que beaucoup de problèmes d'optimisation se situent dans l'espace. Il s'avère donc intéressant de recourir à des modifications sur l'algorithme initial pour pouvoir répondre aux besoins spécifiques aux problèmes d'ordonnancement de type Job Shop flexible (Berzegar et al., 2011 ; Berzegar et al, 2012 ; Berzegar et al., 2013).

Le principe de base de GSA classique est très simple. Dans l'algorithme proposé, une solution est considérée comme un objet et sa performance est mesurée par sa masse. Tous les objets s'attirent entre eux avec une force gravitationnelle, cette force cause un mouvement global de tous les objets vers l'objet ayant la plus grande masse. Les objets ayant les plus grandes masses correspondent aux meilleures solutions, ces derniers se déplacent plus lentement que les plus légers, ce qui garantit l'étape de l'exploitation de l'algorithme.

Dans GSA, chaque objet a deux caractéristiques :

- Sa position qui représente une solution du problème.
- Sa masse qui correspond à la performance de la solution (calculée à partir de la fonction de fitness).

Les notations relatives à ces caractéristiques sont les suivantes :

Envisageons un espace de recherche à N dimensions avec S objets dans lequel la position de l'*i ème* objet est définie comme suit :

$$X_i = (x_i^1, \ldots, x_i^d, \ldots . x_i^N); \text{ i=1, 2, ..., S} \quad (5)$$

Tel que x_i^d représente la position de l'*i ème* agent dans la *d ème* dimension. Basé sur (Rashedi, et al., 2009), la masse gravitationnelle de l' *i ème* agent est calculée après évaluation des fitness de la population courante comme suit:

$$m_i(\text{t}) = \frac{fit_i(t) - Worst(t)}{Best(t) - Worst(t)} \quad (6)$$

$$M_i(\text{t}) = \frac{m_i(\text{t})}{\sum_{j=0}^{N} m_j(\text{t})} \quad (7)$$

Telle que $M_i(\text{t})$ et $fit_i(t)$ représentent la masse gravitationnelle et la valeur de fitness de l'agent i au moment t, respectivement, *Worst (t)* et *Best(t)* sont définis comme suit pour un problème de minimisation :

$$\text{Best(t)} = min\ fit_j(t) \quad (8)$$

$$Worst\ (t) = max\ fit_j(t) \quad (9)$$

Les objets ayant les plus grandes masses attirent les plus légers (**loi de gravitation**). Ces derniers se déplacent selon la loi de mouvement décrite par la formule suivante :

$$A_i^d(\text{t}) = \frac{F_i^d(\text{t})}{M_i(t)} \quad (10)$$

La force totale $F_i^d(\text{t})$ qui agit sur l'*i ème* objet à partir de tous les autres objets *j* est calculée en utilisant l'équation (10), puis cette dernière doit être devisée par la masse gravitationnelle de cet agent, soit $M_i(t)$ en utilisant l'équation (7) afin de calculer l'accélération.

$$F_i^d(\text{t}) = \sum_{j=1\ j\neq i}^{S} rand_j \text{G(t)} \frac{M_j(t)}{R_{ij}(t)+\varepsilon} (x_j^d(\text{t}) - x_i^d(\text{t})) \quad (11)$$

Certains des principaux paramètres de l'équation (11) sont décrits comme suit :

- $rand_j$est un nombre aléatoire uniformément distribuée sur l'intervalle [0,1],
- $R_{ij}(t)$est la distance euclidienne entre les deux agents[3] i et j dans un espace euclidien à N dimensions,
- ε est une très petite valeur utilisée afin d'échapper à la division par zéro dès que la distance euclidienne entre deux objets i et j est égal à zéro.
- G(t)est le coefficient de gravité qui aura une valeur initiale, Ginitial, et il sera réduit avec le temps vers une valeur finale par l'équation (8).

$$G(t) = G(t_0) * \left(\frac{t_0}{t}\right)^{\beta} \quad , \beta < 1 \tag{12}$$

Ensuite, la prochaine vitesse de l'*i ème* agent est calculée par sa vitesse courante ajoutée à son accélération par l'équation (13), et la position suivante de l'*i ème* agent peut être calculée en utilisant l'équation (14) :

$$V_i^d(t+1) = rand * V_i^d(t) + A_i^d(t) \tag{13}$$

$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1) \tag{14}$$

Tel que $rand$ est un nombre aléatoire uniformément distribué sur l'intervalle [0,1].

3.4 NOTRE SOLUTION GSA-FJSP-FLEXSIM

Les problèmes d'ordonnancement d'ateliers de type job-shop flexible sont connus dans la littérature comme étant les problèmes les plus difficiles à résoudre. La difficulté réside dans le choix de la meilleure métaheuristique pour leur résolution ainsi que pour la détermination des meilleurs ordonnancements en des temps raisonnables, les plus proches possibles de la solution optimale. Plusieurs auteurs se sont penchés sur la résolution des problèmes d'ordonnancement job-shop flexible. La méthode que nous avons proposée pour résoudre le FJSP est l'hybridation optimisation-simulation en utilisant l'algorithme de recherche gravitationnelle (GSA) couplé avec le simulateur FlexSim d'où vient le nom **GSA-FJSP-FlexSim**.

[3] Les termes objets, solutions, agents réfèrent à la même chose.

3.4.1 Présentation de la méthode

L'algorithme d'optimisation par recherche gravitationnelle étant défini initialement dans le cas continu, il s'avère évidemment nécessaire d'effectuer une conversion du cas continu vers le cas discret pour la résolution des problèmes d'ordonnancement. La première étape de cette conversion consiste à créer une structure de particules puis à proposer un algorithme présentant la marche à suivre pour une meilleure résolution des problèmes FJSP.

3.4.1.1 Formulation générale des problèmes d'ordonnancement FJSP par GSA

$X_i(t) = (x_i^1, \dots, x_i^d, \dots . x_i^N)$: Une solution (l'ordonnancement).

$M_i(t)$: La masse de la solution i à l'itération t.

G(t) : La gravité de l'espace de recherche.

$F_i(t)$: La force totale appliquée sur l'objet par tous les autres objets.

$A_i(t)$: L'accélération de l'objet.

$V_i(t)$: La vitesse de l'objet, grâce à cette vitesse l'objet peut se déplacer vers les meilleures solutions.

Toutes ces variables sont calculées en utilisant les formules de GSA de base (5) (7) (10) (11) (13) (14) sauf la gravité, la formule exponentielle ne donnait pas de bonne résultats, c'est pour cette raison qu'on a opté pour une formule linéaire.

$$G(t) = G0\ (1-(t/max\_it)) \qquad (15)$$

Exemple : Considérons un espace de 3 agents et 11 dimensions (***Figure 22***).

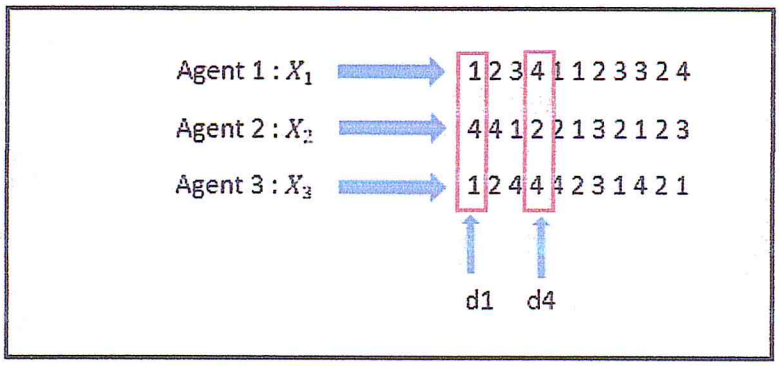

Figure 22: Exemple montrant la signification de la notion dimension.

3.4.1.2 Structure d'un objet (Modélisation d'une solution)

Présenter une structure de l'objet revient à présenter une combinaison des règles d'ordonnancement ; d'un ensemble de règles d'affectation R_p affectées à P produits avec un ensemble de règles de séquencement (priorité) R_m affectées à M machines (***Figure 23***).

➢ **Les règles d'ordonnancement**

Les règles d'affectation utilisées par les produits sont :

1- La machine la plus proche : basée sur le calcul des temps de transport entre les machines.
2- la machine la moins chargée en temps : le produit doit choisir une des machines qui est la plus proche et qui l'exécute au plus tôt.
3- SPT (Smallest Processing Time) : le produit doit choisir parmi les machines admissibles, celle qui a la plus courte durée d'exécution pour l'opération en cours.
4- LPT (Longest Processing Time) : le produit doit choisir parmi les machines admissibles, celle qui a la plus longue durée d'exécution pour l'opération en cours.
5- La machine ayant une file d'attente vide.

Les règles de priorité utilisées par les machines sont :

1- SPT (Smallest Processing Time) : La machine choisit le produit ayant l'opération qui a la plus courte durée d'exécution.
2- LPT (longest Processing Time) : La machine choisit le produit ayant l'opération qui a la plus longue durée d'exécution.
3- FIFO (First In First Out) : La machine choisit le premier produit dans sa file d'attente.
4- LIFO (Last In First Out) : La machine choisit le dernier produit dans sa file d'attente.

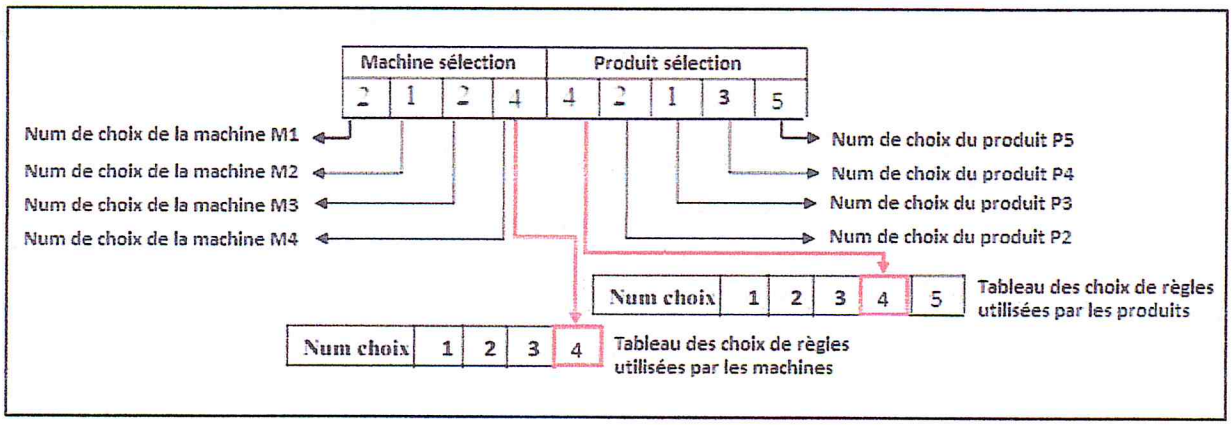

Figure 23: Exemple de la structure d'une solution (4 machines, 5 produits).

3.4.2 Etapes de l'algorithme proposé

Une population initiale, est générée aléatoirement dans une première étape, en affectant des règles d'ordonnancement aux produits et aux machines. En deuxième étape, une valeur de fitness est calculée pour chaque solution de la population. Des changements à effectuer à partir des opérations de l'algorithme qui permettent, en effet, de modifier les affectations des règles d'ordonnancement aux machines ainsi qu'aux produits en vue de rapprocher les solutions de l'optimum.

Ces étapes relatives à l'évolution de l'algorithme GSA, représentées dans la **figure 24**, sont reprises jusqu'à ce qu'un certain nombre d'itérations soit atteint.

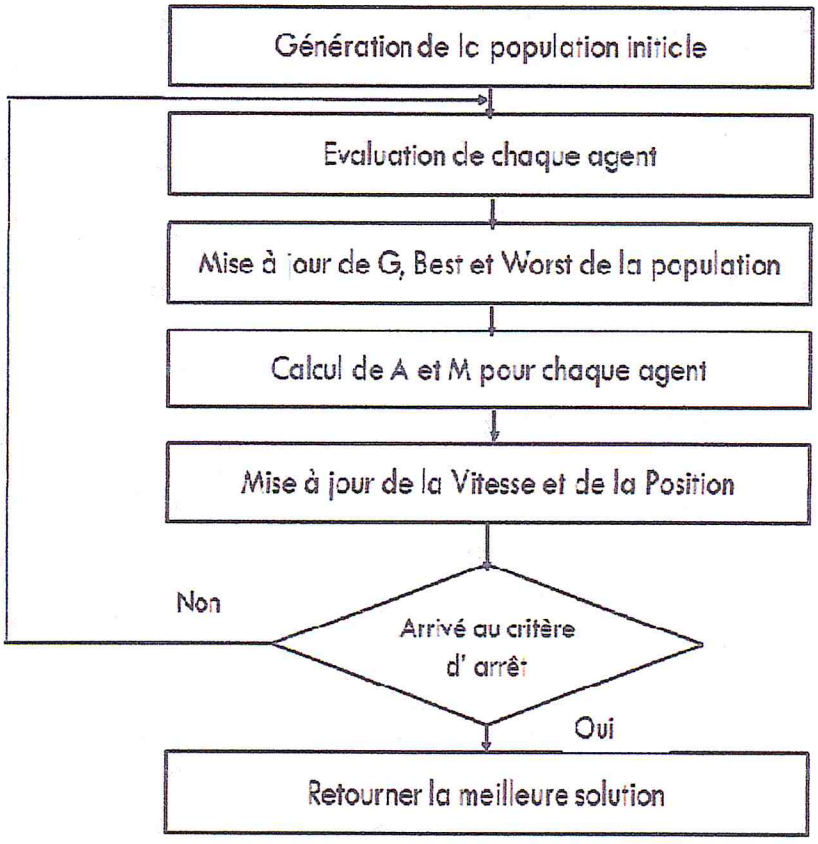

Figure 24: Principe général de GSA.

Les différentes étapes schématisées ci-dessus dans la **figure 24** sont expliquées en détail dans ce qui suit. Pour chaque objet (solution) de la population, ces étapes sont suivies et l'objet présentant la meilleure fonction objectif est considéré comme la solution optimale obtenue.

Le rôle principal de notre application est de lancer l'algorithme de recherche gravitationnelle qui donne en résultat une solution composée du meilleur jeu de règles d'ordonnancement destinées à être utilisés par les produits et par les machine lors de la conduite temps réel du système.

➢ **Les paramètres de dimensionnement de l'espace de recherche**

Les paramètres utilisés par l'algorithme sont :

- Taille de la population : N.
- Nombre de génération : *max_itération*.
- Nombre de produit : P.
- Nombre de machines : M.
- Nombre de règles sur les produits : R_p.
- Nombre de règles sur les machines : R_m.

3.4.2.1 Génération de la population initiale

La population initiale peut être obtenue en générant aléatoirement les chaînes de l'espace de recherche. Cette méthode à l'avantage de commencer la recherche à partir de diverses solutions possibles de l'espace de recherche, cela donne un point fort de plus à notre algorithme d'optimisation.

Une population qui contient N objets (solutions) est une matrice de taille $N * P+M$. Les membres aléatoires de la matrice ainsi générée sont des valeurs comprises entre 1 et le nombre de règle d'ordonnancement.

3.4.2.2 Evaluation des individus (hybridation optimisation-simulation)

La fonction d'adaptation « fitness », mesure la performance de chaque objet. Pour notre problème le seul critère utilisé pour l'évaluation des solutions est de minimiser le makespan. La plus petite valeur de cette fonction correspond à la meilleure solution.

Pour ce faire, on doit optimiser les règles pouvant être utilisées par les produits et par les machines en vue d'atteindre notre objectif (minimisation du temps de production). Ces règles sont des règles d'ordonnancement (allocation et priorité) qu'utilisent les produit et les ressources. Selon une règle d'allocation le produit choisiras la prochaine machine à visiter, la ressource utiliseras quant à elle une règle de priorité pour choisir quel produit à exécuter dans sa file d'attente.

L'optimisation des dites règles est basée sur l'hybridation simulation-optimisation. Le principe de cette hybridation est l'utilisation de la simulation pour l'évaluation des règles sur le système de production, l'algorithme d'optimisation est utilisé pour orchestrer l'optimisation de ces règles. En d'autres termes, selon ce couplage, la simulation est utilisée comme mécanisme d'évaluation au sein de l'algorithme d'optimisation. On présente dans ce qui suit les concepts liés à cette approche.

On utilise dans ce travail un simulateur à évènements discrets nommé FlexSim. La simulation à évènements discrets est ces dernières années de plus en plus souvent couplée à des techniques d'optimisation. La simulation à évènements discrets est utilisée pour modéliser le système en question aux contrôles et pour calculer le résultat de la stratégie de contrôle. Cette stratégie est donnée par un algorithme d'optimisation.

Ainsi, la simulation travaille finalement dans son cadre habituel, celui de test de scénarios (ici les solutions de l'algorithme), même si elle est incluse dans une démarche d'optimisation.

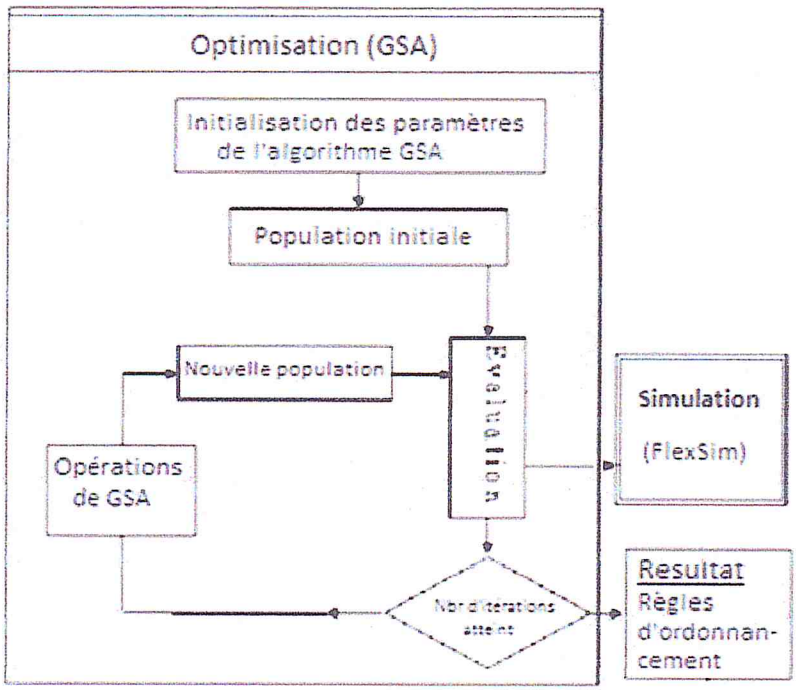

Figure 25 : L'hybridation simulation-optimisation.

➢ **Le noyau de simulation à événements discrets**

Le noyau de simulation est une composante principale de l'approche par simulation-optimisation. Il permet l'évaluation de l'effet de l'utilisation d'un ensemble donné de règles d'ordonnancement sur les performances du système de production, tout en intégrant des facteurs aléatoires.

Le noyau de simulation à événements discrets programmé, permet de simuler l'exécution dans le temps l'enchaînement des différents événements pouvant survenir dans le système et d'apprécier leurs conséquences. Il est basé sur une représentation objet des entités principales du système de production qui sont les produit, les opérations, les files d'attente, et les machines, et sur une gestion synchrone cadencé par une horloge des événements. Le temps est ainsi discrétisé en intervalles égaux. À chaque top d'horloge, le moteur détermine l'évolution du simulateur compte tenu des évènements qui se sont déroulés sur la dernière période et réinitialise l'état interne des différents objets. Les événements peuvent être un début d'exécution d'une opération sur une machine, la fin d'une opération, …etc. La procédure est effectuée à chaque top d'horloge (incrément du compteur de la boucle de simulation) jusqu'à épuisement des événements. Le résultat obtenu représente le makespan.

De ce fait, le calcul de la valeur objective d'un jeu de règle se fait par le déroulement de plusieurs simulations.

La simulation part d'une structure de données initialisée selon une forme défini qui représente les données reçus par le simulateur (les données présentent le jeu de règles encapsulé dans un socket et envoyé par l'algorithme d'optimisation).

On décrit la fonction d'évaluation qui intègre plusieurs déroulements de la simulation dans la figure suivante (**Figure 26**) :

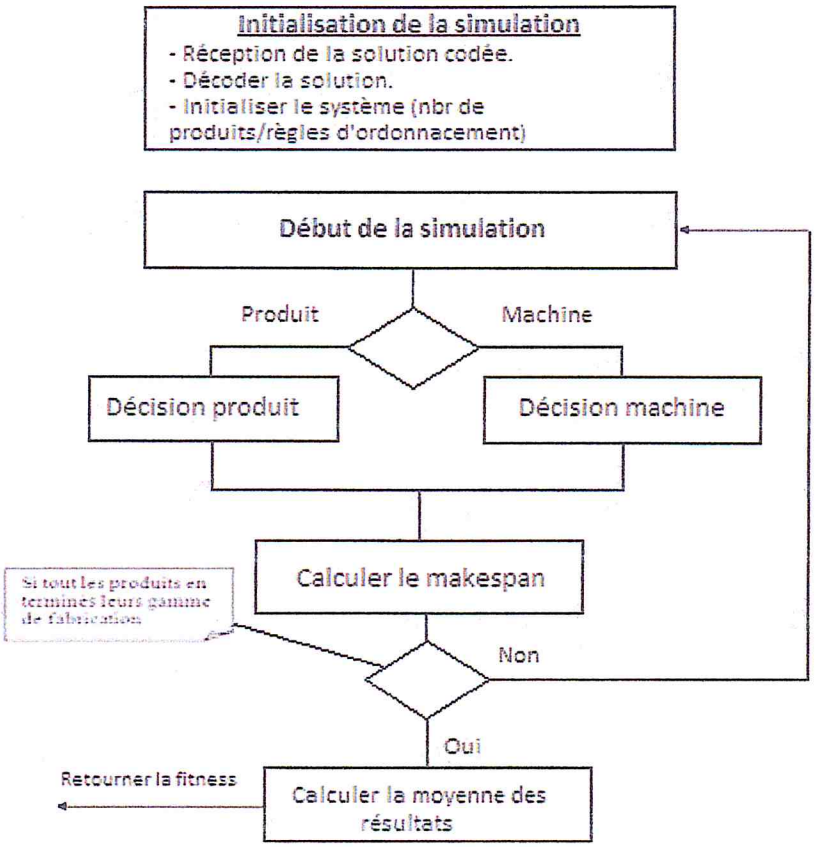

Figure 26 : Déroulement de la fonction d'évaluation.

3.4.2.3 Déplacement des objets dans l'espace de recherche

On calcule la nouvelle position d'un objet par l'équation (14) (voir section 3.3).

Exemple : Déplacement d'un objet pour 4 machines et 4 produits (**Figure 27**).

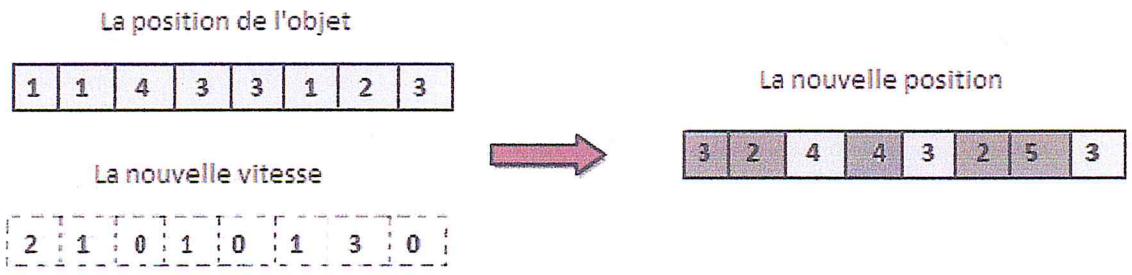

Figure 27 : Exemple du déplacement d'une solution à 4 machines et 4 produits.

3.5 ETUDE DE CAS : APPLICATION SUR LA CELLULE AIP-PRIMECA

La formalisation précédente (section 1.2) du FJSP est suffisamment générique pour envisager son application à notre cas d'étude qui est la cellule de production de l'AIP-PRIMECA de Valenciennes visible sur la **figure 28**.

Le choix de ce cas d'étude plus particulièrement a été imposé par notre organisme d'accueil (CDTA).

Une courte instanciation de ce modèle appliqué à la cellule AIP et les paramètres statiques pertinents sont donnés ci-dessous.

La cellule de l'AIP PRIMECA a été présentée et modélisée de manière formelle dans le benchmark proposé dans (Trentesaux et al. 2013). Ce benchmark propose aussi différents scénarios dynamiques permettant de tester la robustesse des approches de pilotage (ex. apparition d'une nouvelle ressource, problème de qualité). Les données présentées dans cette section reprennent les éléments clés de ce benchmark et sont présentées dans trois sous-sections: les données relatives aux produits, celles relatives aux ressources et celles relatives au système de transport.

Figure 28 : La cellule AIP-PRIMECA.

3.5.1 Données sur les produits à fabriquer

Les cinq composants de base disponibles dans la cellule (« Axe », « I », « L », « r » et « Vis ») ainsi que le plateau accueillant l'assemblage sont représentés sur la **figure 29-a**).

Les composants sont assemblés sur un plateau vierge déposé sur chaque navette (i.e. ressource de transport présenté figure 29- c) lors de l'opération de chargement. Ils sont répartis entre les différentes ressources réalisant des opérations d'assemblage (présentées dans la partie données sur les ressources) et leur stock est supposé suffisant.

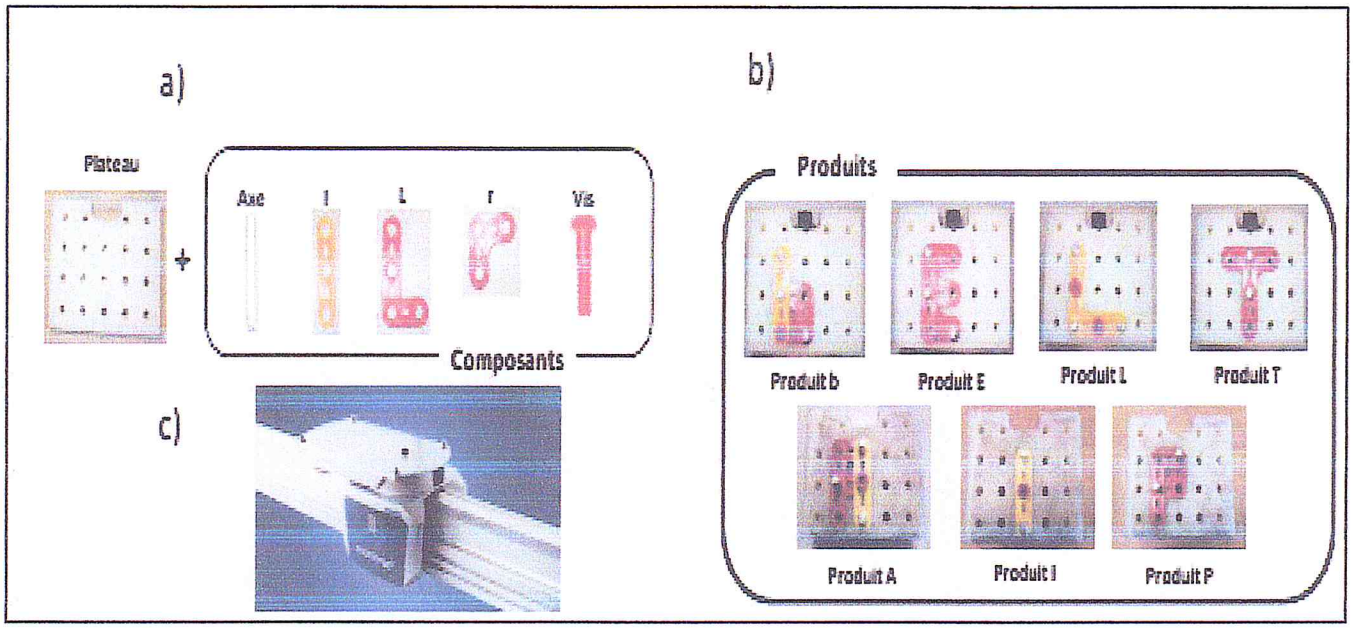

Figure 29 : a) Composants, b) Produits et c) Navette.

Avec ces cinq composants, sept produits différents peuvent être assemblés sur la cellule («B », « E », « L », « T », « A », « I » et « P ») comme illustré ***figure 29-b***). Les produits assemblés sont déchargés avec le plateau à la fin de la séquence de production.

La séquence de production de ces produits est composée d'actions élémentaires, appelées opérations, réalisées sur les ressources de la cellule. Ces opérations sont au nombre de huit («Chargement_Plateau », « Montage_Axe », « Montage_r », « Montage_I », « Montage_L», «Montage_Vis », « Inspection automatisée » et « Déchargement_Plateau »). Par exemple «Montage_Axe » signifie qu'un Axe est pris du stock de la ressource concernée et monté sur le plateau correspondant au produit en cours d'assemblage (l'emplacement du montage est déterminé par la ressource en fonction du produit). L'inspection automatisée est effectuée par un système de vision présenté par la suite.

Une séquence de production est associée à chaque type de produits. Elle est toujours structurée de la même façon : on charge le plateau vierge sur une navette, une série d'opérations d'assemblage est effectuée, puis le produit est inspecté et enfin le plateau

contenant le produit est déchargé. Si deux opérations consécutives d'une séquence de production ne sont pas réalisées sur la même ressource, un transport est alors nécessaire. Le ***tableau 3*** présente la séquence de production associée à chaque produit.

Tableau 3: Séquence de production des produits.

B	E	L	T	A	I	P
Chargement	Chargement	Chargement	Chargement	Chargement	Chargement	Chargement
Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe
Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe
Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe	Montage_Axe
Montage_r	Montage_r	Montage_l	Montage_L	Montage_r	Montage_Vis	Montage_L
Montage_r	Montage_r	Montage_l	Inspection	Montage_L	Inspection	Inspection
Montage_l	Montage_L	Montage_Vis	Déchargement	Montage_l	Déchargement	Déchargement
Montage_Vis	Inspection	Montage_Vis		Montage_Vis		
Inspection	Déchargement	Inspection		Inspection		
Déchargement		Déchargement		Déchargement		

Pour corriger d'éventuels problèmes de qualité sur certains produits, une opération supplémentaire appelée « réparation » peut-être, le cas échéant, insérée juste avant le déchargement. Cette opération est manuelle et corrige les problèmes de qualité avec un taux de réussite de 100%.

Dans les scénarios en fonctionnement normal, les ordres de fabrication provenant des clients peuvent être de trois formes : « BELT », « AIP » et « LATE ». Ces ordres de fabrication contiennent donc trois ou quatre produits qui peuvent être fabriqués dans n'importe quel ordre.

La figure suivante illustre les trois ordres de fabrication.

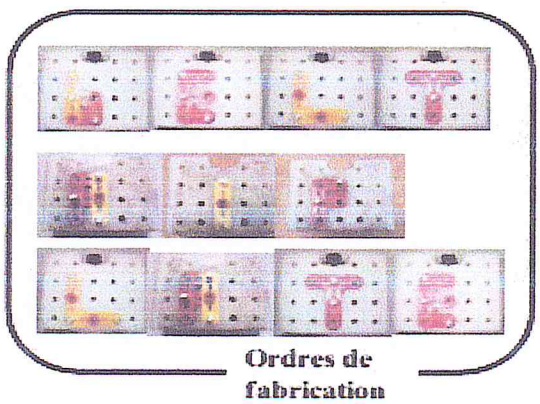

Figure 30 : Ordres de fabrication.

3.5.2 Données sur les ressources

La cellule est composée de sept ressources (voir **figure 31**) :

- Un poste de chargement / déchargement (R1)
- Trois postes d'assemblage (R2, R3 et R4)
- Une unité d'inspection automatisée (R5)
- Une unité de réparation (R6), seul poste manuel
- Une station optionnelle (R7), utilisée dans certains scénarios avec perturbation.

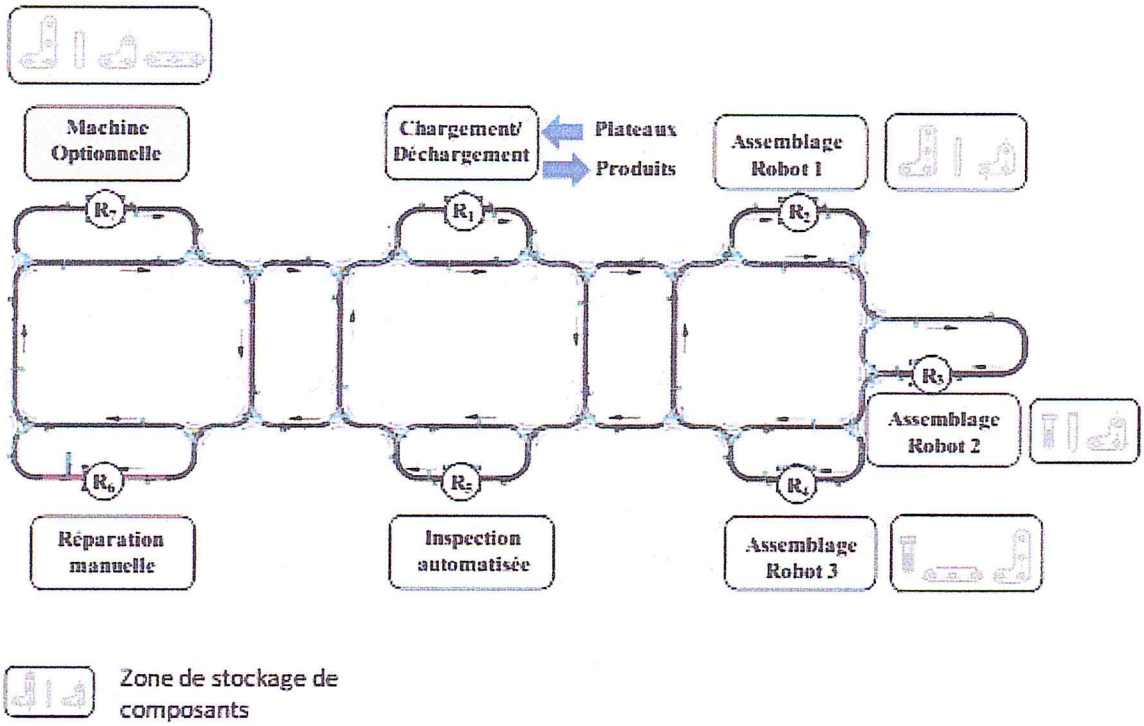

Figure 31 : Localisation des ressources de la cellule.

Chaque ressource est associée à sa propre zone de stockage de composants. Lors de scénarios du benchmark dits « dynamiques », il est possible de causer une perturbation sur une zone de stockage spécifique tout en laissant les autres zones fonctionnelles.

Le **tableau 4** récapitule l'affectation des opérations de production aux ressources. La valeur présentée indique le temps de production en secondes sur cette ressource pour l'opération considérée. Une absence de chiffre signifie que la ressource ne peut pas effectuer l'opération.

La ressource R7 n'est effective que dans certains scénarios dynamiques, lors de fonctionnements normaux elle n'effectue aucune opération. Chaque ressource ne peut traiter qu'un seul produit à la fois.

Tableau 4: Temps de production pour chaque ressource.

Opération	R_1	R_2	R_3	R_4	R_5	R_6	R_7
Chargement	10	-	-	-	-	-	-
Déchargement	10	-	-	-	-	-	-
Montage_Axe	-	20	20	-	-	-	(20)
Montage_r	-	20	20	-	-	-	(20)
Montage_l	-	-	-	20	-	-	(20)
Montage_L	-	20	-	20	-	-	(20)
Montage_Vis	-	-	20	20	-	-	-
Inspection	-	-	-	-	5	-	-
Réparation	-	-	-	-	-	60	-

3.5.3 Données sur le système de transport

Les ressources sont connectées grâce à un système de transport. Ce système de transport est un monorail unidirectionnel avec des aiguillages rotatifs avant chaque divergence/convergence (Montratec AG, 2013). Ainsi, ce système de transport peut être considéré comme un graphe fortement connexe, composé des nœuds suivants :

- R1, R2, R3, R4, R5, R6, R7(en blanc dans la **figure 32**) sont les nœuds ressources,

- n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, (en gris dans la **figure 32**) sont les nœuds décisionnels où des décisions de routage doivent être prises. Par exemple en n1, il faut décider d'aller vers n2 ou n10.

L'ensemble de ces nœuds est présenté sur la **figure 32**.

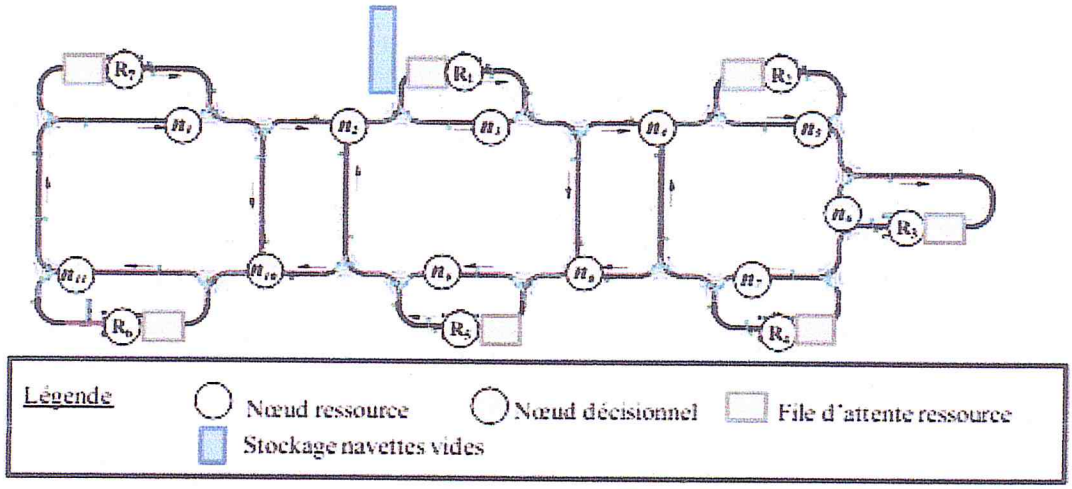

Figure 32. Le système de transport.

Les navettes parcourent le système de convoyage, sont autopropulsées et peuvent chacune transporter un produit de nœud en nœud. Le nombre de navettes est limité et donc, à chaque instant, un nombre (à fixer) maximum de produits peut être en cours d'assemblage sur la cellule. Chaque navette intègre un mécanisme de base permettant de ne pas entrer en collision avec les autres navettes, détecter les aiguillages et s'arrêter en face des ressources. Le convoyeur étant unidirectionnel la première navette qui entre sur un tronçon sera toujours la première à en sortir (règle First In First Out ou FIFO).

Pour des raisons de simplification, les navettes vides sont considérées comme stockées dans une zone dédiée, à côté de la ressource R1, et dont la capacité de stockage de navettes est considérée comme infinie (il est supposé qu'un mécanisme extérieur charge et décharge les navettes à cet endroit). Les navettes entrent et quittent ainsi le système sur cette zone. Elles commencent donc par accueillir un plateau (opération chargement sur R1) avant d'entrer dans le système. Elles transportent ensuite le produit à travers la cellule pour lui permettre d'être assemblé avant de revenir sur R1 afin qu'il soit déchargé. Elles sont ensuite stockées dans cette zone jusqu'à ce qu'elles soient réutilisées. Ainsi, il est impossible qu'une navette vide (sans plateau) circule dans la cellule.

Avant chaque ressource se trouve une file d'attente avec une capacité finie.

Les temps de transport théoriques entre chaque nœud de la cellule sont donnés dans le **tableau 5**. Ces temps sont minima et sont fixés par la vitesse réelle des navettes. En réalité, le transport peut prendre plus de temps à cause de ralentissements ou embouteillages sur la cellule.

Tableau 5: Temps de transport entre chaque nœud.

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	R_1	R_2	R_3	R_4	R_5	R_6	R_7
n1		4	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-
n2	-		4	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-
n3	-	-		4	-	-	-	5	-	-	-	-	-	-	-	-	-	-
n4	-	-	-		4	-	-	-	-	-	-	-	5	-	-	-	-	-
n5	-	-	-	-		3	-	-	-	-	-	-	-	11	-	-	-	-
n6	-	-	-	-	-		4	-	-	-	-	-	-	-	5	-	-	-
n7	-	-	-	5	-	-		4	-	-	-	-	-	-	-	-	-	-
n8	-	-	-	-	-	-	-		4	-	-	-	-	-	-	5	-	-
n9	-	5	-	-	-	-	-	-		4	-	-	-	-	-	-	-	-
n10	-	-	-	-	-	-	-	-	-		4	-	-	-	-	-	7	-
n11	9	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	10
R_1	-	-	-	6	-	-	-	7	-	-	-		-	-	-	-	-	-
R_2	-	-	-	-	-	5	-	-	-	-	-	-		13	-	-	-	-
R_3	-	-	-	-	-	-	6	-	-	-	-	-	-		7	-	-	-
R_4	-	-	-	7	-	-	-	6	-	-	-	-	-	-		-	-	-
R_5	-	7	-	-	-	-	-	-	-	6	-	-	-	-	-		-	-
R_6	12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		13
R_7	-	6	-	-	-	-	-	-	-	7	-	-	-	-	-	-	-	

3.5.4 Données de notre système

Pour la réalisation de notre système, seulement une partie des données décrites précédemment a été prise en compte.

Les deux tableaux suivants illustrent les paramètres pris en considération lors l'élaboration de notre système.

Tableau 6: Paramètres pris en considération lors la génération des solutions.

paramètre	Valeur
Nombre de produits	7 (B, E, L, T, A, I, P)
Nombre de machines	4 (R1, R2, R3, R4)
Règles produits	1 -> SPT ; 2 -> LPT 3 -> la machine moins chargée en temps 4 -> la machine la plus proche 5 -> la machine ayant une file d'attente vide
Règles machines	1 -> FIFO ; 2 -> LIFO ; 3 -> SPT ; 4 -> LPT

Tableau 7: Les paramètres pris en considérations par le simulateur

paramètre	Valeur
Nombre de produits	7 (B, E, L, T, A, I, P)
Opérations pour chaque produit	Voir tableau 3
Nombre de machines	4 (R1, R2, R3, R4)
Opérations pouvant être exécutées par chaque machine	Voir figure 32
Temps de production pour chaque machine	Voir tableau 4
Règles produits	1 -> SPT , 2 -> LPT 3 -> la machine moins chargée en temps 4 -> la machine la plus proche 5 -> la machine ayant une file d'attente vide
Règles machines	1 -> FIFO, 2 -> LIFO, 3 -> SPT, 4 -> LPT.

3.6 CONCLUSION

Nous avons abordé dans ce chapitre notre solution GSA-FJSP-FlexSim. Dans un premier temps nous avons présenté le modèle générique du problème d'ordonnancement de type Job Shop Flexible ainsi que l'algorithme de recherche gravitationnelle.

Dans un troisième temps, nous avons présenté notre solution GSA-FJSP-FlexSim comme méthode d'optimisation des problèmes d'ordonnancement Job Shop flexible en explicitant ses différentes étapes. C'est dans cette section que nous avons introduit le concept de simulation à événement discret et son couplage en ligne avec l'optimisation.

En dernier lieu, nous avons abordé notre cas d'étude : la cellule flexible AIP-PRIMECA de l'Université de Valenciennes en France, qui est une instanciation du modèle générique.

Le chapitre suivant présente la mise en œuvre de notre solution dans l'élaboration de notre système d'ordonnancement d'atelier de type Job Shop flexible.

CHAPITRE 4 : VALIDATION, TESTS ET EXPERIMENTATIONS

4.1 INTRODUCTION

Le développement d'une application nécessite plusieurs phases. L'analyse des besoins étant la première phase, permet une meilleure compréhension de ce qu'on doit faire pour une meilleure organisation lors de la phase réalisation.

L'objectif de ce chapitre est de présenter notre application de pilotage et de donner une synthèse des résultats obtenus par application de l'algorithme de recherche gravitationnelle au problème d'ordonnancement Job Shop Flexible.

De ce fait, nous allons commencer ce chapitre par l'analyse et spécification des besoins, puis nous allons entamer la partie implémentation de notre application où nous présentons l'architecture de l'application ainsi qu'un exemple de déroulement. L'objectif de la troisième partie est de données une synthèse des résultats obtenus par application de l'Algorithme de recherche gravitationnelle comparés avec celles obtenus par l'application de l'algorithme de recherche par harmonie dans le cas classique pour la résolution d'un ensemble de problèmes tests "benchmarks" et éventuellement les résultats obtenus en utilisant l'hybridation optimisation-simulation sur la cellule flexible AIP PRIMECA.

4.2 ANALYSE ET SPÉCIFICATION DES BESOINS

L'analyse et spécification des besoins sont des étapes très importantes. La finalité de cette étape est la description générale des fonctionnalités du système. Par la réponse à ces questions :

- Qu'attendons-nous du système ?
- Quelles sont les fonctions du système ?
- Quels sont les utilisateurs du système ?

4.2.1 Présentation des objectifs de notre application

L'application doit permettre dans un premier temps, le choix de type et de nombre de produit à usiner. Ensuite, elle permet à l'utilisateur de choisir la taille de la population à générer ainsi que le nombre maximal d'itérations avant de lancer l'optimisation. En résultat, elle

retourne la combinaison des jeux de règles qui donne un temps de production minimum (calculé par le simulateur) tout en respectant les contraintes réels du système.

4.2.2 Les fonctionnalités de l'application

- Choisir les produits à fabriquer.
  - A partir d'une liste limitée de produits.
- Préciser la taille de la population ainsi que le nombre max d'itérations.
- Lancer l'algorithme d'optimisation.
- Connecter avec le simulateur.
- Renvoyer la meilleure solution.

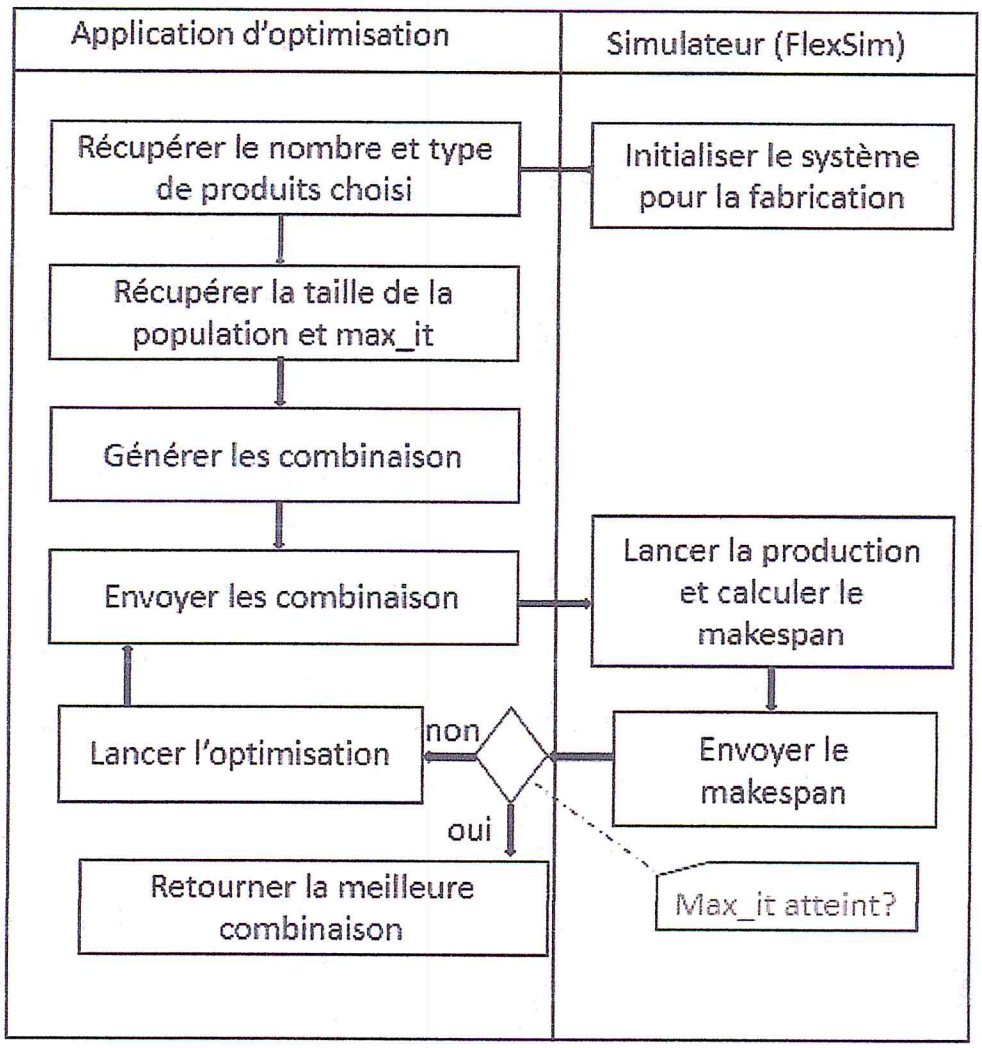

Figure 33: Schéma représentant les fonctionnalités de l'application.

4.2.3 Les acteurs

Un acteur est une entité externe qui agit sur le système (opérateur, autre système…) et qui peut consulter ou modifier l'état du système.

Dans notre cas il y a un seul utilisateur principal (l'utilisateur du système) qui peut être un informaticien ou non.

4.3 IMPLÉMENTATION

Dans cette partie on présentera les plus importantes fonctionnalités de notre application afin de tester si les objectifs fixés lors d'expression des besoins sont atteints.

4.3.1 Stratégie suivie

On a opté pour une architecture à trois niveaux ; ces trois niveaux sont décrits comme suit :

- Niveau présentation : Responsable de l'affichage des interfaces graphiques qui permettent l'interaction entre l'utilisateur et le système.
- Niveau d'application (métier) : S'occupe de l'ordonnancement par l'algorithme implémenté.
- Niveau de simulateur : Pour la simulation de l'exécution de la production.

La figure ci-contre illustre cette organisation :

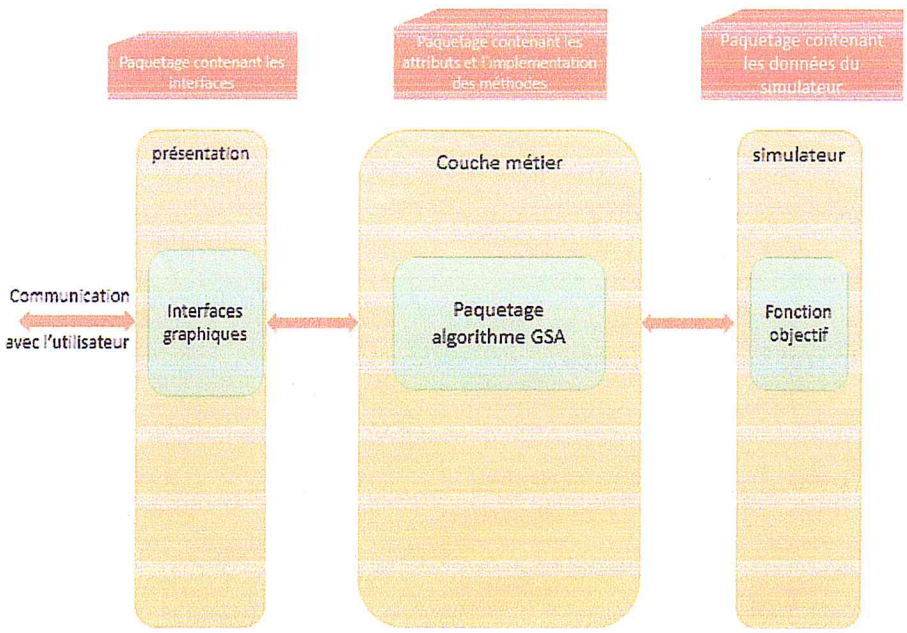

Figure 34: Schéma d'implémentation de l'application.

4.3.2 Présentation des outils de développement et langage

4.3.2.1 Langage de programmation

Pour la réalisation de notre projet on a utilisé le langage de programmation Java (JEE), sous l'éditeur NetBeans.

Figure 35: interface netneans 8.0.2.

Pour les raisons suivantes :

- Nous sommes bien familiarisées avec les notions du langage Java ;
- L'application peut s'exécuter sur n'importe quel système d'exploitation à condition d'avoir la machine virtuelle java installée sur la machine (portabilité) ;
- La disponibilité de la documentation et de l'assistance (forums).
- JAVA est un langage orienté objet qui met à la disposition du développeur plusieurs paquetages prêt à l'utilisation, nous citons parmi les principaux qu'on a utilisés :

  - ✓ La bibliothèque Swing : cette librairie offre un ensemble de composants, graphiques pour la construction des interfaces graphiques ; parmi ces composants on a utilisé : JFrame, JPanel, JDialog, JButton, JTabbedPane etc.…
  - ✓ Java.io: comme tout langage de programmation, java fournit cette bibliothèque de manipulation des flux de données que nous avons utilisée pour manipuler des fichiers Texte (création, lecture, écriture).
  - ✓ Java. util: nous avons utilisées cette bibliothèque pour manipuler les structures de données (les vecteurs) et certaines fonctions mathématiques (random (), sqrt (), abs () etc.…) ;
  - ✓ Java.net.Socket : cette librairie permet d'utiliser les sockets pour la communication avec le simulateur.

4.3.2.2 Outils de simulation

L'outil de simulation que nous avons utilisé pour la simulation du système qu'on travaille dessus est le logiciel FlexSim.

Figure 36: Interface FlexSim 7.5.0.

Le logiciel FlexSim est un outil d'analyse et de simulation de système manufacturier. Il permet, entre autres, de simuler plusieurs alternatives d'aménagement afin de laisser le concepteur choisir celle qui répond à ses besoins. Il s'avère très efficace en ce qui concerne l'optimisation des couts de projets avant leur réalisation, car grâce à la simulation avec FlexSim on peut estimer les dépenses dont on a besoin pour l'aboutissement d'un quelconque projet industriel, en simulant sa production par exemple, on peut aussi estimer le nombre de personnel et de machines de production nécessaires pour arriver à un chiffre de production précis.

On a choisi FlexSim comme outil de simulation, particulièrement pour ça flexibilité, car il offre une interface simple d'utilisation qui accélère la prise en main du logiciel et nous permet, notamment, de faire de la simulation à évènements discrets grâce aux objets discrets que comporte sa bibliothèque. En plus, le CDTA est le seul centre de recherche qui a acheté la licence au niveau national. Il est en adéquation avec notre objectif qui consiste à la simulation-optimisation, car il comporte des fonctionnalités qui permettent de communiquer avec d'autres composants tels que la communication avec Sockets.

4.3.3 Présentation de l'application

Voici les principales interfaces de notre application :

4.3.3.1 Interface d'accueil

Lors du lancement de l'application, l'interface d'accueil apparait (**Figure 37**).

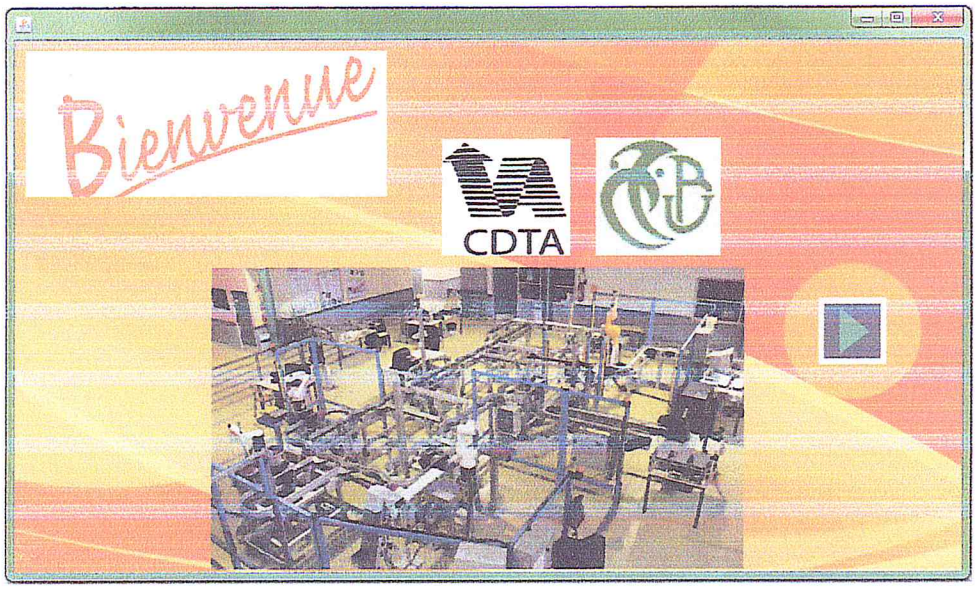

Figure 37: Interface d'accueil.

En appuyant sur le bouton Entrer, l'utilisateur passe à l'interface suivante (**Figure 38**).

4.3.3.2 Interface choix de type et nombre de produits

La figure ci-dessous (**Figure 38**) représente l'interface qui permet de choisir le type ainsi que le nombre de produits à fabriquer. Elle comporte une liste des produits que le système peut produire. Une fois cliqué sur le bouton OK un message portant le choix de l'utilisateur est envoyé au simulateur et l'interface suivante est affichée.

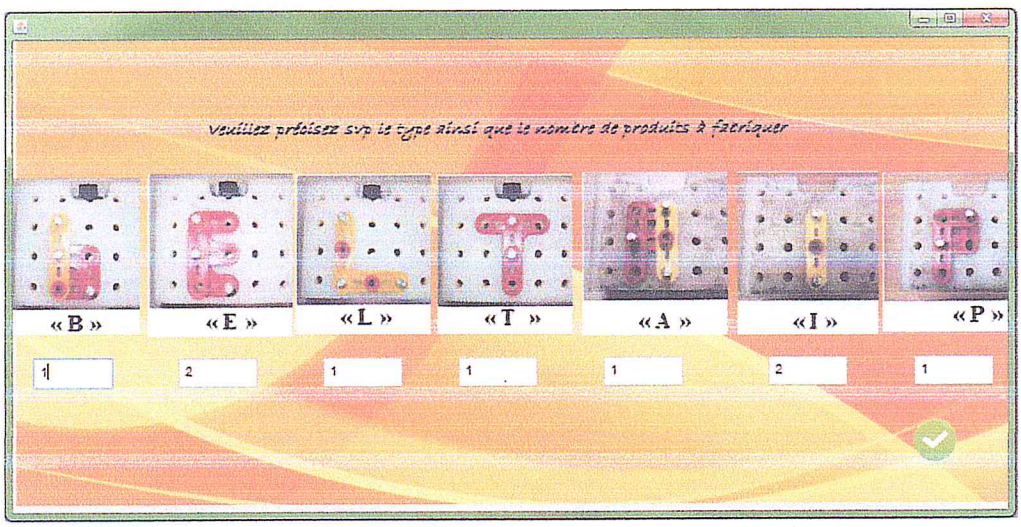

Figure 38: Interface choix type et nombre de produits à fabriquer.

4.3.3.3 *Interface d'optimisation*

Cette interface (**Figure 39**) permet à l'utilisateur de l'application de préciser la taille de la population à générer ainsi que le nombre d'itération de l'algorithme. Lorsque l'utilisateur clique sur le bouton OK, la première population est générer et une connexion avec le simulateur sera établie. Une fois la simulation d'exécution de toutes les solutions est terminée, l'algorithme d'optimisation sera lancé et retourne comme résultat la meilleure solution.

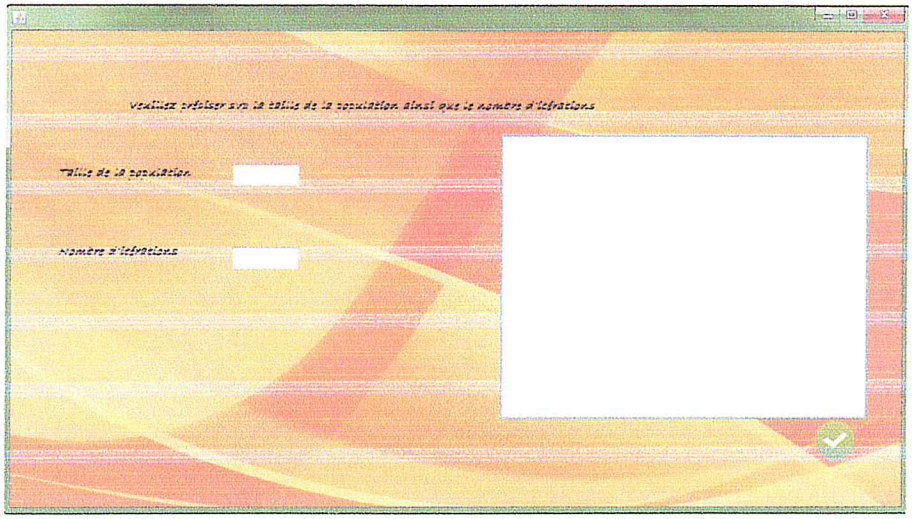

Figure 39: Interface d'optimisation.

4.3.3.4 *Interface du simulateur*

Comme déjà mentionné dans les chapitres précédents, le simulateur joue le rôle de la fonction objectif au sein de l'algorithme d'optimisation. De ce fait, lorsque l'utilisateur appuie sur de bouton OK de l'interface d'optimisation (**Figure 39**), la simulation est lancée (**Figure 40**) et renvoie le makespan de chaque solution.

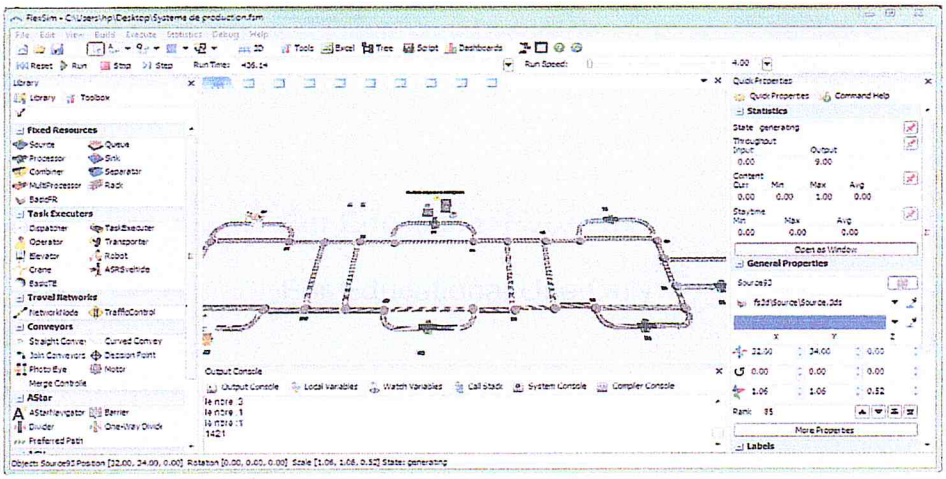

Figure 40: Interface du simulateur FlexSim.

4.3.3.5 Exemple de déroulement

Pour mieux expliquer l'enchainement des différentes interfaces de l'application, la figure ci-dessous présente un exemple de déroulement de l'application.

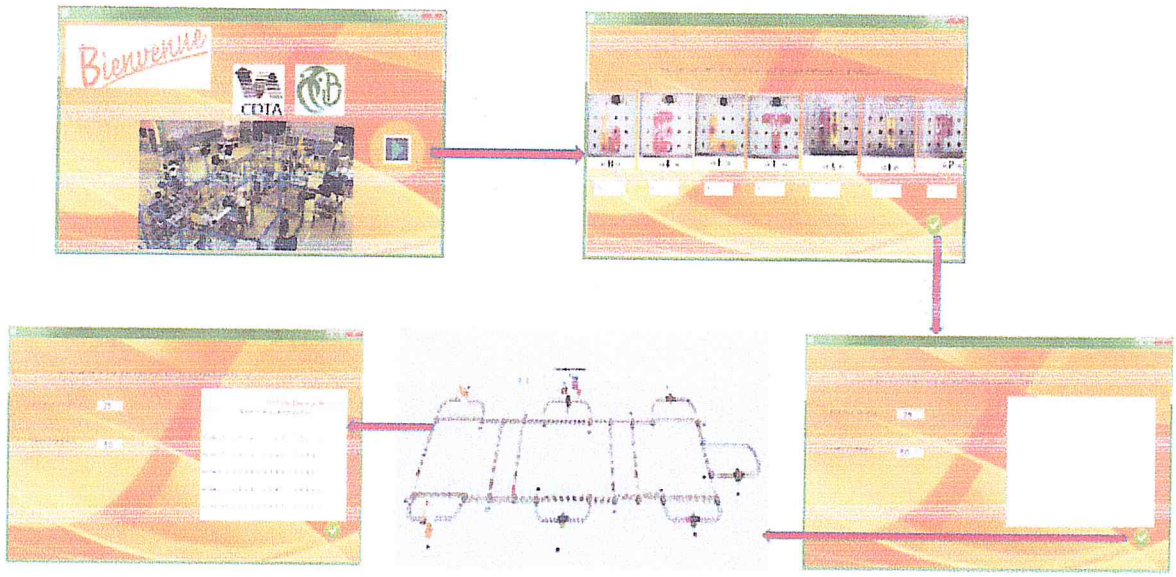

Figure 41: Exemple de déroulement de l'application.

4.4 EXPÉRIMENTATIONS

Afin de tester et valider notre algorithme, nous avons effectuée plusieurs expériementations en utilisant les benchmarks puis l'hybridation optimisation-simulation sur la cellule flexible AIP PRIMECA. Dans les deux cas, les résultats sont comparés avec ceux obtenus par l'application de l'algorithme de recherche par harmonie.

Notons ici que les expérimentations sont réalisées sur un ordinateur portable possédant les caractéristiques suivantes :

- Processeur : Intel(R) Core (TM) i5-2410M CPU @ 2.30 GHz 2.30 GHZ.
- Mémoire installée (RAM) : 4.00 Go.

4.4.1 Tests pour les problèmes Job Shop Flexible classique

En ordonnancement d'ateliers de type Job Shop Flexible, il est souvent utile d'avoir des Benchmarks pour définier les paramètres et tester l'efficacité de la méthode de résolution adoptée.

En effet, les Benchmarks sont des problèmes-types construits par plusieurs auteurs pour tester les performances des approches de résolution, en procédant surtout à des comparaisons sur les mêmes instances. Il s'agit d'instances théoriques (àl'instar des benchmarks des autres problèmes), formulées pour servir à l'étude de Job Shop Flexible. Un grand nombre de benchmarks de Job Shop Flexible sont proposés par plusieurs auteurs. Certains sont largement employés dans les recherches, alors que, d'autre sont moins connus. Dans notre expérimentation, nous avons utilisé les benchmarks MK01-10. Le tableau suivant décrit la taille de chaque instance de Benchmarck :

Tableau 8: Tableau représentant la taille de chaque instance de benchmarck.

Benchmark	Nombre de produits	Nombre de machines
MK 01	10	06
MK 02	10	06
MK 03	15	08
MK 04	15	08
MK 05	15	04
MK 06	10	15
MK 07	20	05
MK 08	20	10
MK 09	20	10
MK 10	20	15

Une comparaison de nos résultats obtenus par les différentes approches implémentées avec d'autres résultats obtenus par l'implémentation de l'algorithme de recherche par harmonie est représentée par les deux tableaux suivants.

L'implémentation de l'algorithme de recherche par harmonie a été réalisée par les membres de l'équipe SRP au CDTA.

Concernant le premier tableau (**Tableau 9**), nous avons fixé le nombre d'itération à 500 et la taille de population à 100. Quant au deuxième, nous avons fixé le nombre d'itération à 200 et la taille de population à 50.

La valeur de Moy est obtenue par le calcul de la moyenne des résultats obtenus en excécutant les deux algorithmes une dizaine de fois.

Tableau 9: Tableau compratif entre HS et GSA avec taille_pop = 100 et max_it=500.

	GSA		Harmony Search	
Problème	Moy	CBest	Moy	CBest
MK01	46	43	44	42
MK02	35	34	34	34
MK03	230	226	221	213
MK04	81	80	77	73
MK05	188	182	189	184
MK06	89	87	95	91
MK07	163	160	182	168
MK08	526	523	523	523
MK09	372	361	363	352
MK10	285	285	278	264

- Temps d'exécution moyen pris par les deux algorithmes : 4 minutes.

Tableau 10 : Tableau compratif entre HS et GSA avec taille_pop = 50 et max_it=200.

	GSA		Harmony Search	
Problème	Moy	CBest	Moy	CBest
MK01	48	43	44	43
MK02	35	34	34	33
MK03	230	223	223	213
MK04	82	73	75	69
MK05	196	194	188	181
MK06	97	93	99	86
MK07	189	184	175	161
MK08	540	532	524	523
MK09	384	339	358	351
MK10	310	305	288	371

- Temps d'exécution moyen pris par les deux algorithmes : 5 secondes.

On observant les résultats des tableaux ci-dessus, on peut remarquer que par la diminution de la taille de population et le nombre maximal d'itérations, les résultats obtenus par l'algorithme Harmony Search s'améliorent tandis que ceux obtenus par GSA-FJSP-FlexSim se détériorent. L'ajout d'itérations permet à l'algorithme de recherche par harmonies

de visiter plus d'endroits dans l'espace de recherche, ce qui n'est pas le cas pour l'algorithme GSA-FJSP-FlexSim. Ceci indique que les solutions de départ de la méthode GSA-FJSP-FlexSim sont très intéréssantes, surtout quand le nombre d'itérations est réduit.Quant au temps d'exécution, les premiers tests prennent plus de temps que les deuxième vu la grande différence de nombre d'itérations ainsi que la taille de populations.

4.4.2 Tests pour la cellule AIP-PRIMECA (optimisation-simulation)

Le tableau ci-dessous présente les résultats obtenus par GSA-FJSP-FlexSim et Harmony Search en utilisant l'approche optimisation-simulation.

Les résultats representent le temps totale de production calculé en secondes.

Tableau 11: Tableau comparatif des résultats trouvés par HS et GSA-FJSP-FlexSim en utilisant l'approche optimisation-simulation.

Max_it = 50	GSA	HS
Taille pop = 25	421	422
Taille pop = 50	435	435
Taille pop = 100	422	422
Taille pop = 500	428	438

- Temps d'éxécution moyen pris par harmony search : 20 minutes
- Temps d'éxécution moyen pris par GSA-FJSP-FlexSim : 45 minutes.

A la lumière des résultats trouvés, on remarque que l'algorithme GSA-FJSP-FlexSim donne souvent de meilleurs résultats que celles obtenus par l'algorithme de recherche par harmonie mais son temps d'éxécution reste critique.

4.5 Conclusion

Nous avons présenté dans ce chapitre les résultats de différentes expérimentations réalisées sur un ensemble de problèmes tests tirés des Benchmarks par deux algorithmes GSA-FJSP-FlexSim et HS. L'objectif est d'explorer les performances des stratégies qu'on a utilisées, d'un côté, et de valider notre implémentation de la métaheuristique adapté à la résolution du problème de Job Shop Flexible dans les deux cas statique et dynamique (optimisation-simulation) en comparant cette dernière avec l'algorithme de recherche par harmonies. Les tests effectués permettent d'apprécier les différents aspects liés à l'application des approches

développées à la résolution des problèmes d'ordonnancement type job shop flexible. Nous pouvons très vite conclure que les deux méta-heuristiques hybridations sont efficaces pour la résolution de ces problèmes. Reste que l'algorithme de recherche par harmonie est plus efficient en termes de temps de calcul.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les systèmes de pilotage jouent un rôle fondamental dans la maîtrise des activités de production et l'amélioration des performances au sein des systèmes industriels. En effet, la globalisation des marchés économiques, la concurrence féroce, la complexification des produits, sont autant de facteurs qui rendent l'environnement dans lequel évolue l'entreprise manufacturière, difficile à gérer.

Les problèmes de l'ordonnancement constituent une fonction importante en pilotage des systèmes de production. Il consiste à allouer dans le temps des taches à des ressources qui existent en quantité limitée, tout en satisfaisant un ensemble de contraintes.

Le problème d'ordonnancement de type Job Shop Flexible est connu dans la littérature comme un problème NP-difficile. De ce fait, l'utilisation de méthodes exactes en vue de l'obtention de solutions optimales semble non réaliste. Le recours à des méthodes approchées est donc devenu incontournable. Parmi ces méthodes, le paradigme des métaheuristiques s'impose comme une approche très prometteuse. En effet, en plus de leur adaptabilité aux différents problèmes combinatoires, les métaheuristiques ont l'avantage de parvenir à une solution acceptable dans un temps de calcul réduit.

L'objectif de notre travail est la réalisation d'une application d'ordonnancement industriel basée sur le model Job Shop Flexible et le développement d'un noyau algorithmique de résolution efficace pour ce problème. Pour ce faire nous avons développé l'approche optimisation-simulation. La méthode d'optimisation étant l'algorithme de recherche gravitationnelle, le simulateur à événements discrets a été intégré en tant que fonction d'évaluation au sein de l'algorithme d'optimisation. L'évaluation devait se faire en déroulant plusieurs simulations. Nous avons apprécié l'influence de ce paramètre (le nombre de simulations) sur l'évolution du processus dévolution. Plus ce paramètre est grand plus le processus d'optimisation est stable mais les temps de calcul augmente aussi. Il faut donc trouver une valeur compromis.

Afin de tester notre application, plusieurs expérimentations sont effectuées sur des Benchmarks retenus comme échantillons de tests pour prouver l'efficacité de l'algorithme GSA proposée et implémentée. Et ce, avant de l'appliquer sur notre cas d'étude qui est la cellule flexible AIP-PRIMECA de l'université de Valenciennes.

Ce travail nous a permis de constater que la méthode de la recherche gravitationnelle donne des résultats satisfaisants pour la résolution du problème d'ordonnancement de type Job Shop Flexible et que l'emploi d'une grande taille de population donne de mauvaises résultats.

A l'issue du travail présenté dans ce mémoire, différentes pistes forment les directions futures à cette recherche. Notamment :

- Enrichir la validation par l'expérimentation de nouvelles règles d'ordonnancement.
- Intégrer l'aspect séquencement des produits dans l'approche d'optimisation. Le séquencement peut être très important, mais il introduit de nouvelles difficultés concernant le codage et l'adaptation de la méthode GSA à ce dernier.
- Les résultats obtenus par l'algorithme GSA sont satisfaisants, mais comme toute méthode d'optimisation il serait intéressant de trouver des améliorations à apporter pour l'améliorer en termes d'efficience.

BIBLIOGRAPHIE

Artiba , A et Elmaghraby, S.E. 1997. *The planning and scheduling of production systèms, methodologies and applications.* London-Weinhein-New York– Tokyo – Melbourne – Madras : CHAPMAN & HALL, 1997.

Baptise, P. 1998. *Une étude théorique et expérimentale de la propagation des contraintes de ressources.* s.l. : Université de Compiègne, 1998.

Barnes , J.W et Chambers , J.B. 1996. *« Flexible job shop scheduling by tabu search».* s.l. : The University of Texas at Austin, Technical Report Series, 1996.

Barzegar, B et Motameni, H. 2011. *"Optimality of the flexible job shop scheduling system based on Gravitational Search Algorithm".* 2011.

Barzegar, B, Motameni, H et Bozorgi, H. 2012."*Solving Flexible Job-Shop Scheduling Problem Using Gravitational Search Algorithm and Colored Petri Net".* 2012, Journal of Applied Mathematics.

Barzegar, B et Motameni, H. 2013."*Solving Flexible Job-Shop Scheduling Problem using Hybrid Algorithm Based on Gravitational Search Algorithm andParticle Swarm Optimization".* 2013, JournalofAdvances in Computer Research .

Blazewicz, J, et al. 1996. *« Scheduling computer and manufacturing process ».* Berlin : Springer, 1996.

Boryczka, U. 2004. *Ant Colony System for JSP.* s.l. : Lecture Notes in Computer Science, 2004.

Boukadi , Khouloud . 2009. *Coopération interentreprises à la demande :Une approche flexible à base de services adaptables.* l'École Nationale Supérieure des Mines de Saint-Étienne : Thèse de Docteur, 2009.

Boukef, Ben Othman . 2011. *Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shopen industries pharmaceutiques : optimisation par algorithmes génétiques et essaims particulaires.* s.l. : Université de Tunis el Manar, 2011.

Carlier, J, Chrétienne, P et Girault, C. 1984.« *Modelling scheduling problems with Petri nets. Advanced studies in Petri nets ».* 1984, Springer.

Carlier , J et Chrétienne, P. 1988. *« Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes ».* Paris : Edition Masson, 1988.

Caumond, Anthony. 2006. "*Le problème de Job Shop avec contraintes : modélisation et optimisation".* Clermont Ferrand II : Université Blaise Pascal, 2006.

Chové, Ethienne. 2010. "*Contributions à l'ordonnancement réactif des instalations de traitement de surface".* s.l. : Université de Nantes, 2010.

Clerc, M et Siarry, P. Avril,2009. *« Une méthode inspirée de comportements coopératifs observés dans la nature : l'optimisation par essaim particulaire ».* Avril,2009, Revue de l'Electricité et de l'Electronique.

Collette, Y et Siarry, P. 2002. *« Optimisation Multiobjectif ».* Paris : Editions Eyrolles, 2002.

Dauzère-Pérès , S et Paulli, J. 1994. *"Solving the general job-shop scheduling problem Rotterdam School of Management"* . Rotterdam : s.n., 1994.

Dréo, J. 2004. *"Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical".* s.l. : Université Paris 12, 2004.

Dréo, J, et al. 2005. *"Métaheuristiques pour l'optimisation difficile".* s.l. : Eyrolles, 2005.

Duvivier, D. 2000. *"Etude de l'hybridation des métaheuristiques, application à un problème de type Job Shop".* LIL, Colais : Université de Litoral Cote d'Opale, 2000.

Fang , H, et al. 1993. *"Promissing Genetic Algorithm approach to Job Shop scheduling, rescheduling and open shop sceduling problems".* California : Morgan Kaufmann, 1993.

Fattahi, P et Fallahi, A. 2009. *« Dynamic Scheduling in Flexible Job Shop System by Considering Simultaneously efficiency and stability».* 2009, CIRP Journal of Manufacturing System nd Technology.

Ferr , A et Barachi, F. 2010. *"Conception et réalisation d'un environnement d'émulation multiagents pour l'évaluation des approches de pilotage par le produit des systèmes manufacturiers".* 2010.

Ferrah , Djahida et Baba, Kahina. 2010. *"Approche évolutionnaire pour la résolution du problème d'ordonnancement de type Job Shop flexible dynamique".* s.l. : Université de Blida, 2010.

Fisher, H et Thompson, G.L. 1963. *«Probabilistic learning combinations of local jobshop scheduling rules ».* Englewood Cliffs : Prentice-Hall, 1963.

Fontanili, F. 1999. *Intégration d'outils de simulation et d'optimisation pour le pilotage d'une ligne d'assemblage multiproduit à transfert asynchrone.* s.l. : Université Paris XIII, 1999.

Gao, J, et al. 2007. *«A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems».* s.l. : Computers and Industrial Engineering, 2007.

Gao, K.Z, et al. 2012. *"Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives".* 2012, Springer.

Garey, M et Johnson, D. 1979. *Computer and Interactibility: a gide to the theory of NP-Completeness.* San Francisco : s.n., 1979.

Ge, et al. 2006. *"A novel particle swarm optimization-based approach for job-shop scheduling".* 2006.

Giard, V. 1988. *"Gestion de la production".* Paris : Economica, 1988.

Glover, F. 1989. *« Tabu search, part I ».* 1989, ORSA Journal of Computing, pp. 190-206.

Gotha. 1993. *« Les problèmes d'ordonnancement »,RAIRO-Recherche Opérationnelle.* 1993.

Guohui , Z, Yang, S et Liang , G. 2008. *«A Genetic Algorithm and Tabou Search for Solving FlexibleJob Shop Schedules ».* China : s.n., 2008.

Hentous, H. 1999. *"Contribution au pilotage des systèmes de production de type Job Shop".* Lyon : s.n., 1999.

Hurink , J, Jurisch, B et Thole , M. 1994. *«Tabu search for the job-shop scheduling problem with multi-purpose machines ».* s.l. : OR Spektrum , 1994.

Jain, A.S et Meeran, S. 1999. *« Deterministic job-shop scheduling : past, present and future ».* s.l. : European Journal of Operational Research, 1999, Vol. 113.

Javel, G. 2004. *"Organisation et gestion de la production".* Paris : DUNOD, 2004.

Jurisch , B. 1992. *«Scheduling Jobs in Shops withMulti-Purpose Machines».* 1992.

Kacem I, I, Hammadi, S et Borne , P. 2002. *«Approach by localization and multiobjective evolutionary optimization for flexible job shop scheduling problems».* s.l. : IEEE Transactions on Systems, Man and Cybernetics, 2002.

Kacem, I. 2003. *"Ordonnancement multicritères des Job Shop flexibles: formulation, bornes inferieures et approche évolutioniste coopérative".* s.l. : Université de Lille 1, 2003.

Kang, et al. 2007. *"MAS Equipped with Ant Colony Applied into Dynamic Job Shop Scheduling".* s.l. : Lecture Notes in Computer Science, 2007.

Kebabla, Mebarek. 2008. *"Utilisation des stratégies Métaheuristiques pour l'ordonnancement des ateliers de type Job Shop".* s.l. : Université de Batna, 2008.

Kennedy, J et Eberhart, R.C. 1995. *« Particle swarm optimization ».* s.l. : Piscataway, 1995.

Kirkpatrick, S, Jr , C.D et Vecchi , M.p. 1983. *« Optimization by simulated annealing ».* s.l. : Science, 1983. p. 1983.

Larabi, Mohand. 2011. *"Le problème de job-shop avec transport : modélisation et optimisation".* s.l. : Université Blaise Pascal, 2011.

Lawrence, S. 1984. *«Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques».* s.l. : Carnegie-Mellon University, 1984.

Le Moigne , J.-L. 1999. *La modélisation des systèmes complexes.* s.l. : Dunod, 1999.

Lei, D. 2008. *"A Pareto archive particle swarm optimization for multi-objectivejob shop scheduling".* s.l. : Computers & Industrial Engineering, 2008.

Letouzey, A. 2001. *"Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs".* s.l. : Institus national de polythechnique de Toulouse, 2001.

Looveren, A.J.Van, Wassehnove, L.F et Gelders. 1986. *"A review of FMS planning model".* s.l. : A. Kusiak, editor, Modelling and design of flexible , Elsevier Science Publishers, 1986. article.

Lopez, P et Esquirol, P. 1999. *"L'ordonnancement".* s.l. : Economia, 1999.

Lopez, P et Roubelat, F. 2001. *"Ordonnancement de la production".* s.l. : Hermès sciences, 2001.

Mastrolilli, M et Gambardella , L. M. 2000. *«Effective neighbourhood functions for the flexible job shop problem».* 2000, Journal of Scheduling 3.

Meziane, Mohamed El Amine. 2011. *"Optimisation par phases pour le problème d'ordonnancement des ateliers de type Job Shop totalement flexibles".* s.l. : Université d'Oran, 2011.

Mirmadi, Samieh. 2009. *"Modélisation du processus de pilotage d'un atelier en temps réel à l'aide de la simulation en ligne couplée à l'exécution".* s.l. : Université de Toulouse, 2009.

Parunak. 1991. *Characterizing the manufacturing scheduling problem.* 3, s.l. : Journal of manufacturing systems, 1991, Vol. 10.

Paschos, V. 2005. *Optimisation combinatoire, Concepts fondamenteaux.* Paris : Hermès Sciences, 2005. Vol. 1.

Pinedo, M. 1955. *« Scheduling : Theory, Algorithms and systems ».* New Jersey : Prentice-Hall, Englewood Clis, 1955.

Rashedi, E, Nezamabadi-pour, H et Saryazdi, S. 2009. *"GSA : A Gravitational Search Algorithm".* Kerman,Iran : ELSEVIER, 2009.

Rodammer, F.A et Preston White, K. 1999. *« A recent survey of production scheduling ».* s.l. : IEEE Transaction on Systems, Man and Cybernetics, 1999.

Sharma, B et Yao, X. 2004. *"Characterising Genetic Algorithm Approaches to Job Shop Scheduling".* s.l. : Loughborough University, 2004.

T'kind, V et Billaut, J-C. 2006. *"Multicreteria scheduling Theory,Models and Algorithms".* Berlin : Springer, 2006.

Tongour Toumi , Fatma. 2007. *"Ordonnancement dynamique dans les industries agroalimentaires".* 2007.

Trentesaux, Damien. 1996. *"Conception d'un système de pilotage distribué, supervisé et multicritère pour les systèmes automatisés de production".* s.l. : l'université de Valenciennes et du Hainaut-Cambrésis, 1996.

Widmer, M. 2001. *« Les Métaheuristiques : Des outils performants pour les problèmes industriels ».* Troyes : s.n., 2001.

Xia, W, Wu et Z. 2006. *"A hybrid particle swarm optimization approach for the job-shop scheduling problem".* 2006, nternational Journal of Advanced Manufacturing Technology.

Yuan, Y et Hua, X. 2013. *A hybrid harmony search algorithm for the flexible job shop scheduling problem.* 2013.