الجـــــــــــــــــــــــــــــــمـهـوريـــــــــة الجـــــــــــــــــــــزائـريـــة الديمـــــــــــقـراطيـــة الـشـــــــــــــعـبـيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــعـليـــم العـــــــــــالي والبـــــــــــــــحث العـــــــلـمـــــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

– جـــــــــــــــــــــــــامعة أبي بكـــــر بـلـقـــايد – تـــلمســـــــــــــــــــــان

Université Aboubakr Belkaïd – Tlemcen –

Faculté de TECHNOLOGIE

MEMOIRE

Présentée pour l'obtention du **diplôme** de **MASTER**

En : Génie Industriel

Spécialité : Ingénierie de la Production

Par : BRAHIMI Ayet

Sujet

Etude de Résolution de Problème D'ordonnancement D'atelier de Type Flow Shop en Utilisant des Métaheuristiques Récents

Soutenu publiquement, le 06/06/2024, devant le jury composé de :

M. SOUIER Mehdi	Professeur	Université de Tlemcen	Président
Mme. MENADJLIA Nardjes	MAA	Université de Tlemcen	Examinatrice
Mme. ABDELLAOUI Wassila	MCB	Université de Tlemcen	Examinatrice
Mme. HOUBAD Yamina	MCB	Université de Tlemcen	Encadrant
Mme. BOUMEDIENE Fatima Zohra	MAB	Université de Tlemcen	Co-Encadrant

Année universitaire : 2023 / 2024

REMERCIMENTS

D'abord, nous tenons à remercier Allah, notre créateur, pour nous avoir donné le courage, la santé, les moyens et la patience nécessaires pour accomplir ce modeste travail.

Je tiens à exprimer ma chaleureuse gratitude envers mon encadrante, **Mme. HOUBAD Yamina**, et ma Co-encadrante, **Mme. BOUMEDIENE Fatima Zohra**, pour leur précieux encadrement, leurs conseils éclairés, leurs qualités humaines et leurs compétences de chercheuses. Leur patience m'a permis de mener à bien cette mémoire. Je tiens à leur exprimer toute ma gratitude pour la confiance qu'elles m'ont accordée.

Je souhaite également remercier les membres du jury, **M. SOUIER Mehdi, Mme. MENADJLIA Nardjes, Mme. ABDELLAOUI Wassila**, pour leur intérêt et leur acceptation d'évaluer ce mémoire, ainsi que pour leurs remarques constructives qui m'ont permis d'améliorer mon travail.

Je remercie du fond du cœur ma famille pour leur soutien inconditionnel et leur patience tout au long de ces années d'études. Leur confiance et leurs encouragements ont été une source de motivation constante.

Je tiens également à exprimer ma reconnaissance envers mes amis et collègues de promotion pour leurs échanges intellectuels stimulants et leur camaraderie. Leur présence a considérablement enrichi mon expérience universitaire.

Enfin, je souhaite exprimer mes sincères remerciements et mon respect à tous les professeurs et au personnel administratif de Faculté de Technologie, Département Génie Industriel, pour leur dévouement et leur aide précieuse. Leur contribution a grandement facilité mon parcours académique.

À tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce mémoire, je vous adresse mes plus sincères remerciements.

DEDICACES

الاهداء

﴿*وآخر دعواهم ان الحمد لله رب العالمين*﴾

فالحمد لله حبا وشكرا وامتنانا على البدء والختام

من قال انا لها نالها وان أبت رغما عنها أتيت بها, وها أنا ارى مرحلتي الدراسية قد شارفت على الانتهاء, عانقت اليوم مجدا عظيما فعلتها بعد تعب ومشقة دامت سنين في سبيل الحلم, حملت في طياتها أمنيات الليالي, وأصبح عنائي اليوم للعين قرة, ها انا اليوم أقف على عتبة تخرجي أقطف ثمار تعبي وأرفع قبعتي بكل فخر, فاللهم لك الحمد قبل أن ترضى ولك الحمد إذا رضيت ولك اك الحمد بعد الرضا

وبكل حب أهدي ثمرة تخرجي ونجاحي

الى العزيز الذي أحمل اسمه بكل فخر, من دعمني بلا حدود وأعطاني بلا مقابل الى من علمني أن الدنيا كفاح وسلاحها العلم والمعرفة, داعمي الاول في مسيرتي, سندي, قوتي وملاذي بعد الله, وبكل اعتزاز أنا لهاذا الرجل ابنة, الى **أبي الغالي** حفظه الله

الى من جعل الله الجنة تحت قدميها, واحتضنني قلبها قبل يديها وسهلت لي الشدائد بدعائها, من كانت الداعمة الأولى والأبدية معلمتي الأولى, **أمي الغالية** حفظها الله

الى من أفتقده, من رحل باكرا ويرتعش قلبي لذكره تاركا غصة لا تزول لآخر العمر, الى من فارقني وروحه مازالت ترفرف في سماء حياتي, الى تلك الروح الطاهرة **جدي** رحمه الله

الى خيرة ايامي وصفوتها ومن بوجودهم أكتسب قوة ومحبة لا حدود لها, الى ضلعي الثابت أماني أيامي أختي وأخواني (**سناء,وليد,أنس**)

الى رفقاء قلبي وصديقات الايام المرة والحلوة...ضماد الروح أنيسات روحي وسرور عمري, الى من راهنو على نجاحي وتذكيري بمدى قوتي واستطاعتي, أخواتي التي أنجبتهم الأيام, رفيقات قلبي **إيناس وعايدة**

الى كل عائلتي الى الذين يبهجهم نجاحي وكل من كان عونا وسندا في هذا الطريق لأصدقاء ورفقاء السنين

ما سلكنا البدايات الا بتيسيره وما بلغنا النهايات الا بتوفيقه وما حققنا الغايات الا بفضله فالحمد لله

Résumé

Ce mémoire présente la résolution du problème d'ordonnancement dans un atelier de type flow shop en utilisant des métaheuristiques récentes afin de minimiser le makespan. Il est structuré en deux parties. La première partie aborde les concepts liés aux systèmes de production et à l'ordonnancement, tout en offrant également un aperçu des méthodes de résolution de ces problèmes. Cette section met en évidence l'importance de l'ordonnancement dans les systèmes de production ainsi que les approches spécifiques pour traiter ces problèmes. La deuxième partie se concentre sur l'adaptation de deux métaheuristiques basées sur une population : l'algorithme génétique et l'algorithme de recherche pingouins, pour résoudre notre problème. Une étude de sensibilité a été réalisée, démontrant l'efficacité des deux algorithmes.

Mots clés : Ordonnancement, Flow Shop, Métaheuristiques, Algorithme Génétique, Algorithme de recherche pingouins.

Abstract

This thesis presents the solution of the scheduling problem in a flow shop using recent metaheuristics to minimize the makespan. It is structured in two parts. The first part covers concepts related to production systems and scheduling, while also providing an overview of methods for solving these problems. This section highlights the importance of scheduling in production systems, as well as specific approaches to dealing with these problems. The second part focuses on the adaptation of two population-based metaheuristics : the genetic algorithm and the penguin search algorithm, to solve our problem. A sensitivity study has been carried out, demonstrating the effectiveness of both algorithms.

Keywords : Scheduling, Flow Shop, Metaheuristics, Genetic Algorithm, Penguin Search Optimization Algorithm.

ملخص

هذا البحث يقدم حل مشكلة الجدولة في ورشة عمل من نوع "Flow shop" باستخدام Métaheuristique الحديثة بهدف تقليل مدة التنفيذ .(makespan) وهو مُقسم إلى جزأين. الجزء الأول يتناول المفاهيم المتعلقة بأنظمة الإنتاج والجدولة، مع تقديم نظرة عامة على طرق حل هذه المشاكل. يُبرز هذا القسم أهمية الجدولة في أنظمة الإنتاج وكذلك الأساليب المحددة لمعالجة هذه المشاكل. الجزء الثاني يركز على تكييف اثنتين من Métaheuristique المعتمدة على السُكان: الخوارزمية الجينية وخوارزمية بحث البطاريق، لحل مشكلتنا. أُجريت دراسة حساسية تُظهر فعالية كلتا الخوارزميتين.

الكلمات المفتاحية: الجدولة،Flow Shop ، Métaheuristique، الخوارزمية الجينية، خوارزمية تحسين البحث البطريق

Sommaire

Introduction générale 1

Chapitre I : l'ordonnancement dans les Systèmes de production

I.1 Introduction 3
I.2 Les Système de production 3
I.2.1 Décomposition de système de production 3
I.2.1.1 Système physique de production 3
I.2.1.2 Système de décision 4
I.2.1.3 Le système d'information 4
I.2.2 Classification des systèmes de production 4
I.2.2.1 Processus continue 4
I.2.2.2 Processus discrets 4
I.2.2.3 Les processus discontinus 5
I.3 L'Ordonnancement 5
I.4 Les Elément de l'ordonnancement 5
I.4.1 Les tâches 5
I.4.2 Les ressources 6
I.4.2.1 Les ressources renouvelables 6
I.4.2.2 Les ressources consommables 6
I.4.3 Les contraintes 6
I.4.4 Les objectifs 7
I.5 Les classes d'ordonnancement 7
I.5.1 Ordonnancement actif 7
I.5.2 Ordonnancement semi- actif 7
I.5.3 Ordonnancement sans délai 8
I.6 Objectif de l'ordonnancement 8
I.7 Classification des problèmes d'ordonnancement 8
I.8 Les problèmes d'ordonnancement 11
I.8.1 Modèle à une opération 11
I.8.1.1 Problème à machine unique 11

I.8.1.2 Problème à machine parallèle 11
I.8.2 Modèle à plusieurs opérations 12
I.8.2.1 Flow-Shop 13
I.8.2.2 Job-Shop 13
I.8.2.3 Open-Shop 14
I.9 Complexité des problèmes d'ordonnancement 14
I.10 Conclusion 16

Chapitre II : Les Méthodes de résolution des problèmes d'ordonnancement

II.1 Introduction 18
II.2 Méthodes de résolution 18
II.2.1 Méthodes exactes 19
II.2.1.1 Programmation linéaire 19
II.2.1.2 Programmation dynamique 19
II.2.1.3 Procédure Evaluation et Séparation (Branch & Bound) 20
II.2.2 Méthodes approchées 22
II.2.2.1 Heuristiques 22
II.2.2.2 Métaheuristiques 23
II.3 Métaheuristiques 23
II.3.1 Métaheuristique à solution unique 23
II.3.1.1 Recherche Tabou 24
II.3.1.2 Recuit Simulé 25
II.3.1.3 Recherche Locale Itérative 26
II.3.2 Métaheuristique à Population de Solution 26
II.3.2.1 Colonie de Fourmis 26
II.3.2.2 L'optimisation par Essaim de Particules 27
II.3.2.3 Algorithme Génétique 27
II.3.2.4 Algorithme d'optimisation de la recherche Pingouins (PeSOA) 30
II.4 Intensification et Diversification 32
II.5 Conclusion 33

Chapitre III : Adaptation de l'algorithme génétique et l'algorithme d'optimisation de la recherche Pinguins (PeSOA)

III.1 Introduction 35
III.2 Problème de Flow Shop 35
III.2.1 Formulation mathématique 35
III.3 Adaptation de l'algorithme génétique 37
III.3.1 Les paramètres d'algorithme génétique 37
III.3.1.1 La taille de population 37
III.3.1.2 Le codage 37
III.3.1.3 La sélection 38
III.3.1.4 Le croisement 38
III.3.1.5 La mutation 39
III.3.1.6 L'évaluation des individus 39
III.3.2 La simulation de l'algorithme génétique 39
III.4 Adaptation de l'algorithme de pingouins pour la résolution du problème d'ordonnancement de type Flow Shop 40
III.4.1 Le principe de l'algorithme 40
III.4.2 Les Paramètres d'Algorithme d'optimisation de la recherche Pingouins 40
III.4.2.1 La POPinitial des pinguins 40
III.4.2.2 Les N groupes 40
III.4.2.3 Un Trou 40
III.4.2.4 Un Niveau 40
III.4.3 L'algorithme PeSOA Adapté 41
III.4.4 Résultats de Simulation de l'algorithme de pingouin 42
III.4.5 Interprétation de résultats de Simulation 42
III.5 Comparaison de résultats de simulation entre l'algorithme génétique et l'algorithme des pinguins 42
III.6 Etude de sensibilité d'algorithme d'optimisation de la recherche Penguins 44
III.7 Conclusion 46
Conclusion Générale 48
Référence Bibliographiques 50

Liste des Figures

Chapitre I : l'ordonnancement dans les Systèmes de production

Figure I. 1: Les sous-systèmes constituant le système de production. [4] 4
Figure I. 2: Caractéristiques d'une Tâche. 6
Figure I. 3: Les relations d'inclusion entre les classes d'ordonnancement. 8
Figure I. 4: Exemple d'un Problème à Machine Unique. 11
Figure I. 5: Exemple d'un problème à machine parallèle. 12
Figure I. 6: Atelier Flow Shop. 13
Figure I. 7: Atelier Job Shop. 14
Figure I. 8: Atelier Open Shop. 14
Figure I. 9: Les principales Classes de Complexité. 15

Chapitre II : Les Méthodes de résolution des problèmes d'ordonnancement

Figure II. 1: Classification des Méthodes de Résolution. 18
Figure II. 2: Organigramme de la Recherche Tabou. 24
Figure II. 3: Organigramme du Recuit Simulé. 25
Figure II. 4: l'expérience de sélection du plus court chemin. 26
Figure II. 5: Organigramme de Colonie de Fourmis. 27
Figure II. 6 : Organigramme d'Algorithme génétique. 28
Figure II. 7: Opérateur de Sélection. 28
Figure II. 8: Croisement en Un Point 29
Figure II. 9: Croisement en Deux Point. 29
Figure II. 10: Opérateur de Mutation. 29
Figure II. 11: Schéma explicatif pour le principe d'algorithme PeSOA 31
Figure II. 12: Algorithme d'optimisation de la recherche Pingouins Générale 32

Chapitre III : Adaptation de l'algorithme génétique et l'algorithme d'optimisation de la recherche Pinguins (PeSOA)

Figure III. 1: Exemple d'un Codage d'un Chromosome 38
Figure III. 2: Exemple de Croisement Appliqué sur deux individus 38
Figure III. 3: Exemple de Mutation 39
Figure III. 4: Algorithme d'optimisation de la recherche Pingouins Adapté 41
Figure III. 5: des graphes présente les résultats de la simulation de PeSOA et de l'AG. 43

Liste des Tableaux

Chapitre I : l'ordonnancement dans les Systèmes de production

Tableau I. 1: Principale valeurs du champ α1 et α2. [6] 9
Tableau I. 2: Principaux sous champs du champ β. [6] 10
Tableau I. 3: Principales notations du champ γ. [6] 10
Tableau I. 4: Description Des Configuration d'atelier. [6] 13

Chapitre III : Adaptation de l'algorithme génétique et l'algorithme d'optimisation de la recherche Pinguins (PeSOA)

Tableau III. 1: Les résultats de simulation de l'Algorithme Génétique 39
Tableau III. 2: Les résultats de Simulation de l'algorithme de pingouin 42
Tableau III. 3: Analyse de sensibilité de L'Algorithme PeSOA pour 5 exemples de 10 jobs 30 machines (N, Fixé et Tr, varier) 44
Tableau III. 4: Analyse de sensibilité de L'Algorithme PeSOA pour 5 exemples de 10 jobs 30 machines (N, varier et Tr, Fixé) 45

INTRODUCTION GENERALE

Introduction générale

Les problèmes d'ordonnancement des ateliers de type Flow Shop sont cruciaux dans la gestion de la production, surtout lorsqu'ils sont liés aux systèmes de production. Ces problèmes sont souvent complexes et difficiles à résoudre, surtout lorsque la taille des instances augmente, ce qui les rend NP-difficiles.

Pour faire face à cette complexité, il est nécessaire d'utiliser des méthodes spécialisées, allant des méthodes exactes aux heuristiques et aux métaheuristiques. Les méthodes exactes ne peuvent souvent pas les résoudre en raison du temps de calcul requis. Heureusement, les méthodes d'approximation (heuristiques et métaheuristiques) se sont révélées être des outils puissants pour trouver des solutions satisfaisantes de haute qualité dans un délai raisonnable.

Le domaine de résolution des problèmes NP-difficiles offre une opportunité passionnante pour l'exploration grâce à l'application de métaheuristiques. Des recherches approfondies ont été menées sur différentes approches métaheuristiques, allant des méthodes de recherche locale de base aux techniques avancées qui sont à la pointe des stratégies de résolution de problèmes. Parmi les métaheuristiques les plus récentes et innovantes, on trouve l'algorithme d'optimisation de la recherche Pingouins (PeSOA).

Ce travail a pour objectif d'utiliser et d'adapter des métaheuristiques pour résoudre les problèmes de Flow Shop tout en minimisant le makespan. Notre contribution consiste à adapter, programmer et hybrider l'algorithme PeSOA (Algorithme de recherche des pingouins).

L'algorithme de recherche des pingouins (PeSOA) est une nouvelle métaheuristique inspirée de la nature, qui s'appuie sur le comportement de chasse collaboratif des pingouins. Les pingouins utilisent une stratégie de chasse basée sur la collaboration et la synchronisation de leurs plongées. Cet algorithme se distingue par sa capacité à équilibrer l'exploration et l'exploitation de l'espace de recherche, ce qui lui permet d'éviter de se retrouver piégé dans des optima locaux et d'améliorer l'efficacité de la recherche globale.

Ce document présente une synthèse de notre travail et est structuré comme suit :

Le premier chapitre se concentre sur les concepts et les définitions pertinents des systèmes de production et de l'ordonnancement, tout en exposant les problèmes liés à l'ordonnancement des systèmes de production.

Le deuxième chapitre présente différentes approches pour résoudre les problèmes de planification. Il aborde les méthodes d'optimisation exactes et approximatives, puis se concentre sur les principes des métaheuristiques les plus couramment utilisées. Ensuite, nous introduisons deux algorithmes : l'algorithme génétique et la recherche de manchots, que nous utilisons pour résoudre notre problème.

Le troisième chapitre présente notre adaptation de ces deux algorithmes, ainsi que les résultats de simulation obtenus. Ces résultats démontrent l'efficacité de notre algorithme de recherche Penguin par rapport aux algorithmes génétiques. De plus, nous avons effectué une analyse de sensibilité sur l'algorithme PeSOA.

En conclusion, nous présentons une synthèse générale qui résume notre travail et propose des perspectives de recherche

Chapitre I

L’ORDONNANCEMENT DANS LES SYSTEMES DE PRODUCTION

I.1 Introduction

Un système de production est un ensemble organisé d'éléments interdépendants conçus pour créer un produit ou fournir un service. Cela implique des processus coordonnés, des ressources et une technologie appropriée ainsi qu’une gestion efficace pour atteindre les objectifs de production. Ils sont utilisés dans divers domaines tels que les industries manufacturières et de services. Cela comprend la conception, la planification, la gestion des ressources, la fabrication, la logistique et d'autres aspects liés à la production. L'ordonnancement dans les systèmes de production fait référence au séquençage et à la planification des opérations de production dans le temps. C'est un processus de décision qui détermine l'ordre dans lequel les différentes tâches de production seront exécutées.

Les systèmes de production et l'ordonnancement sont deux concepts étroitement liés. L’ordonnancement fait partie intégrante du système de production et joue un rôle essentiel dans l’organisation et la planification des activités manufacturières.

Dans ce chapitre, nous discutons des systèmes de production en général et de certains concepts sur l'ordonnancement et ces problèmes. Nous présenterons ce chapitre en trois parties.

Dans la première partie, nous présenterons quelques notions liées aux les systèmes de production.

Dans la deuxième partie, nous nous présenterons l'ordonnancement, les différents éléments qui composent un problème d'ordonnancement et les différents types d'ateliers.

Dans la dernière partie, nous nous intéressons à la complexité des problèmes d'ordonnancement.

I.2 Les Système de production

Un système de production est un ensemble de ressources réalisant une activité de production **[1].** Un système de production rassemble tous les éléments matériels et immatériels dont une entreprise a besoin pour produire des biens ou des services. Le système de production d'une entreprise est le processus d'ajout de valeur à des biens ou des services qui répondent aux objectifs de quantité, de prix, de qualité et de délai.

Les systèmes de production varient en fonction des objectifs fixés pour le système. Ces objectifs évoluent en réponse aux changements du marché et aux progrès technologiques dont dépend le système.

I.2.1 Décomposition de système de production

Un système de production peut être en trois sous-systèmes, le système physique de production, le système de décision et le système d’information.

I.2.1.1 Système physique de production

Un système physique fait référence à un ensemble d'éléments interconnectés qui interagissent entre eux pour réaliser une fonction ou un objectif physique spécifique. Où les matières premières ou les composants sont transformés en produits finis. Il est constitué de ressources humaines et physiques.

I.2.1.2 Système de décision

Il englobe les processus de prise de décision liés à la planification, à la coordination et au contrôle des activités de production (le système physique de production). Il en coordonne et organise les activités en prenant des décisions basées sur les données transmises par le système d'information.

I.2.1.3 Le système d'information

Il s'agit d'un ensemble organisé de ressources telles que les personnes, les procédures, les données, les technologies et les équipements. Ces ressources sont utilisées pour collecter, stocker, traiter, diffuser et utiliser des informations dans le cadre des activités d'une organisation. L'objectif est de soutenir et d'améliorer les processus de production. Le système a pour rôle de collecter, stocker et transmettre des informations de diverses natures.

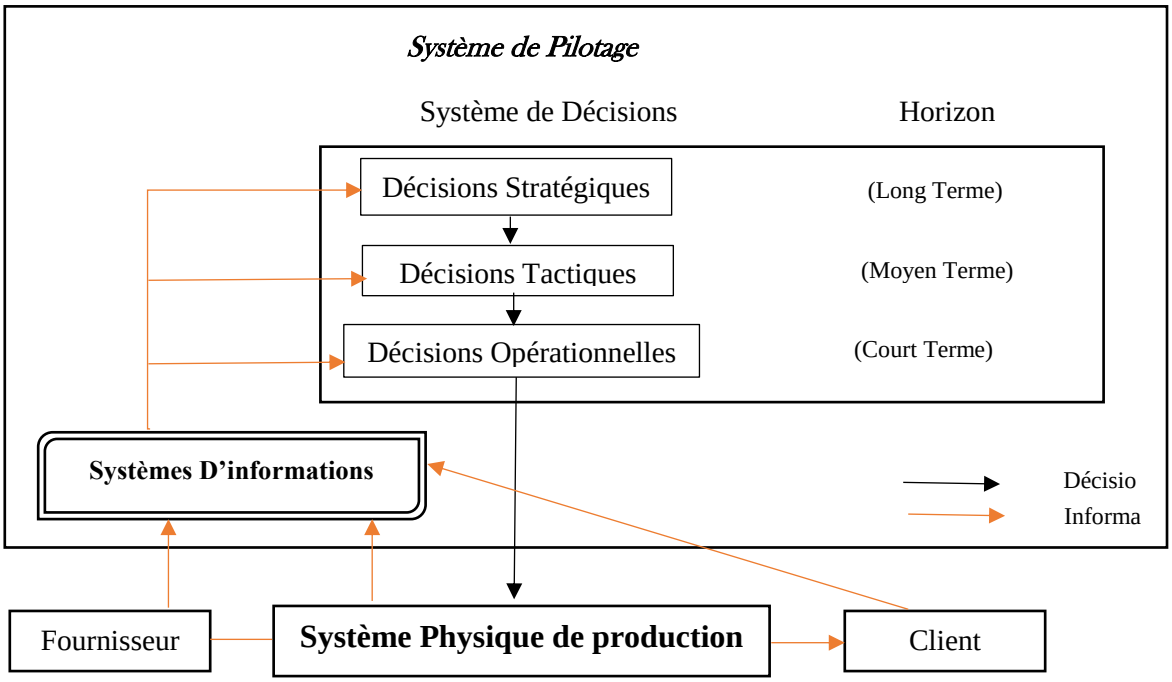

Figure I. 1: Les sous-systèmes constituant le système de production. [4]

I.2.2 Classification des systèmes de production

Les systèmes de production peuvent être classés en trois grandes catégories :

I.2.2.1 Processus continue

Sont utilisés pour la production de biens de grande quantité et de nature homogène, accompagnée d'une automatisation poussée des processus de production et des systèmes de manutention. La production se fait de manière continue, sans interruption. Tels que les raffineries pétrolières, les usines de produits chimiques et les usines de production d'énergie.

I.2.2.2 Processus discrets

Le processus discret, est un type de processus dans lequel les événements se produisent à des instants distincts et séparés dans le temps. Concerne la fabrication de produits distincts et identifiés individuellement.

Contrairement au processus continu, où la production est continue et l'unité de produit n'est pas distincte, le processus discret implique la production d'articles individuels ou de lots distincts. Tels que l'assemblage de produits, industrie pharmaceutique, etc.

I.2.2.3 Les processus discontinus

Sont des méthodes de fabrication où les produits sont réalisés en lots ou en unités séparées plutôt que de manière continue. Chaque lot ou unité passe par différentes étapes de production, avec des pauses entre celles-ci. Cette approche permet de fabriquer des produits variés ou personnalisés en réponse à des demandes spécifiques. La production est continue, mais les produits sont conditionnés de manière distincte.

I.3 L'Ordonnancement

L'ordonnancement est un processus de planification et de prise de décisions., d'organisation, de sélection et de synchronisation de l'utilisation des ressources. Il est largement utilisé dans divers domaines tels que la fabrication, la logistique, l'informatique, la gestion de projet et même la gestion du temps personnel. L'ordonnancement vise à optimiser l'allocation du temps et des ressources matérielles pour atteindre les objectifs de manière efficace et efficiente.

L'objectif principal de L'ordonnancement est d'optimiser l'utilisation des ressources disponibles telles que le temps, la main-d'œuvre, l'équipement et les matières premières afin de maximiser l'efficacité, la productivité et la satisfaction du client.

L'ordonnancement d'atelier couvre un ensemble d'actions qui transforment les décisions de fabrication définies par le programme directeur de production en instructions d'exécution détaillées destinées à piloter et contrôler à court terme l'activité des postes de travail dans l'atelier. **[2]**

Dans les problèmes d'ordonnancement des systèmes de production quatre notions fondamentales sont utilisées : **les tâches**, les machines ou **ressources** et certains **objectifs** à optimiser en respectant des **contraintes**. **[3]**

I.4 Les Elément de l'ordonnancement

I.4.1 Les tâches

Une tâche est le processus le plus élémentaire. Elle est constituée d'un ensemble d'actions à accomplir, dans des conditions fixées, pour obtenir un résultat attendu et identifié, en termes de performances, de coûts et de délais **[3]**. La planification consiste à organiser ces tâches de manière séquentielle ou simultanée pour optimiser l'utilisation des ressources et respecter les contraintes de temps.

En général, une tâche est caractérisée par trois paramètres. [4] :

- **La durée opératoire p_i** (processing time) : c'est la durée d'exécution de la tâche.
- **La date de disponibilité r_i** (release time) : c'est la date de début au plus tôt.
- **La date d'échéance d_i** (due date) : c'est la date de fin au plus tard.

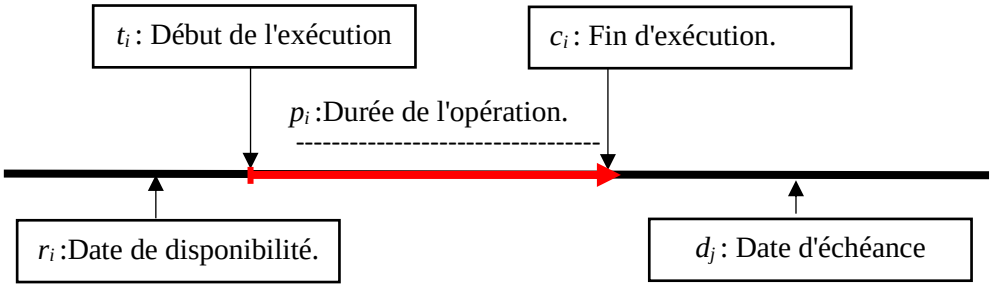

Figure I. 2: Caractéristiques d'une Tâche.

I.4.2 I

« Une ressource R_k est un moyen technique ou humain requis pour la réalisation d'une tâche » **[3]**. Les ressources sont les biens et les moyens, matériels et immatériels, qu'une entreprise possède ou peut mobiliser. Sont les moyens techniques ou humains utilisés pour accomplir une tâche. Ce moyen technique est donc nécessaire et indispensable au fonctionnement du cycle de fabrication. Dans un atelier, plusieurs types de ressources sont :

I.4.2.1 Les ressources renouvelables

Sont des éléments, des matériaux ou des sources d'énergie qui peuvent être exploités de manière durable sans épuiser leurs réserves naturelles à long terme. Les ressources renouvelables usuelles sont les machines, les processeurs, les fichiers, le personnel... **[5]**.

I.4.2.2 Les ressources consommables

Sont des éléments ou des matériaux qui sont utilisés dans un processus ou une activité et qui sont généralement épuisés, altérés ou consommés au fur et à mesure de leur utilisation. Une ressource est consommable si, après avoir été allouée à une tâche, elle n'est plus disponible pour les tâches restant à exécuter. C'est le cas pour l'argent, les matières premières... **[5]**.

I.4.3 Les contraintes

Les contraintes d'ordonnancement précisent la description des tâches et des machines. Sont des restrictions ou des limitations qui doivent être prises en compte lors de la planification et de la séquence des tâches dans un processus de production ou de projet.

Une contrainte exprime les restrictions sur les valeurs que les variables représentant les relations entre les tâches et les ressources peuvent prendre conjointement. **[3]**

On peut distinguer trois grandes catégories de contraintes. **[1]** :

- **Les Contraintes Temporelles**

  Concerne les délais de fabrication imposé.

- **Les Contraintes Technologique**

  Correspond aux contraintes technologiques, en général décrites dans les gammes de fabrication de produit.

- **Les Contraintes de Ressource**

Concerne ma limitation de la quantité de ressource de chaque type.

I.4.4 Les objectifs

Les objectifs sont les résultats attendus de l'ordonnancement. Ils peuvent inclure l'optimisation des performances, la réduction des coûts, l'amélioration de la qualité, la satisfaction des clients, et la réduction des temps d'attente.

Tout ordonnancement repose sur un ou plusieurs objectifs à optimiser. Les objectifs d'un ordonnancement peuvent varier en fonction des besoins. En règle générale, plusieurs catégories d'objectifs peuvent être identifiées pour un ordonnancement donné. **[4]** :

- **Objectifs Temporels**

L'objectif est de réduire le temps d'exécution, le temps de cycle et la durée de chaque tâche afin de respecter les délais fixés pour chaque phase du projet.

- **Objectifs de Coûts**

La minimisation des coûts totaux du projet consiste à réduire les coûts liés au lancement de production, au stockage, au transport, etc., tout en optimisant l'utilisation des ressources pour éviter les coûts supplémentaires liés au temps supplémentaire ou à une utilisation inefficace des ressources.

- **Objectifs de Ressources**

Ils consistent à optimiser l'utilisation des ressources disponibles (Il est important d'équilibrer la charge de travail en fonction des ressources disponibles.), telles que les machines, les outils et les travailleurs, en minimisant les coûts et en évitant les conflits de ressources.

- **Objectifs de Qualité**

Visent à maximiser la qualité des produits ou des services, en minimisant les erreurs, les retards et les révisions nécessaires.

I.5 Les classes d'ordonnancement

L'espace d'ordonnancement peut être divisé en sous-classes en fonction de la façon dont les opérations sont exécutées dans le temps. Ainsi, il est essentiel d'optimiser l'utilisation du temps pour générer des solutions efficaces. **[6]**

I.5.1 Ordonnancement actif

Un ordonnancement est considéré réalisable s'il n'est pas possible de traiter une opération plus tôt sans retarder le début d'une autre opération. **[6]**. Cela signifie que toutes les tâches sont en concurrence pour les ressources disponibles, et qu'il n'y a pas de tâches qui peuvent être exécutées en parallèle sans affecter les autres tâches.

I.5.2 Ordonnancement semi- actif

L'ordonnancement semi-actif est une situation dans laquelle certaines tâches peuvent être exécutées plus tôt sans changer l'ordre d'exécution des autres tâches, Contrairement à l'ordonnancement actif. Elle permet

une certaine flexibilité dans la réalisation de certaines tâches à l'avance sans affecter l'ordre d'exécution des autres tâches.

Dans ce type d'ordonnancement, la seule manière d'améliorer le makespan est de revoir la séquence des opérations sur les machines. **[6]**.

I.5.3 Ordonnancement sans délai

L'ordonnancement sans délai est une méthode qui garantit qu'une machine ne reste jamais inactive si elle peut être utilisée. Où les tâches peuvent être exécutées dès qu'elles sont disponibles, sans attendre une date de début spécifique. Cela signifie que les tâches peuvent être exécutées tant que toutes les ressources nécessaires sont disponibles sans attendre une date de début spécifique.

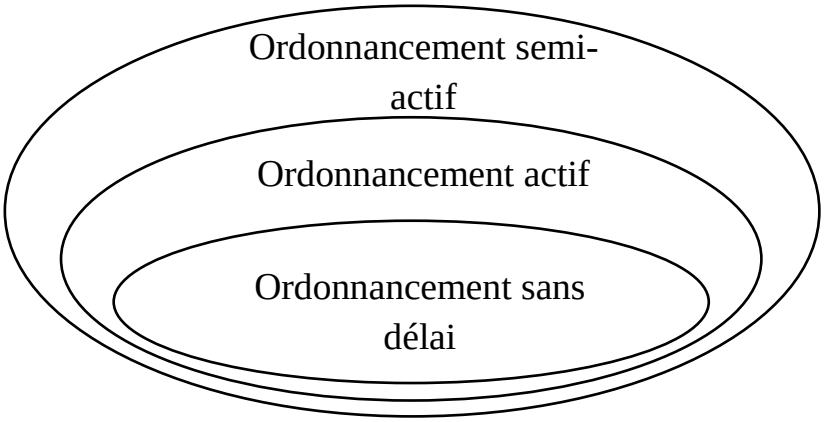

Figure I. 3: Les relations d'inclusion entre les classes d'ordonnancement.

I.6 Objectif de l'ordonnancement

L'objectif de l'ordonnancement est de déterminer la séquence optimale d'exécution des tâches ou des opérations, que ce soit dans le contexte de la production manufacturière ou des systèmes informatiques. Cela implique d'organiser et de planifier différentes tâches, activités ou processus de manière à atteindre les résultats souhaités.

En somme, l'ordonnancement vise à optimiser l'efficacité, la productivité et la rentabilité, que ce soit au niveau de la production ou de l'exécution des processus informatiques, et à atteindre les objectifs fixés.

I.7 Classification des problèmes d'ordonnancement

L'ordonnancement est lié à un ensemble de tâches N qui sont exécutées sur un ensemble de machines M, en respectant certaines contraintes de disponibilité, caractéristiques des opérations, types de ressources à utiliser et surtout l'organisation et la configuration de l'atelier. La notation proposée par Graham et al : $\alpha/\beta/\gamma$, présente la structure du problème (α), les contraintes d'ordonnancement à respecter (β) et l'objectif à optimiser (γ). **[7]**

- **L'Environnement des machines (champ α)**

Le champ α décrit l'environnement de la machine, tandis que le problème est représenté par deux sous-champs : α1 et α2.

- **α1** : fait référence à la configuration de l'atelier. **[7]**
  - Ø : Une seule machine.
  - P : Machines identiques en parallèle : pour une tâche i et M machines, $p_{i,m} = p_i$ (m = 1, ..., M).
  - Q : Machines uniformes en parallèle : pour une tâche i et M machines, $p_{i,m} = q_m p_i$ (m = 1, ..., M), avec q_m un facteur de vitesse relatif à chaque machine.
  - R : Machines indépendantes en parallèle : pour une tâche i et une machine m, le temps de traitement de la tâche i est donné par p_{im}.
  - F : Les machines effectuent des opérations en suivant un cheminement unique pour tous les produits (Flow shop).
  - J : Les machines effectuent des opérations avec un cheminement qui dépend du type de produit (Job shop).
  - : Les machines travaillent sur une structure à cheminement libre (Open shop).

- **α2** : Il décrit le nombre de machines à considérer.

Si la valeur de α2 est un nombre entier, le nombre de machines est constant. Si la valeur est m, le nombre de machines est variable (supérieur à 1). Enfin, si la première partie(α1) est vide, la deuxième partie(α2) reste vide ou égale à 1.

Variante	Notations	Description
α1	**Ø**	Machine unique, la valeur du champ α est alors ‹ 1 ›
	P	Machines parallèles identiques
	Q	Machines parallèles proportionnelles
	R	Machines parallèles non reliées
	F	Flow Shop
	J	Job Shop
	O	Open Shop
	FH	Flow Shop Hybride
	JG	Job Shop généralisé
	OG	Open Shop généralisé
	XAG	Problème de type X (avec X ϵ {P, Q, R, F, J, O}) avec affectation générale
α2	**{}**	Nombre de machines variable
	{m}	Problème à m Machines

Tableau I. 1: Principale valeurs du champ α1 et α2. [6]

- **Les Contraintes d'ordonnancement (champ β)**

Le champ beta décrire des contraintes additionnelles au problème. Il est constitué d'une suite de sous champs.

- pmtn, Ø : Les tâches peuvent être interrompues au moment de leur exécution. Pour tout autre cas, le champ reste vide.

- res, res1, ∅ : La présence de ressources supplémentaires est comprise (res, res1).
- prec, tree, ∅ : Les relations de précédence (prec) sont marquées, avec des précisions si elles sont nécessaires (tree).
- r_i, ∅ : Les dates de disponibilité des tâches sont indépendantes pour chaque tâche.
- P_{im} : Les temps opératoires sont définis par tâche et par machine, par tâche, par machine, ou sont identiques dans tous les cas.

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

Tableau I. 2: Principaux sous champs du champ β. [6]

- **Les Objectifs (champ γ)**

Le champ γ décrit l'objectif à optimiser pour le problème d'ordonnancement. Voici quelque exemple :

- Minimisation du makespan (C_{max}) : où le makespan est la date de fin maximale de toutes les tâches.
- Minimisation du retard ($\sum_{i=1}^{N} Ti$). Le retard est défini par la différence entre la date de fin effective et la date de fin prévue d'une tâche.
- Minimisation du temps total de séjour ($\sum_{i=1}^{N} Ci$). Le temps de séjour est le temps pendant lequel une tâche reste à l'intérieur du système de production.

$C_{max} = max_i \in \{0...n\}\ C_i$	Makespan : la plus grande date de fin
$T_{max} = max_i \in \{0…n\}\ (C_i - d_i, 0)$	Le retard maximum des jobs
$L_{max} = max_i \in \{0…n\}\ (C_i - d_i)$	Le retard algébrique maximum des travaux
$\sum wiCi$	La date de fin pondérée des tâches.
$\sum wiTi$	Le retard pondéré de jobs
$\sum Ti$	Le retard total
$\sum wiLi$	Le retard algébrique pondéré des travaux
$\sum \lvert Li \rvert$	Le retard absolu
$\sum Ui = \lvert Ji \rvert\ if\ (Ci > di)$	Le nombre de jobs en retard

Tableau I. 3: Principales notations du champ γ. [6]

I.8 Les problèmes d'ordonnancement

On dit qu'on a affaire à un problème d'ordonnancement lorsque : « on doit programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution [...]. Les tâches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni toujours immédiate, ni toujours triviale » [Ca-1988]. [8]

Les problèmes d'ordonnancement peuvent exister dans divers domaines d'activité allant de la fabrication à l'informatique et font l'objet de recherches en recherche opérationnelle et en gestion de production. Les problèmes d'ordonnancement consistent à organiser l'exécution des tâches dans le temps, en tenant compte des contraintes de temps et des ressources disponibles. Résoudre des problèmes d'ordonnancement implique de trouver des plannings de tâches sur les ressources en optimisant les objectifs et en respectant les contraintes.

Les problèmes d'ordonnancement sont classés en fonction du nombre d'opérations : ceux qui consistent en une seule opération (problèmes à machine unique ou multi-machines) et ceux qui comprennent plusieurs opérations (flow-shop, open shop et job shop).

I.8.1 Modèle à une opération

Un modèle à une opération consiste à accomplir chaque tâche ou job ne consiste qu'en une seule opération ou étape.

I.8.1.1 Problème à machine unique

Les problèmes d'ordonnancement sur une seule machine "Single Machine Scheduling Problem" constituent une classe spécifique de problèmes d'ordonnancement dans lesquels une seule machine est disponible pour effectuer toutes les tâches ou opérations. Chaque tâche a un temps d'exécution spécifique et une seule tâche peut être exécutée à la fois sur la machine.

L'objectif principal des problèmes mono-machine (Machine Unique) est de déterminer l'ordre optimal des tâches de traitement sur une machine, en minimisant certains critères tels que le temps d'exécution total, le temps d'attente des tâches ou la latence.

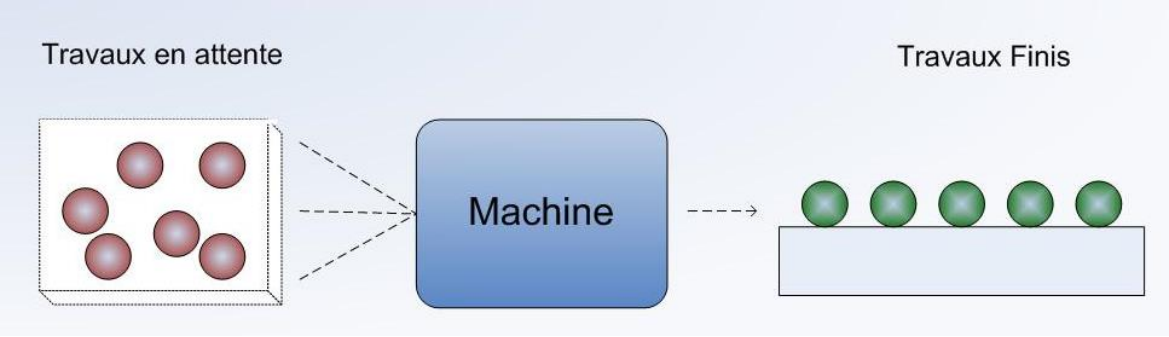

Figure I. 4: Exemple d'un Problème à Machine Unique.

I.8.1.2 Problème à machine parallèle

Le problème d'ordonnancement à machines parallèles est un type de problème d'ordonnancement où plusieurs machines parallèles qui effectuent la même tâche peuvent travailler simultanément. Contrairement

au problème à machine unique où une seule machine est disponible. Chaque tâche ayant un temps d'exécution spécifique et chaque machine ayant une capacité limitée.

Chaque machine ayant une capacité limitée. Les machines parallèles sont classées en fonction de leur vitesse. On peut distinguer trois classes de machines :

- **Machine parallèle identique** (Identical Parallel Machines) : Toutes les machines possèdent une vitesse de traitement identique et exécutent les mêmes tâches.
- **Machines parallèles uniforme** (Uniform Parallel Machines) : Les machines ont des vitesses de traitement variables, mais elles sont toutes linéaires. Chacune des machines est en mesure de traiter n'importe quelle tâche.
- **Machines parallèles indépendantes** (Unrelated Parallel Machines) : Les vitesses des machines sont indépendantes.

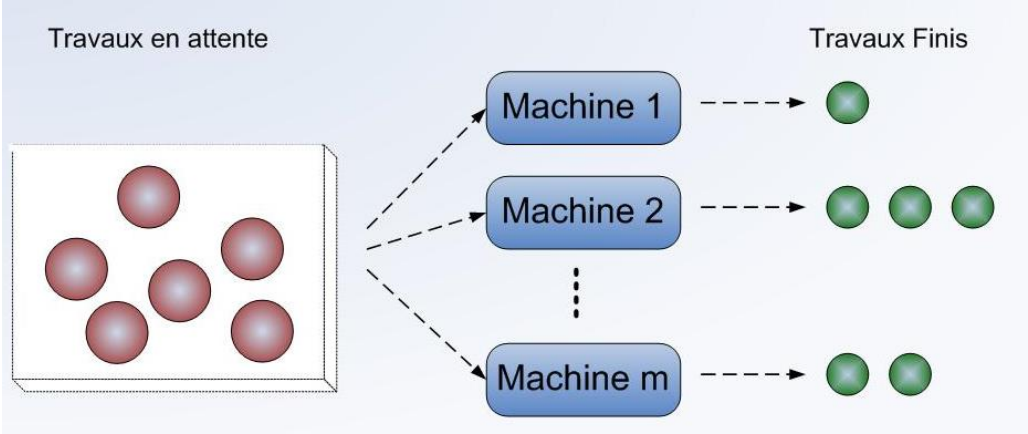

Figure I. 5: Exemple d'un problème à machine parallèle.

I.8.2 Modèle à plusieurs opérations

Les problèmes d'ordonnancement à plusieurs opérations (multi-opérations) incluent Chaque tâche doit passer par plusieurs étapes successives, contrairement au problème d'ordonnancement à machine unique où chaque tâche consiste en une seule opération. Puisqu'il y a généralement plusieurs opérations à effectuer, chaque tâche est caractérisée par une séquence spécifique d'opérations qu'elle doit effectuer, et chaque opération peut avoir des contraintes et des exigences spécifiques, telles que des délais, des prérequis, des priorités, des ressources nécessaires en attente. Ils peuvent être rencontrés dans divers contextes industriels, de production, de fabrication et autres domaines.

Trois modèles sont distingués en fonction de l'ordre de passage des tâches sur les machines, à savoir les modèles de flow-shop, job-shop et open-shop.

Atelier	Description
Machine unique	L'atelier se compose d'une seule machine et toutes les pièces doivent être traitées sur cette machine. C'est le type de configuration d'atelier le plus simple.
Flow Shop	Ateliers à cheminements unique, il contient le nombre « m » de machines en série, dans lesquelles toutes les pièces doivent entre usinées par le même routage, bien qu'il ne soit pas nécessaire que toutes les pièces soient affectées à toutes les machines.
Job Shop	Ateliers à cheminements multiples, il se compose de « n » nombre de pièce à usiner sur « m » nombre de machines et chaque pièce a son propre routage à traiter.

Open Shop	Ateliers à cheminements libres, il contient un nombre « m » de machines et le routage des pièces à travers la machine n'est pas limité. Les pièces peuvent utiliser lies machines dans n'importe quel ordre. Cependant, le planificateur est autorisé à déterminer un routage pour chaque pièce et chaque pièce peut avoir un routage différent.
Mixed-Shop	Il se compose d'un nombre « m » de machines et certaines pièces ont leur propre routage (fixe) et d'autres pas.

Tableau I. 4: Description Des Configuration d'atelier. [6]

I.8.2.1 Flow-Shop

Les ateliers de type Flow-Shop appelés également ateliers à cheminement unique, il s'agit d'un ensemble de m machines disposées en séries. Toutes les opérations de tous les jobs passent par les machines dans le même ordre. Chaque machine ne peut effectuer qu'une seule opération à la fois et toutes les pièces doivent suivre la même séquence d'opérations. Auparavant appelés systèmes de production à k niveaux, les flow shop n'ont reçu ce nom qu'à partir des travaux d'Heller en 1960 [Gupta et Stafford (2006)]. Dans le flow shop de base, tel qu'étudié dans l'article de [Johnson (1954)], l'objectif était d'ordonnancer n jobs sur m machines dans le but de minimiser un certain objectif. Les jobs devaient être traités par toutes les machines exactement une fois, et ce en suivant la même route à travers le système. Chaque job était alors traité par la machine M_1, la machine M_2, et ainsi de suite jusqu'à la machine M_m. **[9]**

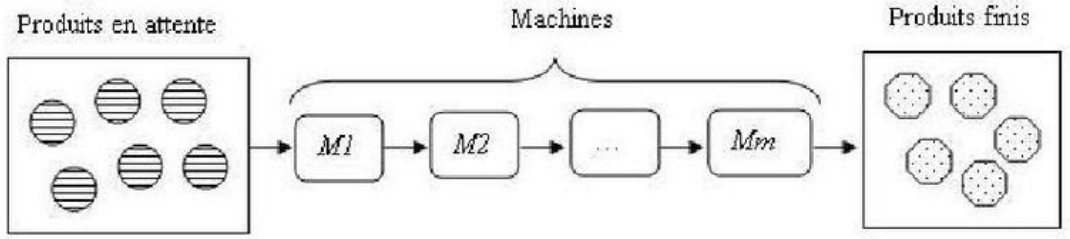

Figure I. 6: Atelier Flow Shop.

I.8.2.2 Job-Shop

Atelier de type job shop appelés aussi ateliers à cheminements multiples, où chaque tâche (job) peut avoir sa propre séquence d'opérations spécifique sur des machines différentes à différentes étapes. En d'autres termes, l'ordre des opérations peut varier d'une tâche à l'autre. Cela rend le problème d'ordonnancement plus complexe par rapport à l'atelier flow shop.

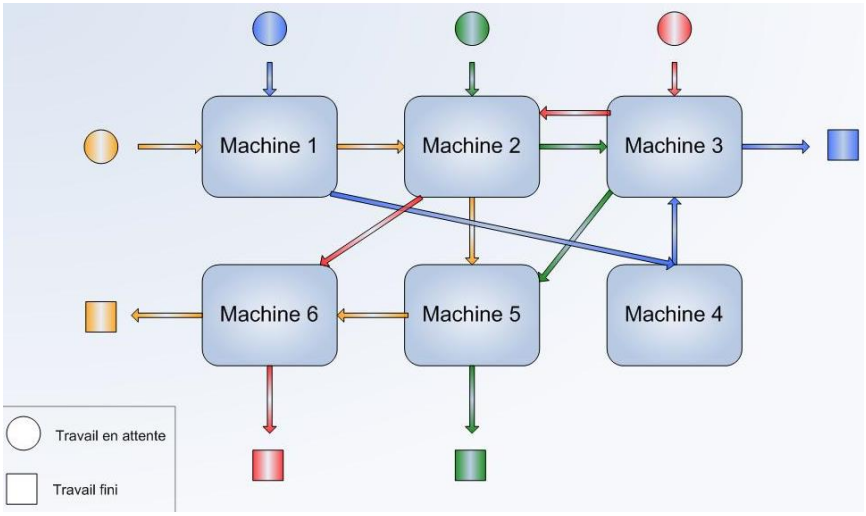

Figure I. 7: Atelier Job Shop.

I.8.2.3 Open-Shop

Les ateliers open shop, appelés aussi ateliers à cheminement libre, les gammes opératoires des différents jobs ne sont pas fixées a priori contrairement au problème d'atelier Job-Shop. Les opérations d'un même job peuvent donc être exécutées dans un ordre quelconque. Les opérations peuvent être exécutées dans n'importe quel ordre sur différentes machines, en fonction des besoins et des contraintes spécifiques. Le problème consiste d'une part à déterminer le cheminement de chaque job et d'autre part à ordonnancer les jobs en tenant compte des gammes trouvées.

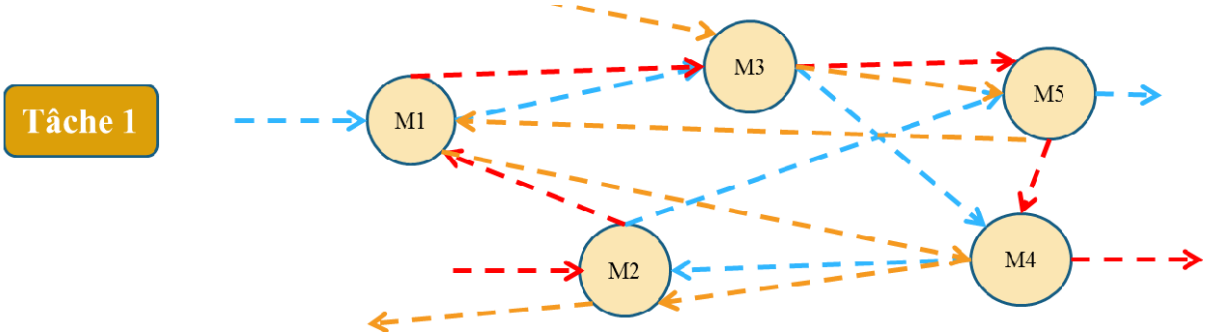

Figure I. 8: Atelier Open Shop.

I.9 Complexité des problèmes d'ordonnancement

La difficulté théorique d'un problème d'ordonnancement dépend de plusieurs facteurs, tels que la nature du problème, le nombre de tâches à planifier, les contraintes imposées, la complexité de l'algorithme, etc. L'un des concepts importants de la théorie de l'ordonnancement est la complexité du problème. Le niveau de difficulté du problème est mesuré par cette mesure. En général, la difficulté d'un problème d'ordonnancement est évaluée en termes de complexité algorithmique, qui est étroitement liée à la complexité et à l'efficacité de l'algorithme utilisé pour sa résolution.

Les problèmes d'ordonnancement d'ateliers sont des problèmes combinatoires notoirement complexes. Aucune méthode universelle n'existe pour résoudre tous les cas. Ces problèmes peuvent être classés selon leur complexité, mais se divisent en deux familles distinctes : ceux qui ont une complexité polynomiale et ceux qui ont une complexité exponentielle (problèmes NP-complets ou NP-difficiles). **[10]**

On distingue alors la classe P des problèmes qui sont calculables en temps polynomial par une machine de Turing déterministe. Face à la classe P, il y a la très célèbre classe NP (Non-déterministe Polynomial) qui est définie à partir d'un modèle de calcul non déterministe. De façon équivalente, c'est la classe des problèmes qui sont calculables par une machine de Turing non déterministe en temps polynomial (c'est-à-dire qu'on peut tester la validité d'une solution du problème en un temps polynomial). **[11]**

- **La Classe P (Polynomial)**

La classe P (temps polynomial) regroupe les problèmes de décision qui peuvent être résolus par des algorithmes déterministes en temps polynomial par rapport à la taille d'entrée. Un problème est appelé problème polynomial s'il existe un algorithme de complexité polynomiale tel que la question posée dans le problème peut recevoir une réponse quelles que soient les données du problème. La classe P correspond à l'ensemble de tous les problèmes d'identification polynomiale.

- **La Classe NP (Non-Deterministic Polynomial)**

Dans cette classe, on étudie les problèmes qui peuvent être résolus par un algorithme polynomial non déterministe. Il existe deux sous-classes distinctes dans cette catégorie. **[10]** :

- **La classe NP-complet :** Si un algorithme polynomial existe pour résoudre un problème NP-complet, il est également possible de trouver des algorithmes polynomiaux pour résoudre tous les autres problèmes de cette classe.
- **La classe NP-difficile :** Ces problèmes ne sont pas résolubles en temps polynomial, mais il est possible d'utiliser des algorithmes pseudo-polynomiaux ou des méthodes approximatives. Lors de leur exécution, ces algorithmes effectuent des choix dont l'optimalité ne peut pas être prouvée.

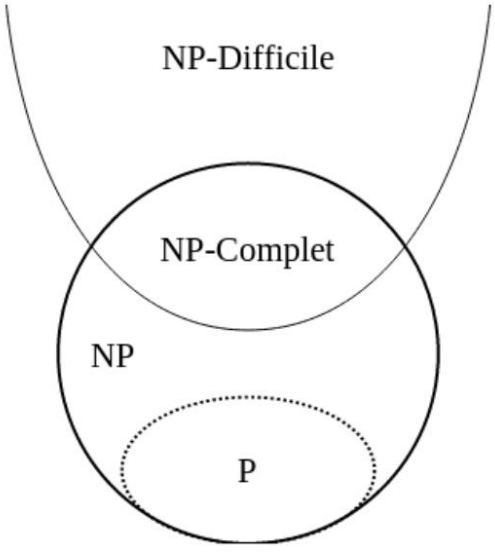

Figure I. 9: Les principales Classes de Complexité.

I.10 Conclusion

Ce chapitre nous a permis de donner des définitions relatives aux systèmes de production et à leurs différentes classes. Nous avons abordé la notion générale de systèmes de production en introduisant la décomposition et la classification de ces systèmes. Cela nous a permis de situer notre problématique par rapport aux autres problèmes des systèmes de production. En effet, notre thème de recherche concerne l'ordonnancement des ateliers.

Nous avons ensuite présenté l'ordonnancement et ses différents types d'ateliers. À ce stade, nous avons fourni des définitions, des notations et une classification des problèmes d'ordonnancement. Pour résoudre efficacement ces problèmes, il est important de prendre en compte les spécificités de chaque situation, c'est pourquoi nous avons introduit la notion de théorie de la complexité.

Chapitre II

LES METHODES DE RESOLUTION DES PROBLEMES D'ORDONNANCEMENT

II.1 Introduction

Les problèmes liés aux systèmes de production, tels que l'ordonnancement des tâches, sont souvent considérés comme des problèmes d'optimisation combinatoire. Leur complexité augmente avec la taille du problème et aucun algorithme connu ne peut les résoudre. Avec les avancées actuelles, les méthodes d'approximation, notamment les métaheuristiques, deviennent des alternatives intéressantes pour résoudre ces problèmes. Elles permettent d'obtenir des solutions de haute qualité tout en réduisant le temps de calcul requis.

Dans ce chapitre Nous commençons par une présentation générale des méthodes de résolution, Ces méthodes peuvent être classées en deux grandes catégories : Les méthodes exacte et les méthodes approché.

Ensuite, une partie sera dédiée aux principes des métaheuristiques les plus largement adoptées.

II.2 Méthodes de résolution

Les problèmes d'ordonnancement d'atelier sont des problèmes d'optimisation combinatoire qui nécessitent d'effectuer un nombre important de calculs pour obtenir un ordonnancement admissible (ou réalisable) qui optimise le (ou les) critère(s) retenu(s) en tenant compte des contraintes **[12].**

La résolution des problèmes d'ordonnancement implique la planification et l'organisation efficaces des différentes tâches ou activités dans le temps de manière optimale, en tenant compte de diverses contraintes et objectifs.

Les méthodes de résolution des problèmes d'ordonnancement sont des approches algorithmiques utilisées pour trouver des solutions efficaces aux problèmes d'ordonnancement, et le choix de la méthode dépend souvent de la complexité du problème. Ils peuvent être classées en deux catégories principales : les méthodes exactes et les méthodes approchées. Les méthodes exactes cherchent à trouver une solution optimale, tandis que les méthodes approchées cherchent à trouver une solution satisfaisante en un temps raisonnable.

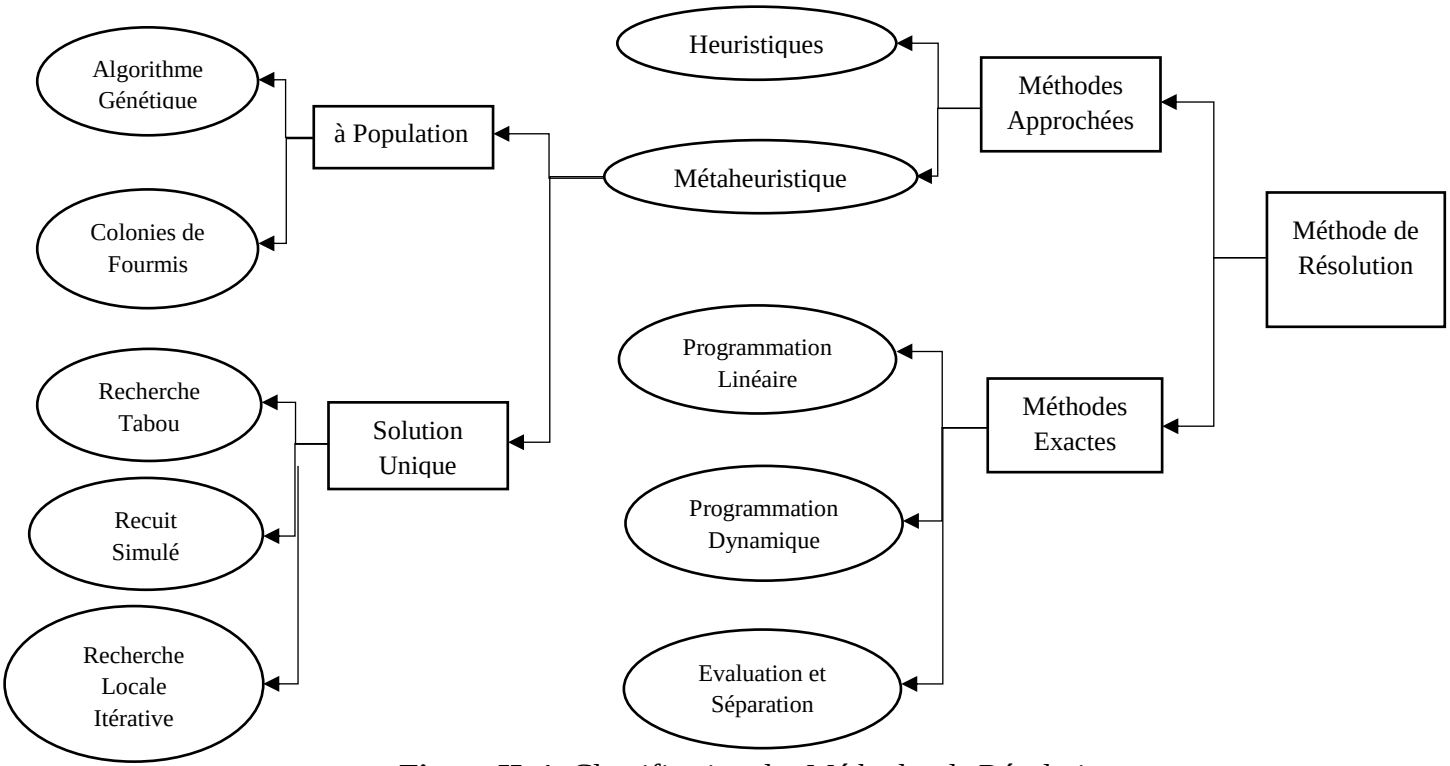

Figure II. 1: Classification des Méthodes de Résolution.

II.2.1 Méthodes exactes

Les méthodes exactes sont utilisées pour résoudre des problèmes d'ordonnancement simples et bien structurés. Visent à trouver la solution optimale d'un problème d'ordonnancement en examinant de manière exhaustive l'ensemble des solutions possibles ou en utilisant des techniques mathématiques pour déterminer la meilleure solution.

Elles peuvent être basées sur des techniques de programmation linéaire, de programmation dynamique ou de recherche exhaustive pour trouver une solution optimale.

Cependant, ces méthodes ne sont pas adaptées aux problèmes d'ordonnancement complexes, car leur complexité croît exponentiellement avec le nombre de tâches et de ressources. Ces méthodes garantissent généralement la recherche de la solution optimale, mais elles peuvent devenir couteuses en temps d'exécution, notamment pour les problèmes NP-difficiles.

II.2.1.1 Programmation linéaire

On appelle Programmation Linéaire, le problème mathématique qui consiste à optimiser (maximiser ou minimiser) une fonction linéaire de plusieurs variables qui sont reliées par des relations linéaires appelées contraintes **[13].**

La programmation linéaire permet de modéliser les problèmes d'optimisation dans lesquels critères et contraintes sont des fonctions linéaires des variables. Les deux types d'algorithmes les plus importants pour traiter un programme linéaire à variables continues sont la méthode du Simplex et la méthode des points intérieurs. L'inconvénient majeur réside dans le nombre important de variables et de contraintes nécessaires **[3].**

La programmation linéaire est l'une des techniques classiques de recherche opérationnelle. Consiste à maximiser ou minimiser (optimiser) une fonction linéaire appelée fonction objective et soumise à un ensemble de contraintes linéaires. Les contraintes sont exprimées sous forme de relations linéaires entre les variables du problème.

II.2.1.2 Programmation dynamique

La programmation dynamique est une technique mathématique développée par Bellman en 1957. Elle facilite la prise de décisions successives indépendantes en décomposant les problèmes avec des contraintes, tout en optimisant localement la fonction objectif (Gondran et Minoux en 1984). **[6]**. La programmation dynamique est une méthode algorithmique qui consiste à résoudre des problèmes d'optimisation. Elle consiste à décomposer un problème en sous-problèmes, puis à résoudre ces sous-problèmes, des plus petits aux plus grands, en stockant les résultats intermédiaires.

Cette méthode décompose un problème de dimension n en n problèmes de dimension. Le système est alors constitué de n étapes que l'on résout séquentiellement, le passage d'une étape à une autre se faisant à partir des lois d'évolution du système et d'une décision. Le principe d'optimalité de Bellman est basé sur l'existence d'une équation récursive permettant de décrire la valeur optimale du critère à une étape en fonction de sa valeur à l'étape précédente. Ainsi, pour appliquer la programmation dynamique à un problème combinatoire, le calcul du critère pour un sous-ensemble de taille k nécessite la connaissance de ce critère pour chaque sous-ensemble de taille k -1 et porte le nombre de sous-ensembles considérés à 2n (où n est le nombre d'éléments considérés dans le problème) **[13].**

II.2.1.3Procédure Evaluation et Séparation (Branch & Bound)

La procédure de séparation et évaluation, également connu sous le nom de "Branch and Bound" (B&B), est une technique algorithmique utilisée pour résoudre des problèmes d'optimisation combinatoire. Il convient aux problèmes pour lesquels l'espace de solution est vaste et nécessite une exploration systématique. Le Branch and Bound divise l'espace de solution en sous-ensembles plus petits, évalue ces sous-ensembles et élimine les sous-ensembles qui ne contiennent pas la solution optimale.

Repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des nœuds, et des feuilles. Le branch-and-bound est basé sur trois axes principaux **[14]** :

- L'évaluation.
- La séparation.
- La stratégie de parcours.

➢ **Principe de la Méthode :**

Il est aisé en général de déterminer l'ensemble de solutions réalisables du problème. Mais cet ensemble est généralement trop grand pour qu'il soit possible d'en extraire la solution optimale. En conséquence, on procède à une subdivision de cet ensemble S en un nombre fini de sous-ensembles $S_{i=\cdots P}^{(i)}$ de plus en plus petits en veillant à ce que **[15] :**

$$\bigcup_{i=1}^{p} S^{(i)} = S$$

Jusqu'à l'obtention d'un sous-problème suffisamment restreint "ensemble sondé" pour qu'on puisse extraire la solution optimale. La méthode de branch and bound, est essentiellement présentée par les trois procédures suivantes **[15]** :

➢ **La procédure de séparation :**

La séparation consiste à diviser le problème en sous-problèmes. Son principe doit satisfaire aux trois règles suivantes **[15]** :

- La règle de finitude : le nombre total de nœuds engendrés doit être fini.
- La règle de conservation : aucune solution d'un sous-problème ne peut être éliminée par la séparation c'est-à-dire,

$$\bigcup_{k=1}^{p} S^{(ik)} = S^{(i)}$$

Et qu'il vérifie également $S^{(ik)} \cap S^{(il)} = \emptyset \quad k \neq l$, où $S^{(ik)} \quad k = 1, \ldots, p$ représentent les sous-ensembles (enfants)du sous-ensemble (parent) $S^{(i)}$.

- La Règle d'arrêt : Un nœud ou sous-ensemble terminal de l'arborescence noté $S^{(t)}$ est défini comme un nœud qu'il n'est plus possible de séparer, pour un tel sous-ensemble :
  Soit $S^{(t)} = \emptyset$

Soit qu'il est possible de déterminer une solution optimale du problème $p^{(i)}$ qui est le sous-problème réduit de (P) et qui est défini comme suit :

$$p^{(i)} \begin{cases} \min & F(x) \\ & x \in S^{(i)} \end{cases}$$

- **La procédure d'évaluation :**

L'évaluation d'un sous-problème $P^{(i)}$ de (P) consiste à évaluer la valeur optimale de sa fonction économique. Plus précisément à déterminer une borne supérieure si la fonction économique est à maximiser respectivement une borne inférieure si elle est à minimiser. L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous-ensembles qui ne contiennent pas la solution optimale et elle a pour but de déterminer le sous problème qu'on doit séparer. L'exploration d'une branche est éliminée si **[15]** :

- Le sommet de l'arborescence ne peut être séparé c.à.d que le sommet(nœud)de l'arborescence est associé à une solution entière.
- Le sous-problème est non-réalisable.

- La valeur de Z associée au sous-problème est supérieure à Z^{opt} si le problème est à maximiser respectivement inférieur à Z^{opt} si le problème est à minimiser.
- La valeur Z d'un sommet associée à une solution non-entière inférieure ou égale à une valeur Z d'une solution entière respectivement supérieure pour un problème de minimisation.

- **La stratégie de parcours :**

Il est évident d'examiner la totalité des sommets de l'arborescence pour réaliser une énumération implicite efficace. Pour savoir quel sommet doit-on séparer on utilise des stratégies, on peut distinguer trois d'entre elles **[15]** :

- La largeur d'abord :

Cette stratégie favorise les sommets les plus proches de la racine (nœud père) en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies.

- La profondeur d'abord :

Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

- Le meilleur d'abord :

Cette stratégie consiste à explorer les sous-problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

II.2.2 Méthodes approchées

Une méthode approchée est une méthode d'optimisation qui vise à trouver une solution réalisable à la fonction objective dans un laps de temps raisonnable, mais sans garantie d'optimalité. **[16].**

Les méthodes approchées sont utilisées pour résoudre des problèmes d'ordonnancement complexes (NP-difficile) où la recherche exhaustive ou les méthodes exactes peuvent être impraticables en raison de la taille de l'espace de recherche. Contrairement aux méthodes exactes qui garantissent de trouver la meilleure solution possible, les méthodes approchées fournissent une bonne solution (de qualité raisonnable) dans un temps raisonnable, mais qui n'est pas nécessairement optimale.

Lorsque l'on dispose d'un temps de calcul limité ou lorsqu'on est confronté à des problèmes difficiles ou de taille importante, on peut avoir recours aux méthodes approchées, en se contentant de rechercher une solution de « bonne qualité » ; dans ce cas le choix est parfois possible entre une heuristique spécialisée, entièrement dédiée au problème considéré, et une métaheuristique [Dréo et al., 2003] **[11].**

Les heuristiques sont dédiées à des problématiques spécifiques alors que les métaheuristiques sont plus génériques **[7].**

II.2.2.1 Heuristiques

Le terme "heuristique" dérive du mot grec "eurisko" qui signifie "je trouve". Les heuristiques sont des heuristiques pratiques permettant de résoudre efficacement des problèmes complexes en fournissant une solution réalisable, même si elle n'est pas forcément optimale ou exacte.

Les approches heuristiques, contournent le problème de l'explosion combinatoire en faisant délibérément des impasses et n'explorent qu'une partie de l'espace des combinaisons. Une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre **[11].**

Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable. Parmi ces stratégies, nous distinguons **[13]** :

- **FIFO (First In First Out) :** où la première tâche arrivée est la première à être ordonnancée **[13].**
- **SPT (Shortest Processing Time) :** où la tâche ayant le temps opératoire le plus court est traitée en premier **[13].**
- **LPT (Longest Processing Time) :** où la tâche ayant le temps opératoire le plus important est traitée en premier **[13].**
- **EDD (Earliest Due Date) :** où la tâche ayant la date due la plus petite est la plus prioritaire **[13].**
- **SRPT (Shortest Remaining Processing Time) :** cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs **[4].**
- **ST (Slack Time) :** à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative **[4].**

II.2.2.2 Métaheuristiques

Le mot métaheuristique est dérivé de la composition de deux mots grecs : méta, du grec μοτα signifiant « au-delà » (ou « à un plus haut niveau ») et heuristique qui signifie « je trouve ».

Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, les métaheuristiques ont fait leur apparition. Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage **[17]**.

Les algorithmes de résolution métaheuristiques sont des heuristiques génériques, nécessitant une adaptation à chaque problème spécifique. Ils sont utilisables dans de nombreux problèmes d'optimisation combinatoire, et permettent d'incorporer diverses contraintes pratiques. En outre, ils sont relativement faciles à implémenter et produisent des solutions satisfaisantes.

II.3 Métaheuristiques

Les métaheuristiques consistent à explorer de manière intelligente l'espace des solutions possibles afin de trouver la meilleure ou la plus proche solution. Elles sont apparues au début des années 1980 avec un objectif commun : résoudre de manière optimale les problèmes d'optimisation difficiles. Depuis leur apparition, elles ont joué un rôle essentiel dans les méthodes approchées et ont ouvert de nouvelles perspectives passionnantes dans la conception de méthodes heuristiques pour l'optimisation combinatoire. **[10]**. Ils utilisent des méthodes de recherche stratégique afin d'explorer plus efficacement l'espace de recherche et se concentrent souvent sur les régions prometteuses. Ces méthodes commencent par un ensemble de solutions initiales ou une population initiale. Ensuite, elles examinent étape par étape une séquence de solutions pour atteindre, ou se rapprocher de, la solution optimale du problème. Les métaheuristiques offrent plusieurs avantages par rapport aux algorithmes traditionnels. Les deux avantages les plus importants sont leur simplicité et leur flexibilité. **[16]**.

Les métaheuristiques sont en général non-déterministes, elles peuvent ne pas trouver la solution optimale, et encore moins prouver l'optimalité de la solution trouvée. On peut distinguer les métaheuristiques qui font évoluer une seule solution sur l'espace de recherche à chaque itération et les métaheuristiques à base de population de solutions. En général, les métaheuristiques à base de solution unique sont plutôt axées sur l'exploitation de l'espace de recherche, on n'est donc jamais sûr d'obtenir l'optimum. Les métaheuristiques à base de population sont plutôt exploratoires et permettent une meilleure diversification de l'espace de recherche **[11]**.

II.3.1 Métaheuristique à solution unique

Les métaheuristiques à base de solution unique, également connues sous le nom de méthodes de trajectoire, traitent une seule solution à la fois pour trouver la solution optimale. Contrairement aux métaheuristiques à base de population, les métaheuristiques à solution unique débutent la recherche avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Elles utilisent le concept de voisinage pour optimiser la qualité de la solution courante.

Les différentes méthodes de trajectoire incluent le recuit simulé, la recherche tabou, la recherche locale itérée et leurs variations.

II.3.1.1 Recherche Tabou

La méthode de recherche avec tabous, ou simplement recherche tabou (TS : Tabu Search) a été introduite par Fred Glover en 1986 [Glover, 1986] Hansen (Hansen 1986), Glover et Laguna dans (Glover et Laguna 1997), et est devenue très conventionnelle dans l'optimisation combinatoire. Cette méthode est une métaheuristique itérative, que l'on peut appeler recherche locale au sens large. La différenciation se fait en utilisant la mémoire des solutions explorées lors de la recherche de la meilleure solution. Cette mémoire s'appelle la liste taboue, elle contient des solutions déjà visitées et les retours via cette solution sont interdits, afin d'éviter de rester prisonnier d'un minimum local ou de se retrouver bloqué dans un cycle répétitif.

La liste tabou, enregistre les dernières solutions rencontrées vers lesquelles il est interdit de se déplacer. Ce procédé simple de mémoire permet de choisir le meilleur voisin non tabou, même si celui-ci dégrade la fonction-objectif f **[11]**. La taille de la mémoire, le nombre d'éléments mémorisés, est un paramètre de l'algorithme qui permet de prendre en compte l'équilibre de diversification et d'intensification. En effet, si la mémoire est faible alors elle va favoriser l'intensification, car un nombre de solutions restreint seront interdites. Cependant, plus la taille de la mémoire augmente, et plus la diversification est favorisée, car l'algorithme pourra de moins en moins visiter les régions précédentes qui ont de grandes chances d'être voisines à la solution actuelle **[18]**.

La procédure de cette métaheuristique débute avec une solution initiale s appartenant à un ensemble local de solutions, S. Ensuite, des sous-ensembles de solutions, N(s), sont générés à partir du voisinage S. En utilisant la fonction d'évaluation, nous sélectionnons la solution qui améliore la valeur de f parmi les solutions voisines de N(s). **[4]**

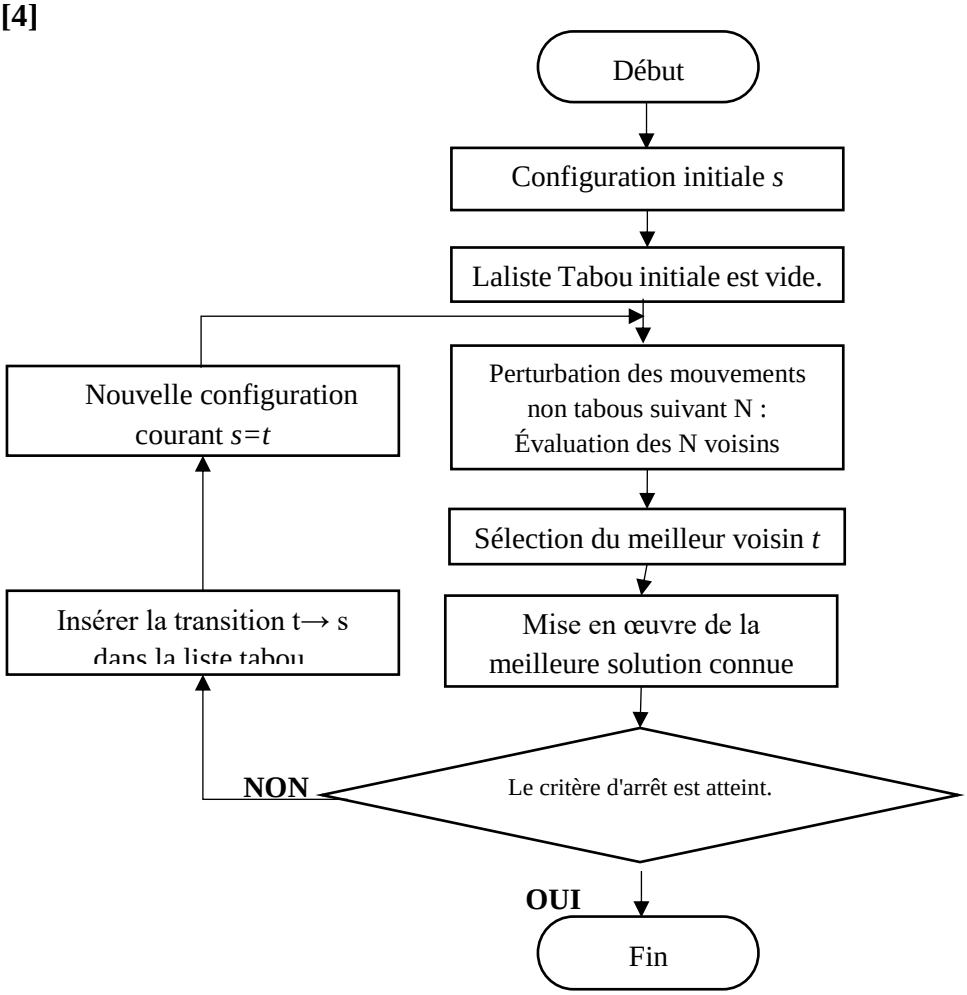

Figure II. 2: Organigramme de la Recherche Tabou.

II.3.1.2 Recuit Simulé

Le recuit simulé, simulated annealing (SA), tire son principe de la métallurgie. Il consiste à effectuer des cycles de refroidissement lent et de réchauffage d'un matériau afin de minimiser son énergie **[18]**, inspirée du processus de recuit physique utilisé en métallurgie, reposant sur les lois de thermodynamique énoncées par Boltzmann. Elle a été mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983 [Kirkpatrick et al., 1983], et indépendamment par Cerny en 1985 [Cerny, 1985] **[11]**.

Le principe du recuit simulé est de parcourir de manière itérative l'espace de solution. L'algorithme commence par générer une solution initiale (soit de façon aléatoire ou par une heuristique). À chaque nouvelle itération, une solution s′ est générée de manière aléatoire dans le voisinage N (s) de la solution courante s. La solution s′ est retenue si elle est de performance supérieure ou égale à celle de la solution courante, i.e., f (s′) ≤ f(s).

Dans le cas contraire, s′ est acceptée avec une probabilité $\exp(-\frac{\Delta f}{T})$. Cette probabilité dépend de deux facteurs : d'une part l'importance de la dégradation ($\Delta f = f(s') - f(s)$) – les dégradations plus faibles sont plus facilement acceptées ; d'autre part, un paramètre de température T – une température élevée correspond à une probabilité plus grande d'accepter des dégradations. Ainsi, au début de la recherche, la probabilité d'accepter des dégradations est élevée et elle diminue progressivement **[11]**.

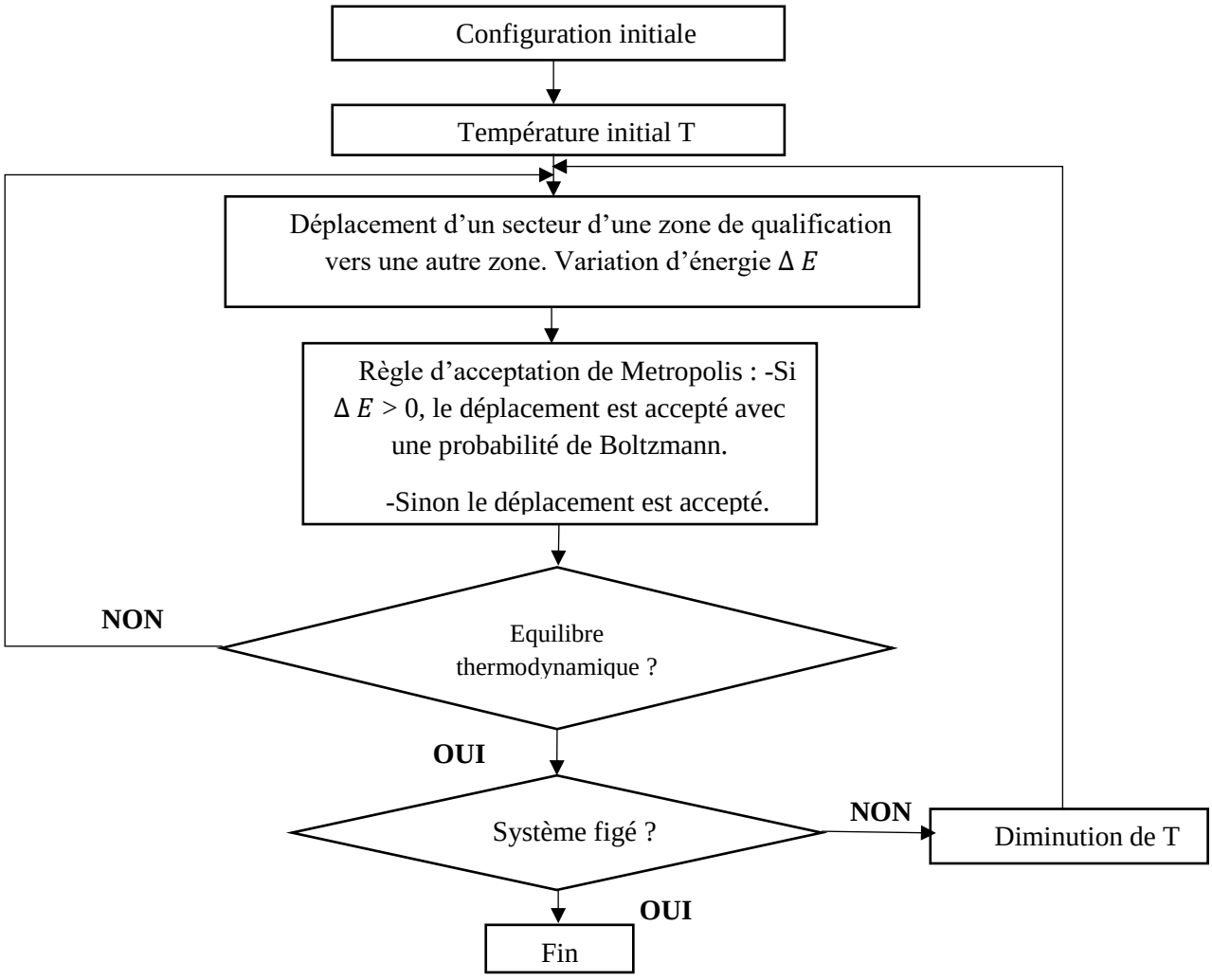

Figure II. 3: Organigramme du Recuit Simulé.

II.3.1.3 Recherche Locale Itérative

La recherche locale itérée (ILS, Iterated Local Search) est un algorithme d'optimisation métaheuristique qui associe la recherche locale avec des perturbations aléatoires afin d'explorer de manière plus approfondie l'espace de recherche. Cette approche est couramment utilisée pour résoudre des problèmes d'optimisation complexes pour lesquels il est difficile de trouver l'optimum global.

En quelques mots, le concept de base de la métaheuristique de recherche locale itérée peut être résumé efficacement, on construit itérativement une séquence de solutions générées par l'heuristique intégrée, ce qui permet d'obtenir de bien meilleures solutions que si l'on utilisait des essais aléatoires répétés de cette heuristique. Deux éléments principaux font d'un algorithme une recherche locale itérée : (i) une seule chaîne doit être suivie (ce qui exclut les algorithmes basés sur la population) ; (ii) la recherche de meilleures solutions se fait dans un espace réduit défini par la sortie d'une heuristique de boîte noire. Dans la pratique, la recherche locale a été l'heuristique intégrée la plus fréquemment utilisée, mais en fait n'importe quel optimiseur peut être utilisé, qu'il soit déterministe ou non **[19].**

L'objectif d'ILS est de naviguer à travers différents points optimaux locaux. Afin d'explorer de nouveaux optima, il est nécessaire que la perturbation soit suffisamment importante pour s'éloigner du point optimal actuel, tout en restant suffisamment modérée pour éviter de s'éloigner des régions favorables de l'espace de recherche **[20].**

II.3.2 Métaheuristique à Population de Solution

II.3.2.1 Colonie de Fourmis

L'optimisation des colonies de fourmis (Ant Colony Optimization, ACO) a été initiée dans les années 1990 par Marco Dorigo. Il est dérivé du comportement des fourmis réelles à la recherche de nourriture et visait à résoudre des problèmes combinatoires complexes. L'ACO et appliquée la première fois pour résoudre le problème du Voyageur de commerce.

Le principe de cette métaheuristique repose sur le comportement spécifique des fourmis. Elles communiquent localement et indirectement en utilisant une hormone volatile appelée phéromone. Lorsqu'elles se déplacent, les fourmis laissent une trace de phéromone derrière elles. Ensuite, elles choisissent leur chemin de manière aléatoire, en fonction de la quantité de phéromone déposée précédemment. **[21]**. La Figure II.4 présente l'expérience de sélection du chemin le plus court.

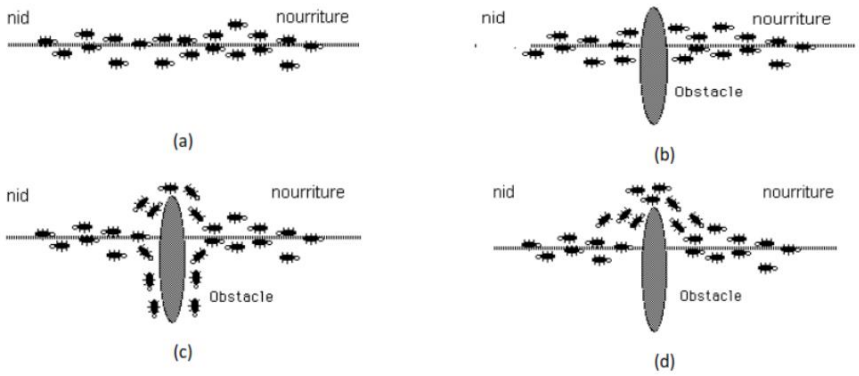

Figure II. 4: l'expérience de sélection du plus court chemin.

L'ACO est un Ant System orienté, Les colonies de fourmis artificielles exploitent cette caractéristique pour construire des solutions à un problème d'optimisation. Elles construisent des solutions en parcourant un graphe de construction complet GC (V, E) dont les nœuds V sont des composants de solutions, et les arêtes E sont des connexions entre les composants. **[11]**

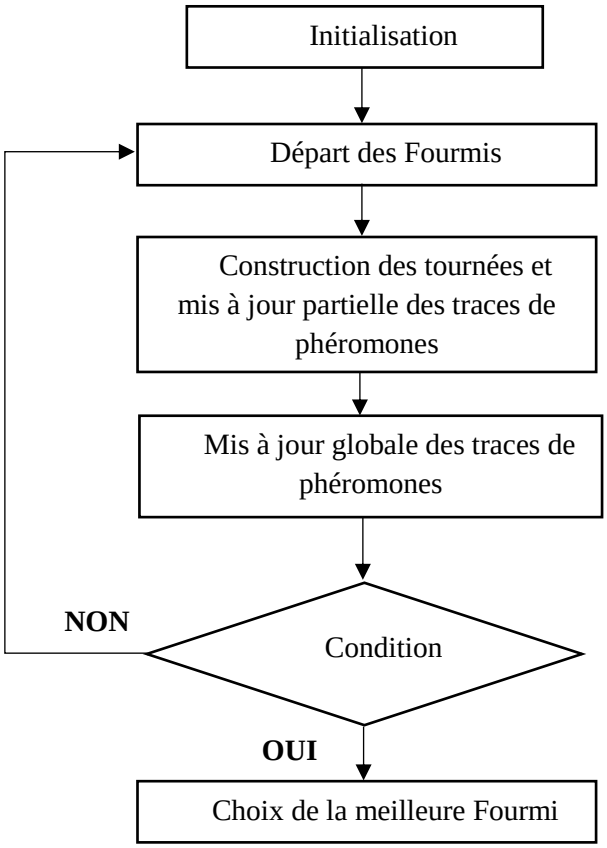

Figure II. 5: Organigramme de Colonie de Fourmis.

II.3.2.2 L'optimisation par Essaim de Particules

La technique d'optimisation par essaim de particules (Particle Swarm Optimization, PSO), développée par James Kennedy et Russell Eberhart en 1995, est une métaheuristique qui s'inspire du comportement social des essaims d'oiseaux et de poissons. Elle est utilisée comme méthode d'optimisation stochastique pour résoudre des problèmes d'optimisation continue, discrète et combinatoire.

Dans l'algorithme PSO une population d'individus, appelés "particules", évolue dans un espace de recherche multidimensionnel. Dans lequel elle est basée sur un essaim de particules, chacune représentant une solution à un problème à N dimensions, avec des paramètres ajustés itérativement pour converger vers la solution optimale.

II.3.2.3 Algorithme Génétique

L'algorithme génétique (AG) l'un des algorithmes évolutionnaires et le plus populaires. Les AG ont été conçus dès le début des années 1970 grâce à John Holland, puis David Goldberg. Le principe s'inspire des mécanismes de sélection naturelle et de la génétique, et consiste à simuler le principe d'évolution biologique.

Les algorithmes génétiques sont des approches d'optimisation qui utilisent des techniques dérivées de la science génétique et de l'évolution naturelle (la sélection, la mutation et le croisement). Ils reposent sur une population de solutions potentielles, représentées sous forme de chaînes de gènes, pour représenter les individus de la population, et évoluent de génération en génération. Pour mener des recherches de haute qualité dans un espace de recherche.

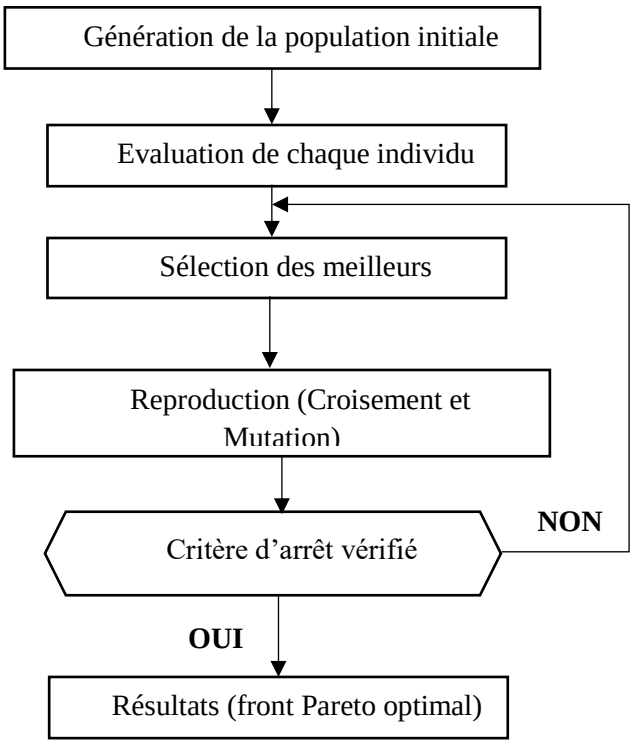

Figure II. 6 : Organigramme d'Algorithme génétique.

Les algorithmes génétiques s'appuient alors sur trois opérateurs d'évolution :

- La sélection : permet d'identifier les individus les plus performants d'une population et d'éliminer les moins performants lors de la transition entre les générations. Ce processus est basé sur la performance individuelle. Il existe différents modes de sélection. **[6]** :
  - La sélection des tournois.
  - La sélection proportionnelle (Roulette) (Goldberg 1989).
  - La sélection de classement.
  - La sélection en régime permanent.

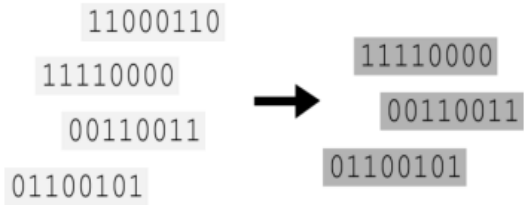

Figure II. 7: Opérateur de Sélection.

- Le Croisement : L'opérateur de croisement fusionne les caractéristiques d'un ensemble de parents sélectionnés (généralement deux) et génère des descendants. Il existe plusieurs opérateurs de croisement disponibles. **[11] :**
    - Le croisement en un point.
    - Le croisement en n-points ($n \geq 2$).
    - Le croisement uniforme.

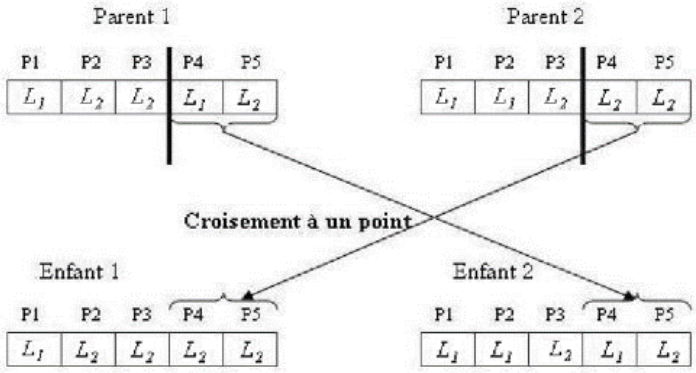

Figure II. 8: Croisement en Un Point

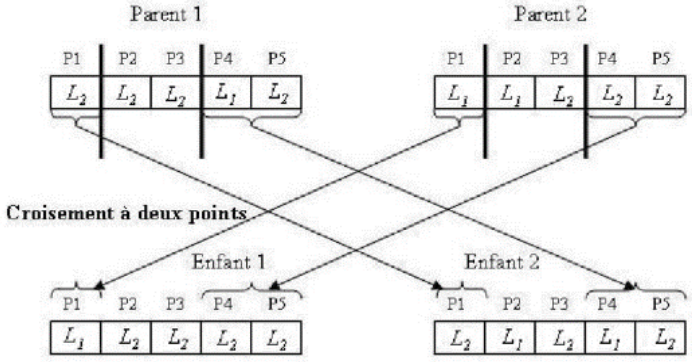

Figure II. 9: Croisement en Deux Point.

- La Mutation : Leur génotype est aléatoirement modifié en utilisant l'opérateur de mutation. **[11].**

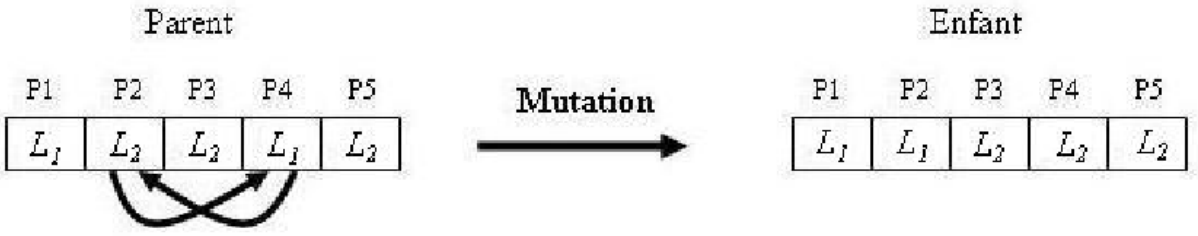

Figure II. 10: Opérateur de Mutation.

II.3.2.4 Algorithme d'optimisation de la recherche Pingouins (PeSOA)

L'algorithme PeSOA est une nouvelle métaheuristique inspirée de la nature, qui repose sur le comportement de chasse collaboratif des pingouins. Les pingouins utilisent une stratégie de chasse basée sur la collaboration et la synchronisation de leurs plongées, afin d'optimiser l'énergie globale dans le processus de chasse collective et de nutrition.

II.3.2.4.1.1 La description d'algorithme PeSOA

L'algorithme d'optimisation basé sur le comportement de chasse des pingouins peut être décrit de différentes manières. Les règles ci-dessous résument les principes fondamentaux de cet algorithme et guident la stratégie de recherche des pingouins **[22]** :

- **Règle 1** : Généré une population des Penguins subdivisé en plusieurs groups.
- **Règle 2** : Chaque groupe est composé d'un nombre variable de pingouins pouvant varier en fonction de la disponibilité de nourriture dans un endroit spécifique.
- **Règle 3** : Ils chassent en groupe et se déplacent de manière aléatoire jusqu'à ce qu'ils trouvent de la nourriture lorsque leurs réserves d'oxygène ne sont pas épuisées.
- **Règle 4** : Ils peuvent effectuer des plongées simultanées à une profondeur identique.
- **Règle 5** : Chaque groupe de pingouins commence sa recherche dans une position spécifique (le trou "i") et à des niveaux aléatoires (niveaux "j1, j2, ..., jn").
- **Règle 6** : Chaque pingouin cherche de la nourriture de manière aléatoire et individuellement dans son groupe, et après un certain nombre approximatif de plongées, les pingouins reviennent sur la glace pour partager avec leurs compagnons la localisation (représentée par le niveau ou la profondeur de la plongée) où ils ont trouvé de la nourriture en abondance (représentée par la quantité de poissons mangés). Cette règle assure la communication intra-groupe.
- **Règle 7** : À un niveau donné, on peut avoir de 0 à N pingouins (pingouin ou tout autre groupe) en fonction de l'abondance de nourriture.
- **Règle 8** : Si le nombre de poissons dans un trou n'est pas suffisant (ou nul) pour le groupe, une partie du groupe (ou tout le groupe) migre vers un autre trou. (Cette règle assure la communication inter-groupes).
- **Règle 9** : Le groupe qui a mangé le plus de poissons nous donne l'emplacement de la nourriture abondante représentée par le trou et le niveau.

II.3.2.4.1.2 Le principe d'algorithme PeSOA

- Dans l'algorithme, chaque pingouin est représenté par le trou "i" et le niveau "j", ainsi que par le nombre de poissons mangés.
- La distribution des pingouins est basée sur les probabilités d'existence de poissons dans les trous et les niveaux.
- Les pingouins sont divisés en groupes (sont pas nécessairement la même cardinalité).
- La recherche de nourritures commence aléatoirement (à des position aléatoires).
- Après un nombre fixe de plongées (recherche), les pingouins retournent sur la glace pour partager avec leurs semblables la profondeur (niveau) et la quantité (nombre) de nourriture trouvée (communication intergroupe).

- Les pingouins d'un ou de plusieurs groupes avec peu de nourriture suivent lors de la plongée suivante, les pingouins qui ont chassé beaucoup de poissons. Ce déplacement est exprimé par la formule suivante **[22]** :

$$D_{new} = D_{LastLast} + \text{rand}() \, |X_{LocalBest} - X_{LocalLast}| \quad \ldots\ldots\ldots (1)$$

Nous avons introduit le principe de l'algorithme des pingouins en utilisant un schéma explicatif illustré par la figure II.11.

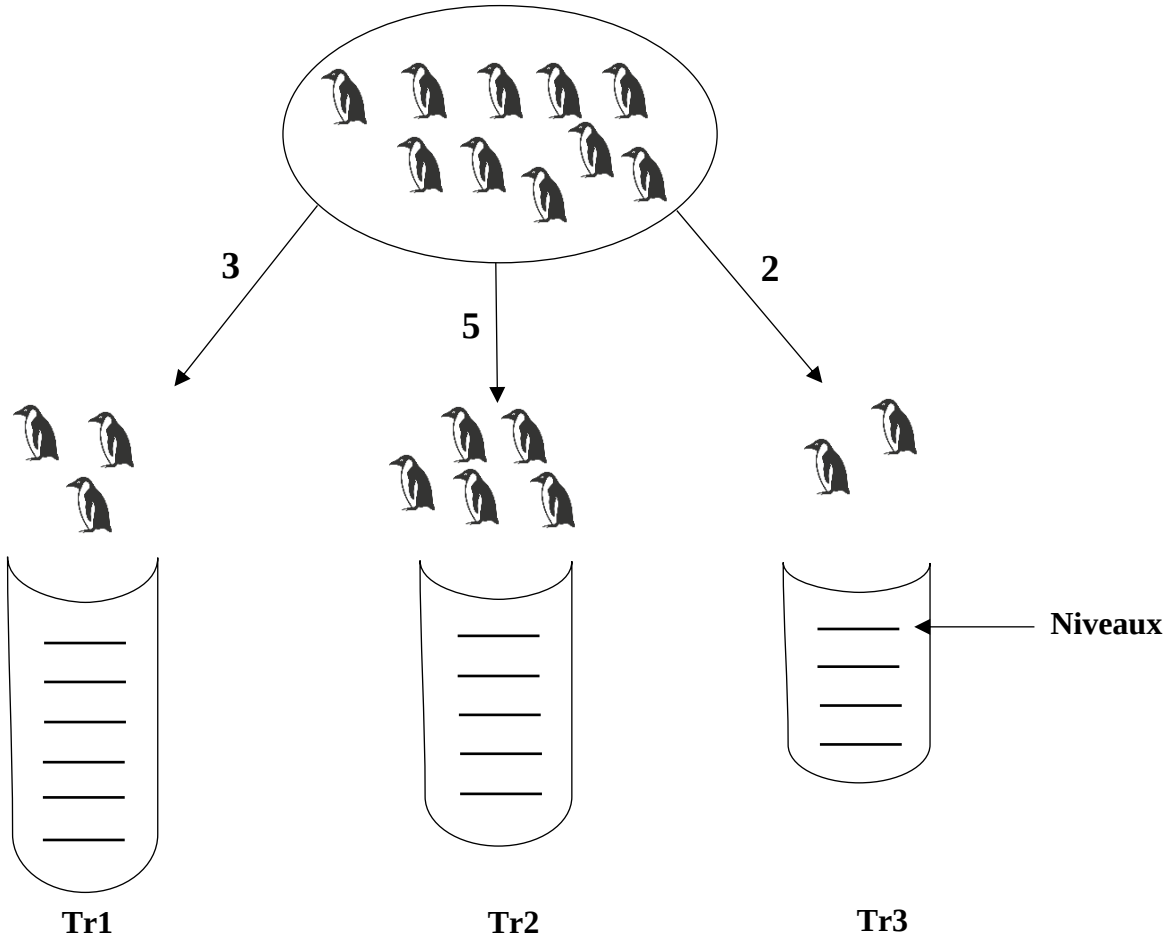

Figure II. 11: Schéma explicatif pour le principe d'algorithme PeSOA

Cette figure représente le comportement de recherche de nourriture d'un groupe de 10 pingouins. Ces derniers explorent trois trous situés à différents niveaux, variant de 4 à 6.

II.3.2.4.1.3 L'algorithme PeSOA

Algorithm de PeSOA

1 : Générer une population aléatoire POPinitl de P solution (penguins) ;

2 : Diviser la POP en N groupes ;

3 : Initialiser la probabilité d'existence de poissons dans les trous et les niveaux ;

4 : Pour i=1 jusqu'à critère d'arrêt faire

5 : Pour chaque individu i de chaque groupe (dans P) faire

6 : Tant que les réserves d'oxygène ne sont pas épuisées faire

7 : - Prendre une direction aléatoire.

8 : - Améliorer la position du pingouin en utilisant les équations (1).

9 : - Mettre à jour les quantités de poissons mangés par ce pingouin.

10 : Fin tant que

11 : Fin pour chaque

12 : - Mettre à jour les quantités de poissons mangés dans les trous, les niveaux et le meilleur groupe.

13 : - Redistribuer les probabilités (les pingouins dans les trous et les niveaux (ces probabilités sont calculées en fonction du nombre de poissons mangés)).

14 : - Mettre à jour la meilleure solution.

15 : Fin pour

Figure II. 12: Algorithme d'optimisation de la recherche Pingouins Générale

II.4 Intensification et Diversification

Les métaheuristiques reposent sur les concepts d'intensification et de diversification, et leur objectif est de découvrir de nouvelles solutions. Les principes d'intensification et de diversification sont des éléments critiques des méthodes d'approximation, permettant d'éviter les optima locaux.

- L'intensification : également connue sous le nom d'exploitation, représente la stratégie consistant à concentrer les efforts de recherche sur les régions prometteuses de l'espace de recherche. Cette approche implique l'utilisation des informations déjà acquises pour identifier et explorer les zones les plus susceptibles de contenir l'optimum global. En se focalisant sur ces régions, on cherche à améliorer la solution actuelle en exploitant les connaissances accumulées.
- Diversification : La diversification, ou exploration, consiste à étendre le domaine de recherche afin d'éviter de se limiter à des optima locaux et de trouver des solutions potentiellement meilleures. Cette approche permet d'explorer de manière plus exhaustive l'espace des solutions et d'éviter de rester bloqué sur une solution sous-optimale.

II.5 Conclusion

Dans ce chapitre, nous avons discuté en détail des principales méthodes de résolution des problèmes d'ordonnancement. Ces méthodes ont été largement étudiées et continuent de jouer un rôle central dans la résolution de divers problèmes NP-difficiles, y compris le problème d'ordonnancement.

La présentation globale de ce chapitre présente les principales métaheuristiques, ainsi que leurs principes fondamentaux et leurs algorithmes de base.

Chapitre III

ADAPTATION DE L'ALGORITHME GENETIQUE ET L'ALGORITHME D'OPTIMISATION DE LA RECHERCHE PINGUINS (PeSOA)

III.1 Introduction

Les ateliers de type flow shop se distinguent par une production linéaire où toutes les opérations sont effectuées dans une séquence identique pour tous les produits. L'objectif de cette étude est d'optimiser le makespan (Cmax) en utilisant deux métaheuristiques : l'algorithme génétique et l'optimisation par recherche des manchots. L'objectif est d'obtenir des solutions optimales.

Ce chapitre présente de manière concise le principe et l'adaptation de ces deux méthodes pour résoudre le problème d'ordonnancement de type flow shop proposé. De plus, une comparaison des résultats obtenus avec ces deux techniques est également effectuée.

Le chapitre est divisé en trois sections. La première section décrit le problème, tandis que les deux dernières sections se concentrent sur l'adaptation des deux méthodes retenues et les différents paramètres de chaque méthode. Ensuite, les résultats de simulation respectifs sont présentés et le chapitre se termine en interprétant ces résultats.

III.2 Problème de Flow Shop

La plupart des problèmes d'ordonnancement sont classés comme problèmes d'optimisation NP-difficiles. Les problèmes d'ordonnancement sont généralement définis par quatre éléments principaux : les tâches, les ressources, les contraintes et les objectifs.

Les problèmes d'ordonnancement des systèmes de production ont été largement analysés et élaborés selon différentes approches (Błazewicz, et al (1996) ; Allahverdi et al (1999) ; Kis (2003) ; Allahverdi et al (2008) ; Kis and Kovacs (2012) et Allahverdi (2015). Les résultats dans ce domaine ont contribué à l'amélioration des systèmes de fabrication (Błazewicz, et al (2007). **[23]**

Le Problème d'Ordonnancement Flow Shop (FSSP), Ce problème a été proposé pour la première fois en 1954 par Johnson et classifié comme un problème NP-difficile **[24]**. Est un composant crucial de l'organisation des tâches sur une période spécifique en tenant compte des contraintes temporelles (comme les délais et les contraintes d'enchaînement) ainsi que des limitations de l'utilisation et de la disponibilité des ressources **[25].** Les configurations de type flow-shop sont courantes dans les environnements de fabrication où un ensemble de jobs N = {1, 2, …, n} sont traités par un ensemble de machines M = {1, 2, …, m}. Chaque travail passe par les machines dans le même ordre technologique, c'est-à-dire qu'il commence à la machine 1, puis passe à la machine 2, … jusqu'à la machine m. La décision à prendre est de choisir l'ordre dans lequel les différents travaux passeront par les machines. **[23]**

Chaque travail j contient une séquence d'opérations O = {O_{ji}, j ∈ {1, ..., n} et i ∈ {1, ..., m}} qui doivent être exécutées dans un ordre donné. Suivant cet ordre, chaque opération O_{ij} doit être réalisée sur une machine spécifiée k pendant un temps spécifié P_{ijk}. Chaque machine ne peut traiter qu'un travail à la fois, et chaque travail doit passer par chaque machine une seule fois et une seule fois. Le FSSP consiste à trouver un ordonnancement pour effectuer les opérations sur les machines qui minimise le Cmax (temps d'exécution maximum), c'est-à-dire le temps nécessaire pour réaliser tous les travaux. **[24]**

III.2.1 Formulation mathématique

- **Indices :**

i : indice de job, *i*=1...n.

j : indice de machines, j=1...m.

n : nombre de job

m : nombre de machine.

k : indice de position, k=1...n.

- **Paramètre :**

$P_{i,j}$= temps de traitement du job i sur la machine j.

- **Variables de Décision :**

$X_{i,k}$= si le job i est attribué à la position k de toutes les machines, alors c'est 1, sinon 0.

$P_{[k],j}$= temps de traitement du job assigné au position k dans la machine j.

$C_{[k],j}$= temps d'achèvement du travail assigné au position k dans la machine j.

$S_{[k],j}$= heure de début du job assigné au position k dans la machine j.

C_{max}= makespan of the schedule.

- **La fonction Objectif :**

$$Min\ z = C_{max} \quad (1)$$

- **Contraintes :**

$$C_{max} \geq C_{[n],m} \quad (2)$$

$$\sum_i^n Xi, k = 1 \quad \forall k \quad (3)$$

$$\sum_k^n Xi, k = 1 \quad \forall i \quad (4)$$

$$C_{[k],j} \geq S_{[k],j} + P_{[k],j} \quad \forall k, j \quad (5)$$

$$S_{[k],j} \geq C_{[k],j} - 1 \quad \forall k, j = 2, \dots, m \quad (6)$$

$$S_{[k],j} \geq C_{[k-1],j} \quad \forall j, k = 2, \dots, m \quad (7)$$

$$P_{[k],j} = \sum_i^n Xi, k * Pi, j \quad \forall k, j \quad (8)$$

$$C_{[0],1} = 0 \quad (9)$$

$$C_{max} \geq 0 \quad (10)$$

$$C_{[k],j}, P_{[k],j}, S_{[k],j} \geq 0 \quad \forall k, j \quad (11)$$

$$X_{i,k} \in \{0,1\} \quad \forall i, k \quad (12)$$

La fonction objectif (1) minimise le temps maximum d'exécution du planning. La contrainte (2) est de déterminer le Makespan. Les contraintes (3) et (4) garantissent que chaque job doit être affecté à une seule position et que chaque position doit être utilisé pour un seul job. La contrainte (5) montre la relation entre les temps d'achèvement, de début et de traitement de la tâche affectée au position *k*. La contrainte (6) montre que

l'heure de début d'un job dans la machine j doit être supérieure ou égale au temps d'achèvement du même job dans la machine $(j-1)$. La contrainte (7) montre que l'heure de début du job en position k doit être supérieure supérieur ou égal au temps d'achèvement du job en position $(k-1)$ dans la même machine. La contrainte (8) détermine le calcul du temps de traitement du job en position k dans la machine j. La contrainte (9) exprime que tous les travaux peuvent être traités au temps zéro sur la première machine. Les contraintes (10-12) montrent les domaines nécessaires des variables de décision. **[26]**

III.3 Adaptation de l'algorithme génétique

Les Algorithmes Génétiques sont largement reconnus par de nombreux chercheurs comme une méthode hautement appropriée pour résoudre le problème de type Flow Shop, même s'ils pourraient ne pas atteindre l'optimum dans certains cas complexes.

Nous présentons une adaptation de l'algorithme génétique pour résoudre le problème mentionné précédemment "FSSP". Notre application porte uniquement sur l'Algorithme Génétique standard, c-à-d, la version de base caractérisé par une population d'individus générés aléatoirement de sélection, un opérateur de croisement, et un opérateur de mutation.

III.3.1 Les paramètres d'algorithme génétique

La mise en œuvre de notre algorithme génétique implique :

- La détermination de la Taille de Population.
- L'implémentation d'une représentation codée adaptée au problème.
- L'implémentation des différents paramètres de la méthode "La Sélection, Le Croisement, et La Mutation".
- L'évaluation des individus.

III.3.1.1 La taille de population

Avant de commencer l'Algorithme Génétique, il est nécessaire de créer une population initiale. Cette population initiale doit être hétérogène afin de servir de base pour les générations suivantes. Dans notre application, nous générons aléatoirement cette population initiale en utilisant des chromosomes aléatoires au lieu de véritables séquences ordonnées. Cette approche est applicable à tous les types de codage utilisés.

III.3.1.2 Le codage

Le codage est un élément crucial pour garantir l'efficacité de la méthode. Dans le contexte de l'ordonnancement, il fait référence à la transformation d'un ordonnancement réel en un code adapté qui permet la mise en œuvre des différents opérateurs génétiques.

En conséquence, chaque chromosome généré, qui représente également une solution potentielle, correspond forcément à un ordonnancement concret, c'est-à-dire à une séquence potentielle des différentes opérations effectuées sur les différentes machines.

Dans cette étude, nous appliquons le codage basé sur les tâches. Dans ce codage, chaque gène d'un chromosome représente un travail. Un chromosome, dans le contexte d'un algorithme génétique, correspond à une solution potentielle du problème, également connue sous le nom d'"individu".

Dans un problème d'ordonnancement Flow Shop, une solution réalisable se traduit par une séquence de jobs. Par exemple, si nous avons 5 jobs à traiter (j1..., j5), la figure II.1 présente un exemple du codage proposé dans notre étude.

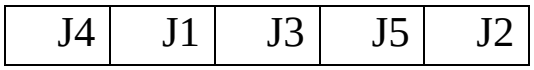

Figure III. 1: Exemple d'un Codage d'un Chromosome

III.3.1.3 La sélection

La sélection est un processus visant à favoriser au fil du temps la reproduction des individus les mieux adaptés. Elle consiste à choisir deux individus, appelés "parents", pour créer deux autres individus, appelés "enfants".

Il existe différentes méthodes de sélection. Dans notre approche, nous avons opté pour la sélection par roulette, où les individus sont choisis successivement, deux par deux.

III.3.1.4 Le croisement

Le croisement, un opérateur essentiel dans les Algorithmes Génétiques, est appliqué à chaque paire de parents sélectionnés. Il existe différentes façons de l'appliquer, telles que le croisement à un point ou à deux points. Dans notre cas, nous utilisons le croisement à un point, où chaque chromosome des parents est divisé en deux parties pour créer deux nouveaux enfants.

La manière dont nous appliquons cet opérateur de croisement peut varier en fonction du problème. La Figure illustre la technique de croisement.

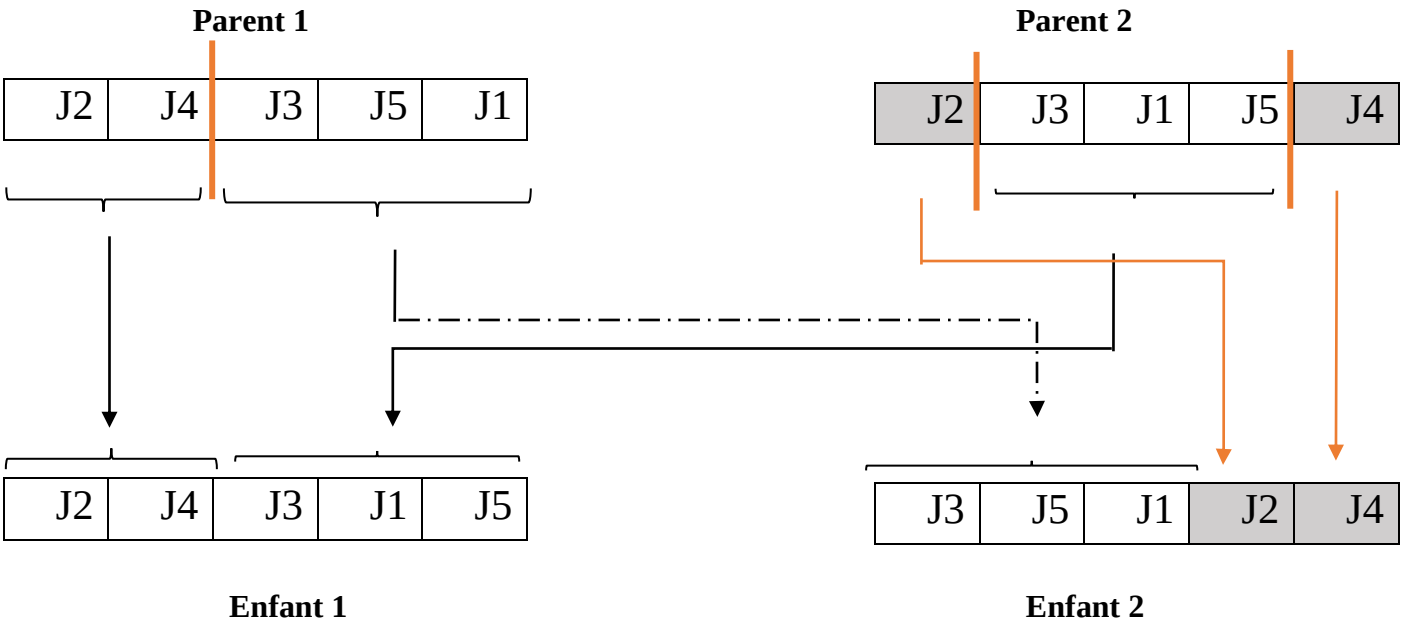

Figure III. 2: Exemple de Croisement Appliqué sur deux individus

III.3.1.5 La mutation

Le rôle de la mutation est d'introduire de la diversité au sein de la population et de prévenir une convergence trop rapide vers un seul phénotype. Cela se produit généralement par une légère modification de chaque chromosome, qui se manifeste le plus souvent par l'échange de deux ou plusieurs allèles. Dans notre étude, nous avons choisi la méthode standard de mutation basée sur la permutation de deux allèles sélectionnés au hasard le long du chromosome considéré.

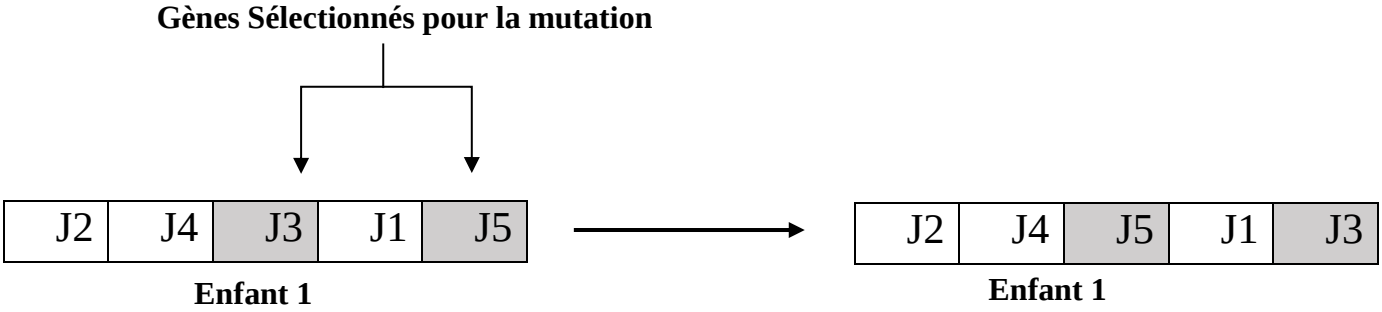

Figure III. 3: Exemple de Mutation

III.3.1.6 L'évaluation des individus

Nous avons utilisé le makespan comme seul critère pour évaluer les individus. Le fitness des individus est donc estimé en fonction de ce critère. L'évaluation du fitness des chromosomes consiste à calculer le makespan, qui représente la fonction objectif (Cmax).

III.3.2 La simulation de l'algorithme génétique

Pour réaliser les simulations, nous avons généré un ensemble d'instances basées sur un système de production qui présente différentes classes de problèmes de différentes tailles. Dans cette étude, nous avons pris en compte quatre classes de problèmes différentes : 10 jobs/30 machines, 20 jobs/30 machines, 40 jobs/30 machines et 50 jobs/30 machines. Chaque classe de problèmes comprend cinq exemples avec des temps de traitement différents. Nous avons fixé une taille de population de 60 et un coefficient de mutation de 10. Le tableau présente les résultats de la simulation de l'algorithme génétique classique.

	Cmax			
	10jobsx30machines	**20jobsx30machines**	**40jobsx30machines**	**50jobsx30machines**
Exemple 1	727	1051	1488	1731
Exemple 2	771	1069	1479	1704
Exemple 3	730	1012	1482	1708
Exemple 4	776	1032	1472	1734
Exemple 5	790	1077	1483	1759

Tableau III. 1: Les résultats de simulation de l'Algorithme Génétique

III.4 Adaptation de l'algorithme de pingouins pour la résolution du problème d'ordonnancement de type Flow Shop

III.4.1 Le principe de l'algorithme

L'algorithme d'optimisation par recherche des pingouins, développé par Gheraibia et Moussaoui en 2013, est un algorithme d'intelligence computationnelle qui s'inspire du comportement de recherche de nourriture des pingouins dans leur milieu naturel. Dans le cadre de cet algorithme, les pingouins sont regroupés et chaque groupe se voit attribuer une région distincte dans l'espace de recherche. Durant leur quête de nourriture, chaque pingouin explore de manière aléatoire et indépendante la région qui lui a été assignée, jusqu'à épuisement de ses réserves d'oxygène. Lorsqu'ils retournent sur la glace, les pingouins partagent les informations sur les endroits où ils ont trouvé de la nourriture (trou et niveau) avec les autres membres de leur groupe. Ces informations sont ensuite utilisées pour améliorer le mécanisme de recherche, permettant ainsi à chaque pingouin de suivre à la fois l'optimum local individuel et l'optimum global du groupe, afin de trouver l'espace de solution optimal.

Dans cette section, nous expliquons les différents paramètres de l'Algorithme pour notre adaptation, à savoir : la POPinitial, Un pinguins, un trou et un niveau.

III.4.2 Les Paramètres d'Algorithme d'optimisation de la recherche Pingouins

La mise en œuvre de notre algorithme PeSOA :

- La Détermination de la POPinitial.
- La Division de POP en N group.
- Création des Trous.
- Affectation des Niveaux (Séquences) dans chaque trou.

III.4.2.1 La POPinitial des pinguins

Représente la population initiale des pinguins. Un pinguins représente dans notre adaptation une procédure de recherche. L'ensemble de la population des pinguins représente l'ensemble des recherches parallèles.

III.4.2.2 Les N groupes

Représentent les N recherches parallèles.

III.4.2.3 Un Trou

Représente un ensemble de solutions possibles pour notre problème. Le nombre de trous représente le nombre des sous-ensembles de l'espace de recherche. Chaque trou contient un nombre de niveaux.

III.4.2.4 Un Niveau

Le niveau représente une solution possible pour le problème (dans notre cas, il représente une séquence de jobs).

Le nombre de niveaux n'est pas le même dans tous les trous. Le nombre de niveaux dans le même trou peut changer lors de l'étape de redistribution (Etape 13).

III.4.3 L'algorithme PeSOA Adapté

Algorithm de PeSOA Adapté

1 : Générer une population aléatoire POPinitl des penguins P (Recherche) ;

2 : Diviser la POP en N groupes ;

3 : Initialiser la probabilité d'existence de poissons dans les trous et les niveaux (Compteur) ;

4 : Tant que le critère d'arrêt n'est pas attenu faire

5 : Pour chaque individu de P faire

6 : Tant que les réserves d'oxygène (Compteur inverse) ne sont pas épuisées faire

7 : - Prendre une direction aléatoire (Orientation des penguins vers les niveaux).

8 : - Améliorer la position du pingouin (Classer les recherches selon la qualité des solution obtenus, puis chaque mauvais penguins doit suivre un nouveau group).

9 : - Mettre à jour les quantités de poissons mangés par ce pingouin (nombre d'améliorations des solutions).

10 : Fin tant que

11 : Fin pour

12 : - Mettre à jour les quantités de poissons mangés dans les trous, les niveaux et le meilleur groupe (classer les solutions et crée les nouveaux niveaux).

13 : - Redistribuer (les pingouins et les niveaux (Classer les solutions et remplacer les mauvais)).

14 : - Mettre à jour la meilleure solution.

15 : Fin pour

Figure III. 4: Algorithme d'optimisation de la recherche Pingouins Adapté

III.4.4 Résultats de Simulation de l'algorithme de pingouin

Dans cette section, nous présenterons les résultats des simulations pour les diverses catégories de problèmes étudiées.

	Cmax			
	10jobsx30machines	**20jobsx30machines**	**40jobsx30machines**	**50jobsx30machines**
Exemple 1	723	1035	1499	1723
Exemple 2	756	1054	1481	1702
Exemple 3	748	1010	1465	1699
Exemple 4	768	1022	1457	1685
Exemple 5	803	1051	1448	1716

Tableau III. 2: Les résultats de Simulation de l'algorithme de pingouin

III.4.5 Interprétation de résultats de Simulation

Après avoir effectué la simulation, nous avons constaté que l'algorithme des pingouins est stable et performant dans différents scénarios de simulation pour notre problème. Le tableau montre que cet algorithme gère efficacement les tâches sur les machines, avec un makespan qui augmente progressivement en fonction du nombre de tâches.

Il est intéressant de noter que l'algorithme des pingouins s'adapte bien à l'augmentation progressive du nombre de tâches, avec un Cmax qui augmente proportionnellement.

Les variations du Cmax entre les différents exemples, pour un même nombre de tâches, restent assez limitées, ce qui témoigne de sa fiabilité.

À mesure que le nombre de tâches augmente, le Cmax augmente également, conformément aux attentes. Cependant, cette augmentation semble linéaire et maîtrisée, ce qui suggère que l'algorithme des pingouins est évolutif et peut gérer une augmentation du nombre de tâches sans dégradation excessive des performances.

III.5 Comparaison de résultats de simulation entre l'algorithme génétique et l'algorithme des pinguins

Dans cette section, nous présentons les résultats des simulations sous forme de graphiques pour l'algorithme des pingouins et l'algorithme génétique (voir la Figure III.5).

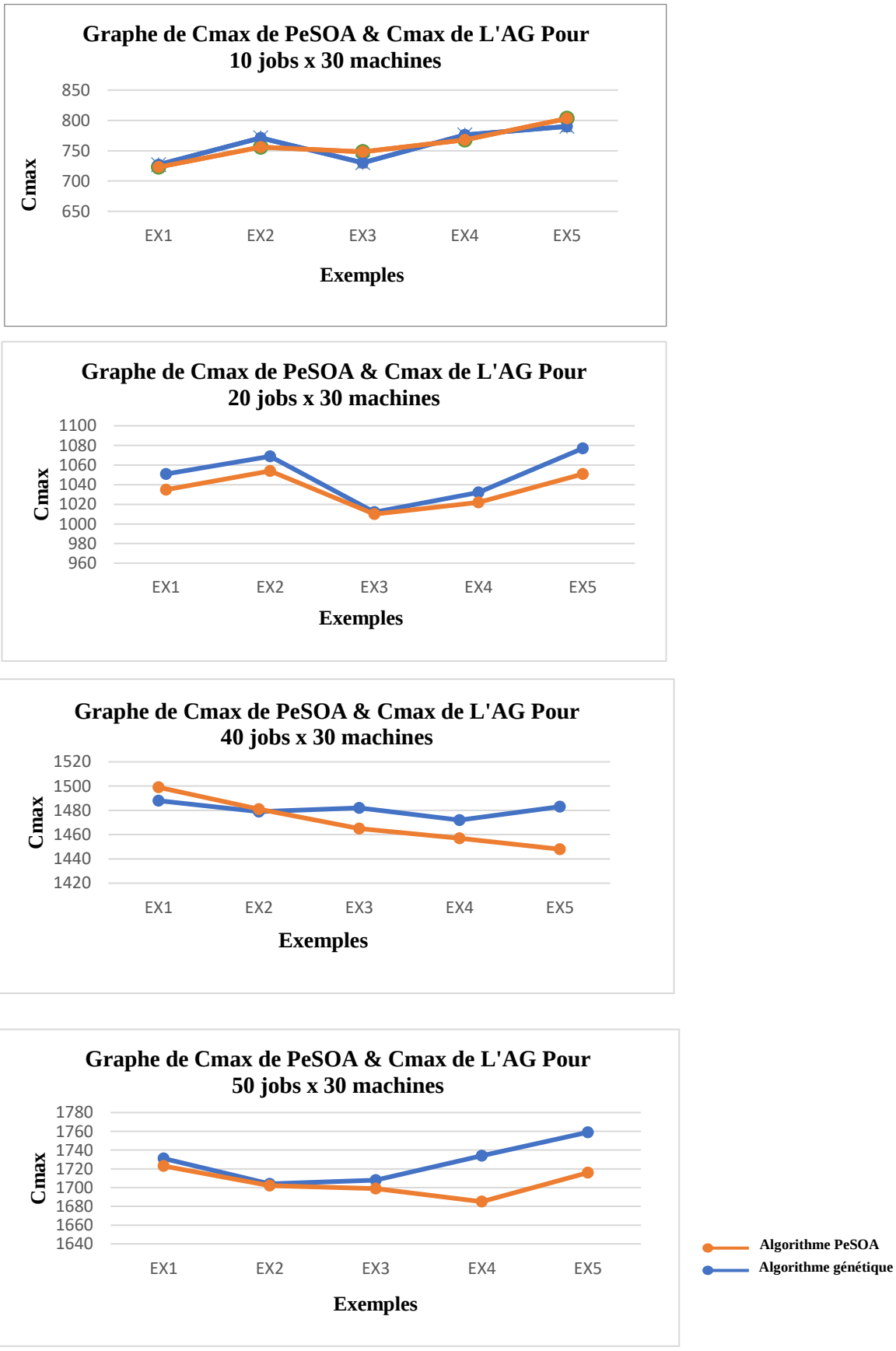

Figure III. 5: des graphes présente les résultats de la simulation de PeSOA et de l'AG.

Après avoir effectué des simulations et analysé les résultats présentés dans les tableaux III.1 et III.2, nous avons constaté que l'algorithme des pingouins présente un léger avantage en termes de minimisation du makespan (Cmax), avec des résultats proches de la solution optimale.

Il est intéressant de noter que l'algorithme des pingouins montre généralement de meilleures performances, voire des performances comparables, à celles de l'algorithme génétique pour les configurations données. Dans la plupart des cas, les valeurs de Cmax sont plus basses ou équivalentes.

Cependant, l'algorithme des pingouins semble être légèrement plus efficace pour analyser l'évolutivité des configurations.

La Figure III.5 présente les résultats de simulation sous forme de graphiques, illustrant les différences entre les différents algorithmes.

III.6 Etude de sensibilité d'algorithme d'optimisation de la recherche Penguins

L'analyse de sensibilité consiste à étudier la réaction de l'algorithme aux variations de ses paramètres. Nous réalisons cette étude en utilisant 05 exemples de la classe 10jobs/30machines.

L'étude de sensibilité de l'algorithme des pingouins est réalisée en fonction du nombre de pingouins et du nombre de trous.

Pour cette analyse, nous utiliserons deux exemples. Dans le premier exemple, nous maintiendrons la valeur de N (nombre de pingouins) constante et ferons varier la valeur de Tr (nombre de trous) dans le tableau III.3. Dans le deuxième exemple, nous maintiendrons la valeur de Tr constante et ferons varier la valeur de N dans le tableau III.4.

	10 jobs x 30 machines			
	Cmax			
Tr (Trous)	**4**	**6**	**8**	**10**
Exemple 1	719	715	712	717
Exemple 2	762	762	751	758
Exemple 3	746	735	730	740
Exemple 4	770	765	760	758
Exemple 5	808	796	784	788

Tableau III. 3: Analyse de sensibilité de L'Algorithme PeSOA pour 5 exemples de 10 jobs 30 machines (N, Fixé et Tr, varier)

Dans ce tableau, nous fixons le nombre de pingouins à 80 et faisons varier le nombre de trous entre 4 et 10.

Remarque : Si le nombre de trous dépasse 10, le nombre de recherches parallèles augmente, ce qui entraîne une simulation plus longue pour obtenir des résultats. Nous limitons donc le nombre de trous entre 4 et 10.

	10 jobs x 30 machines			
	Cmax			
N (Nombre de Pinguins)	**50**	**60**	**70**	**80**
Exemple 1	730	724	716	712
Exemple 2	785	772	752	751
Exemple 3	743	746	753	730
Exemple 4	762	755	763	760
Exemple 5	798	791	800	784

Tableau III. 4: Analyse de sensibilité de L'Algorithme PeSOA pour 5 exemples de 10 jobs 30 machines (N, varier et Tr, Fixé)

Dans ce tableau, on fixe le nombre de Trous à 8, et on fait varier le nombre de Pinguins entre 50 et 80.

Les deux tableaux présentés illustrent les résultats des analyses de sensibilité de l'algorithme PeSOA (Penguin Search Optimization Algorithm) dans le contexte des problèmes d'ordonnancement de tâches dans un atelier flow shop. Ces problèmes consistent à exécuter 10 jobs sur 30 machines. Les tableaux examinent spécifiquement l'impact des variations de certains paramètres sur le Cmax, qui représente le temps de complétion maximal des tâches. Cela permet de fournir une interprétation pour cette étude.

Le tableau III.3 présente les résultats de l'analyse de sensibilité de l'algorithme PeSOA en fixant le nombre de pingouins à 80 (N) et en variant le nombre de trous entre 4,6,8 et 10 (Tr). Les résultats montrent que lorsque le nombre de trous augmente, l'algorithme parvient à trouver des solutions plus efficaces pour certains exemples, bien que cette amélioration soit marginale dans certains cas.

En ce qui concerne le tableau III.4, qui présente les résultats de l'analyse de sensibilité de l'algorithme PeSOA en fixant le nombre de trous à 8 (Tr) et en variant le nombre de pingouins entre 50,60,70 et 80(N), on constate une légère amélioration du Cmax à mesure que le nombre de pingouins augmente. Cela indique que l'augmentation du nombre de pingouins (agents de recherche) permet à l'algorithme d'explorer plus efficacement l'espace des solutions, ce qui conduit à de meilleures performances pour certains exemples, même si les gains restent relativement modestes.

Ces analyses de sensibilité sont cruciales pour optimiser les paramètres de l'algorithme PeSOA, et elles montrent que l'augmentation du nombre de trous et du nombre de pingouins semble légèrement améliorer les performances en termes de réduction du Cmax. Cela peut être dû à une meilleure diversification des solutions explorées, ainsi qu'à une exploration plus approfondie et plus efficace de l'espace des solutions.

III.7 Conclusion

Dans ce chapitre, nous avons exploré l'utilisation de deux algorithmes pour résoudre le problème de l'ordonnancement de la production. Nous avons adapté l'algorithme génétique et l'algorithme des pingouins à notre problème spécifique.

Une fois les deux algorithmes adaptés et les simulations effectuées, nous avons analysé les résultats obtenus afin de déterminer les meilleurs paramètres pour chacun d'entre eux.

En conclusion, nous pouvons affirmer que l'algorithme des pingouins peut être appliqué aux problèmes d'ordonnancement de la production dans un atelier flow shop, en fournissant des résultats satisfaisants et efficaces.

CONCLUSION GENERALE

Conclusion Générale

Le travail présenté dans ce mémoire concerne l'étude d'un problème industriel relatif à l'ordonnancement d'ateliers, qui est classifié comme un problème NP-difficile. Pour résoudre ces problèmes, plusieurs approches sont utilisées, notamment les métaheuristiques.

Les métaheuristiques sont principalement inspirées de la nature et du comportement des organismes vivants. Ces algorithmes peuvent s'adapter au problème posé et être étudiés afin d'améliorer les solutions proposées.

En effet, il existe un grand nombre de métaheuristiques qui permettent d'obtenir des solutions optimales pour nos problèmes, en fonction de leur complexité. Parmi les problèmes couramment rencontrés dans l'ordonnancement de la production, on trouve les problèmes d'atelier flow shop.

Les problèmes d'atelier flow shop sont des problèmes d'ordonnancement où une seule séquence d'opérations doit être déterminée pour l'ensemble des tâches, dans le but de minimiser le temps total d'exécution (makespan).

Dans notre étude, nous avons examiné la résolution du problème d'ordonnancement dans un atelier de type flow shop. Nous avons appliqué l'algorithme d'optimisation de la recherche des pingouins (PeSOA : Penguins Search Optimization Algorithm) et l'algorithme génétique, puis comparé les résultats obtenus.

L'algorithme d'optimisation de la recherche des pingouins (PeSOA : Penguins Search Optimization Algorithm) est un nouvel algorithme inspiré du comportement de chasse des pingouins pour la recherche de nourriture. Cette métaheuristique offre des approches flexibles et adaptatives pour trouver des solutions de qualité à notre problème.

Après avoir adapté chaque algorithme à notre problème, nous avons réalisé des simulations en utilisant le langage Matlab. Nous avons obtenu plusieurs résultats, que nous pouvons résumer de la manière suivante :

- ✓ En général, l'algorithme des pingouins montre de meilleures performances et semble être légèrement plus efficace pour analyser l'évolutivité des configurations. Il a réussi à trouver rapidement des solutions de bonne qualité.
- ✓ L'algorithme génétique a également montré de bonnes performances en termes d'exploration de l'espace de recherche et de recherche de solutions globales. Il a réussi à trouver des solutions de qualité, mais avec un temps d'exécution relativement long.

En conclusion, nous avons constaté que l'algorithme de recherche des pingouins a été plus performant que l'algorithme génétique traditionnel.

REFERENCES BIBLIOGRAPHIQUES

Référence Bibliographiques

[1] Letouzey A., Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs, Thèse de Doctorat, Institut National Polytechnique de Toulouse, 2001.

[2] Javel G., Organisation et Gestion de la Production, 3ème édition, DUNOD, Paris 2004.

[3] Nassima Aissani. Pilotage adaptatif et réactif pour un système de production à flux continu : application à un système de production pétrochimique. Automatique / Robotique. Ecole doctorale de Lille ; Université d'Oran, Algérie, 2010. Français.

[4] LAIEB Ahmed, GRINE Amir ; Ordonnancement d'un atelier Flow shop par Métaheuristique récente, Mémoire de Master, Université Abou Baker Belkaid-Tlemcen, 2017.

[5] Groupe d'Ordonnancement Théorique et Appliqué, "Les problèmes d'ordonnancement", R.A.I.R.O. Recherche opérationnelle/Opérational Research, 27(1), 1993, pp. 77-150.

[6] KEDDARI Nassima. ; intégration de la planification des processus et l'ordonnancement d'un système de production par les métaheuristiques, thèse de doctorat, Université Abou Baker Belkaid-Tlemcen, 2020.

[7] Guillermo CAMPOS CIRO. ; Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique, Thèse de doctorat de l'UTT, UNIVERSITE DE TECHNOLOGIE DE TROYES, 2015.

[8] Vignier, A. ; Billaut, J.-C. ; Proust, C. Les problèmes d'ordonnancement de type « flow-shop » hybride : état de l'art. RAIRO - Operations Research - Recherche Opérationnelle, Tome 33 (1999)

[9] IMÈNE BENKALAI. ; ordonnancement d'ateliers en présence d'opérateurs, thèse de doctorat, UNIVERSITÉ DU QUÉBEC À CHICOUTIMI, 2018.

[10] HOUBAD Yamina. ; Ordonnancement de système de production par métaheuristiques, Thèse de Doctorat, Université Aboubakr Belkaid– Tlemcen –, 2023.

[11] Ilhem Boussaid. Perfectionnement de métaheuristiques pour l'optimisation continue. Autre. Université Paris-Est ; Université des Sciences et de la Technologie Houari-Boumediène (Alger), 2013. Fr

[12] SAOUDI Bakir., REMITA Fares. ; Résolution du problème d'ordonnancement conjoint de la production et de la maintenance de type Flow Shop, Mémoire de Master, Université Abou Bekr Belkaid Tlemcen, 2016.

[13] Adjiri Hadjer ; Les problèmes d'ordonnancement d'atelier : M-Machine identique en parallèle, Mémoire de Master, UNIVERSITE MOHAMED BOUDIAF - M'SILA, 2017/2018.

[14] Bellache N. E. I., Développement et implémentation d'un solveur bio inspiré pour la résolution d'un problème d'ordonnancement d'atelier, Mémoire Master en informatique, Université de M'sila, 2016 /2017.

[15] Cours de la Méthode de séparation et Evaluation (BRANCH AND BOUND), Université Bejaïa, Capitre2.

[16] CHERGUI A., DAHMANI A. N., ADDA ABBOU A., Ordonnancement d'un flow-shop par métaheuristique hybride, Université Abou Bekr Belkaid, Tlemcen, 2016/2017.

[17] R. L. Solso. Cognitive psychology. Harcourt Brace Jovanovich, Inc., New York.436, 1979.

[18] Asmaa GHOUMARI ; Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage, UNIVERSITE PARIS EST, 2018.

[19] "Handbook of Metaheuristics", Ed. F. Glover and G. Kochenberger, ISORMS 57, p 321-353 (2002), Kluwer.

[20] Vincent HÉNAUX, Génération d'algorithmes de recherche locale, THÈSE DE DOCTORAT, UNIVERSITÉ D'ANGERS,2021.

[21] Ines Alaya, Christine Solnon, Khaled Ghedira. Optimisation par colonies de fourmis pour le problème du sac-à-dos multi-dimensionnel. Revue des Sciences et Technologies de l'Information - Série TSI : Technique et Science Informatiques, 2007.

[22] Gheraibia, Y., & Moussaoui, A. (2013). Penguins search optimization algorithm (PeSOA). Computing Department, Mohamed Cherif Messadia Univercity, SOUK AHRAS 41000, Computing Department, Ferhat Abbas Univercity, SETIF 19000, Algeria.

[23] Rossit, D. A., Tohmé, F., & Frutos, M. (2018). The non-permutation flow-shop scheduling problem : a literature review. *Omega*, *77*, 143-153.

[24] Mzili, I., Riffi, M. E., & Benzakri, F. (2020). Discrete penguins search optimization algorithm to solve flow shop scheduling problem. *International Journal of Electrical and Computer Engineering (IJECE) Vol*, *10*, 4426-4435ISSN

[25] Mzili, T., Mzili, I., Riffi, M. E., Pamucar, D., Simic, V., Abualigah, L., & Almohsen, B. (2024). Hybrid genetic and penguin search optimization algorithm (GA-PSEOA) for efficient flow shop scheduling solutions. *Facta Universitatis, Series : Mechanical Engineering*, *22*(1), 077-100.

[26] Arık, O. A. (2021). Genetic algorithm application for permutation flow shop scheduling problems. *Gazi University Journal of Science*, *35*(1), 92-111.