MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Batna 2

Faculté de Technologie

Département de Génie Industriel

Thèse

Présentée au Laboratoire d'Automatique et Productique

Pour l'obtention du diplôme de Doctorat

3eme Cycle LMD

Spécialité : Génie industriel

Par

DRISS IMEN

Master en Génie Industriel

Option Informatique Industrielle et Productique

Thème

Analyse D'un Système Job Shop Aspect Ordonnancement

Soutenue publiquement le : 10/05/2016

Devant le jury composé de :

Pr	MOUSS	Leila Hayet	Présidente	Université de Batna2
Pr	MOUSS	Kinza Nadia	Rapporteur	Université de Batna 2
Pr	YALAOUI	Farouk	Examinateur	Université de Technologie de Troyes
Pr	SARI	Zaki	Examinateur	Université de Tlemcen
Pr	KAZAR	Okba	Examinateur	Université de Biskra
MCA	SMADI	Hacene	Examinateur	Université de Batna 2

REMERCIEMENTS

Le présent travail a été réalisé au LAP (Laboratoire d'Automatique et Productique) à l'Université de Batna 2.

Au nom de dieu je commence mes remerciements.

Je tiens à remercier tout d'abord Pr MOUSS KINZA NADIA, directrice de thèse, pour toute l'aide et la disponibilité qu'elle ma offerte, sa générosité, son écoute patiente et ses conseils précieux. Veuillez trouver Madame l'expression de ma sincère reconnaissance.

Je tiens à remercier également les membres de jury

Merci à Pr MOUSS Leila Hayet pour m'avoir accueilli au LAP et fait l'honneur d'accepter la présidence de ce jury.

Merci Pr YALAOUI Farouk, Pr SARI Zaki , Pr KASAR Okba et Dr SMADI Hacene , examinateurs de ma thèse, pour m'avoir fait l'honneur d'examiner et de participer au jury de ma soutenance. Veuillez trouver l'expression de ma sincère gratitude.

Un grand merci à mon amie et ma sœur LAGGOUN ASSIA pour son aide son soutien et sa présence constante à mes côtés.

Dédicaces

Toutes les lettres ne sauraient trouver les mots qu'il faut… Tous les mots ne sauraient exprimer la gratitude, l'amour, Le respect, la reconnaissance…

Aussi, c'est tout simplement que Je dédie cette Thèse…?

Dédicaces

A MA TRÈS CHÈRE MÈRE : SAIDI MERIEM

Autant de phrases aussi expressives soient-elles ne sauraient montrer le degré d'amour et d'affection que j'éprouve pour toi. Tu m'as comblé avec ta tendresse et affection tout au long de mon parcours. Tu n'as pas cessé de me soutenir et de m'encourager durant toutes ces années de mes études, tu as toujours été présente à mes cotés pour me consoler quand il fallait. En ce jour mémorable, pour moi ainsi que pour toi, reçoit ce travail en signe de ma vive reconnaissance et ma profonde estime. Puisse le tout puissant te donner santé, bonheur et longue vie afin que je puisse te combler à mon tour.

Dédicaces

A MON TRÈS CHER PÈRE : DRISS MUSSADA

Autant de phrases et d'expressions aussi éloquentes soit-elles ne sauraient exprimer ma gratitude et ma reconnaissance. Tu as su m'inculquer le sens de la responsabilité, de l'optimisme et de la confiance en soi face aux difficultés de la vie. Tes conseils ont toujours guidé mes pas vers la réussite. Ta patience sans fin, ta compréhension et ton encouragement sont pour moi le soutien indispensable que tu as toujours su m'apporter. Je te dois ce que je suis aujourd'hui et ce que je serai demain et je ferai toujours de mon mieux pour rester ta fierté et ne jamais te décevoir. Que Dieu le tout puissant te préserve, t'accorde santé, bonheur, quiétude de l'esprit et te protège de tout mal

Dédicaces

A MON TRÈS CHÈR MARI : BARKAT REDHA

Ton encouragement et ton soutien étaient la bouffée d'oxygène qui me ressourçait dans les moments pénibles, de solitude et de souffrance. Merci d'être toujours à mes côtés, par ta présence, par ton amour dévoué et ta tendresse, pour donner du goût et du sens à notre vie de famille En témoignage de mon amour, de mon admiration et de ma grande affection, je te prie de trouver dans ce travail l'expression de mon estime et mon sincère attachement. Je prie Dieu le tout puissant pour qu'il te donne bonheur et prospérité.

Dédicaces

A MA TRÈS CHÈRE FILLE : BARKAT AFNAN ZEINBE

C'est le fruit et l'espoir de toute ma vie

Dédicaces

A MA DEUXIEME CHERE MERE : BARKAT ZEINBE

Ton souvenir reste à jamais gravé dans ma mémoire. YARHAMOUHA ALLAH

Dédicaces

A MES TRES CHERES FRERES : AHMED, HAMZA, MOUNIB

Dédicaces

A MES TRES CHERES SŒURS : KHADIJA, SAFA, ZEINBE, ELBATOULE LYNA, SOUAD

En souvenir d'une enfance dont nous avons partagé les meilleurs et les plus agréables moments. Pour toute la complicité et l'entente qui nous unissent, ce travail est un témoignage de mon attachement et de mon amour.

Résumé

Les problèmes d'ordonnancement sont souvent classés NP-Difficiles. Leur résolution nécessite des méthodes dédiées à leur degré de complexité ; pour cette raison plusieurs heuristiques et méta-heuristiques ont été conçues.

Dans ce travail : nous avons développé deux approches basées sur les algorithmes génétiques et la recherche Coucou pour la résolution du problème job shop classique et /ou job shop flexible afin de minimisons la date de fin de toutes les opérations .Dans la première approche nous proposons une nouvelle représentation du chromosome pour le problème job shop flexible et nous utilisons différentes stratégies pour les opérateurs de croissement et mutation . Pour la deuxième approche nous proposons une nouvelle représentation du nid et différentes stratégie de tri. On a vérifié les performances des deux approches par application à plusieurs benchmarks et comparaison des résultats de nos approches avec ceux obtenus par différents algorithmes évolutionnaire trouvés dans la littérature. Les résultats obtenus sont satisfaisant

Mots clefs : Ordonnancement, Job Shop, Algorithmes Génétique, Recherche Coucou

Abstract

Job-shop scheduling problem, which is proved to be NP-hard, Their resolution, requires methods dedicated to their complexity degree; for this reasons several heuristics and meta-heuristics have been designed.

In this work, we propose two approaches one base a genetic algorithm and cuckoo search algorithm to solve JSSP and /or FJSP to minimize the makespan. This new algorithm uses a new chromosome representation and adopts different strategies for crossover and mutation. The proposed algorithm is validated on a series of benchmark data sets. For the second approach we propose a new representation of the nest and different sorts strategy, performance of this method has been compared with that obtained by other evolutionary computing algorithms by tested on a series of benchmark datasets The results are satisfactory.

Key words: Scheduling, Job Shop, Genetic Algorithm, Cuckoo Search

ملخـــــــص

إن مشاكل الجدولة في الغالب تعتبر من مشاكل التحسين الصعبة ن ب، حلها يتطلب أساليب مختصة نظرا لتعقيداتها. لهذا السبب تم تصميم العديد من الاستدلال وفوقية الاستدلال.

في هذا العمل اقترحنا استخدام الخوارزميات الجينية و خوارزميات البحث الوقواقي كطريقتين مقربتين لحل مسألة الجدولة جوب شوب الكلاسيكية و المرونية. للتقليص من الوقت الكلي للتنفيذ كهدف لهذه المسألة. في النهج الأول نقترح تمثيل جديد للصبغي الذي يعبر عن حل مسألة الجدولة المرنة حيث استخدمنا استراتيجيات جديدة للعوامل الجينية مثل ظاهرة العبور و ظاهرة الطفرة. اما في النهج الثاني نقترح التمثيل الجديد لاستراتيجيات العش وأنواع مختلفة من التصنيف. و للتحقق من صحة الطريقة قمنا بتطبيـق خوارزمية البحث الوقـــواقي المقترحة ومقارنة نتائجها مع تلك التي تم الحصول عليها عن طريق نهج الخوارزميات التطورية المختلفة التي وجدت في المراجع العلمية في هذا المجال. وكانت النتائـــج مرضيـــة.

الكلمات المفتاحيـة: الجدولـــــة , جـوب شوب , خوارزميـــة جينيـــة , خوارزمية البحث الوقـــواقي.

TABLE DES MATIERES

Liste des tableaux I

Liste des figures III

Liste des algorithmes IV

Liste des abriviations VII

Introduction Générale VIII

CHAPITRE 1 : LES PROBLÈMES D'ORDONNANCEMENT D'ATELIERS

1.Introduction 2

2.Concept de base du problème d'ordonnancement 2

2.1.Définition du problème d'ordonnancement 2

2.2.Définition des éléments du problème d'ordonnancements 3

2.2.1.Les taches 3

2.2.2.Les ressources 3

2.2.3.Variables de décision et contraintes 4

2.2.4.Les objectifs 5

2.2.5.Les critéres d'optimisation 6

3.Classification des problèmes d'ordonnancement 7

4. Organisation d'atelier de production 8

5. Complexité des problèmes 9

6. Les méthodes de résolution 12

6.1. Les méthodes exactes 14

6.1.1. Procédure par séparation et évaluation 14

6.1.2. La programmation dynamique 16

6.2. Les méthodes approchées : 16

6.2.1.Les heuristiques 16

6.2.2.Les métaheuristiques 18

6.3. Les méthodes hybrides 26

6.3.1. Coopération méta/méta 27

6.3.2. Coopération méta/exact 28

7. Conclusion 28

CHAPITRE 2 : DESCRIPTION DU PROBLÈME

1.Introduction 31

2.Déscription du probléme 31

2.1. Probléme Job shop classique 31

2.2 .Probléme Job shop flexible 33

3.Modélisation du probléme 34

3.1 .Modélisation du Probléme Job shop classique 35

3.1.1. Modélisation mathématique 35

3.1.2 .Modélisation graphique 36

3.1.3. Principe de réseaux de Petri 36

3.1.4. Modélisation par les réseaux de Petri de JSC 40

3.2. Modélisation du probléme job shop flexible 43

3.2.1. Modélisation mathématique du FJSP 43

4. Représentation des solution du problèmes d'ordonnancement job shop 44

5.Conclusion 45

CHAPITRE 3 : ÉTAT DE L'ART

1.Introduction 47

2.Les probléme job shop classique 47

2.1. Travaux relatifs à la complexité des JSC 47

2.2.Travaux relatifs à l'utilisation des AGs pour l'étude des JSC 47

2.2.1.Présentation des différents travaux suivant les approches de Cheng 48

2.2.2.Présentation des travaux relatifs aux codages des chromosomes 53

2.2.3. Présentation des travaux relatifs à l' hybridation des AGs 54

3.Les problèmes d'ordonnancement job shop flexible 55

3.1.Complexité des problèmes d'ordonnancement job shop flexible 56

3.2 .Travaux relatifs à l'utilisation des AGs pour les FJSP 56

4.Conclusion 59

CHAPITRE 4 : LA RESOLUTION DU PROBLEME PAR LES AGs

1.Introduction 61
2. Principes des Algorithmes Génétiques 61
2.1. Historique 61
2.2.Terminologie 61
2.3.Concepts de bases 62
2.3.1. Le codage 64
2.3.2 .Sélection 65
2.3.3. Le croissement ou crossover 68
2.3.4. La mutation 69
3. Application des AGs pour la résolution du problème FJSP 71
3.1. Codage du chromosome 73
3.2.Opérateurs de l'algorithme génétique proposé (AGP) 76
3.2.1 .Opérateur de sélection 76
3.2.2 Opérateur de croissement 77
3.2.3 Opérateur de mutation 78
4. Validation de l'appoche proposée 79
4.1.Comparaison du AGP avec différent AG pour les instances de Brandimarte80
4.2. Comparaison de l'AGP avec différent AG pour les instances de Chambers83
5. Conclusion 85

CHAPITRE 5 : RESOLUTION DU PROBLEME PAR LA RECHERCHE COUCOU

1.Introduction 87
2.Présentation de l'algorithme de la recherche coucou 87
2.1 Les variantes de la recherche coucou 88
2.2.Algorithme de la recherche coucou 89
3.La résolution de problème d'ordonnancement de job shop par RC 90
3.1 La recherche coucou appliqué à un JSSP 90
3.2. Validation de l'approche pour un JSSP 92
4. La recherche coucou appliqué à un FJSP 94
4.1. Validation de l'approche pour un FJSP 96

4.2 Comparaison de l'RCP et de l'AGP pour les inst de Chambers et Barnes 98

5. Conclusion 99

CONCLUSION GÉNERALE ET PERSPECTIVES 101

BIBIOGRAPHIE

LISTE DES TABLEAUX

CHAPITRE 1	
Tableau 1.1. Variable de décision et contraintes	5
Tableau 1.2. Critère d'optimisation	6
Tableau 1.3. Classe de complexité des problèmes d'ordonnancement (monocritère)	11
CHAPITRE 2	
Tableau 2.1. Un exemple du problème job shop avec quatre machines et cinq jobs	32
Tableau 2.2. Définition des transitions et des places du modèle	41
CHAPITRE 3	
Tableau 3.1. Etat de l'art relatifs au niveau de complexité des problèmes d'ordonnancement JSC	51
CHAPITRE 4	
Tableau 4.1. Temps d'exécution d'un job shop flexible partiel	73
Tableau 4.2. Makespan obtenu par application de différent AG pour les instances de Brandimarte	80
Tableau 4.3. Valeur du C_{max} et du temps de calcul en utilisant l'AGP et d'autre AG pour les instances de Chambers et Barnes	83
CHAPITRE 5	
Tableau 5.1. Les variantes de la recherche Coucou	89
Tableau.5.2. Problème job shop 3*4	90
Tableau.5.3. Validation de l'algorithme de la recherche coucou par lesbenchmarks de Fisher et Thompson	93
Tableau 5.4. Validation de l'algorithme de recherche coucou par les benchmarks de Lawrence	93
Tableau 5.5. Temps d'exécution du problème job shop flexible partiel (P-FJSP)	94
Tableau 5.6. Comparaison entre les makespan dans les différentes approches en fonction des instances de KACEM	97
Tableau 5.7.Comparaison entre les makespan et le temps de calcul dans les approches proposées en fonction des instances de Chambers et Barnes	99

LISTE DES FIGURES

Figure I.1. Organisation hiérarchique fonctionnelle X

CHAPITRE 1

Figure 1.1. Une tâche 3

Figure 1.2. Classification des types d'ateliers 4

Figure 1.3. Relations entre les différentes organisations 9

Figure 1.4. Classification des méthodes de résolution des problèmes d'ordonnancement d'atelier 13

Figure 1.5. L'organigramme du recuit simulé 19

Figure 1.6. L'organigramme de la recherche tabou 20

Figure 1.7. Principales étapes d'un algorithme génétique 22

Figure 1.8. Le comportement des fourmis pour trouver le plus court chemin 23

Figure 1.9. Schéma de principe du déplacement d'une particule. 25

Figure 1.10. Principe de l'algorithme d'optimisation par essaims particulaires 25

Figure 1.11. Principales étapes de l'algorithme de la recherche coucou 26

CHAPITRE 2

Figure 2.1. Représentation d'un système de type Job-shop 32

Figure 2.2. Représentation d'un système de type Job-shop flexible à deux étages 33

Figure 2.3. Exemple d'un RdP 37

Figure 2.4. Modèle d'atelier type job shop par RdP 42

Figure 2.5. Présentation de Makespan de tous les jobs 42

CHAPITRE 3

Figure 3.1. Représentation des algorithmes génétique pour le problème d'ordonnancement Job Shop classique 49

Figure 3.2. Type d'Algorithmes génétique pour un problème d'ordonnancement job shop 52

Figure 3.3. Les deux approches proposées par Brandimarte 57

CHAPITRE 4

Figure 4.1. Les Cinq niveaux d'organisation dans les algorithmes génétiques 61

Figure.4.2. Codage des variables 62

Figure 4.3. Présentation de la population à une génération i 62

Figure 4.4. Principes des algorithmes génétiques. 63

Figure 4.5. Codage binaire d'un chromosome 64

Figure 4.6. Codage réel d'un chromosome 64
Figure 4.7. Codage par valeur d'un chromosome 65
Figure 4.8. La méthode de sélection de la loterie biaisée 66
Figure 4.9. Croisement avec un point de crossover 68
Figure 4.10. Croisement uniforme 69
Figure 4.11. Une mutation 69
Figure 4.12. Mutation uni-point 70
Figure 4.13. Mutation uniforme 70
Figure 4.14. Mutation par valeur 70
Figure 4.15. Organigramme de l'algorithme génétique pour un Job Shop Flexible 72
Figure 4.16. Codage du chromosome 73
Figure 4.17. Deux permutations différentes de chromosome 74
Figure 4.18. Makespan obtenu par application de différents AG pour les instances de Brandimarte 81
Figure 4.19. Régression du makespan et le Diagramme de Gantt de MKO1 82
Figure 4.20. Le temps de calcul de deux approches en fonction des instances de Chambers et Barnes 84

CHAPITRE 5

Figure 5.1. Oiseau de coucou 87
Figure 5.2. Les variantes de la recherche coucou 88
Figure 5.3. Codage du nid. 90
Figure 5.4. Population initiale. 91
Figure 5.4. Décodage du nid 91
Figure 5.6. Régression du makespan par l'algorithme de recherche coucou pour l'instance FT06 93
Figure 5.7. Diagramme de Gantt FT06 95
Figure 5.8. Codage du nid 95
Figure 5.9. Régression de makespan et Diagramme de Gantt (4*5) 97
Figure 5.10. Régression de makespan et Diagramme de Gantt (8*8). 98
Figure 5.11. Régression de makespan et Diagramme de Gantt (10*10) 98

LISTE DES ALGORITHMES

CHAPITRE 1

Algorithme 1.1. Procédure par séparation et évaluation 15

Algorithme 1.2. Algorithme de colonie de fourmis 23

Algorithme 1.3. Algorithme mimétique 27

CHAPITRE 4

Algorithme 4.1. Procédure de codage du chromosome PJ 75

Algorithme 4.2. Procédure de décodage du chromosome PJ 75

Algorithme 4.3. Sélection pour AGP 76

Algorithme 4.4. Croissement pour AGP 77

Algorithme 4.5. Mutation pour AGP 78

Algorithme 4.6. Algorithme Génétique Propose AGP 78

CHAPITRE 5

Algorithme 5.1. Algorithme de la recherche Coucou 89

Algorithme 5.2. Algorithme de recherche coucou pour un job shop classique 91

Algorithme 5.3. Algorithme de recherche coucou pour un job shop flexible 95

LISTE DES ABREVIATIONS

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

CS: La recherche coucou (de l'anglais: Cuckoo Search)

GA : L'algorithme génétique (de l'anglais: Genetic Algorithm)

IS : Le système immunitaire (de l'anglais: Immun System)

KP : Le problème du sac à dos **(**de l'anglais: Knapsack Problem)

LS : La recherche locale (de l'anglais: Local Search)

MA : L'algorithme mimétique (de l'anglais: Memetic Algorithm)

PSO : L'optimisation par essaim de particules (de l'anglais: Particle Swarm Optimization)

SA : Le recuit simulé (de l'anglais: Simulated Annealing)

TS : La recherche tabou (de l'anglais: Tabu Search)

JSSP : Le problème d'ordonnancement job shop (de l'anglais:Job shop scheduling problem)

VNS : La recherche à voisinage variable (de l'anglais: Variable Neighborhood Search)

ILS : la recherche locale réitérée (de l'anglais: Iterated Local Search)

GLS : la recherche locale guidée (de l'anglais: Guided Local Search)

FJSP : Le problème d'ordonnancement job shop flexible (de l'anglais: flexible Job shop scheduling problem)

FJSP-P : Le problème d'ordonnancement job shop flexible partiel (de l'anglais : flexible Job shop scheduling problem partiel)

FJSP-T : Le problème d'ordonnancement job shop flexible total (de l'anglais: flexible Job shop scheduling problem total)

FIFO: First In First Out

SPT : Shortest Processing Time

LPT : Longest Processing Time

EDD : Earliest Due Date

SRPT : Shortest Remaining Processing Time

ST : Slack Time

RdP : Réseaux de Pétri

INTRODUCTION GENERALE

Introduction Générale

L'environnement actuel des entreprises est caractérisé par des marchés confrontés à une vive concurrence, à partir de laquelle les exigences et attentes des clients sont de plus en plus élevées en termes de temps de qualité, de coût et de délais de livraison. Cette évolution est d'autant plus renforcée par le développement rapide de nouvelles technologies d'informations et de communication qui permettent une connexion directe entre les entreprises et entre les entreprises et leurs clients [LOP 08].

Dans ce contexte, la performance des entreprises repose sur deux dimensions. Une dimension technologique, dont le but est de développer la performance intrinsèque des produits commercialisés pour satisfaire aux exigences de qualité et de faible coût de la propriété de ces produits. Une dimension organisationnelle destinée au développement et l'amélioration des performances en termes de temps de cycle de production, respect des délais de livraison , l'adaptation et la réactivité aux variations de commandes commerciales

Cette dimension est d'autant plus importante que les marchés sont de plus en plus volatiles et progressive, et nécessitent des temps de réponse plus courts des entreprises. Par conséquent, les entreprises doivent disposer de méthodes et d'outils puissants pour assurer un meilleur pilotage et contrôle de la production.

Pour atteindre ces objectifs, une organisation de l'entreprise repose sur la mise en œuvre d'un certain nombre de fonctions y compris la planification, qui joue un rôle essentiel (figure I.1). En effet, la fonction d'ordonnancement est destinée à l'organisation des ressources humaines et à l'utilisation des ressources technologiques au niveau des ateliers pour pouvoir répondre aux exigences des clients.

Le problème d'ordonnancement est l'un des problèmes les plus étudiés dans le domaine de la recherche opérationnelle. Ce problème consiste à trouver une séquence optimale pour l'exécution de n jobs sur m machines afin d'optimiser une fonction objectif et de déterminer également les dates de début et de fin d'exécution de chaque job.

Parmi les problèmes d'ordonnancement, nous trouvons les problèmes dits d'ateliers. Ces derniers appartiennent aux problèmes d'ordonnancement de la production. Dans ces types de problèmes, chaque job est caractérisé par une séquence qui définit l'ordre de passage des opérations sur les machines. Ces séquences sont appelées dans la littérature gamme opératoire. On trouve principalement trois types d'atelier : Les ateliers de type flow-shop, job-shop et open-shop.

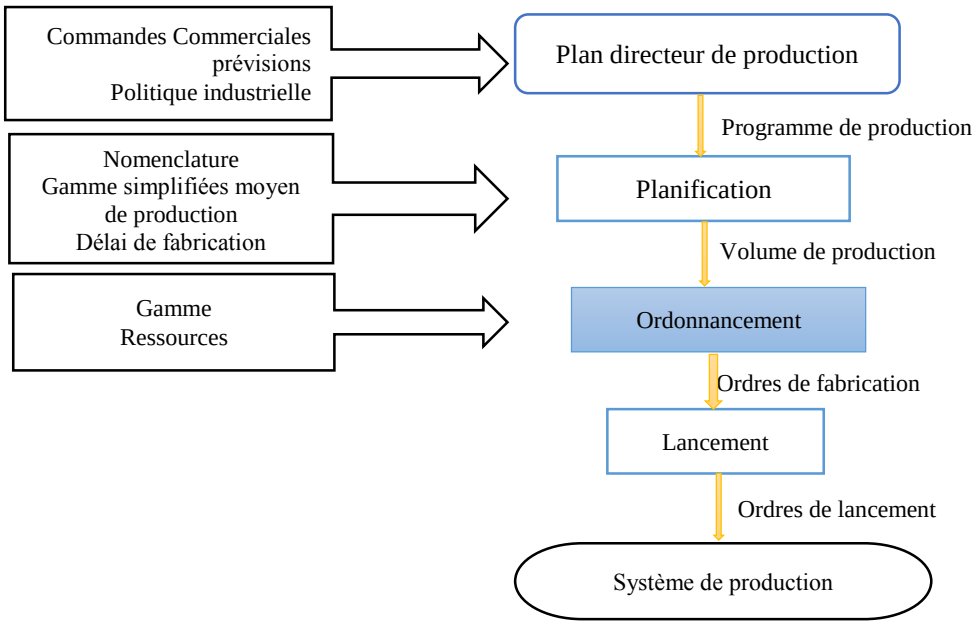


Figure I.1 .Organisation Hiérarchique Fonctionnelle [ESQ 99]

Les problèmes d'ordonnancement les plus difficiles et les plus étudiés, sont les problèmes d'ordonnancement d'atelier de type job shop. Considérés comme un modèle d'ordonnancement intéressant, il est très représentatif du domaine général de production. Ce problème correspond, réellement, à la modélisation d'une unité de production disposant de moyens polyvalents utilisés suivant des séquences différentes en fonction du produit [KEB 08].

Afin d'augmenter la capacité de production, il est possible de mettre en parallèle des machines qui peuvent exécuter les mêmes opérations. Dans ce cas, on parle de systèmes flexibles. L'intérêt majeur des entreprises est d'augmenter la productivité tout en réduisant les coûts au maximum.

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, …) Ces méthodes garantissent en général l'optimalité de la solution fournie, cependant certains problèmes nécessitent la construction de méthodes de résolution approchées, qu'on appelle méthodes heuristiques ou méta heuristique [HOU 11], efficaces pour les problèmes de type NP-difficile.

Les problèmes d'ordonnancement d'atelier de type Job shop sont la plupart des problèmes NP difficile.

L'une de leur résolution nécessitent d'utiliser des heuristiques ou méta-heuristiques. Ces méthodes ne fournissent aucune garantie sur optimalité des résultats, mais ils ont permis de

résoudre avec succès plusieurs problèmes d'optimisation et en particulier une multitude de problèmes d'ordonnancement.

Cette thèse concerne de manière générale l'évaluation des performances et l'ordonnancement dans des systèmes flexibles, principalement les problèmes de type job-shop, Il s'agit d'un problème d'ordonnancement composé de tâches regroupées en jobs et utilisant un ensemble de ressources (machines, opérateurs, …). Les machines sont disponibles en un ou plusieurs exemplaires. Chaque job doit passer sur un ensemble de machines. L'ordre de passage sur les ressources dans la gamme opératoire peut être différent d'un produit à l'autre (tous les produits ne passent pas nécessairement sur toutes les ressources), sous différentes contraintes imposées. L'objectif de notre travail est la minimisation du makespan .

Ainsi et pour atteindre cet objectif, nous avons proposé dans le cadre de cette thèse deux approches évolutionnistes pour la résolution du problème d'ordonnancement job shop et job shop flexible. Une approche basée sur les algorithmes génétiques et l'autre basée sur l'algorithme de la recherche coucou.

La validation des approches proposées a été faite par comparaisons des résultats obtenus par l'approche proposée avec ceux d'autres AGs et cela pour différents benchmarks

Aussi et pour assurer une chronologie de l'évolution de notre travail, le manuscrit de cette thèse contient cinq chapitres organisés de la manière suivante :

Le premier chapitre est réservé à la présentation des problèmes d'ordonnancement de manière générale. Nous présentons en première lieu une vue d'ensemble sur ce type de problème des systèmes de production et sur leur complexité. Nous terminons notre chapitre par la présentation de différentes méthodes de résolution, des problèmes d'ordonnancements aussi bien exactes qu'approchées.

La deuxième chapitre est réservé à la formulation mathématique pour le problème d'ordonnancement Job shop classique et le problème d'ordonnancement job shop flexible afin de minimiser le makespan. Nous avons modélisé notre problème d'ordonnancement job shop utilisant les RdP.

Le troisième chapitre est compte à lui réservé à la présentation d'un état de l'art des problèmes d'ordonnancement type job shop classique et flexible mais spécifiquement ceux relatifs à l'utilisation des AGs comme méthode de résolution.

Dans le quatrième chapitre, nous présentons d'une part l'AG proposé et de l'autre part sa validation. Validation obtenue par comparaison entre les valeurs du makespan et du

temps de calcul, obtenu par application de l'AG proposé et par application d'autre AG de la littérature et cela pour les même benchmarks.

Le cinquième chapitre présente la seconde contribution de notre travail. Elle consiste en la proposition d'un algorithme basée sur la recherche coucou. D'une manière analogue que pour la première contribution (AG) nous validons notre travail par comparaison entre le makespan obtenu avec la RC et ceux d'autres approches pour les mêmes instances.

Nous terminons la présentation de cette thèse par une conclusion, résumant les travaux développés, les résultats obtenus et quelques perspectives de recherche ouvertes par ces travaux.

CHAPITRE 1

NOTIONS SUR LES PROBLEMES D'ORDONNANCEMENT DES ATELIERS

Nous proposons dans ce chapitre des notions de base des problèmes d'ordonnancement d'atelier. Dans la première partie du chapitre nous présentons les concepts, les éléments et la classification des problèmes d'ordonnancement. La seconde partie est consacrée aux notions liées à la complexité et aux méthodes de résolution de ces problèmes.

La chute n'est pas un échec ; l'échec c'est de rester là où on est tombé.

Socrate

1. Introduction

L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison. L'ordonnancement est un élément crucial dans l'ensemble des tâches liées au pilotage d'atelier.

Dans ce chapitre nous présentons quelques concepts de base de l'ordonnancement et des problèmes d'ordonnancement dans les ateliers de production à savoir les types d'ateliers. Les critères d'optimisation ainsi que la notion de complexité des problèmes d'ordonnancement .Nous terminons ce chapitre par la description de certaines méthodes exactes ou approchées de résolution du problème d'ordonnancement.

2. Concept de base du problème d'ordonnancement

2.1 Définition du problème d'ordonnancement

Nombreuses sont les définitions proposées au problème d'ordonnancement. Dans ces définitions on retrouve l'aspect commun de l'affectation de ressources aux tâches, en recherchant une certaine optimisation. Nous pouvons citer quelques unes :

Eric Pinson définit l'ordonnancement comme étant « *l'organisation dans le temps de l'exécution d'un projet* ». [GOT 93] ;

Pour Norman Sadeh c'est « *l'allocation dans le temps de ressources permettant l'exécution d'un ensemble de tâches*» [LOP 01] [GOT 93].

Pour Gotha, c'est « *programmer l'exécution des tâches en leur allouant les ressources requise et en fixant leur dates de début* » [GOT 93] [LAC 05]

Le problème d'ordonnancement consiste donc, à programmer dans le temps l'exécution de certains tâches, en leur allouant les ressources requise, et en respectant les contraintes imposées afin d'atteindre certains objectifs.

D'une façon générale, il y a un problème d'ordonnancement quand il existe :

- Un ensemble de travaux (tâche, jobs ou lots) à réaliser
- Un ensemble de ressources à utiliser par ces travaux
- Un programme (calendrier) à utiliser, pour allouer convenablement les ressources aux tâches.

2.2 Définition d'Eléments du problème d'ordonnancement

Les principaux éléments d'un problème d'ordonnancement sont les suivants : les tâches, les ressources, les contraintes et les objectifs.

2.2.1 Les tâches

Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unité de temps (sa durée) et d'unités de chaque ressource.

Une tâche (i) est localisée dans le temps par une date de début (S_i) et/ou de fin (C_i), et par une durée d'exécution (P_i) : telle que $P_i = C_i - S_i$ (figure 1.1)

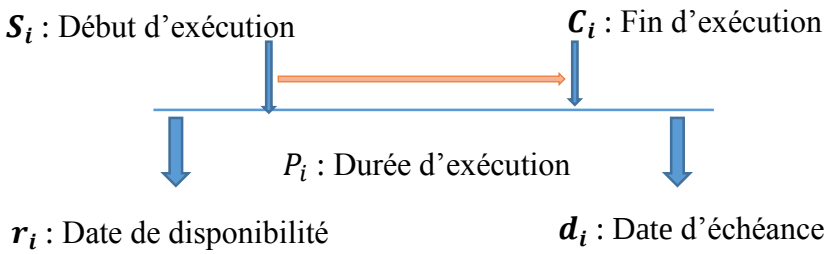


Figure 1.1. Une tâche

Si les tâches sont, liées entre elles par des conditions d'origines diverses on dit alors que les tâches sont *interdépendantes*, dans le cas contraire, elles sont dite *indépendantes*.

De même, les tâches sont dites *interruptibles* si elles peuvent être exécutées par morceaux par une ou plusieurs ressources ; dans le cas contraires, si on ne peut interrompre une tâche une fois celle-ci commencée, on parle alors de taches non préemptif.

2.2.2 Les ressources

Dans le contexte industriel, les ressources peuvent être des machines, des ouvriers, des équipements, des locaux ou encore de l'énergie, des budgets, etc.

Une ressource est un moyen technique ou humain destinée à être utilisée pour la réalisation d'au moins une tâche et disponible en quantité limitée.

Les ressources peuvent être classées soit selon leurs disponibilités au cours du temps, et on parle dans ce cas de ressources renouvelables ou consommables, soit selon

leurs capacités et ceux sont des ressources disjonctives (non partageables) ou cumulatives (partageables).

- **Ressource renouvelable** : il s'agit d'une ressource pouvant redevenir disponible en même quantité après avoir été allouée à une tâche (les machines, les hommes, l'équipement,…).
- **Ressource consommable** : il s'agit d'une ressource dont la disponibilité décroît après avoir été allouée à une tâche, par exemple la matière première.
- **Ressource disjonctive** : il s'agit d'une ressource qui ne peut exécuter qu'une seule tâche à la fois.
- **Ressource cumulative** : il s'agit d'une ressource qui peut être utilisée simultanément par plusieurs tâches.

Lorsque plusieurs ressources sont nécessaires simultanément pour assurer la réalisation d'une tâche, on parle de problèmes d'ordonnancement multi ressources.

En revanche, un problème d'ordonnancement est dit *mono-ressource* lorsqu'une seule ressource est nécessaire et suffisante pour la réalisation de chaque tâche.

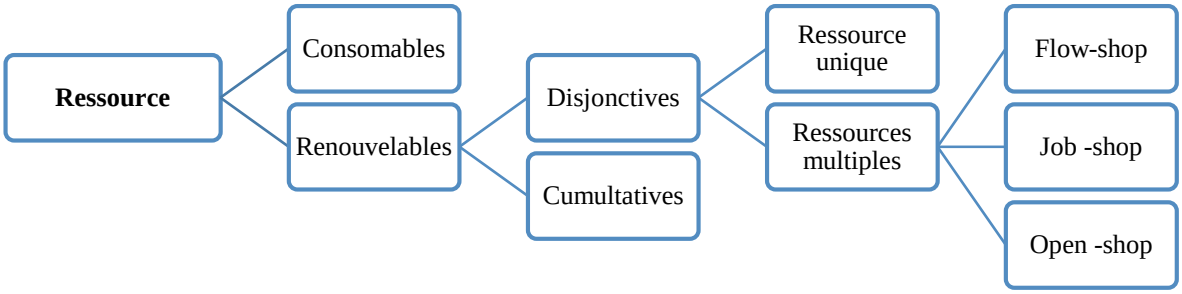


Figure 1.2. Classification des types d'ateliers [ESQ 99]

2.2.3 Variables de décision et contraintes

Les variables de décision sont des quantités numériques pour les quelles des valeurs sont à choisir. Les variables des décisions sont liées au temps ou aux ressources. On parle respectivement de *variables temporelles* et de *variables d'affectation*.

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement une ou plusieurs variables de décision. On distingue deux sortes de contraintes en ordonnancement, les contraintes temporelles et les contraintes de ressources. Le tableau 1.1 illustre ces deux types de contraintes.

Contrainte	Type	Illustration
Contrainte temporelles	Contrainte de temps absolu	Cette contrainte permet de représenter la limitation des valeurs possible pour les dates.
	Contrainte de temps relatif	Cette contrainte est relative aux contraintes de cohérence technologique telle que les contraintes de gamme dans lesquelles il faut représenter le positionnement relatif entre tâches
Contrainte de ressources	Contrainte de capacité	Contrainte disjonctive : Cette contrainte impose la réalisation disjointe de deux tâches (i) et (j)
		Contrainte cumulative : Cette contrainte interdit, pour une ressource donnée, la réalisation simultanée d'un certain nombre de tâches utilisant une quantité de la ressource excédant sa capacité
	Contraintes d'affectation	Contrainte de domaine : Cette contrainte représente l'ensemble des ressources candidates pour l'exécution d'une tâche
		Contrainte de différence : Cette contrainte impose l'utilisation de ressources différentes pour la réalisation d'un certain nombre de tâche

Tableau 1.1. Variable de décision et contraintes [BEN 11]

2.2.4 Les objectifs

Tout ordonnancement est guidé par un ou plusieurs objectifs, à optimiser. Ces objectifs ont été groupés en classe :

- Les objectifs liés au temps : à titre d'exemple la minimisation :
  - du temps total d'exécution
  - du temps moyen d'achèvement

- des durées totales de réglage
- des retards totales par rapport aux dates de livraison

- Les objectifs liés aux ressources : à titre d'exemple :
  - maximiser la charge d'une ressource
  - minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches.

2.2.5 Les critères d'optimisation

Le choix d'un ordonnancement , parmi les ordonnancements réalisables , se fait en fonction d'un ou plusieurs critère(s) que l'on cherche à optimiser.

Généralement, les critères considérés sont des fonctions des dates de fin C_i des produits. Ces critères peuvent être de type Max ou de type Somme. Il s'agit de considérer le maximum ou la somme (éventuellement pondérée) sur toutes les tâches d'une certaine mesure ou d'une combinaison de plusieurs mesures. Les mesures les plus intéressantes, considérées le plus souvent dans la littérature, pour un travail J_i, sont données dans le tableau 1.2.

Critères	Maximum	Somme	Somme pondérée
Fin de traitement	$C_{max} = \max_{i=1,n}(C_i)$	$\bar{C} = \frac{1}{n}\sum_{i=1}^{n} C_i$	$\bar{C}_w = \sum_{i=1}^{n} w_i C_i$
Décalage	$L_{max} = \max_{i=1,n}(L_i)$	$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i$	$\bar{L}_w = \sum_{i=1}^{n} w_i L_i$
Retard	$T_{max} = \max_{i=1,n}(T_i)$	$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i$	$\bar{T}_w = \sum_{i=1}^{n} w_i T_i$
Avance	$E_{max} = \max_{i=1,n}(E_i)$	$\bar{E} = \frac{1}{n}\sum_{i=1}^{n} E_i$	$\bar{C}_w = \sum_{i=1}^{n} w_i E_i$
Durée de flot	$F_{max} = \max_{i=1,n}(C_i)$	$\bar{F} = \frac{1}{n}\sum_{i=1}^{n} F_i$	$\bar{F}_w = \sum_{i=1}^{n} w_i F_i$
Nombre de retard		$\bar{U} = \frac{1}{n}\sum_{i=1}^{n} U_i$	$\bar{C}_w = \sum_{i=1}^{n} w_i U_i$

Tableau 1.2.Critère d'optimisation

3. Classification des problèmes d'ordonnancement

Il existe une très grande variété de problèmes d'ordonnancement. Pour leur identification et leur classification , nous adoptons la notation proposée par Graham et al [GRA79] [BRU94]. Cette notation est constituée de trois champs $\alpha|\beta|\gamma$.

- Le premier champ α est constitué de deux éléments : $\alpha = \alpha_1\alpha_2$ il décrit l'environnement des machines utilisées :

- le paramètre $\alpha_1 \epsilon$ {∅, P, Q, R, F, J, O} décrit le type des machines utilisées tel que ;
- **Une machine (∅)** : chaque travail est constitué d'une seule opération. Dans ce cas, on appelle indifféremment travail et tache ;
- **machines parallèles** : elles remplissent, a priori, toutes les mêmes fonctions. Selon leur vitesse d'exécution, on distingue :
  - **machines identiques (P)** : la vitesse d'exécution est la même pour toutes les machines $M_j$et pour tous les travaux J_i;
  - **machines uniformes (Q)** : chaque machine M_ja une vitesse d'exécution propre et constante. La vitesse d'exécution est la même pour tous les travauxJ_id'une même machine M_j;
  - **machines indépendantes (R)** : la vitesse d'exécution pourrait être différente pour chaque machine $M_j$et pour chaque travail J_i.
- **Les ateliers de type flow-shop (F) :** Appelés également ateliers à cheminement unique, ce sont des ateliers où une ligne de fabrication est constituée de plusieurs machines en série ; toutes les opérations de toutes les tâches passent par les machines dans le même ordre.
- **Les ateliers de type job-shop (J) :** Appelés également ateliers à cheminement multiple, ce sont des ateliers où les opérations sont réalisées selon un ordre bien déterminé, variant selon la tâche à exécuter.
- **Les ateliers de type open-shop (O) :** Ce type d'atelier est moins contraignant qu'un atelier de type flow-shop ou de type job-shop, L'ordre des opérations n'est pas fixé a priori. Le problème d'ordonnancement consiste, d'une part, à déterminer le cheminement de chaque produit et, d'autre part, à ordonnancer les produits en tenant compte des gammes trouvées, ces deux problèmes pouvant être résolus simultanément.
- le paramètre α_2 indique le nombre de machines utilisées. Lorsque α_2n'est pas précise, le nombre de machines est quelconque.

- Le deuxième champ $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7$ décrit les caractéristiques des travaux et des machines. Il est utilisé pour préciser les contraintes du problème et les différentes hypothèses sur le mode d'exécution des tâches (préemption, précédence, etc.),
- Le dernier champ γ indique le critère d'optimisation, il peut donc prendre de nombreuses valeurs et peut être une combinaison de plusieurs critères.

A titre d'exemple la notation :

$$J,5,9|Prec,r_i|\ C_{max}$$

Signifie qu'il s'agit d'un problème d'ordonnancement d'un atelier de type job-shop (J), de cinq jobs à neuf machines(5,9). Les jobs présentent une contrainte de précédence, (*Prec),* et une contrainte r_i de dates de début au plus tôt. Avec pour objectif est de minimiser le makespan (C_{max}).

4. Organisation d'ateliers

Pour se rapprocher le plus des ateliers réels de production une extension aux types classiques d'ateliers (une machine, flow shop, job shop, open shop) a été ajoutée. Il s'agit de la flexibilité des machines .elle consiste à associer à chaque opération un ensemble de machines. Ainsi cette opération peut être exécutée (ou traitée) par une machine quelconque de cet ensemble. Aussi de nouveaux types d'ateliers ont fait leur apparition.

- les ateliers à « machines parallèles » où les opérations indépendantes et sont traitées par le même ensemble de machines,
- les ateliers de type «flow-shop flexible » constituant une extension du problème du type flow-shop, où la première opération de chaque produit est traitée par le premier ensemble, la deuxième opération par le deuxième ensemble, et ainsi de suite.
- les ateliers de type « job-shop flexible » est une extension du modèle job-shop classique. Sa particularité essentielle réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations, ou une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération [GZA 01].

Les organisations présentées (classiques ou flexible) ne sont pas indépendantes .En effet des relations de ressemblance, de réduction et de différenciation existent entre elles. La figure 1.3 illustre ces différentes relations : chaque arc de « B » vers « A » s'interprète que « A » est un cas particulier de « B ». Le job shop classique est un cas particulier de job shop flexible.

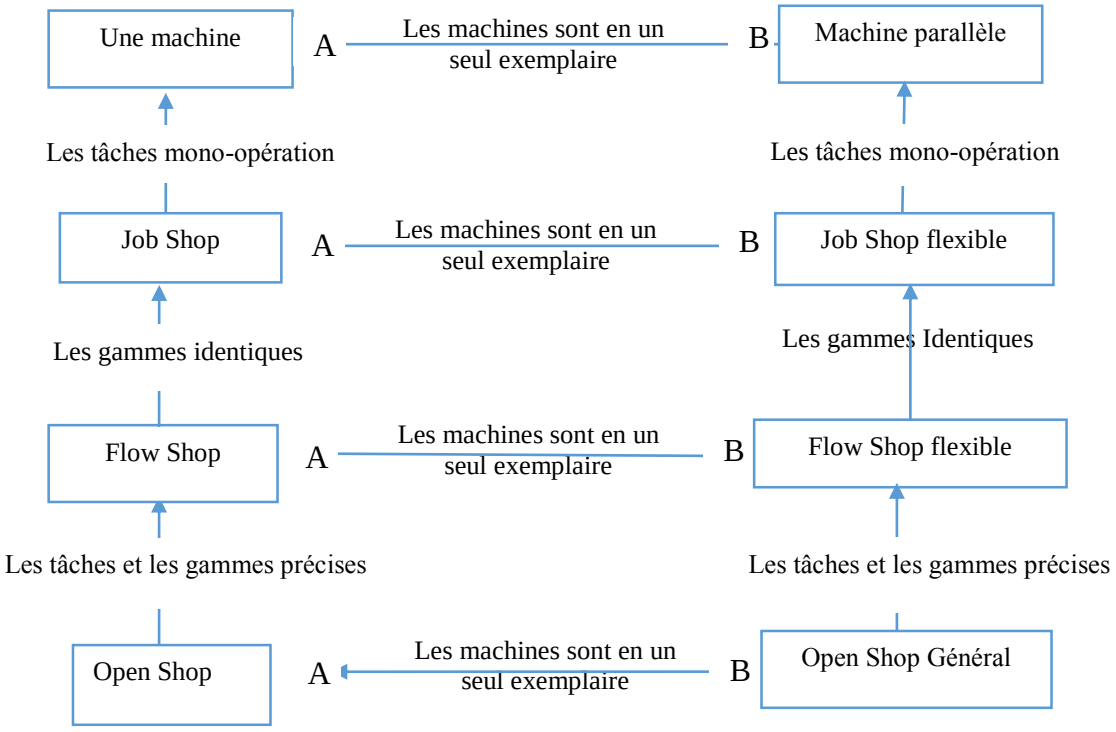


Figure 1.3. Relations entre les différentes organisations [KEB 08]

5. Complexité des problèmes d'ordonnancement

D'une manière générale, les problèmes d'ordonnancement d'ateliers étant des problèmes combinatoires difficiles, il n'existe pas d'algorithmes ou de méthodes universelles permettant de résoudre tous les cas. Plusieurs algorithmes peuvent être utilisés pour résoudre un problème d'ordonnancement .Tous ces algorithmes ne sont pas équivalents, cependant on peut les différencier par le moyen des critères suivants [BOU 11]

- L'efficacité de l'algorithme en terme de durée d'exécution ; un algorithme est dit plus efficace qu'un autre si pour les mêmes données, il s'exécute en temps plus court ;
- L'efficacité de l'algorithme en espace mémoire de stockage ; un algorithme est dit plus efficace qu'un autre si pour résoudre le même problème, il utilise moins d'espace mémoire ;
- La fiabilité de l'algorithme ; plus un programme est complexe, plus il y a des risques d'existence de bugs, les bugs étant des erreurs plus ou moins évidentes qui se manifestent lors de la mise en exploitation d'un programme. Un programme est jugé plus fiable ou plus stable qu'un autre s'il présente moins de bugs ;

- La robustesse de l'algorithme ; elle mesure son degré de tolérance aux erreurs des utilisateurs et sa résistance aux attaques des pirates ; un programme est plus robuste qu'un autre s'il résiste mieux aux erreurs de manipulations des utilisateurs plus ou moins bien attentionnés.

Il est à noter qu'il n'y a pas de méthode ou d'échelle de mesure permettant d'évaluer la fiabilité ou la robustesse d'un algorithme. C'est à l'usage que ces qualités sont mesurées. Par contre, il existe des méthodes rationnelles et rigoureuses pour évaluer l'efficacité en temps ou en espace d'un algorithme. Ces méthodes d'évaluation portent le nom d'analyse de complexité des algorithmes.

Deux types de complexité peuvent être cités [KAA 04] [BOU 11] :

- La complexité méthodologique, qui exprime une fonction du nombre d'opérations élémentaires de calcul effectuées par la méthode ou par l'algorithme de résolution en fonction du nombre des données du problème traité,
- La complexité problématique, liée à la difficulté du problème à résoudre et au nombre des opérations élémentaires qu'un algorithme déterministe peut effectuer pour trouver l'optimum en fonction de la taille du problème.

Selon leurs degrés de complexité on classe les problèmes d'ordonnancement en deux classes principales : la classe P (Polynomial time) et la classe NP (Non deterministic Polynomial time). La classe NP est elle-même subdivisée en deux sous classes : NP-Complet et NP-Difficile.

1. **Les problèmes de la classe P :** pouvant être résolus par une machine de Turing déterministe en temps de calcul polynomial par rapport à la taille de l'instance du problème à traiter.

2. Les problèmes de la classe *NP :* Cette classe regroupe l'ensemble de problèmes qui peuvent être résolus avec une machine de Turing non-déterministe et admettent un algorithme polynomial pour tester la validité d'une solution du problème traité

2.1. Les problèmes NP-Complets :La classe NP-Complet regroupe les problèmes de décision pour lesquels il n'existe pas d'algorithme permettant leur résolution en un temps polynomial.

- Selon Garey et Johnson [GAR 79] : un problème *A* est un problème NP-Complet s'il appartient à la classe NP, et si quel que soit le problème *Á* qui appartient aussi à la classe NP, on peut le réduire au problème A en un temps polynomial.
- Selon Ferro et al [FER 05], un problème *A* est un problème NP-Complet s'il appartient à la classe NP, et si on peut le réduire à un problème connu dans NP-Complet. Par conséquent, la NP-Complets est un problème décisionnel.

2.2 Les problèmes NP-Difficiles : La classe de problèmes NP-Difficiles englobe les problèmes de décision et les problèmes d'optimisation. Si un problème de décision associé à un problème d'optimisation P est NP Complet alors P est un NP-Difficile [CHA 96].

La plupart des problèmes d'ordonnancement sont des problèmes d'optimisation qui appartiennent à la classe des problèmes NP-Difficiles (tableau 1.3) [PON 09]. La résolution d'un problème d'ordonnancement, on cherche souvent à optimiser un ou plusieurs critères [BOU 06] tels que : le temps total d'exécution, le temps moyen d'achèvement d'un ensemble de tâches, les différents retards (maximum, moyen, somme …etc.) ou avances par rapport aux dates limites fixées.

Problèmes / Critère		C_{max}	F_{max}	T_{max}	L_{max}	f_{max}	$\sum C$	$\sum C_w$	$\sum T$	$\sum T_w$	$\sum U$	$\sum U_w$
Une seule machine		P	P	P	P	P	P	P	NP^*	NP	P	NP^*
Machines parallèles	**P**	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
	Q	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
	R	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
Flow shop (F)		NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
Flow shop flexible		NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
Job shop (J)		NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
Job shop flexible		NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP
Open shop (O)		NP	NP	NP	NP	NP	NP	NP	NP	NP	NP	NP

NP^* : Classe NP-difficile au sens faible NP : Classe NP-difficile au sens fort

Tableau 1.3. Classe de complexité des problèmes d'ordonnancement d'atelier (monocritère) [SAK 09].

6. Les méthodes de résolution du problème d'ordonnancement

Au fil des années, de nombreuses méthodes de résolution de problèmes ont été proposées. Ainsi, une grande variété de concept et principe, de la stratégie et des performances ont été discernées. Cette variété et ces différences ont permis de regrouper les différentes méthodes de résolution de problèmes NP-difficiles en deux classes principales : la classe de méthodes exactes et la classe des méthodes approchées. (Voir figure 1.4)

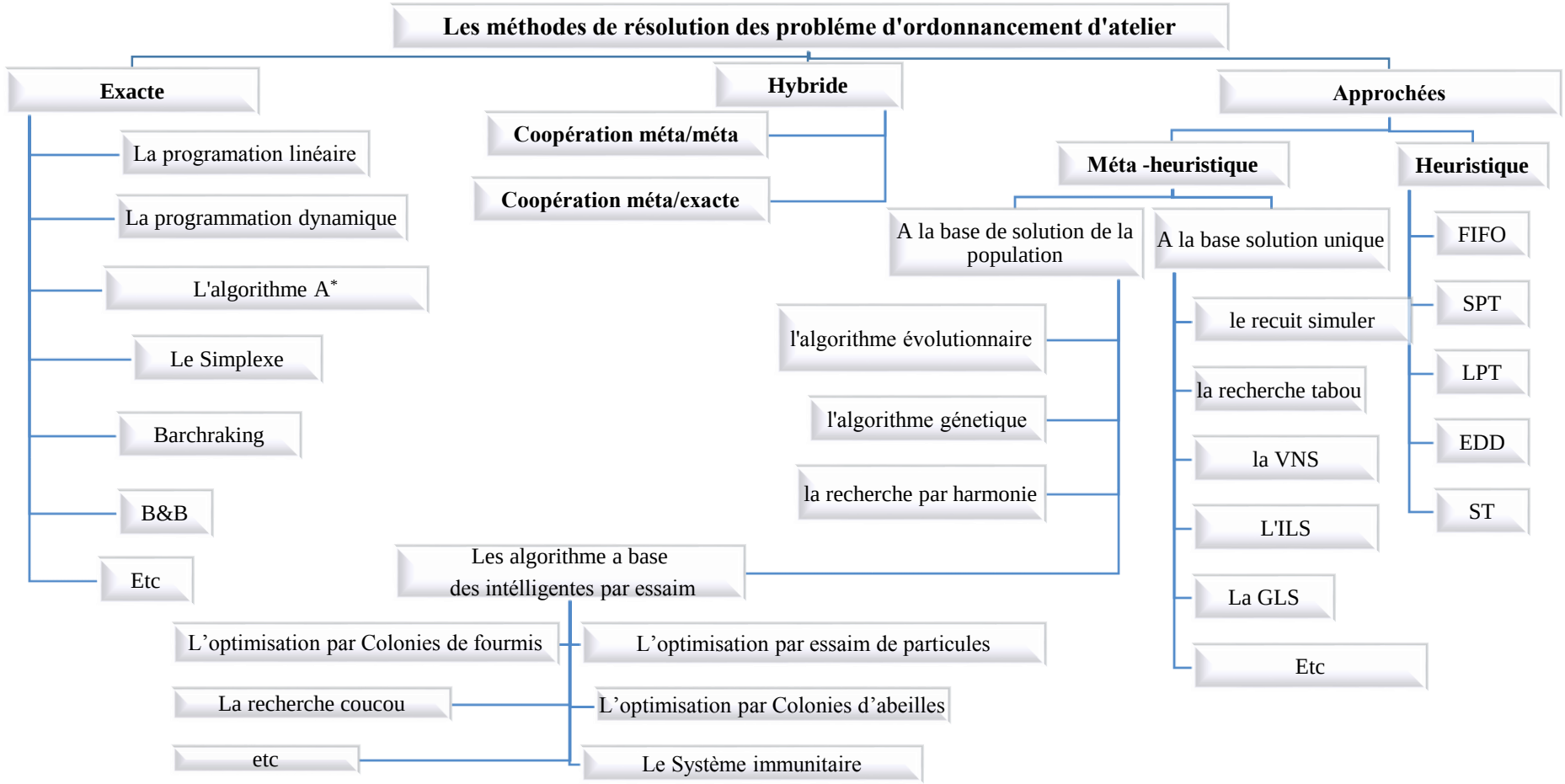


Figure 1.4. Classification des méthodes de résolution des problèmes d'ordonnancement d'atelier

6.1 Les méthodes exactes

L'intérêt des méthodes exactes réside dans le fait qu'elles assurent l'obtention de la solution optimale du problème traité. En fait, elles permettent de parcourir la totalité de l'ensemble de l'espace de recherche de manière à assurer l'obtention de toutes les solutions ayant le potentiel d'être meilleures que la solution optimale trouvée au cours de la recherche. Cependant, les méthodes exactes sont très connues par le fait qu'elles nécessitent un coût de recherche souvent prohibitif en termes de ressources requises. En effet, le temps de recherche et/ou l'espace mémoire nécessaire pour l'obtention de la solution optimale par une méthode exacte sont souvent trop grands, notamment avec des problèmes de grandes tailles [GHE 13]. De ce fait, la complexité de ce type d'algorithme augmente avec la taille de l'instance à traiter.

Il existe de nombreuses algorithmes exacts y compris l'algorithme du simplexe, la programmation dynamique, la programmation linéaire, l'algorithme A*, les algorithmes de séparation et évaluation (Branch and Bound), les algorithmes de retour arrière (Backtracking) . Nous nous limitons dans le cadre de ce travail à présenter quelques-unes.

6.1.1 Procédure par Séparation et Evaluation (PSE)

Les procédures d'optimisation par séparation et évaluation (PSE ou branch and bound) sont des procédures d'exploration par énumération implicite de l'ensemble des solutions. Elles conduisent à construire une arborescence ou chaque sommet représente un sous problème et ou les arcs issus d'un même sommet représentent la décomposition d'un problème ou sous problème de taille réduite [ESQ 99]. L'objectif est de parvenir à un sommet correspondant à un problème résolu de façon simple (solution unique par exemple) et en ayant généré le moins possible de sommets.

Les PSE reposent sur quatre composants essentiels :

- la technique de séparation ; qui permet de décomposer un problème en sous problèmes de taille réduite ;
- la méthode d'évaluation qui associe une borne, en fonction du critère d'optimisation, à l'ensemble des solutions d'un sous problème (par exemple la borne inférieur dans le cas de minimisation) ;

- la méthode de sondage, qui permet de déterminer si un sommet est terminal (il ne contient pas de solution admissible, ou c'est une solution optimale ou on peut obtenir polynomialement la solution optimale de ce sous- problème) ou il mérite d'être séparé ;
- la méthode de sélection, ou stratégie d'exploration, qui décrit comment choisir le sous problème à séparer, s'il y a plusieurs. Deux grandes stratégies existent. La stratégie du type *meilleur d'abord*, consiste à sélectionner puis a séparé le sous problème « pendant » ayant la meilleure évaluation, c'est-à-dire la plus petite ; la méthode arborescente ainsi développée est appelée PSEP (procédure par séparation et évaluation progressives). Dans une stratégie de type *profondeur d'abord* de retour arriéré, les sommets pendants sont gérés selon une pile, la procédure est une PSES (procédure par séparation et évaluation séquentielles)

L'algorithme 1.1 décrit la procédure d'une PSE [ESQ 99]

Algorithme 1.1. Procédure par séparation et évaluation

```
Début
   Calculer une borne supérieure BS
  Tant qu'il existe des sommets non explorés faire
       Sélectionner le sommet à séparer et créer ses fils
      Pour tous les sommets S crées faire
            Si S représente une solution complète ou si on connait une
solution optimale de
               S alors
                Calculer sa valeur du critère d'optimisation et mettre jour BS

            Sinon
                Calculer la borne inferieure BI
                   Si BI < BS alors
                        Placer S dans la liste des sommets a exploré
                   Fin si
              Fin si
      Fin pour
     Eliminer tout sommet tel que BI > BS
 Fin tant que
Fin
```

Les performances de la méthode dépendent évidemment de la qualité des bornes inferieures et supérieures mais aussi de la rapidité, en terme de temps de calcul, à obtenir ces bornes. Par conséquent, plusieurs améliorations de l'algorithme B&B ont été proposées, y

compris les algorithmes : Branch and Cut (B&C) [PAD 91], Branch and Price (B&P) [BRA 05], Branch and Cut and Price (B&C&P).

De nombreuses PSE ont été proposées dans la littérature de l'ordonnancement d'atelier [BRA 91][YAL 02][SHI 08][SHI 09] [FAT 14]

6.1.2 La programmation dynamique

La programmation dynamique est une méthode d'optimisation opérant par séquences Elle a été introduite au début des années 50 par Richard Bellman. Son efficacité repose sur le principe d'optimalité de Bellman [BEL 86]. Ce principe permet une résolution ascendante qui détermine une solution optimale d'un problème à partir des solutions de ses sous problèmes. Cette méthode nécessite une formulation du critère sous forme d'une relation de récurrence entre deux niveaux successifs. Elle est destinée à résoudre des problèmes d'optimisation à vocation plus générale que la méthode de séparation et d'évaluation sans permettre pour autant d'aborder des problèmes de tailles importantes.

6.2 Les méthodes approchées

La résolution d'un problème d'ordonnancement, de taille comparable à ceux rencontrés dans la pratique, se heurte à des tailles mémoire et de temps de calcul trop importants. L'objectif n'est plus alors d'obtenir systématiquement l'optimum mais plutôt d'obtenir une solution proche de l'optimum ou de « bonne qualité » en un temps minimal. Ainsi, au lieu d'effectuer une recherche exhaustive, les méthodes approchées échantillonnent l'espace de recherche et n'en considèrent qu'une partie, et fournissent ainsi, en un temps raisonnable, la meilleure configuration rencontrée. On distingue deux types de méthodes : les heuristiques et métaheuristiques.

6.2.1 Les Heuristiques

Nombreuse sont les définitions proposées aux heuristiques.

On peut citer quelques une :

- Pour Feigenbaum « *Une méthode heuristique est une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillibles de ce qui est la meilleure chose à faire*» [FEI 63] ;

- Pour Slagle « *Une heuristique est une règle d'estimation, une stratégie, une méthode ou astuce utilisée pour améliorer l'efficacité d'un système qui tente de découvrir les solutions des problèmes complexes*.»[SLA 71] ;
- Pour Newell « *Les heuristiques sont des règles empiriques et des morceaux de connaissances, utiles (mais non garanties) pour effectuer des sélections différentes et des évaluations*» [NEW 80] ;
- Pour Pearl « *Les heuristiques sont des critères, des méthodes ou des principes pour décider qui, parmi plusieurs d'autres plans d'action promet d'être le plus efficace pour atteindre un certain but* » [PEA 84] ;
- Pour Solso « *Les heuristiques sont des ensembles de règles empiriques ou des stratégies qui fonctionnent, en effet, comme des règles d'estimation* » [SOL 79] ;

Les heuristiques sont donc des méthodes empiriques basées sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de ces méthodes est d'intégrer des stratégies de décision pour construire une solution proche de l'optimum, tout en essayant de l'obtenir en un temps de calcul raisonnable [BEL 01] [LOP 01] [KAC 03].

De nombreuses méthodes heuristiques ont été proposées dans la littérature pour résoudre les problèmes d'ordonnancement d'atelier. On distingue [CHU 96] [ESQ 99] [WAN 99] :

- FIFO (First In First Out) : la première tâche qui vient est la première tâche ordonnancée,
- SPT (Shortest Processing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu,
- LPT (Longest Processing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu,
- EDD (Earliest Due Date) : cet algorithme choisit parmi les tâches exécutables celle dont le délai est échu le plus tôt. Si aucune tâche n'est disponible, alors un temps libre est généré ,
- 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,
- 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.

6.2.2 Les Métaheuristique

Selon la définition proposée par Osman « *Une métaheuristique est un processus itératif qui subordonne et guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales* » [OSM 96].

L'ensemble des métaheuristiques proposées dans la littérature sont partagées en deux catégories : des métaheuristiques à base de solution unique et des métaheuristiques à base de population de solutions. Nous présentons dans ce qui suit quelques métaheuristiques des deux catégories.

6.2.2.1. Les Métaheuristiques à base de solution unique

Les métaheuristiques à base de solution unique débutent la recherche avec une seule solution initiale. Elles se basent sur la notion du voisinage pour améliorer la qualité de la solution courante. En fait, la solution initiale subit une série de modifications en fonction de son voisinage. Le but de ces modifications locales est d'explorer le voisinage de la solution actuelle afin d'améliorer progressivement sa qualité au cours des différentes itérations.

De nombreuses méthodes à base de solution unique ont été proposées dans la littérature. Parmi lesquelles : la descente, le recuit simulé, la recherche tabou, la recherche à voisinage variable (VNS: Variable Neighbourhood Search), la recherche locale réitérée (ILS: Iterated Local Search), la recherche locale guidée (GLS: Guided Local Search)…etc quelques métaheuristique

6.2.2.1.1 Le Recuit Simulé (Simulated Annealing)

Le Recuit Simulé [KIR 83] trouve ses origines dans le phénomène thermodynamique de recuit des métaux. Cette méthode imite une procédure utilisée par les métallurgistes qui, pour obtenir un alliage exempt de défauts, chauffent d'abord à blanc leur morceau de métal et laissent ensuite l'alliage se refroidir très lentement de manière à ce que les atomes aient le temps de s'ordonner régulièrement. Cependant, lorsque la baisse progressive de la température est trop rapide (comme pour la méthode de la Trempe), le solide présente alors des défauts, et un minimum local d'énergie est obtenu. Se basant sur l'évolution d'un système

thermodynamique, l'algorithme du recuit simulé accepte, pour sortir d'un minimum local, une dégradation de la fonction objectif avec une certaine probabilité. [DRE 03] [KAR 07].

L'algorithme illustré par la figure 1.5, se base sur deux résultats de la physique statistique [DRE 03].

1. Lorsque l'équilibre thermodynamique est atteint à une température donnée T, la probabilité, pour un système physique, de posséder une énergie donnée E, est proportionnelle au facteur de Boltzmann : $\exp(^{-E}/_{K_B T})$ ou K_B est la constante de Boltzmann.
2. Le deuxième résultat s'appuie sur l'algorithme de Metropolis [MET 53] pour simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température T donnée. Le principe de cet algorithme est le suivant : partant d'une configuration donnée, nous faisons subir au système une modification élémentaire ; si cette transformation a pour effet de diminuer la fonction objectif f (ou « énergie »), elle est acceptée ; si elle provoque au contraire une augmentation ΔEde la fonction objectif, elle est tout de même acceptée, mais avec la probabilité $\exp(^{f}/_{K_B T})$.

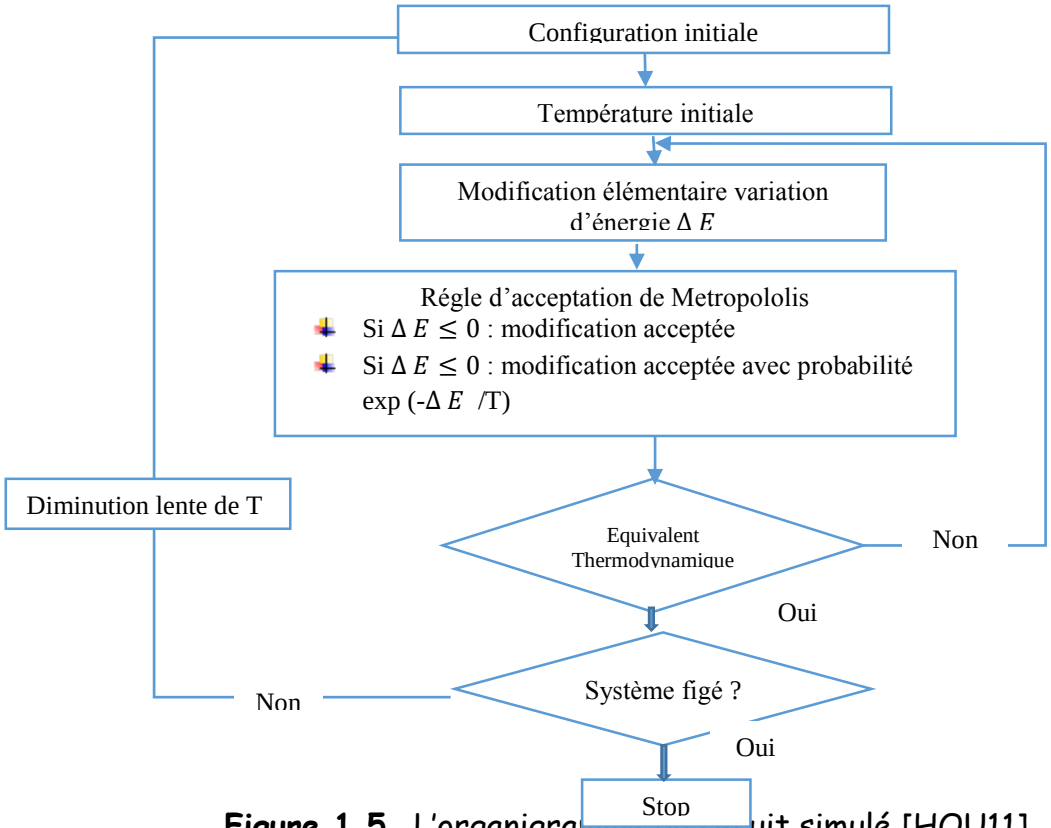


Figure 1.5. L'organigramme du recuit simulé [HOU11]

Plusieurs problèmes d'ordonnancement d'atelier ont été traités par la méthode de recuit simulé [RAA 00] [JOL 13].

6.2.2.1.2 La Recherche Tabou (Tabu Search)

Bien que son origine remonte à 1977, la recherche tabou n'est proposée qu'à la fin des années 80 par Fred Glover [GLO 89], [GLO 90]. Cette méthode, développée pour résoudre des problèmes combinatoires, la plupart NP-difficiles, propose de surmonter le problème des optima locaux par l'utilisation d'une mémoire.

La méthode tabou est une procédure itérative qui, partant d'une solution initiale, tente de converger vers la solution optimale en exécutant, à chaque pas, un mouvement dans l'espace de recherche. Chaque pas consiste d'abord à engendrer un ensemble de solutions voisines de la solution courante pour ensuite en choisir la meilleure, même si ce choix entraîne une augmentation de la fonction objectif à minimiser.

Le principe de cette méthode est définit par l'algorithme de la figure 1.6

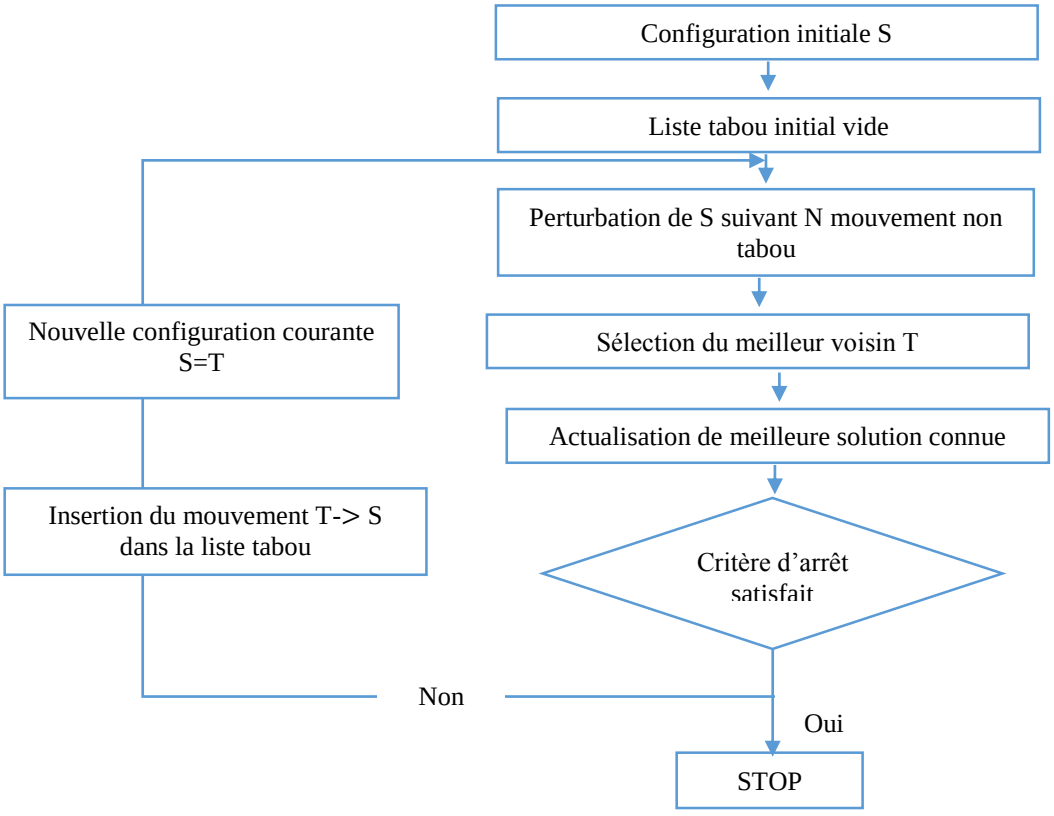


Figure 1.6. L'organigramme de la recherche tabou [HOU 11]

6.2.2.2 Les métaheuristiques à base de population de solutions

Les métaheuristiques à base de population de solutions débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure (l'optimum global) qui représentera la solution du problème traité. L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité.

Une grande variété de métaheuristiques basées sur une population de solutions a été proposée dans la littérature, Nous présentons dans la partie suivantes ceux ayons connus un développement remarquable ces deux dernières décennies à savoir les algorithmes génétiques, et les algorithmes à base d'intelligence par essaims : l'algorithme d'optimisation par essaim de particules, l'algorithme de colonies de fourmis, et la recherche coucou.

6.2.2.2.1 Les Algorithmes Génétiques (Genetic Algorithms)

Cette classe d'algorithmes a été inventée par Holland dans les années 70 [HOL73], pour imiter les phénomènes d'adaptation des êtres vivants. L'application aux problèmes d'optimisation a été développée ensuite par Goldbey [GOL 89].

On part d'une population initiale de N solution aléatoires, chacune étant codée par une chaine de caractères appelée chromosome. Holland recommande un codage sous forme de chaine de bits, mais ce n'est pas obligatoire.

Chaque chromosome et muni d'un mesure d'adaptation (fitness). En optimisation, il s'agit tout simplement de la fonction objectif . Un algorithme génétique choisit des paires de chromosome parents, en favorisant les plus adaptés (ceux de meilleur fitness), et engendre de nouvelles solutions (enfants) en appliquant des opérateurs de croissement (crossover) et de mutation. On espère ainsi que les bonnes solutions vont échanger par croisement leurs caractéristiques et engendrer des solutions encore meilleurs. Les principales étapes d'un algorithme génétique sont données par la figure 1.7.

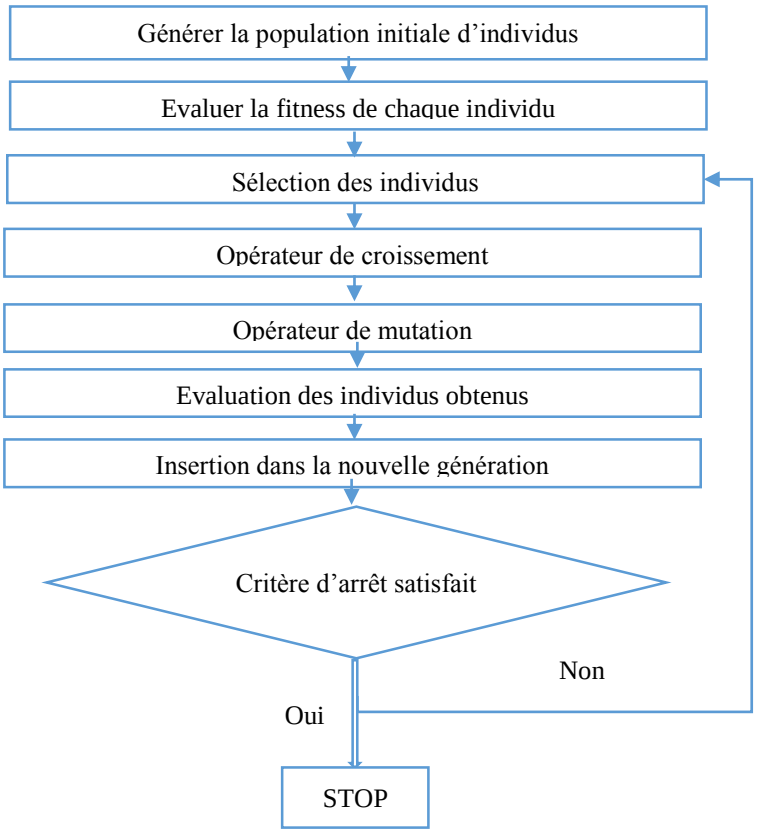


Figure 1.7 Principales étapes d'un algorithme génétique

6.2.2.2.2 Les Colonies de Fourmis (Ant Colony Optimization)

L'optimisation par l'approche colonie de fourmis est relativement récente. Le principe de l'optimisation est apparu au début des années 90. Il est le fruit des travaux de recherche de M. Dorigo, V. Maniezzo et A. Colorni. [DOR 91].

Les fourmis sont capables de résoudre collectivement des problèmes complexes. Pour cela, elles communiquent entre elles de façon locale et indirecte. Au cours de leur progression, les fourmis déposent une trace de phéromone ; elles choisissent ensuite leur chemin, selon une probabilité dépendant de la quantité de phéromone précédemment déposée par les autres fourmis.

L'exemple présenté dans la figure 1.8, montre la capacité d'adaptation des fournis. [KHA 10]

- Au départ, les fourmis disposent d'un chemin pour aller de leur nid, la fourmilière, à la source de nourriture, figure 8. (a)

- Un obstacle est placé entre le nid et la source de nourriture, les fourmis vont alors commencer par se séparer de part et d'autre de l'obstacle en déposant de la phéromone sur leur passage, figure 8. (b). Les fourmis les plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui ont emprunté les deux branches les plus courtes à l'aller et au retour. Ainsi la quantité de phéromone présente sur le plus court trajet est plus importante que celle présente sur le chemin le plus long.
- Finalement la quantité de phéromone sur le chemin le plus court fait que ce chemin finit par être emprunté par la plupart les fourmis, figure 8. (c)

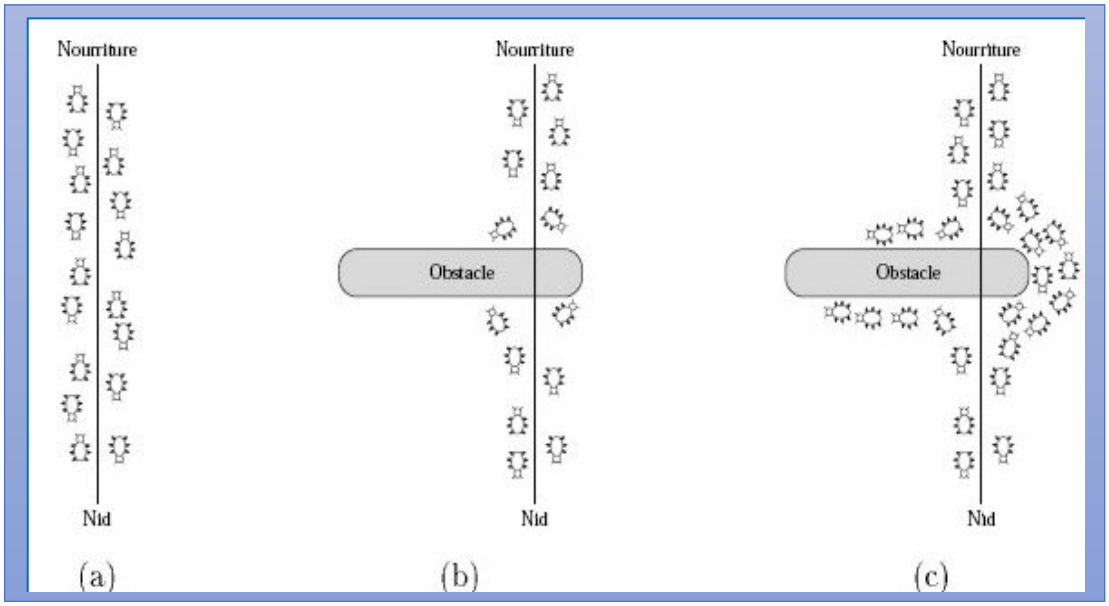


Figure 1.8. Le comportement des fourmis pour trouver le plus court chemin

Ant system (AS) est le premier algorithme de fourmi reposant sur le comportement de fourragement des fourmis [DOR91]. Cet algorithme a pour but de reproduire le comportement naturel des fourmis pour retrouver le meilleur chemin possible vers un objectif donné. On peut résumer ce mécanisme par l'algorithme 1.2

Algorithme 1.2. Algorithme de colonie de fourmis

(1) Initialiser

(2) Répéter

(3) Pour **chaque fourmi** faire

(4) Construire **une solution basée sue une procédure de construction, dépendant de la trace de phéromone**

(5) Mettre à jour **la trace de phéromone en se basant sur la qualité de la solution trouvée**
(6) Fin pour
(7) Jusqu'à **satisfaction d'un critère d'arrêt**

Beaucoup travaux ont été proposé pour résoudre le problème d'ordonnancement d'atelier nous citons : pour une seul machine [DEN 00] [GAG02], machines parallèle [SAN 05][SRI 09], atelier type flow shop [GAJ 06] [STU 98] [YIN 04] [YIN 07] atelier type job shop [COL 94] [ZHA 06],atelier type open shop [BLU 05], atelier flow shop flexible [ALK 07] [KHA 08a] [KHA 08b] et le job shop flexible [NAI 05] .

6.2.2.2.3 Les essaims particulaires (Particle Swarms Optimization)

L'Optimisation par essaim particulaire (OEP) est une méthode née aux Etats –Unis en 1995 sous le nom de Particle Swarms Optimization (PSO) .Ces concepteurs à l'origine cherchaient à modéliser des interactions sociales entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun , chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information . La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Ce comportement fait rappeler particulièrement celui des essaims d'abeilles, du fait qu'une abeille ayant trouvé un site promoteur sait informer certaine de ses 'consœurs' et que celles –ci vont tenir compte de cette information pour leur prochain déplacement [BER 01] [BON 99] [CLE 02].

Le fonctionnement de l'OEP fait que c'est une méthode qui peut être rangée dans les méthodes itératives (on approche peu à peu de la solution) et stochastiques (un fait appel au hasard).

Le principe de l'algorithme est : au départ un essaim est reparti au hasard de l'espace de recherche chaque particule a une vitesse aléatoire également, ensuite à chaque pas de temps : (figure 1.9)

- Chaque particule est capable d'évaluer la qualité de la position qu'elle a atteind jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser).

- Chaque particule est capable d'interroger un certain nombre de ses congénères (ses informatrices, dont elle- même) et d'obtenir de chacune d'entre elles sa propre meilleure performance (et la qualité afférente).
- Chaque particule choisit la meilleure des meilleures performances dont elle a connaissance, modifie sa vitesse en fonction de cette information et de ces propres données et se déplace en conséquence.

Figure 1.9. Schéma de principe du déplacement d'une particule. [CRA 01]

Les principales étapes de l'algorithme d'optimisation par essaims particulaires sont données par la figure 1.10

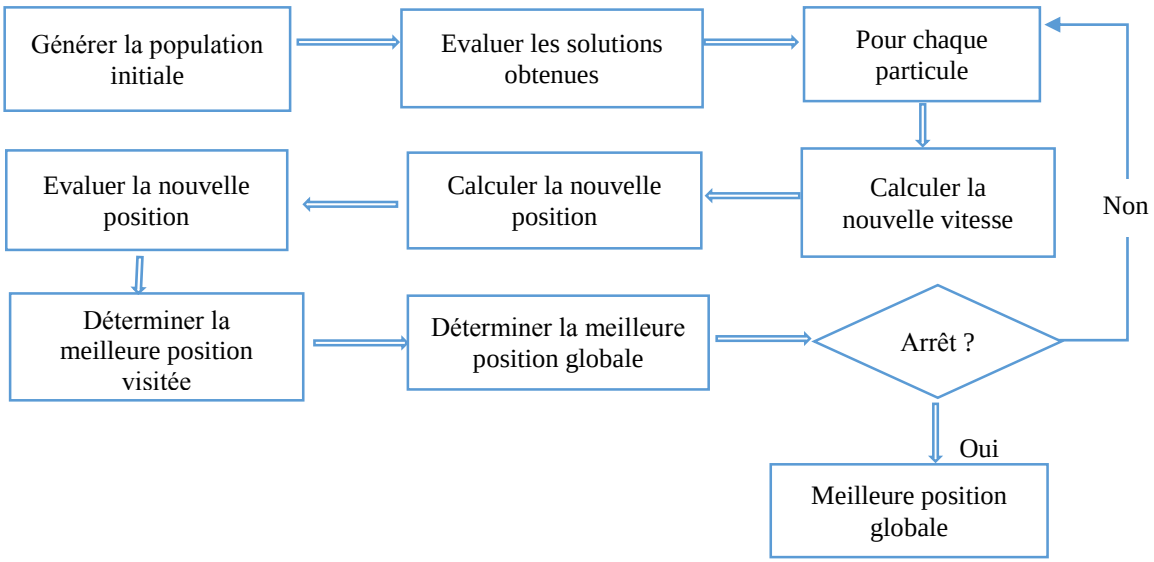


Figure 1.10. Principe de l'algorithme d'optimisation par essaims particulaires

Parmi les travaux en PSO a été proposé pour le problème d'ordonnancement des ateliers on cite [ALL 06] [PON 09].

6.2.2.2.4 L'algorithme de la recherche coucou

La recherche coucou a été proposée en 2009 par Yang et Deb sous le nom de Cuckoo Search (CS) [YAN 09] [YAN 10]. Elle s'inspire du comportement de reproduction d'une espèce spéciale d'oiseaux parasites de nids appelés « Coucous ».

Dans l'algorithme de la recherche coucou, une solution possible est appelée « nid » ou « coucou ». En fait, la recherche coucou part du principe que chaque nid comporte un seul coucou. Au cours du processus de la recherche de l'algorithme CS, chaque coucou crée son propre poussin en fonction de sa représentation actuelle (le coucou lui-même) et du vol de Lévy. L'évaluation de la qualité du poussin et de son père permet de sélectionner lequel d'entre eux survivra et subira quelques modifications dans le but d'améliorer sa qualité.

Les principales étapes d'un algorithme de la recherche coucou sont donnés par la figure 1.11 et au chapitre 6 nous représentons plus de détail sur cet algorithme.

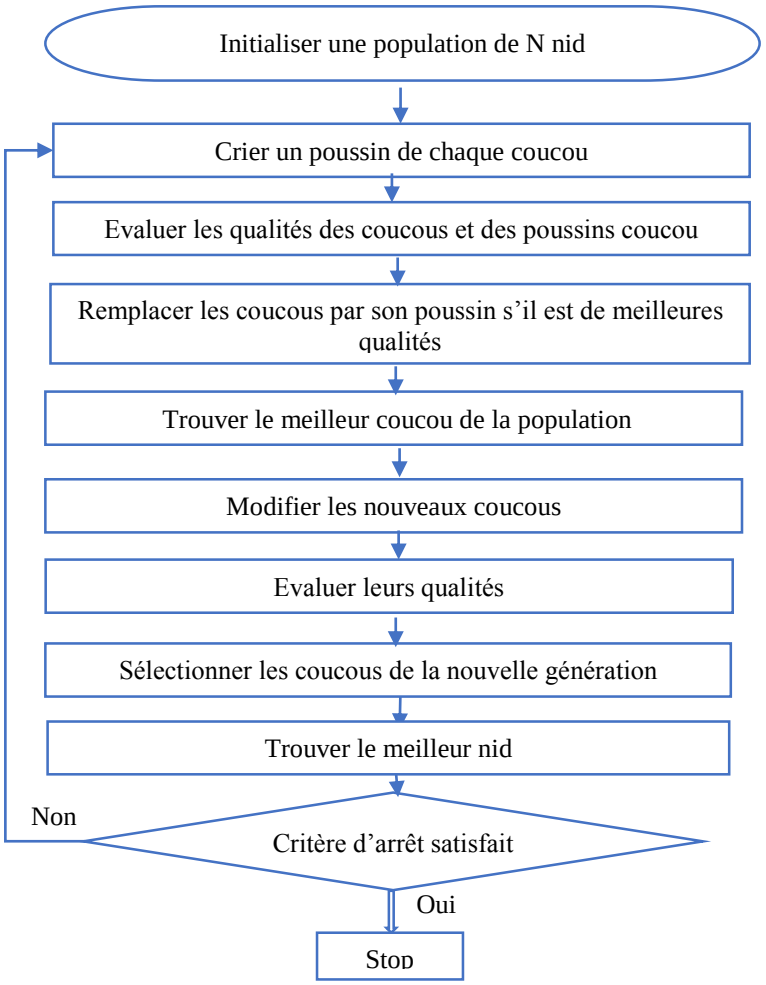


Figure 1.11. Principales étapes de l'algorithme de la recherche coucou

6.3 Les méthodes hybrides

Les méthodes hybrides c'est la combinaison des méthodes exactes et/ou des méthodes approchées pour créer de nouvelles méthodes appelés les méthodes hybrides.

L'idée est basée sur le théorème « No free lanch » qui indique qu'aucune méthode n'est efficace pour tous les types de problèmes. En fait, chaque méthode propose des avantages dont on cherche à maximiser et des lacunes dont on cherche à combler. Partant de ce principe, beaucoup de chercheurs ont envisagé la combinaison des méthodes de résolution des problèmes afin de tirer profit des points forts de chacune et de proposer des alternatives plus efficaces et plus performantes.

6.3.1 Coopération méta/méta

Les coopérations étaient à l'origine essentiellement réalisées entre différentes métaheuristiques. A peu près tous les types d'approches ont été proposés pour ce type de coopération, fait que les métaheuristiques hybrides sont devenues maintenant assez classiques dans le domaine de l'optimisation [BAS 05] .Par exemple : algorithme mimétique

6.3.1.1 L'algorithme mimétique

Les algorithmes mimétiques ou encore connus sous le nom d'algorithmes génétiques hybrides ou genetic local search ont été introduits par Moscato [MOS 89] en 1989. Cette heuristique est une hybridation des AG et de la recherche locale.

Ces deux méthodes sont complémentaires car l'une permet de détecter de bonnes régions dans l'espace de recherche alors que l'autre se concentre de manière intensive à explorer ces zones de l'espace de recherche [MOS 99]. Ainsi, on peut explorer rapidement les zones intéressantes de l'espace de recherche pour les exploiter en détail.

Algorithme 1.3. Algorithme mimétique

(1) Initialiser : générer une population initiale P de solutions

(2) Appliquer une procédure de recherche locale sur chaque solution de P

(3) Répéter

(4) Sélection : choisir deux solutions $\boldsymbol{x}$ et $\acute{\boldsymbol{x}}$

(5) Croisement : combiner deux solutions parents $\boldsymbol{x}$ et $\acute{\boldsymbol{x}}$ pour former une solution $\boldsymbol{y}$

(6) Recherche locale : appliquer une procédure de recherche locale sur y

(7) Mutation : appliquer un opérateur de mutation

(8) Choisir un individu y' pour être remplacé dans la population

(9) Remplacer $\acute{\boldsymbol{y}}$ par y dans la population

(10) Jusqu'à satisfaire un critère d'arrêt.

Dans cet algorithme, l'opérateur de mutation assure la diversification de la méthode, d'autre part, l'intensification est produite par l'application de la méthode de recherche locale.

6.3.2 Coopération méta/exacte

Nous avons vu que les méthodes exactes permettaient de résoudre des petits problèmes tandis que les (méta) heuristiques sont capables d'appréhender de grands problèmes sans pouvoir donner la solution optimale ou prouver que la solution fournie est optimale. Cependant les méthodes exactes peuvent néanmoins être utiles lorsque des sous-problèmes peuvent être extraits du problème global. Leur résolution permet en effet de contribuer à la recherche de la solution globale, soit en combinant judicieusement différents sous-problèmes, soit en hybridant résolution exacte de sous-problèmes et résolution heuristique du problème complet [DHA 05].

Cette constatation constitue le point de départ d'une approche hybride, assez originale dans le contexte des problèmes d'optimisation combinatoire visant à combiner résolutions exactes et métaheuristiques permettant de conserver aux mieux les avantages de chacune des approches.

7. Conclusion

Après la présentation des problèmes d'ordonnancement et de leurs principales caractérisations, différentes méthodes, exactes et approchées, pouvant être utilisées pour la résolution de ces problèmes sont introduites dans ce chapitre. Pour les méthodes exactes, nous avons présenté la programmation linéaire, la programmation dynamique et Procédure par séparation et évaluation.

Recherche tabou, recuit simulé et les algorithmes génétiques, les algorithmes de colonies de fourmis, les algorithmes d'optimisation par essaim particulaire, recherche coucou ont fait l'objet de la présentation pour les méthodes approchées ou méta- heuristiques.

La description de notre problème d'ordonnancement job shop classique et job shop flexible est présenté dans le chapitre suivant.

CHAPITRE 2

DESCRIPTION DU PROBLEME

Ce chapitre est scindé en deux parties , la première est consacré à la description du problème d'ordonnancement job shop classique et job shop flexible . Dans la seconde partie nous présentons une modélisation mathématique du problème par les RdP.

Ce que tu veux me dire, est-ce vrai? Est-ce bien? Est-ce utile? Sinon je ne veux pas l'entendre.

Socrate

1. Introduction

Dans ce chapitre, nous présentons plus en détails les systèmes de production de type job shop classique et job shop flexible qui font l'objet de cette étude. Ensuite, nous décrivons la formulation mathématique de ces deux types de problèmes. Nous présentons à la fin de la modélisation graphique par les RdP ce type de problème d'ordonnancement job shop.

2. Description du problème

2.1 Problème Job shop classique

Le problème de type job-shop est l'un des problèmes les plus étudiés dans la littérature de l'ordonnancement. Son importance théorique ainsi que la modélisation de nombreuses applications industrielles sous forme de systèmes de type job-shop le rendent très intéressant.

La particularité des problèmes de type job shop classique par rapport aux problèmes de production de type flow shop est la gamme opératoire qui n'est pas fixe (atelier à cheminement libre), ainsi que le nombre d'opérations qui n'est pas forcément le même pour tous les jobs. Ces systèmes sont considérés comme des systèmes fortement combinatoires.

Ils sont composés d'un ensemble de n jobs, chacun composé d'un ensemble de n_i opérations qui peuvent être exécutées sur m machines en respectant les contraintes suivantes :

- L'ordre de passage des opérations d'un job sur les différentes machines est fixe pour chaque job et peut être différent d'un job à un autre.
- Une machine ne peut exécuter qu'une seule opération à la fois et à l'instant initial toutes les machines sont disponibles.
- Une opération ne peut être exécutée que sur une seule machine et sans interruption.
- La capacité de stockage entre les machines est considérée comme infinie.
- La préemption des opérations n'est pas autorisée.

Le tableau 2.1 présente un problème de type de job shop classique composé de 5 jobs et 4 machines.

Numéro de job	Type de machine	Temps d'exécution
1	1	8
	2	7
	3	14
	4	9
2	4	6
	1	13
	2	10
	3	17
3	1	18
	4	16
	3	11
	2	12
4	3	12
	4	9
	2	15
	1	11
5	2	11
	1	7
	4	16

Tableau 2.1. Un exemple du problème job shop
Quatre machines et Cinq jobs

La représentation de job shop à 5 jobs et 4 machines (tableau 2.1) peut être représenté par la figure 2.1 .

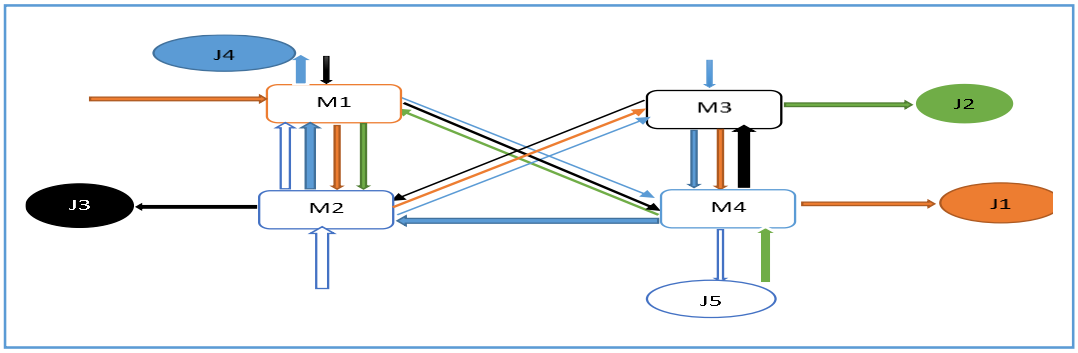


Figure 2.1.Représentation d'un système de type Job-shop classique

2.2 Problème de job shop flexible

Une façon d'augmenter la productivité d'un atelier est d'augmenter sa flexibilité, c'est de multiplier le nombre de machines qui réalisent la même tache. Le modèle résultant est connu dans la littérature comme un job-shop flexible. Dans ce modèle, les machines qui effectuent la même opération sont groupées dans un même étage.

Les problèmes de type job-shop Flexible (*FJSP*) sont une extension de deux problèmes d'ordonnancement : le problème de job-shop classique et le problème d'ordonnancement des machines parallèles. Le *FJSP* peut être vu comme un problème de job shop qui possède une ou plusieurs machines parallèles par étage. La machine nécessaire pour exécuter une opération n'est pas connue a priori. Par conséquent, le problème se compose de deux parties dont la première est de trouver la séquence des jobs sur les étages, la deuxième partie est de trouver la machine nécessaire à l'exécution d'un job tout on respectant les contraintes du job-shop.

Un exemple de ce problème à *m* étages et trois machines maximum par étage, est donné dans la Figure 2.2.

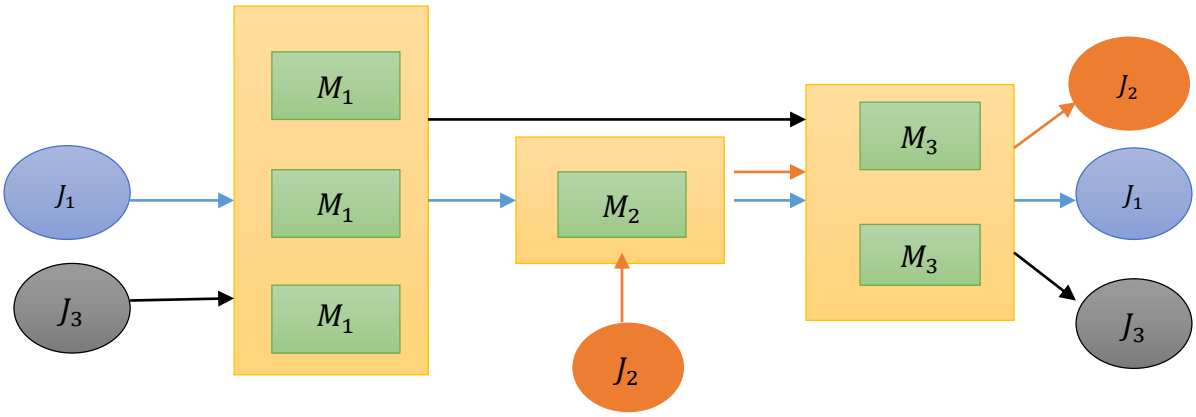


Figure 2.2.Représentation d'un système de type Job-shop flexible à deux étages

Une remarque importante dans les problèmes de type job-shop flexible est que le nombre de machines peut varier d'un étage à l'autre, donc on distingue deux types de flexibilité de machines :

- La flexibilité partielle (*P- FJSP*) : si chaque opération peut être exécuté sur quelque machines candidates, parmi l'ensemble de machines

- La flexibilité totale (*T- FJSP*) : si chaque opération peut être exécuté sur n'importe quelle machine (l'ensemble des machines candidates égal à l'ensemble globale des machines)

Les performances des machines ne sont pas forcément les mêmes pour toutes les machines d'un même étage. Ainsi on distingue :

- Machines identiques : La durée d'exécution d'une tâche est la même sur toutes les machines.
- Machines uniformes : La durée d'exécution d'une tâche est similaire sur toutes les machines.
- Machines indépendantes : La durée d'exécution d'une opération dépend de la machine sur laquelle elle est exécutée.

  Le travail présenté dans ce mémoire est relatif à des machines indépendantes.

3. Modélisation du problème

La modélisation, est en général, une étape très importante dans la résolution d'un problème d'ordonnancement. C'est une écriture simplifiée de toutes les données du problème permettant d'en traduire tous les détails pour mieux représenter la réalité des choses.

Notation :

Dans ce qui suit nous allons donner les notations des données utilisées dans le cadre de notre problème :

$\boldsymbol{\Omega}$: Ensemble de toute les machines disponibles

$\boldsymbol{n}$: Nombre totale de jobs

$\boldsymbol{m}$: Nombre totale de machines

$\boldsymbol{i}$: indice du $\boldsymbol{i^{eme}}$ job

$\boldsymbol{j}$: Indice de la $\boldsymbol{j^{éme}}$ opération du job $\boldsymbol{j_i}$

$\boldsymbol{J_{io}}$: Nombre totale des opérations dans le job $\boldsymbol{J_i}$

$\boldsymbol{O_{ij}}$: $\boldsymbol{j^{éme}}$ Opération du job $\boldsymbol{j_i}$

$\boldsymbol{\Omega_{ij}}$: Ensemble des machines disponibles pour l'opération $\boldsymbol{O_{ij}}$

$\boldsymbol{p_{ijk}}$: temps d'exécution de l'opération $\boldsymbol{O_{ij}}$sur la machine k

$\boldsymbol{S_{ijk}}$: Date de début de l'opération $\boldsymbol{O_{ij}}$sur la machine k

$\boldsymbol{E_{ijk}}$: Date de fin de l'opération $\boldsymbol{O_{ij}}$ sur la machine k

L=$\sum_1^n \boldsymbol{J_{io}}$: somme de toute les opérations dans tous les jobs

3.1 Modélisation du problème Job shop classique

Deux méthodes de modélisation sont possibles pour notre problème job shop classique : la modélisation mathématique et la modélisation graphique

3.1.1 Modélisation mathématique

Le modèle que nous donnons est basé sur la formulation proposée par [PAN 09] dans le cas d'un job shop classique.

La fonction Objectif :

$$\text{Minimise}\left[\text{Max}\left(C_{1J_1}, C_{2J_2}, \dots C_{nJ_n}\right)\right] \quad (1)$$

Sous contraintes :

$$C_{ij} - S_{ij} - P_{ij} = 0 \qquad \forall\, i,j \quad (2)$$

$$C_{i'j'} - C_{ij} + H(1 - Y_{iji'j'}) \geq P_{i'j'} \qquad \forall (i,j), \forall (i',j') : O_{ij} \in N_k , O_{i'j'} \in N_k \quad (3)$$

$$C_{ij} - C_{i'j'} + H\left(Y_{iji'j'}\right) \geq P_{i'j'} \; \forall (i,j), \forall (i',j') : O_{ij} \in N_k , O_{i'j'} \in N_k \quad (4)$$

$$S_{ij} \geq 0 \qquad \forall\, i,j \quad (5)$$

$$S_{ij+1} - C_{ij} \geq 0 \qquad \forall\, i,j = 1, \dots, J_i - 1 \quad (6)$$

$$Y_{iji'j'} = \begin{cases} 1, & \text{si opération } O_{ij} \text{ procède } O_{i'j'} \\ 0, & \text{sinon} \end{cases} \quad (7)$$

Signification des équations.

Contrainte 2 : contrainte de précédence entre les opérations d'un même job : Une opération ne peut commencer à être exécutée avant la fin du traitement de l'opération précédente du même job.

Contraintes 3 ,4 : contrainte de partage de ressource. Une machine ne peut traiter qu'une seule opération à la fois,

Contrainte 5 : cette contrainte signifie que la date de début de chaque opération doit être positive ou nulle.

Contrainte 6 : cette contrainte impose que la valeur du makespan doit être supérieure ou égale aux dates de fin des dernières opérations pour tous les jobs.

3.1.2 Modélisation graphiques

Un problème d'ordonnancement d'atelier de production type job shop classique peut être modélisé par un graphe, dit graphe potentiel - tâches, constitué par :

- des nœuds représentant les tâches,
- des arcs conjonctifs illustrant les contraintes de précédence (en indiquant les durées des tâches),
- des arcs disjonctifs indiquant les contraintes de ressources [ROY70], [GOT93] et [JAI99].

Les méthodes graphiques ont connu une importante évolution surtout avec l'apparition des Réseaux de Petri (RdP) [CHR83], [CAR84b] qui permettent de traduire plusieurs notions fondamentales qui ont un lien avec les problèmes d'ordonnancement telles que :

- les conflits sur les ressources,
- les durées opératoires certaines (RdP temporisés),
- les durées opératoires aléatoires (RdP stochastiques),
- les gammes,
- les disponibilités, les multiplicités et les capacités des ressources (RdP synchronisés et à capacités), la répétitivité (RdP cycliques).

3.1.3 Principe des réseaux de Petri

A. Définition mathématique

Un RdP R = (P, T, L) est un triplé, défini sur deux ensembles finis et distincts de sommets.

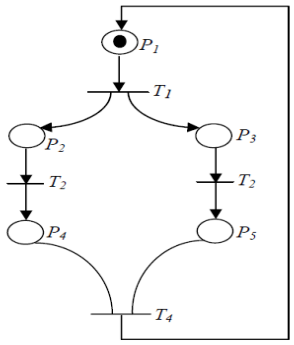
- Les places, P /P = $\{P_1 , P_2 , P_3 \ \dots\dots , P_i , \dots P_n \}$ avec $n = |P|$ qui représente les états du système

- Les transitions, T /T = $\{T_1, T_2, T_3 \ldots\ldots, T_j, \ldots T_m\}$ avec $m = |T|$ qui représente l'ensemble des événements dont l'apparition modifie l'état du système ;
- Les liens ou flèches, L satisfont à L ⊂ (P × T) ∪ (T× P)
  - Pré : P ×T → N : application d'incidence avant ;
  - Post : P×T → N : application d'incidence arrière.

b. Représentation graphique

Un RdP est un graphe orienté biparti comportant deux types de nœuds, illustrant respectivement les places et les transitions.

Traditionnellement, les places sont illustrées par des cercles, et les transitions par des rectangles ou trais (Figure 2.3).



$P = \{p_1, p_2, p_3, p_4, p_5\}$
$T = \{t_1, t_2, t_3, t_4, t_5\}$
$M = [1,0,0,0,0]$

Figure 2.3. Exemple d'un RdP.

c. Notation

On appelle : * ${}^0T_j$ l'ensemble des places d'entrées de la transition T_j tel que :

${}^0T_j = \{\forall P_i \epsilon P : Pré(P_i, T_J) > 0\}$; ${}^0T_j$: représente également les données d'entrées pour l'évènement T_j .

*T_j^0 : l'ensemble des places de sortis de la transition T_j tel que :

$T_j^0 = \{\forall P_i \epsilon P : Pré(P_i, T_J) > 0\}$; T_j^0 : Représente également les données de sortis pour l'évènement T_j .

d. Matrice d'incidence

Il est possible de représenter le Pré et Post par des matrices n × m où Pré (P_i, P_j) (respectivement Post (P_i, P_j)est un élément de la matrice Pré (respectivement Post).

On appelle matrice d'incidence avant la matrice :

$$W^- = [w_{i,j}^-], \text{ou } w_{i,j}^- = Pré\ (P_i, T_j)$$

On appelle matrice d'incidence arrière la matrice :

$$W^+ = [w_{i,j}^+], \text{ou } w_{i,j}^+ = Post\ (P_i, T_J)$$

On appelle matrice d'incidence la matrice

$$W = W^+ - W^-$$

e. Marquage d'un RdP

Le marquage d'un RdP est une fonction M : P→N, où N est l'ensemble des entiers. Graphiquement, il est présenté par une distribution de jetons dans les places de RdP.

Un marquage initial M_0 peut être associe au RdP R = (P, T, L) , on parle alors de RdP marqué, et on le note par (R, M_0) ou bien par (P, T, L, M_0).

Le marquage représente l'état (global) du système. La dynamique du RdP est exprimée par des franchissements (ou des tirs) de transitions.

f. franchissement d'une transition

Une transition T_j est franchissable si est seleument si toutes les places d'entrée de cette transition contiennent un nombre de marquage au moins égale aux poids de l'arc reliant la place à T_j . Ou encore

$$T_j \text{ est franssichable } \Rightarrow \forall P_i \epsilon P : M(P_i) \geq Pré\ (P_i, T_J)$$

Le franchissement de T_j consiste à enlever de toutes places $P_i \epsilon\ {}^0T_j$ le nombre de marque égales au $Pré\ (P_i, T_J)$ et d'ajouter à chaque places $P_i \epsilon T_j^0$ le nombre de marques egales au $Post\ (P_i, T_J)$

g. graphe de marquage

Le graphe des marquages d'un RdP R = (P, T, L, M_0) est le graphe orienté (K, G), ou l'ensemble des sommets est l'ensemble des marquages accessibles ${}^0K = {}^*M_0$, et l'ensemble des flèches G ⊂ K ×T × K est défini par $((M, T_j, M') \in \Leftrightarrow\ G \leftrightarrow M[T_j > \acute{M})$.

h. L'extension des RdP

De nombreuses extensions ont été apportées depuis le développement des RdP pour permettre une représentation de certains comportements ou structures. Nous présentons ici quelques-unes de ces extensions.

1. Le RdP à capacité

Dans un RdP ordinaires, la capacité des places n'est pas limitée. Ici, on affecte une capacité, défini par un nombre entier positif associé aux places. Ainsi, le franchissement d'une transition est conditionné par la capacité des places en aval.

2. Le RdP généralisé

On affecte un poids, (nombre entier positif associé aux arcs). Ce poids indique le nombre de jetons consommés ou crées lors du franchissement d'une transition. On retrouve ces poids dans la matrice d'incidence. Il s'agit d'une simplification par agrégation d'un RdP ordinaire.

3. Les arcs inhibiteurs

L'arc inhibiteur permet de simplifier la représentation graphique d'un RdP. Il s'agit d'un arc orienté qui part d'une place P_i et arrive à une transitionT_j, et telle que la transition $T_j$est valide seulement si la place P_i est vide.

4. Le RdP coloré

Dans un RdP coloré, les marques peuvent être différenciées par des couleurs. Si le nombre de couleurs est fini, alors on peut se ramener à un RdP classique.

5. Le RdP continu

Dans un RdP continu [HOU 08] le marquage des places n'est plus un nombre entier mais un nombre réel positif. Ce type de RdP permet d'analyser des performances en terme de débit. Il est fortement utile lorsque le nombre de marquages dans un RdP classique devient trop important ou pour représenter des processus continus.

6. Le RdP hybride

Ces RdP sont utilisés pour représenter des SdP ayant à la fois des caractéristiques discrètes et continues.

7. Les RdP non autonomes

a. Le RdP à prédicats

Le prédicat permet d'affecter des actions de transformation sur les jetons lors du franchissement d'une transition.

b. Le RdP synchronisé

c. On associe en plus des conditions de franchissement d'un RdP ordinaire (présence de marques dans les places en amont) un événement de franchissement. Ces événements sont externes au système du RdP.

d. Le RdP T-temporisé

Dans les RdP T-temporisés, une durée de franchissement est affectée aux transitions. A chaque transition T_j est associée une temporisation dj constante.

Cette particularité permet de décrire des systèmes dont le fonctionnement est dépendant du temps. Ces RdP sont utiles pour l'évaluation de performance.

e. Le RdP P-temporisé

L'aspect temporel est ici affecté aux places. Aussi, à chaque place P_i est associée une temporisation d_i constante.

f. Le RdP P-interprété

Un RdP interprété est un RdP P-temporisé et synchronisé qui comporte une partie opérative. Ainsi à chaque transition T_j est associé un événement E_j et une condition C_j de franchissement. A chaque place P_i, on associe une temporisation di et une opération O_i

g. Le RdP stochastique

Dans un RdPS, la durée affectée n'est plus déterministe mais aléatoire. On peut grâce à cette particularité prendre en compte des événements probabilistes comme la panne d'une machine.

3.1.4 Modélisation par les réseaux de Petri de problème d'ordonnancement Job shop classique

La modélisation d'un job shop classique se fait de la manière suivante : On commence par modéliser le processus de fabrication de chaque job. Pour cela, chaque opération réalisée sur un job dans sa gamme de fabrication est représentée par une transition T. Les différentes transitions associées aux opérations successives que subit le job sont séparées les unes des autre par des places symbolisant des stocks tampons ou plus généralement des moyen de stockage. Ces derniers sont d'abords supposés infini .A chaque transition est associé un temps de franchissement. Celui égal au temps que prend la machine pour réaliser l'opération représentée par la transition. Enfin la gamme de la fabrication comporte une transition puits dont le franchissent représente la fin de fabrication d'une unité de produit .Pour chaque machine on

définit une place ressource contenant exactement un jeton .On modélise le mode opératoire cyclique de l'atelier, on suppose qu'une nouvelle job rejoint le système dès qu'une job du même type est terminé. Modèle d'atelier type Job Shop classique par RdP de l'exemple de tableau 2.1 est présentée dans la figure 2.4.

3.1.4.1 Présentation du modèle

Le modèle RdP de l'atelier job shop classique donné par le tableau 2.1 est représenté par la figure 2.4. et le graphe de l'évolution de marquage en fonction de makespan donnée dans la figure 2.5. L'implémentation est faite au logiciel SYRFICO sur une machine avec un processeur 2.3 GH et 4 GO de RAM.

Définition des transitions et des places du modèle

Transitions / places	Type	Déscription
T_1 , T_7 ,T_{13} ,T_{19} ,T_{25}	Transitions sources	Lancemement de la production
T_2 , T_3 ,T_4 ,T_5 ,T_8 ,T_9 , T_{10} ,T_{11} ,T_{14} ,T_{15} ,T_{16} , T_{17} ,T_{20} ,T_{21} ,T_{22} ,T_{23} , T_{26} ,T_{27} ,T_{28}	Transtions temporisées	Processing time de chaque machine dans chaque jobs
T_6 , T_{12} ,T_{18} ,T_{24} ,T_{29}	Transitions puits	Fin de la production
$p_1, p_2, p_3, p_4, p_5, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{25}, p_{26}, p_{27}, p_{28}, p_{29}, p_{33}, p_{34}, p_{35}, p_{36}$	Places	Stocks taupous
$p_6, p_{51}, p_8, p_{40}, p_{14}, p_{15}, p_{16}, p_{42} p_{22}, p_{23}, p_{24}, p_{41}, p_{31} p_{30}$, $p_{31}, p_{32}, p_{44}, p_{43}$, p_{45}, p_{38}, p_{39}	Places	la contrainte disjonctive
$p_{37}, p_{40}, p_{41}, p_{42},$	Places de décision SP	Les décisions concernant la séquence d'ordre d'entrée des jobs dans une chaque machine

Tableau 2.2. Définition des transitions et des places du modèle

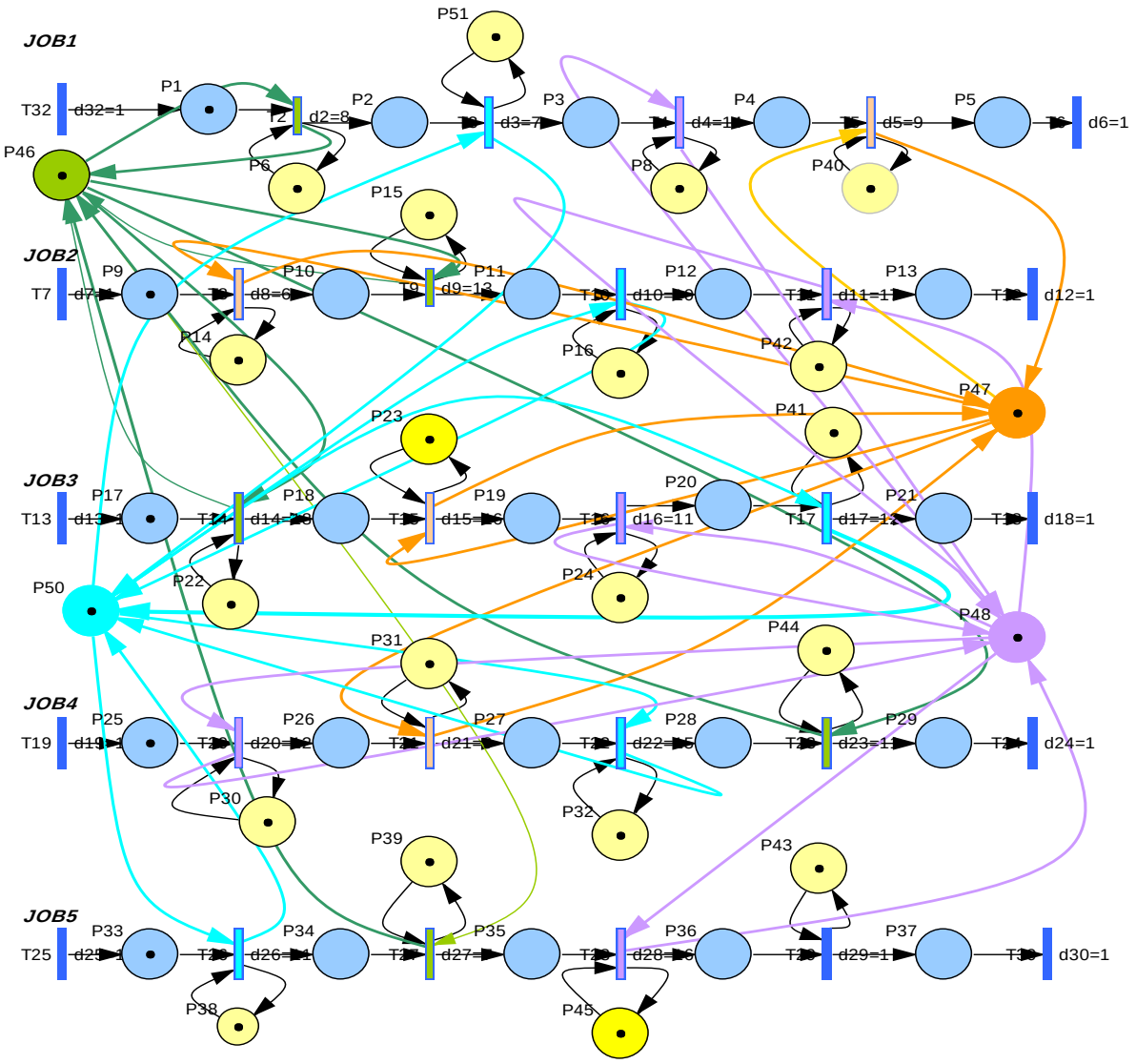


Figure 2.4 . Modèle d'atelier type Job Shop classique par RdP

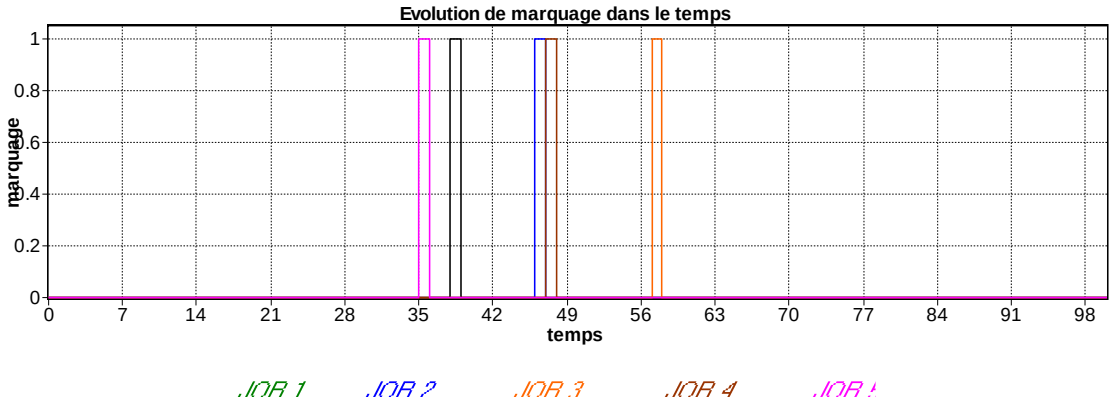


Figure 2.5. Présentation de makespan de tous les jobs

3.2 Modélisation du problème Job shop flexible

D'un maniéré analogue que pour le Job shop classique, la modélisation du problème job shop flexible, peu se faire soit mathématique ou graphique. Dans le cadre de notre travail nous ne présentons que la modélisation mathématique.

3.2.1 Modélisation mathématique du problème job shop flexible

Le modèle que nous donnons est basé sur la formulation proposée par [PAN 09] dans le cas d'un job shop flexible

La fonction Objectif :

Minimise $\left[Max\left(C_{1J_1}, C_{2J_2}, \dots C_{nJ_n}\right)\right]$ (8)

Sous contraintes :

C_{ij}-S_{ij}-$\sum_{k:O_{ij\in\Omega_{ij}}}\left(P_{ijk}.X_{ijk}\right) = 0 \quad \forall\, i,j$ (9)

$C_{i\grave{j}}$-C_{ij}+H$\left(1 - Y_{iji\grave{j}k}\right) + H\left(1 - X_{ijk}\right) + \left(1 - X_{i\grave{j}k}\right) \geq P_{i\grave{j}k}$ (10)

$C_{i\grave{j}}$-$C_{i\grave{j}}$+H$\left(Y_{iji\grave{j}k}\right) + H\left(1 - X_{ijk}\right) + \left(1 - X_{i\grave{j}k}\right) \geq P_{i\grave{j}k}$ (11)

$S_{ij} \geq 0 \qquad \forall\, i,j$ (12)

S_{ij+1}-$C_{ij} \geq 0 \forall\, i,j$, j=1…J_{i-1} (13)

$\sum_{k:O_{ij\in\Omega_{ij}}}\left(X_{ijk}\right) = 1$, $\forall\, i,j$ (14)

$$X_{ijk} = \begin{cases} 1 \; si \; opération \; O_{ij} \; traite \; par \; la \; machine \; k \\ 0 \; sinon \end{cases} \quad (15)$$

$$Y_{iji\grave{j}k} = \begin{cases} 1 \; si \; opération \; O_{ij} \; précéde \; O_{i\grave{j}} \; sur \; la \; machine \; k \\ 0 \; sinon \end{cases} \quad (16)$$

La fonction objectif (8) minimise le makespan, la contrainte (9) impose qu'une opération ne peut commencer avant la fin du traitement de l'opération précédente du même

job. Les contraintes (10.11) impose qu'une machine ne peut traiter qu'une seule opération à la fois, la contrainte (12) signifie que la date de début de chaque opération doit être positive ou nulle. La contrainte (13) impose que la valeur du makespan doit être supérieure ou égale aux dates de fin des dernières opérations pour tous les jobs. La contrainte (14) signifie qu'une opération ne peut être traiter que sur une seul machine.

4. Représentation des solutions du problème d'ordonnancement Job Shop

La solution d'un problème d'ordonnancement est représentée généralement par le digramme de Gantt. Le diagramme de Gantt est un outil élaboré en 1917 par Henry L. Gantt, qui permettant de représenter l'ordonnancement des tâches nécessaires à la réalisation d'un atelier de production type job shop. Ce diagramme présente en ordonnée la liste des tâches, notées J_i à exécuter par les machines notées M_j et en abscisse l'échelle du temps. Le digramme de Gantt peut avoir deux représentations :

- la représentation (a), « Produits ou jobs »,
- la représentation (b), « Machines ».

Nous utilisons la représentation (b), les machines pour présenter notre solution dans ce qui suit.

5. Conclusion

Dans ce chapitre, nous avons présenté le cadre dans lequel est réalisée cette thèse. Nous avons tout d'abord défini les systèmes de production de type job-Shop classique et job shop flexible. Nous avons ensuite présenté une modélisation mathématique de problème d'ordonnancement Job Shop .Nous avons utilisé les réseaux de Petri pour résoudre ce problème, nous avons abouti à des résultats satisfaisants mais à une certaine limite de la taille de problème (nombre de job et le nombre de machine). Mais dans le cas où la taille augmente il faudra chercher une heuristique qui économise plus de temps et de l'espace de stockage.

Dans le chapitre suivant, nous nous consacrons à l'état de l'art sur l'application des algorithmes génétiques sur le job shop classique et le job shop flexible.

CHAPITRE 3

ÉTAT DE L'ART

Les problèmes d'ordonnancement des ateliers ont été très étudiés dans la littérature depuis plus de 50 ans. Nous présentons un état de l'art, non exhaustif, de ces problèmes relatifs principalement aux ateliers job shop classiques et job shop flexible et traité par les algorithmes Génétiques.

Nos expressions doivent toujours être le miroir de nos pensées.
Socrate

1. Introduction

Dans ce chapitre, nous dressons un état de l'art portant sur l'application des algorithmes génétiques aux problèmes d'ordonnancement d'atelier. Il est organisé en deux parties. Dans la première, nous présentons des travaux existants sur la complexité et les approches développées pour résoudre les problèmes d'atelier à cheminements multiples (job-shop classique). Les travaux relatifs aux ateliers à cheminements multiples flexible (job-shop flexible) font l'objet de la seconde partie.

2. Les Problèmes D'ordonnancement Job Shop Classique

Les problèmes d'ordonnancement de type job-shop font partie des problèmes les plus étudiés dans la littérature [JAI 98]. Cela s'explique par deux raisons : la première résulte du fait que leur modélisation théorique est difficile, la seconde du fait que dans la réalité de nombreux systèmes sont de type Job shop.

Le job shop est une généralisation directe du problème d'or²donnancement d'atelier de type flow-shop, pour lequel la gamme opératoire des jobs peut varier d'un job à l'autre. Le problème est donc fortement combinatoire.

2.1. Travaux relatifs à la complexité des problèmes d'ordonnancement job shop classique

Le problème d'ordonnancement job shop avec le makespan comme objectif est un problème d'optimisation combinatoire classique qui a reçu une attention considérable dans la littérature.

Parmi les travaux existant sur la complexité de ces problèmes, nous trouvons les travaux de Garey et Johnson (1979) et Gonzalez et Sahni (1978) [GON78] [SOT 95]. Les auteurs ont classé le problème de job-shop parmi les problèmes NP-difficiles au sens fort Rinnooy Kan [GAR 76] confirme le fait que les JSSP sont des problèmes NP-difficile par la réduction polynômiale d'un problème de flow-shop vers un problème de voyageur de commerce. Cependant il existe certains cas de JSSP de type polynomiale autrement dit solvable qui sont :

- Le cas de l'ordonnancement de deux Jobs par la méthode graphique comme décrit dans [BRU88] et introduit par Akers [AKE56]. L'approche a été reprise pour calculer les bonnes bornes inférieures [CAR82] [BRU93] ;

- Cas de flow shop à deux machines [JOH 54][GON78] ;
- Le problème de l'atelier de type job shop a deux machine où chaque job se compose au plus de deux tâches [JAC56] ;
- Le cas de l'atelier de job shop de deux machines avec des temps de traitement égale à l'unité [HEF82] [KUB94] ;
- Le cas d'atelier de job shop a deux machines avec un nombre fixe de job [BRU94].

2.2 Travaux relatifs à l'utilisation des AGs pour l'étude des problèmes d'ordonnancement job shop classique

L'état de l'art que nous présentons l'application des algorithmes génétiques pour la résolution des problèmes d'ordonnancement d'atelier job shop classique.

2.2.1. Présentation des différents travaux suivant les approches définis par Cheng

Plusieurs approches ont été proposées pour représenter le problème de job shop classique avec les algorithmes génétiques. Cheng et al 1996 ont recensé neuf qui sont [CHE 96] [KEB 08] (figure 2.1) :

1. Représentation basée sur l'opération (Opération based representation)
2. Représentation basée sur la tache (job) (Job based representation)
3. Représentation basée sur relations de pair de tache (Job pair relation based représentation)
4. Représentation basée sur le temps d'achèvement (Completion time based representation)
5. Représentation basée sur des clés aléatoires (Rendom keys représentation)
6. Représentation basée sur des listes de préférences (Preference list based representation)
7. Représentation basée sur les règles de priorité (Priority rule based representation)
8. Représentation basée sur le graphe disjonctif (Disjunctive graph based representation)
9. Répresentation basée sur la machine (Machine based representation)

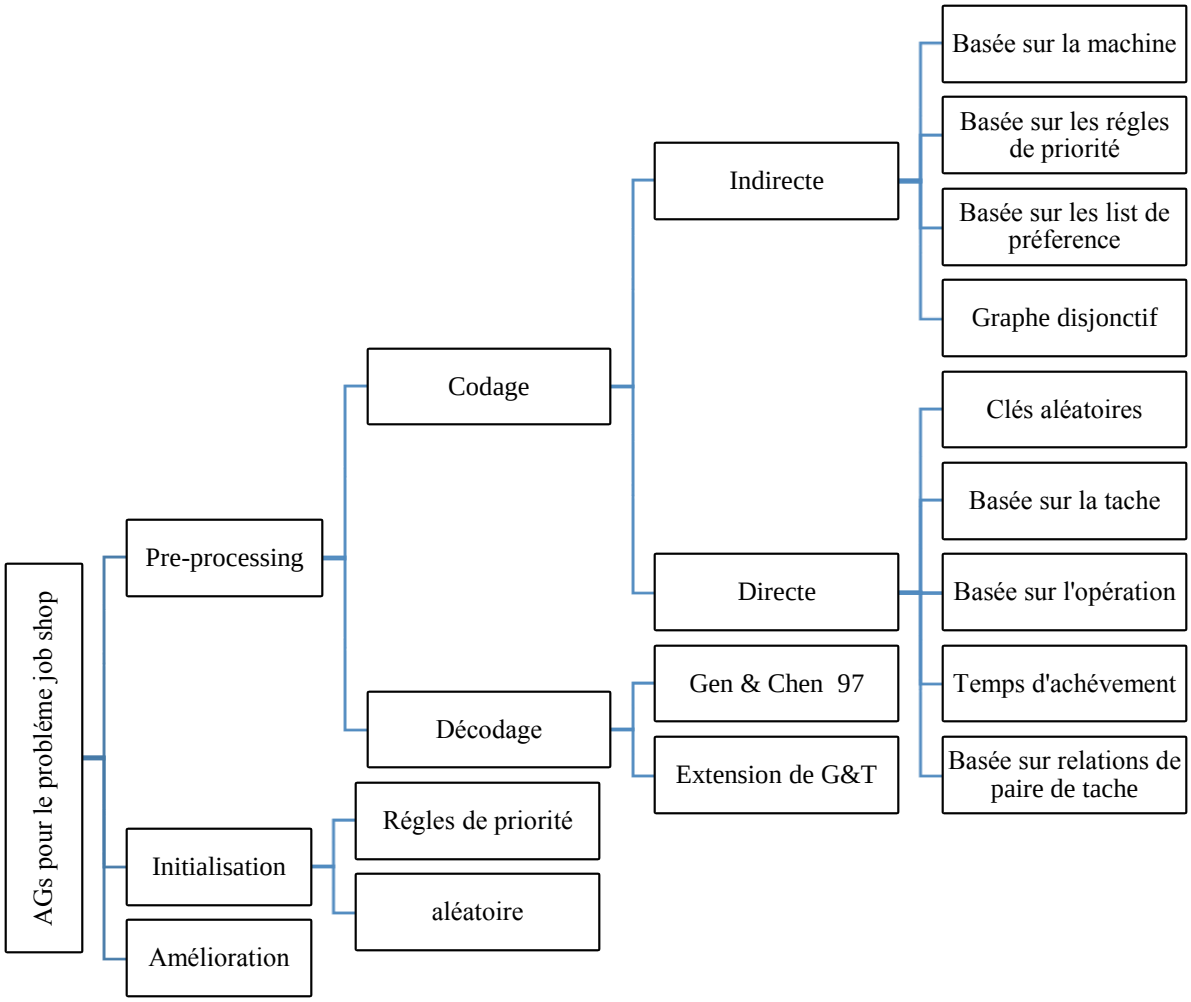


Figure 3.1.Représentation des algorithmes génétique pour le problème d'ordonnancement Job Shop classique [CHE 96] [GEN 99] [ABD14]

Les cinq premières approches sont des représentations directes du problème, tandis que les autres sont indirectes qui nécessitent implantation de décodeur.

A partir de cette classification nous présentons dans le tableau 3.1 un résumé des différents travaux de recherche relatif à ce domaine.

Représentation	Codage	Exemple de la littérature	Avantage	Inconvénient	Espace de recherche	Décodage
Représentation directe	Basée sur l'opération	[BIE 95] [PAR03] [SAK 99]	Toutes les permutations des opérations donnent toujours un ordonnancement faisable.	Exiger un décodeur aux points codés en une liste ordonnée d'opération	$(n * m)!$	Relation simple
	Basée sur le Job	[JAI98] [KUM 96]	Toute permutation des jobs correspond à un ordonnancement faisable.	On ne peut pas couvrir tout l'espace de solutions réalisables	$n!$	Relation simple
	Basée sur la relation de pair de job	[NAK 91] [XIA 06]	pour les algorithmes qui utilisent les concepts binaires dans leur procédure.	très complexe	$(2^{n*m})^m$	Relation simple
	Basée sur le temps d'achèvement	[LIN 01] [SAK 99] [SAK 00]	Représentation simple	Ne convient pas à la plus part de méta heuristique. Risque de donner un ordonnancement faux	$(n * m)^\infty$	Pas de décodeur
	Basée sur les clés aléatoires	[LEI 09] [LEI 10] [SAK 99]	Utile pour les métas heuristiques avec une fonction continue	Les séquences des jobs données une représentation peut voler la contrainte de précédence	$(n * m)!$	Relation simple

Représentation indirecte	Basée sur les listes de préférences	[BOU 91] [DEL 95] [SAK 99]	Couvre l'espace de solution réalisable	Toutes les permutations de cette représentation peuvent ne pas représenter un ordonnancement actif	$(n!)^m$	Heuristique simple
	Basée sur les règles de priorité	[DOR 95] [SAK 99]	Identifie une séquence pour l'affectation des jobs impossibles	Nécessite un programme d'interprétation en un ordonnancement faisable	$(n!)^m$	Heuristique simple
	Basée sur le graphe disjonctif	[TAM92] [PET 04]	La simplicité de la structure permet avoir une représentation visuelle de la relation entre les opérations	Arcs disjonctif arbitraire peut produire un graphique cyclique signifie que l'ordonnancement	$(2^{n*m})^m$	Heuristique complexe
	Basée sur la machine	[FAY 05] [FAY 73]	Utile pour les méta–heuristique présent un « goulot d'étranglement» des procédures dans leur algorithme.	Nécessite de reoptimiser la séquence de Job sur chaque machine.	$m!$	Heuristique complexe

Tableau 3.1. Etat de l'art relatif au niveaux de complexité de problème d'ordonnancement JSC

Une autre classification que celle de Cheng fut présentè en 2010 par Tamer Abdelmagid [ABD 10]. Cette classification est basée sur deux grandes catégories .Une basée sur le modèle et l'autre sur l'algorithme. Dans la seconde, la structure de chromosome est dépond la définition des variables de décision du modèle mathématique spécifique au job shop classique. Un chromosome peut être directement à lie une solution faisable ou non. Dans les représentations à base d'algorithmes, le génotype est utilisé pour stocker les informations de guidage pour être utilisé l'algorithme afin de générer l'ordonnancement réaliste, et il n'y a aucune garantie pour obtenir des solutions optimales.

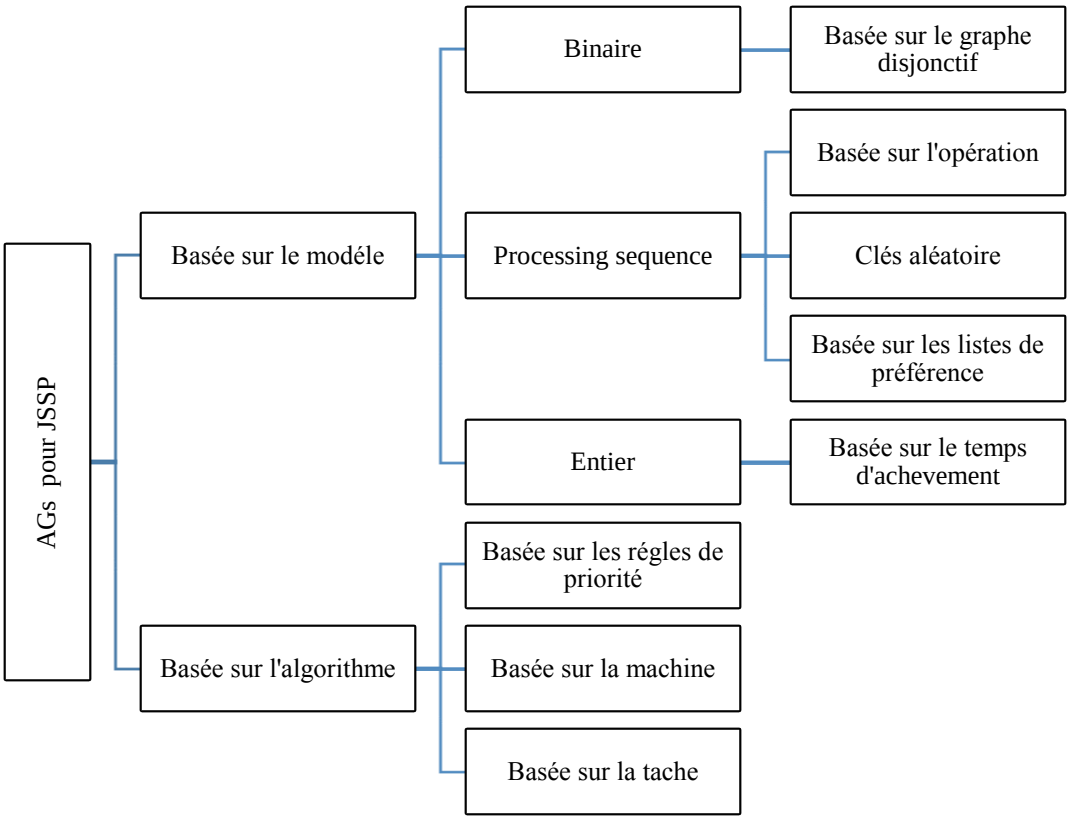


Figure 3.2. Type d'Algorithmes génétique pour le problème d'ordonnancement job shop classique [ABD 10]

Dans la dernière décennie, les recherches sur ce sujet se sont concentrées sur la manière d'adapter les AGs aux problèmes d'ordonnancement Job-Shop. A cet effet deux questions importantes ont été largement étudiées.

La première relative au codage de solution du problème d'ordonnancement job shop classique dans un chromosome. La seconde est comment améliorer les performances des AGs par l'hybridation avec une méthode heuristique.

2.2.2 Présentation de quelques travaux relatifs aux codages des chromosomes

Pour le codage , plusieurs approches ont été proposées :

La première approche fut celle de Davis en 1985 [DAV 85]. Il s'agit d'une approche indirecte qui emploie une technique de codage basée sur des listes d'opérations déterminant l'ordre de préférence pour chaque machine .Cette stratégie fut étendue par Falkenauer et Bouffouix en 1991 en utilisant les listes de préférence de job [JAI 98] [GOL 89].

Une des premières approches directes est celle de Nakano et Yamada qui utilise un codage binaire basée sur les relations de précédence entre les opérations s'exécutant sur la même machine .Le codage dans ce cas est basé sur le temps d'achèvement des opérations [NAK 91].

Tamaki et Nishikawa en 1992 appliquent une représentation indirecte fondée sur le graphe disjonctif. Un chromosome est constitué d'une chaîne binaire correspondant à une liste ordonnée de préférence de nœuds relatifs au graphe disjonctif. [TAM 92]

Un autre codage proposé par Holsapple et al en 1993 repose sur des listes de tâches : l'ordonnancement porte sur toutes les opérations d'un job donné avant de passer au suivant.
Dorndorf et al en 1995 proposent une approche indirecte basée sur les règles de priorité, le chromosome est une suite de gènes désignant chacun une règle de priorité [DOR 95].
Croce et al en 1995 adoptent aussi cette stratégie d'encodage et opérateur de croisement mais avec une méthode de look-ahead afin de générer des d'ordonnancements actifs [CRO 95].
Kobayashi et al. (1995) proposent en 1995 une nouvelle approche ou un chromosome est une suite de symboles de longueur n et chaque symbole identifie l'opération devant être traitées sur une machine [JAI 99] [KEB08].
Bierwirth (1995) crée un algorithme de permutation génétique afin d'améliorer les méthodes existantes .Un chromosome représente une permutation des jobs [BIE 95].
Dans un autre travail Bierwirth et al en 1996 analysent trois opérateurs de croisement, qui préservent respectivement l'ordre : relative, de position et absolue de permutation d'opérations [BIE 96].

Someya en 2001 a proposé un AG avec la méthode d'adaptation de l'espace de recherche GSA (Genetic algorithm with Search area Adaptation) de la fonction d'optimisation en utilisant un opérateur de croissement TMX (Trimodal Distribution Crossover) et un opérateur de mutation BNDX (Bimodal Normal Distribution Crossover) [SOM 01].

Park et al en 2003 ont développé une méthode basée sur un AG simple (SGA) et AG parallèle (PGA). [PAR 03]

Mattfeld et Bierwirth en 2004 ont proposé un algorithme génétique pour le problème job shop multi-objectif avec relise date et due date et le tardiness comme objectifs [MAT 04].

Watanabe et al en 2005 ont proposé des améliorations aux travaux de Someya [WAT 05].

Li et al en 2010 ont développé deux heuristiques basées sur les algorithmes génétiques pour les problèmes de job shop pour minimiser le makespan. Les procédures sont basées sur le procédure de travail (working procedure) et la distribution de machine pour obtenir la solution initial [LI 10].

Ren et al en 2012 ont proposé un nouvel AG pour le problème job shop pour minimiser le makespan .Cet algorithme est constitué par trois opérateurs, conçus selon le modèle graphique de job shop, un opérateur de sélection mixte basée sur la valeur du fitness et la valeur de la concentration. Cet opérateur augmente la diversité de la population, un opérateur de croisement basé sur la machine et un opérateur de mutation basé sur le chemin critique [ZHA12].

2.2.3. Présentation de quelques travaux relatifs à l'hybridation des AGs avec un méta - heuristique

Dans les travaux sur les problèmes d'ordonnancement, la vitesse de convergence des AGs ne constitue pas pour les chercheurs un objectif. Aussi et afin de l'améliorer , plusieurs travaux basés sur l'hybridation des AGs avec des méthodes locale ont vu le jour à partir des années 2000.Nous citons quelques

Les premières travaux dans ce domaine sont ceux de Wang et Zheng [WAN 01] qui ont développé une stratégie d'optimisation hybride pour les problèmes d'ordonnancement job shop ont combinant les AGs avec le recuit simulé afin de minimiser le makespan .

En 2005 Zhang et al [ZHA 05] proposent un nouvel AG (GASA) basé sur deux facteurs le FAS (Full Active Schedule) basé sur le recuit simulé et le POX (Precedence Operation Crossover) basé sur les opérateurs des AGs

Liu et al en 2005 toujours [LIU 05] ont présentent un algorithme hybride Taguch Génetique algorithm (HTGA) pour aussi minimiser le makespan pour le problème d'ordonnancement job shop Xu et Li en 2007 [XU 07] ont proposé un nouvel AG (IGA) pour résoudre le problème d'ordonnancement job shop.

Vilcot et al en 2008 [VIL08] ont proposé une hybridation entre les AGs et plus particulièrement NSGA-II (Non dominat Sorting Genetic Algorithm) et la recherche tabou. La particularité de ce travail est qu'il a été traité pour un problème d'ordonnancement job shop classique multi objectifs multi contraintes.

Surekha et al en 2010 [SUR 10] ont présenté une base de connaissance comme une approche pour le problème d'ordonnancement Job Shop en utilisant les AGs et les colonies de fourmi afin de minimiser le makespan avec contrainte de priorité.

Toujours dans le même axe de recherche Kalantari et al [KAL 12] ont proposé en 2012 une approche d'hybridation à trois étages : un algorithme mimétique combiné aux algorithmes génétiques en plus de recuit simulé comme une recherche local afin de minimiser le makespan

Pour ce qui est de l'hybridation des AGs et de la logique floue pour la résolutions des problèmes d'ordonnancement job shop classique nous citons à titre indicatifs les travaux de : Tsujimura et al (1995) [TSU 95] ,Ghrayeb (2000) [GHR 00a][GHR 00b] , Li et al (2012) [LI 12] , Sakama et Mori (2000) [SAK 00] , Sakawa et Kubota (2001) [SAK 01] Fayad et Petrovic (2005) [FAY 05] , Lei (2010) [LEI 10a][LEI 10b] .

3. Les Problèmes d'ordonnancement Job Shop Flexible

Le job-shop flexible est une extension du problème d'ordonnancement de type job-shop classique pour lequel une opération peut être traitée par l'une des machines d'un étage, ce qui crée une complexité supplémentaire. Les travaux portant sur les problèmes de type job-shop hybrides ne sont pas nombreux. Dans ce qui suit , nous présentons quelques travaux existants dans la littérature.

Comme précédemment, nous commençons par présenter les travaux existants sur la complexité des problèmes de job shop hybride puis les méthodes de résolution.

3.1 Complexité des problèmes d'ordonnancement job shop flexible

Pour le problème de job shop flexible, Jurisch (1995) a proposé un algorithme permettant de résoudre d'une façon polynomiale la relaxation du problème à deux jobs afin de trouver des bornes inférieures. Le problème de job-shop flexible devient NP-difficile à partir trois jobs et deux machines [DEM 12] [GOR 11] [XIA 05].

3.2 Travaux relatifs à l'utilisation des AGs pour les problèmes d'ordonnancement job shop flexible

Brucker et Schlie 1990 [BRU 90]

Le premier article traitant le problème Job shop flexible est celui de [BRU 90]. Les auteurs proposent un algorithme polynomial pour le problème de job shop avec machine à usage multiples (multi-purpose machine) .

Jurisch 1992 [**JUR 92]**

Jurisch a proposé une procédure par séparation et évaluation (PSE) pour la résolution du problème de job shop avec machines multiple.

Brandimarte 1993 [BRA 93]

Brandimarte a proposé une approche hiérarchique pour résoudre le problème de job shop flexible. La première étape consiste à résoudre, le problème d'affectation. La seconde étape consiste à résoudre le problème de job shop restant. L'auteur propose plusieurs méthodes basées toutes la recherche Tabou. L'auteur considéré indépendamment deux critères le makespan et sur la somme pondérée des retards [BRA 93]. La figure 2.3 présente le schéma général de chaque approche.

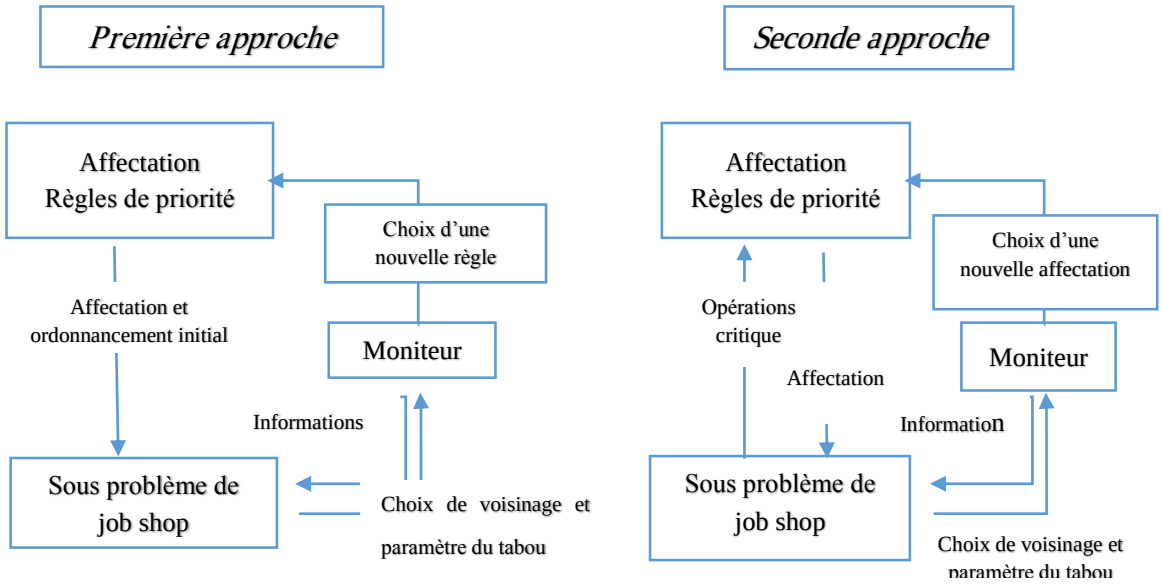


Figure 3.3. Les deux approches proposées par Brandimarte 93 [BRA 93] [VIL 07]

Paulli 1995 [PAU 95]

L'auteur prend en compte une contrainte supplémentaire : le nombre de jobs présents simultanément dans l'atelier est limité à la valeur P. Ainsi si l'atelier est plein, un nouveau travail ne pourra commencer que l'lorsqu'un des travaux en cours dans l'atelier sera terminé. L'objectif est de minimiser le makespan. [PAU 95][VIL 07].

Perregaad 1995 [PER 95]

Perregaad considére dans sa thèse les problèmes de job shop hybride et de flow shop hybride .La différence de ce type de problème avec le job shop classique est que les ressources sont rassemblées en étages de machines parallèles. Une opération pourra donc s'exécuter sur n'importe quelle ressource de l'étage considéré. C'est un cas particulier du job shop flexible.

Brucker, Jurish et Kramer 1997 [BRU 97]

Brucker, Jurish et Kramer ont étudié la complexité de plusieurs problèmes d'ordonnancement d'atelier où il est question d'affectation .La première partie est consacrée aux problèmes à machines parallèles. La seconde partie s'intéresse au job shop , flow shop open shop .

Mesghouni 1998 [MES 99]

Mesghouni a étudié le problème de job shop flexible. il a proposé deux approches évolutionnistes pour résoudre ce problème.

Kacem ,Hammadi et Borne 2002 [KAC 02]

Kacem ,Hammadi et Borne ont proposé un AG multi critère pour le problème de job shop flexible , utilisant la logique floue pour l'évaluation des individus .Les critères qu'il considère sont le makespan , la charge maximale des ressources et la somme des charges des ressources

Dupas 2004 [DUP 04]

Dupas dans son mémoire de HDR, considère les problèmes d'ordonnancement d'atelier de nature cyclique et flexible .

Son étude s'intéresse à la résolution de ces problèmes via des algorithmes évolutionnistes. Pour ce qui est des job shop flexible , l'auteur reprend l'algorithme proposé par Mesghouni avec une approche multicritère basée sur NSGA-II.

Nait Tahar, Yalaoui ,Amodeo et Chu 2004 [NAI 04]

Nait Tahar, Yalaoui, Amodeo et Chu se sont intéressés à un problème d'ordonnancement de fabrication de carton pour un problème de job shop flexible avec des machines parallèles identiques comme une étude de cas. Les auteurs proposent un AG.

Zhang et Gen 2005 [ZHA 05]

Zhang et Gen ont proposé un AG pour le problème job shop flexible multi-objectif (le makespan , la charge maximale des ressources et la somme des charges des ressources).

Ho, Tay et Lai 2007 [HO 07]

Ho, Tay et Lai ont proposé un AG pour le job shop flexible . L'originalité de leur approche vient de l'ajout d'un module d'apprentissage au fil des génération, Ce module apprend à reconnaitre certaines bonnes caractéristiques des chromosomes.

Pezzella ,Morganti , Giaschetti 2008 [PEZ 08]

Pezzella et Morganti ont présenté un algorithme génétique pour le problème d'ordonnancement job shop flexible pour minimiser le makespan , cet algorithme intègre différentes stratégies pour la population initiale composé deux procédures d'affectations (minimum global et les permutations aléatoires de machine et des jobs) et différent règles de priorité (MWR ,MOR) .

Ponnambalan et al 2009 [PAN09]

Ponnambalan et al ont proposé un algorithme génétique basé sur la procédure de Giffler et Thompson, un ordonnancement actif pour le makespan comme un objectif, la représentation du chromosome est composé par deux chaines, la taille de chaque chaine est égale aux nombre total d'opérations dans tous les jobs. La première chaine est relative l'affectation des machines et la deuxième pour les permutations des jobs.

Giovanni et Pezzella 2010 [GIO 10]

Giovanni et Pezzella ont présenté un algorithme génétique améliorée pour résoudre les problèmes d'ordonnancement job shop distribuer et flexible afin de minimiser le makespan.

Zhang , Gao , Shi 2011 [ZHA 11]

Zhang et al ont proposé un algorithme génétique efficace pour résoudre un problème d'ordonnancement job shop flexible. L'auteur crée une population initiale par une sélection locale et une sélection globale pour garantir la diversité de la population initiale. Les critères qu'ils considèrent sont le makespan .

Zhang et al 2012 [ZHA 12a]

Zhang et al ont proposé un algorithme génétique avec procédure de la recherche tabou pour le problème job shop flexible afin de minimiser le makespan avec la contrainte de transport et de temps de traitement (Bounded processing time) [ZHA 12a]

4. Conclusion

Ce chapitre a été consacré à la présentation d'un certains nombres de travaux sur les problèmes d'ordonnancement de type job shop classique et flexible et plus particulièrement ceux relatifs à l'utilisation des AGs comme technique d'optimisation ou l'hybridation des AGs avec une métaheuristique.

Dans le chapitre suivant nous abordons la présentation de notre contribution, à savoir la résolution du notre problème par les algorithmes génétiques que nous avons proposées

CHAPITRE 4

RESOLUTION DU PROBLEME PAR LES ALGORITHMES GENETIQUES

Le présent chapitre présente l'approche de résolution du problème par les AGs. Nous présentons en premier le principe des AGs, nous décrivons ensuite les étapes codage décodage et les opérations pour les AGs proposés pour la résolution du problème d'ordonnancement job shop flexible.La validation de l'AGP se fait pas de comparaison des valeurs du C_{max} obtenu avec notre l'AGP et d'autres AGs trouvés dans la littérature pour les même benchmarks.

Ce que tu veux me dire, est-ce vrai? Est-ce bien? Est-ce utile? Sinon je ne veux pas l'entendre.

Socrate

1. Introduction

Ce chapitre est réservé à la présentation AGP et sa résolution. Sa validation à consister à la comparaison des résultats obtenus par l'AGP avec celles obtenues par d'autres AG de la littérature et cela pour les même instances.

2. Principe des Algorithmes Génétiques

2. 1 Historique

En 1859 les bases de l'évolution étaient posées par C. Darwin avec son idée de la sélection naturelle (« dans tout espèce les meilleurs sont sélectionnés »). En 1901 ceux sont les bases de la génétique qui étaient posées par De-Vries suite à sa théorie du mutationnisme. En 1975 Jhon Holland proposa l'Algorithme Génétique. En 1989 Goldberg exposa les fondements mathématiques des algorithmes génétiques.

2.2 Terminologie

Les algorithmes génétiques sont basés sur cinq niveaux d'organisation comme présenté sur la figure 4.1.

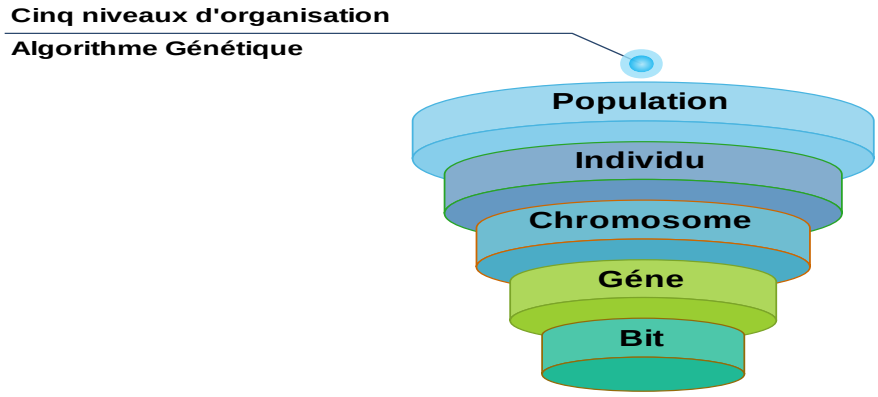


Figure.4.1. Les cinq niveaux d'organisation dans les algorithmes génétiques

Bit : Est la plus petite unité pour présenter une donnée.

Gène : A chaque variable d'optimisation xi, nous faisons correspondre un gène.

Chromosome : Un chromosome est un ensemble de gènes, (figure 4.2). Chaque dispositif est représenté par un chromosome.

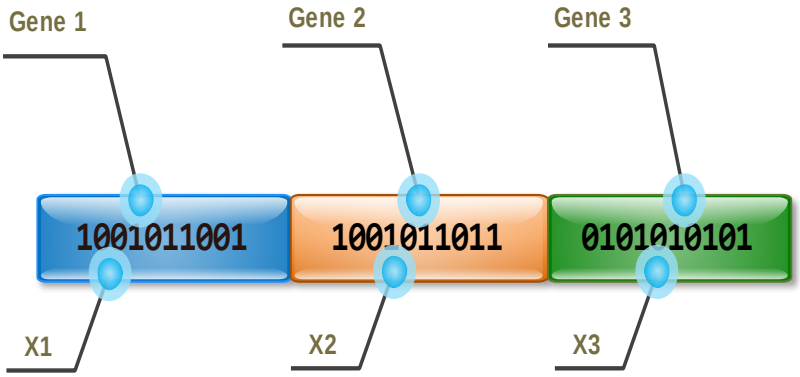


Figure.4.2. Codage des variables [LAG 13]

Individu : Un individu est constitué d'un ou plusieurs chromosomes

Population : Une population est un ensemble de N individus qui vont évoluer.

Génération : C'est la population à l'étape i. A titre d'exemple, la figure 4.3 représente une génération.

Figure 4.3. Présentation de la population à une génération *i* [LAG 13]

2.3. Concepts de base

Un AG recherche le ou les extrema d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité
2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations

futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers la solution optimale ;

3. Une fonction à optimiser. Celle-ci prend ses valeurs dans R+ et est appelée fitness ou fonction d'évaluation de l'individu. elle est utilisée pour sélectionner et reproduire les meilleurs individus de la population ;
4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état ;
5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Sur la figure 4.4, nous avons mis en évidence les principes généraux des AGs . En bleu, nous représentons les étapes liées à la pression sélective. En vert, ce sont les étapes liées à la variabilité et à la transmission des traits de caractère

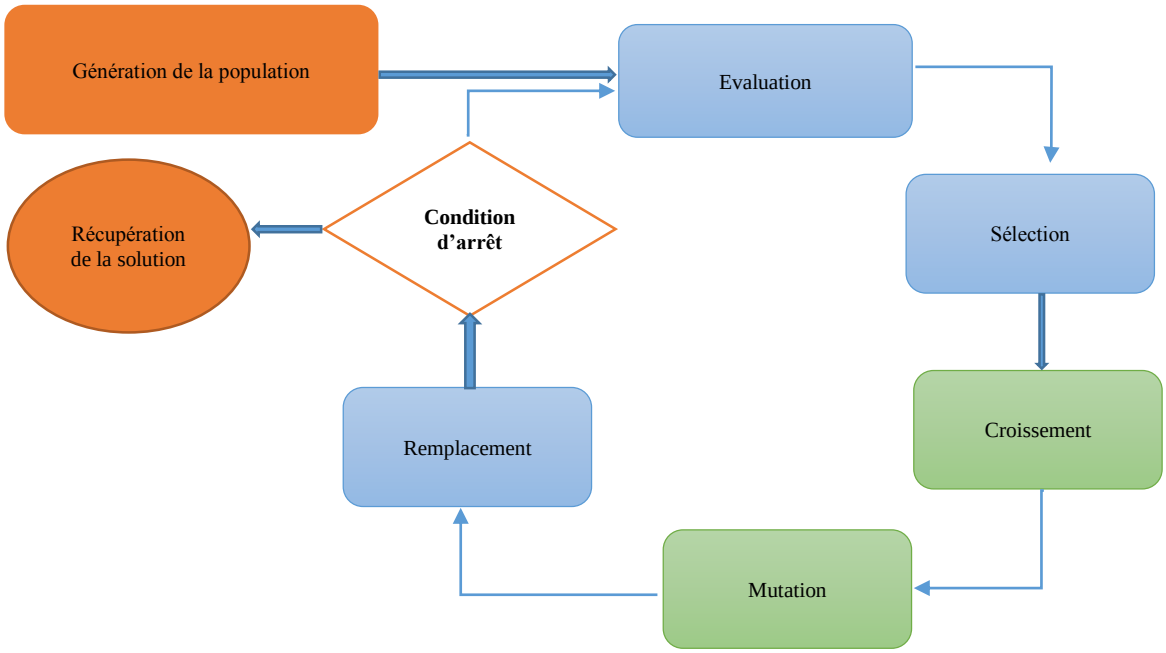


Figure 4.4. Principes des algorithmes génétiques

2.3.1 Le codage

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée).
Dans la littérature, nous pouvons rencontrer trois grandes familles de codage d'une solution

2.3.1.1 Le codage binaire

Le codage binaire est le plus utilisé. Chaque gène dispose du même alphabet binaire {0,1}. Un gène est alors représenté par un entier long (32 bits), les chromosomes qui sont des suites de gènes sont représentés par des tableaux de gènes et les individus de notre espace de recherche sont représentés par des tableaux de chromosomes. Ce cas peut être généralisé à tout alphabet allélique.

1	**0**	**1**	**1**	**1**	**0**

Figure 4.5. Codage binaire d'un chromosome

Ce type de codage s'adapte bien aux problèmes de type binaire, tel que le problème du MAX SAT ou du sac à dos

2.3.1.2 Le codage réel

Ce type de codage est beaucoup plus efficace pour représenter des problèmes de type continu. Il représente les solutions par des suites de type réel, comme le montre la figure 4.3.

11.5	**2.3**	**-0.25**	**56.21**	**32.5**	**0.48**

Figure 4.6 .Codage réel d'un chromosome

On peut distinguer un autre type spécial du codage réel, c'est le codage entier ou discret. Il utilise des entiers au lieu de réels. Ce type est beaucoup plus efficace pour la représentation de certain type de problèmes discrets, comme les problèmes d'ordonnancement et le problème du voyageur de commerce

2.2.1.3 Le codage par valeur

Dans ce type de codage, chaque gène est codé par une valeur qui appartient à un ensemble fini ou infini. Ces valeurs sont liées au problème à résoudre.

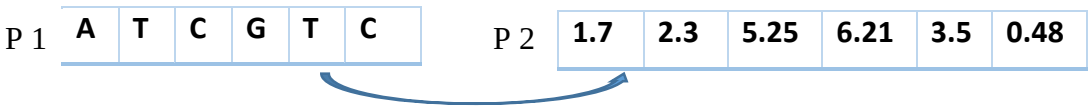


Figure 4.7. Codage par valeur d'un chromosome

2.3.2 Sélection

Cet opérateur est chargé de définir quels seront les individus d'une population P qui vont être dupliqués dans la nouvelle population P' et vont servir de parents ; ce dernier permet aussi aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.

On trouve essentiellement quatre types de méthodes de sélection différentes :

- La méthode de la "loterie biaisée" (roulette wheel) de GoldBerg,
- La méthode "élitiste",
- La sélection par tournois,
- La sélection universelle stochastique.

2.3.2.1La loterie biaisée ou roulette Wheel

Dans cette méthode chaque individu a une chance d'être sélectionné proportionnelle à sa performance, donc plus les individus sont adaptés au problème, plus ils ont de la chance d'être sélectionnés.

Par analogie à la "roue du forain", chaque individu se voit attribué un secteur dont l'angle est proportionnel à son adaptation, son "fitness". On fait tourner la roue, à son arrêt un secteur est désigné par un curseur l'individu choisi et celui correspondant au secteur.(figure 4.8).

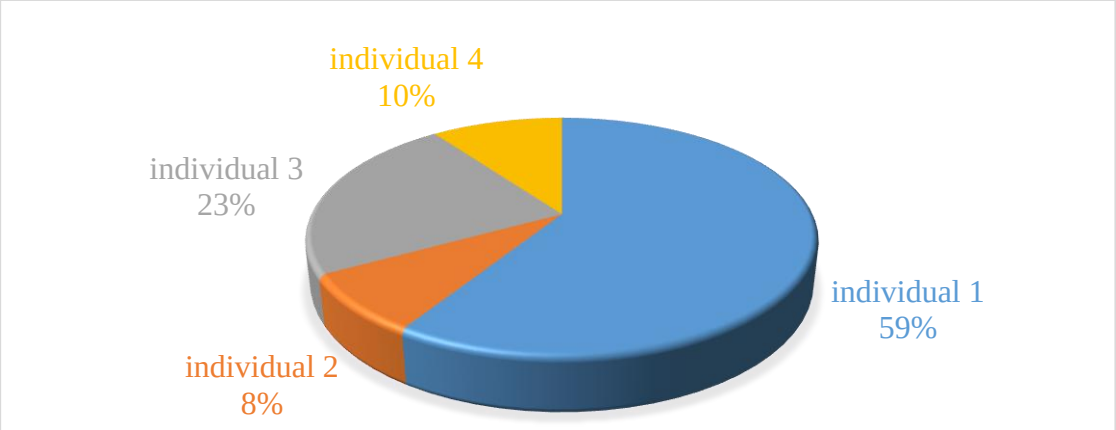


Figure 4.8. La méthode de sélection de la loterie biaisée [LAG 13]

Les Inconvénients de cette méthode :

- En effet, elle a une forte variance. Il n'est pas impossible que sur n sélections successives destinées à désigner les parents de la nouvelle génération P', la quasi-totalité, pire encore la totalité des n individus sélectionnés soient des individus ayant un fitness vraiment mauvais et donc pratiquement aucun individu ayant un fort fitness ne fasse partie des parents de la nouvelle génération
- A l'inverse, on peut arriver à une domination écrasante d'un individu "localement supérieur". Ceci entrainant une grave perte de diversité. Imaginons par exemple qu'on ait un individu ayant une fitness très élevée par rapport au reste de la population, il n'est pas impossible qu'après quelques générations successives on se retrouve avec une population ne contenant que des copies de cet individu. Cependant la valeur de cette fitness est relative, c'est-à-dire élevée par rapport aux autres individus On se retrouve donc face à problème connu sous le nom de "convergence prématurée ; l'évolution se met donc à stagner et on atteindra alors jamais l'optimum, on restera bloqué sur un optimum local.

Il existe certaines techniques pour essayer de limiter ce phénomène, comme par exemple le "scaling", qui consiste à effectuer un changement d'échelle de manière à augmenter ou diminuer de manière forcée le fitness d'un individu par rapport à un autre selon leur écart de fitness.

2.3.2.2 La méthode élitiste

Cette méthode consiste à sélectionner les n individus dont on a besoin pour la nouvelle génération P' en prenant les n meilleurs individus de la population P après les avoir triée de manière décroissante selon la fitness de ses individus.

Il est inutile de préciser que cette méthode est encore pire que celle de la loterie biaisée dans le sens où elle amènera à une convergence prématurée encore plus rapidement et surtout de manière encore plus sûre que la méthode de sélection de la loterie biaisée ; en effet, la pression de la sélection est trop forte, la variance nulle et la diversité inexistante, du moins le peu de diversité qu'il pourrait y avoir ne résultera pas de la sélection mais plutôt du croisement et des mutations.

2.3.2.3 La sélection par tournois

La sélection par tournoi est l'une des sélections les plus utilisées dans les algorithmes génétique Le principe consiste à choisir un sous-ensemble d'individus (S individus) aléatoirement dans la population, puis à sélectionner le meilleur individu dans ce groupe en fonction de sur fitness. Ce processus est répète jusqu'à l'obtention du nombre d'individus requis .Le nombre de participants a un tournoi (S), appelé taille du tournoi, est utilisé pour faire varier la pression de cette sélection. Si ce nombre est grand, alors la pression sera forte et « les faibles individus auront une petite chance d'être choisis.

En général, un seul gagnant est choisi parmi les participants à un tournoi. Ce gagnant peut être choisi d'une façon déterministe ou probabiliste. Dans le cas déterministe, qui est pratiquement le plus utilisé le gagnant est l'individu de meilleure qualité (meilleure fitness). Dans le cas probabiliste, chacun des participants peut être choisi en tant que gagnant avec une probabilité proportionnelle à sa fitness.

2.3.2.4 La sélection universelle stochastique

La sélection stochastique universelle, ou SUS, est une méthode de sélection élimant le biais de la sélection par roulette.

A chaque individu de la population est associé un segment d'une longueur égale à son fitness. L'ensemble des segments est placé de manière contiguë pour former une ligne de longueur égale à la somme des fitness, comme dans la sélection par roulette.

La différence réside, ici, en l'ajout à intervalles réguliers d'un ensemble de pointeurs situés au-dessus de la concaténation des segments. Le nombre de pointeurs doit être égal au nombre d'individus à sélectionner. La distance les séparant équivaut, ainsi, à l'inverse du nombre d'individus à sélectionner. La position du premier pointeur est donnée par un nombre aléatoire compris entre 0 et 1/N.

2.3.3 Le croissement ou crossover

Le croissement utilisé par les algorithmes génétiques est la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents.

Son rôle fondamental est de permettre la recombinaison des informations présentes dans le patrimoine génétique de la population.

En effet, plus le nombre de points de croisements sera grand et plus la probabilité de croisement sera élevée plus il y aura d'échange de segments, donc d'échange de paramètres, d'information, et plus le nombre de points de croisements sera petit et plus la probabilité de croisement sera faible, moins le croisement apportera de diversité.

2.3.3.1 Croisement avec un point de crossover

Le croisement un point consiste à choisir un seul point de coupure, puis échanger les fragments situés après ce point de coupure. Comme le montre la figure 4.9

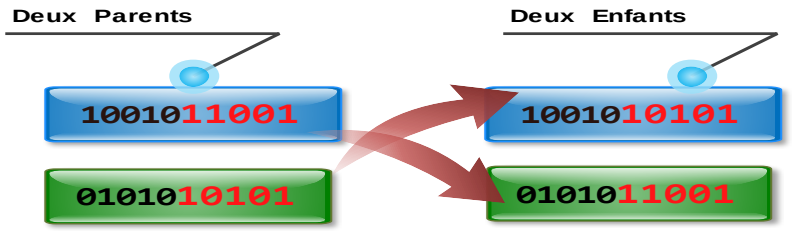


Figure 4.9. Croisement avec un point de crossover [LAG 13]

2.3.3.2 Croisement uniforme

Ce type de croisement est fondé sur la probabilité. En fait, il permet la génération d'un enfant en échangeant chaque gène des deux parents avec une probabilité égale à 0.5, comme le montre la figure 4.10.

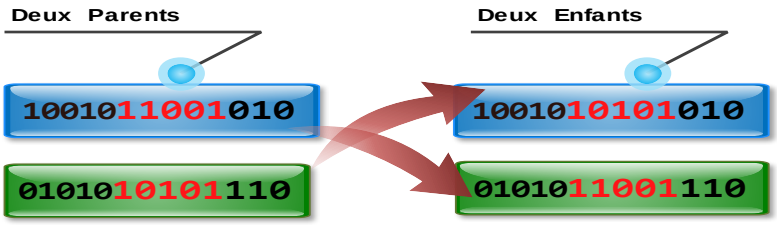


Figure 4.10 : croisement uniforme [LAG 13]

L'opérateur de croisement assure donc le brassage du matériel génétique et l'accumulation des mutations favorables. En termes plus concrets, cet opérateur permet de créer de nouvelles combinaisons des paramètres des composants. Malgré tout, il est possible que l'action conjointe de la sélection et du croisement ne permette pas de converger vers la solution optimale du problème.

2.3.4 La mutation

Cet opérateur consiste à changer la valeur allélique d'un gène avec une probabilité pm très faible, généralement comprise entre 0.01 et 0.001.

Une mutation consiste simplement en l'inversion d'un bit (ou de plusieurs bits, mais vu la probabilité de mutation c'est extrêmement rare) se trouvant en un locus bien particulier et lui aussi déterminé de manière aléatoire; on peut donc résumer la mutation de la façon suivante :

Figure 4.11. Une mutation

On trouve plusieurs stratégies de mutation :

2.3.4.1 La mutation uni- point

Ce type de mutation se fait par altération d'une seule valeur sur chromosome. Comme le montre la figure 4.12.

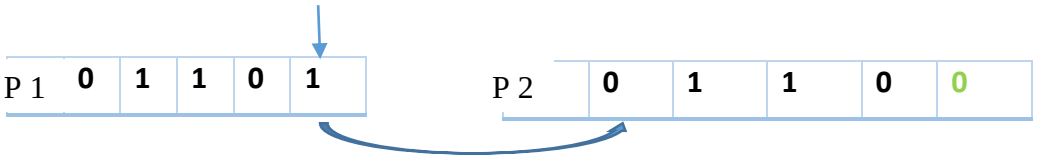


Figure 4.12. Mutation uni – point

2.3.4.2 La mutation uniforme

Ce type de mutation se fait par altération de plusieurs valeurs sur chromosome. Comme le montre la figure 4.13 .

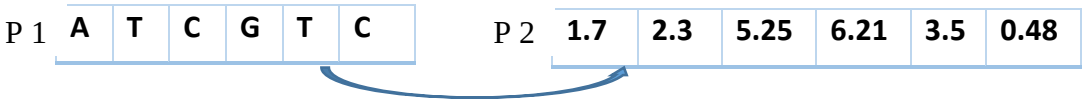


Figure 4.13. Mutation uniforme

2.3.4.3 La mutation par valeur

Se fait d'une valeur donnée en une autre valeur déterminée, sur tous les gènes du chromosome Comme le montre la figure 4.14.

Figure 4.14 .Mutation par valeur

L'opérateur de mutation modifie donc de manière complètement aléatoire les caractéristiques d'une solution, ce qui permet d'introduire et de maintenir la diversité au sein de notre population de solutions. Cet opérateur joue le rôle d'un "élément perturbateur", il introduit du "bruit" au sein de la population.

Cet opérateur dispose de 4 grands avantages :

- Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques.

- Il permet d'éviter un phénomène connu sous le nom de dérive génétique. On parle de dérive génétique quand certains gènes favorisés par le hasard se répandent au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. Le fait que l'opérateur de mutation puisse entraîner de manière aléatoire des changements au niveau de n'importe quel locus permet d'éviter l'installation de cette situation défavorable.
- Il permet de limiter les risques d'une convergence prématurée causée par exemple par une méthode de sélection élitiste imposant à la population une pression sélective trop forte. En effet, dans le cas d'une convergence prématurée on se retrouve avec une population dont tous les individus sont identiques mais ne sont que des optimums locaux.
- La mutation entrainant des inversions de bits de manière aléatoire permet de réintroduire des différences entre les individus et donc d'éviter de cette situation.
- La mutation permet d'atteindre la propriété d'ergodicité. L'ergodicité est une propriété garantissant que chaque point de l'espace de recherche puisse être atteint.

3. Application des AG pour la résolution du problème d'ordonnancement job shop flexible

Pour l'application des AG, pour la résolution du problème d'ordonnancement job shop flexible, et selon le modèle mathématique qui a été présente au chapitre 3, nous avons opté pour trois variables de décision :

- date de début S_i de l'opération i
- date de fin C_i de l'opération i
- capacité machine

La démarche à suivre est dans l'organigramme de la figure 4.15.

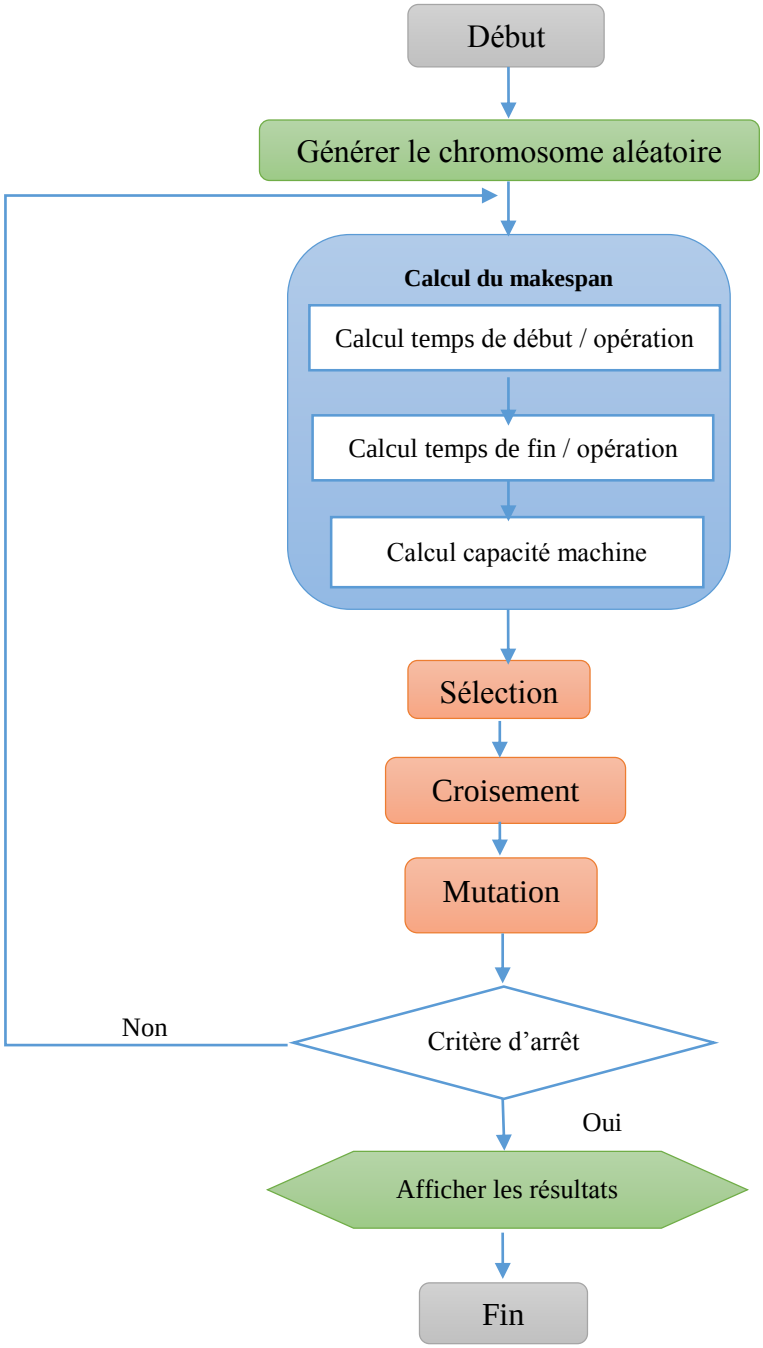


Figure 4.15. *Organigramme de l'algorithme génétique proposé pour un Job Shop Flexible.*

3.1 Codage du chromosome

Dans le cadre de notre application, le chromosome représente les permutations des jobs (PJ). Vous le reprenez en suivante. La taille du chromosome est égal à L tel que $L=\sum_1^n J_{io}$, suite aux hypothèses considérées nous avons deux parties dans notre chromosome.

- La première partie c'est le séquencement des opérations ou l'affectation des opérations d'une façon séquentielle sur le même job. Afin de respecter la contrainte de précédence nous utilisons un codage entier.
- La deuxième partie est relative à l'affectation des machines d'une façon aléatoire. Chaque bit de ce chromosome prend les valeurs possible à choix de Ω_{ij} $tel\ que\ \Omega_{ij} \in \Omega$.

A titre d'exemple : le tableau 4.1, donne un exemple du problème job shop flexible partielle (P-JSF) avec quatre machines et deux jobs, chaque cellule de tableau représente le temps d'exécution d'opération O_{ij} sur la machine M_k.La figure 4.16 suivante désigne le codage de notre chromosome dans cet exemple

Job	*Opérations*		M_1	M_2	M_3	M_4
J_1	O_{11}		5	2	3	1
	O_{12}		8	-	-	8
	O_{13}		-	-	4	3
J_2	O_{21}		4	5	1	-
	O_{22}		3	2	-	1

Tableau 4.1. Temps d'exécution d' job shop flexible partiel(P-FJSP)

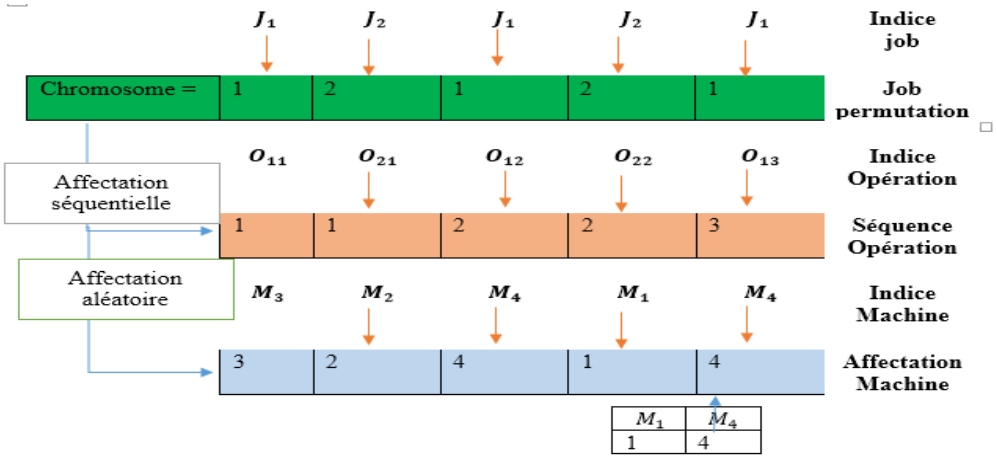


Figure.4.16. Codage du chromosome PJ

Chromosome1 (chrom)	**1**	**2**	**1**	**1**	**2**
Donné d'entréé					
Séquence d'opération (Oseq)	1	1	2	3	2
Affectation des machine aléatoire	M3	M2	M4	M1	M4
Processing time(PTime)	2	5	8	4	1

Ordonnancement 1 = O_{11},O_{21},O_{12},O_{13},O_{23}

Chromosome2 (chrom)	**1**	**2**	**1**	**2**	**1**
Donné d'entréé					
Séquence d'opération (Oseq)	1	1	2	2	3
Affféctation des machine aléatoire	M1	M3	M4	M2	M4
Processing time(PTime)	5	5	1	2	3

Ordonnancement 2 = O_{11},O_{21},O_{12},O_{22},O_{13}

Figure.4.17. Deux permutations différentes de chromosome

Pour calculer le makespan de l'ordonnancement 2 (O_{21},O_{22},O_{11},O_{23},O_{12}) de l'exemple du tableau 4.1 on a besoin des matrices suivantes :

- PTime $= \begin{matrix} & O_1 & O_2 & O_3 \\ J_1 & 5 & 1 & 3 \\ J_2 & 5 & 2 & 0 \end{matrix}$: Matrice du temps de traitement pour chaque opération.
- $STime = \begin{matrix} & O_1 & O_2 & O_3 \\ J_1 & 0 & 5 & 9 \\ J_2 & 0 & 5 & 0 \end{matrix}$ Matrice du temps de début des opérations
- $ETime = \begin{matrix} & O_1 & O_2 & O_3 \\ J_1 & 5 & 6 & 9 \\ J_2 & 5 & 7 & 0 \end{matrix}$ Matrice du temps de fin des opérations
- $EMachine = \begin{matrix} M_1 & M_2 & M_3 & M_4 \\ 5 & 7 & 5 & 9 \end{matrix}$ Matrice de capacité machine

Makespan = max(ETime) = max (EMachine)= 9

La procédure de codage et le décodage du chromosome (PJ) pour notre problème est illustré dans l'algorithme 4.1 et l'algorithme 4.2

Algorithme 4.1 .Codage du chromosome PJ

Entrée :

Nombre total des machines **m ,**

Nombre totale des jobs**nj,**

Nombre total des opérations pour chaque job**no_j,**

Matrice traitement **Time,**

Nombre total des opérations**NumO,**

Sortie :

Vecteur permutation des jobs **Chrom**,

Matrice affectation des machines **Machine**,

Matrice temps de traitement **PTime,**

Début

Générer aléatoirement **Chrom,**

Pour l=1 : NumO **Faire**

% choisir aléatoirement une machine de Time

Machine(l) ,

PTime(l);

Fin Pour

Sortie CHROM, Machine, PTime,

Fin

Algorithme 4.2 .Décodage du chromosome PJ

Entrée : m ,nj, noj, Chrom, Machine, Time,

Sortie : Meilleur **ordonn,Makespan**

Début

Remplir le vecteur Oseq

Remplir le vecteur Machine

Remplir PTime

Calculer Makespan

Sortie : Meilleur **ordonn,** Makespan

Fin

3.2 Opérateurs de l'AGP

3.2.1 Opérateur de sélection

Cet opérateur est chargé de définir quels seront les individus de la population P qui vont survivre dans la nouvelle population P' et vont servir de parents. Nous utilisons dans notre travail la méthode de sélection par la roulette de Wheel modifié. Les différentes étapes de la procédure sont données par dans l'algorithme 4.3.

Algorithme 4.3. Sélection pour AGP

Entrée :

m, nj, noj ,

Nombre total des individus dans une population np,

Makespan pour chaque individu Mspan(np),

La population initiale Population

Sortie : les meilleurs individus ,Newpopulation

Début

trier Mspan par ordre croissant ;

Newpopulation=Population ,

Pour l=1: np **Faire**

Prob(l) = Mspan(l)/(sum(Mspan))

Fin Pour

Probp= cumsun(Prob) % somme cumulative,

%Sélectionnerles meilleurs individus

Perc= np*50% ,

%Garder 50% des meilleurs individus

rdn% probabilité aléatoire

Pour l=1:np **faire**

Pour k = 1 : perc **faire**

Si rdn<Probp(k) **alors**
Newpopulation(l)=Population(k)
Break ,
Fin Si
Fin Pour
Fin Pour
Sortie Newpopulation avec 50% des meilleurs individus.
Fin

3.2.2 Opérateur de croissement

Cet opérateur pour objectif de recombiner les chromosomes des jobs d'une paire d'individu sélectionnés (parents), afin de crée une nouvelle paire d'individus (enfants) qui héritent de certaines caractéristiques de leurs parents. Nous utilisons une hybridation de deux méthodes de croissement : un croissement uniforme et un croissement de type POX, la procédure de croissement est illustré dans l'algorithme 4.4.

Algorithme 4.4. Le croissement pour AGP

Entrée :
np, Mspan(np), Newpopulation
Sortie : Newpopulation,
Debut
Choisir aléatoirement 2 individus parent1, parent2 deNewpopulation
Générer uniformément un vecteur binaireunif(np)
Pour l=1 : 50% de la population **Faire**
Si unif(l)=1 **alors**
Offset1(l)=parent1(2), % Offset1 le premier enfant
Offset2(l)=parent1(1), % Offset2 le deuxième enfant
Fin Si
Replace parent1,2 avec offset1,2
Fin Pour
Sortie : Newpopulation
Fin

3.2.3 Opérateur de Mutation

Cet opérateur consiste à changer la valeur d'un individu par une autre valeur. Dans notre cas puisque les valeurs des jobs sont limitées, nous avons simplement permuté deux jobs. Nous choisissions la mutation par valeur notre chromosome (values mutation Job). La procédure de mutation est définit dans l'algorithme 4.5.

Algorithme 4.5. La mutation pour AGP

Entrée :

np, Mspan(np), Newpopulation,

Sortie : Newpopulation,

Debut

l=1,

Choisir aleatoirement 2 individus deNewpopulation

Permuter ces deux valeurs

Répéter pour 50% de la population

Sortie Newpopulation

Fin

Ainsi la structure de l'algorithme génétique proposé (AGP) pour le problème Job Shop Flexible est illustrée par l'algorithme 4. 6.

Algorithme 4.6. Algorithme Génétique Proposé AGP

Entrée :

m, **nj**, **no$_j$ np,**

nombre total des generations **ng**,

Sortie: ordonnancement ,Makespan

Debut

Generate aléatoire initial géneration,

Calculer Makespan

Garder la meilleure valeur Best,

For g=1 :ng **faire**

Calculer Makespan

Si existe une valeur w < Best **alors**

Best = valeur,

Fin si

sélection

croisement

mutation

Fin Pour

Sortie : ordonnancement ;Makespan

Fin

Afin d'améliorer les résultats on a opté pour deux stratégie basée sur :

- Affecter aux opérations les valeurs minimales de temps d'exécution sur le même étage.
- Faire des réparations sur l'ordre des opérations sur chaque machine pour occuper au minimum les écarts entres les opérations.

4. Validation de l'approche proposée

Pour valider l'AG que nous avons proposé, pour un job-shop Flexible, nous l'appliquons à des benchmarks connus dans la littérature (Brandimarte (1993), Hurink et al. (1994), Dauzère-Pérès et Paulli (1994), Chambers et Barnes (1996), Kacem et al. (2002), et Fattahi et al. (2007)), puis nous comparons les valeurs du temps de calcul et du makespan obtenu avec notre algorithme avec ceux obtenus avec les même benchmarks mais avec d'autre AG tel que.

- M & G : l'approche proposée par Mastrolilli et Gambardella (2000).
- GENACE : l'approche proposée par Ho et Tay (2004)
- Zhang et al : l'approche proposée par Guohui Zhang et Ling Gao (2011).
- Chen et al : l'approche proposée par Chen H and Ihlow J (1999).
- Pessella et al : l'approche proposée par Pezzelle et Morganti (2008).
- HGTS : l'approche proposée par J. J. Palacios, A. González (2014).

L'implémentation est faite avec le langage MATLAB version 9 (2011), sur une machine avec un processeur 2.3 GH et 4 GO de RAM.

A. Les benchmarks de Brandimarte sont composées de dix problèmes nommé de Mk01, Mk02….Mk10 et caractérisé par :

- Le nombre de job : entre 10 et 20 jobs
- Le nombre de machine par étage entre 4 et 15 machines
- Le nombre d'opérations par job est choisi aléatoirement dans l'intervalle [m/2, m]
- Les gammes opératoires ont été générées aléatoirement.
- Les durées d'exécution des opérations ont été générées uniformément

B. Les benchmarks de Chambers et Barnes : Chambers et Barnes (1996) sont composées de 21 problèmes pour le cas du job shop flexible [CHA 96]. Elles sont divisées en trois ensembles :

- Les problèmes de « MT10 »sont basés sur les instances de Fisher et Thompson [FIS 63].
- Les problèmes «setb4 »et "cas de « seti5 »sont basées sur les instances Lawrence [LAW84]

4.1 Comparaison de l'AGP avec différents AG pour les instances de Brandimarte

Le tableau 4.2 présente les valeurs du makespan obtenu pour les instances de Brandimarte appliquée aux différents AG choisi et bien sur l'AG que nous avons proposé.

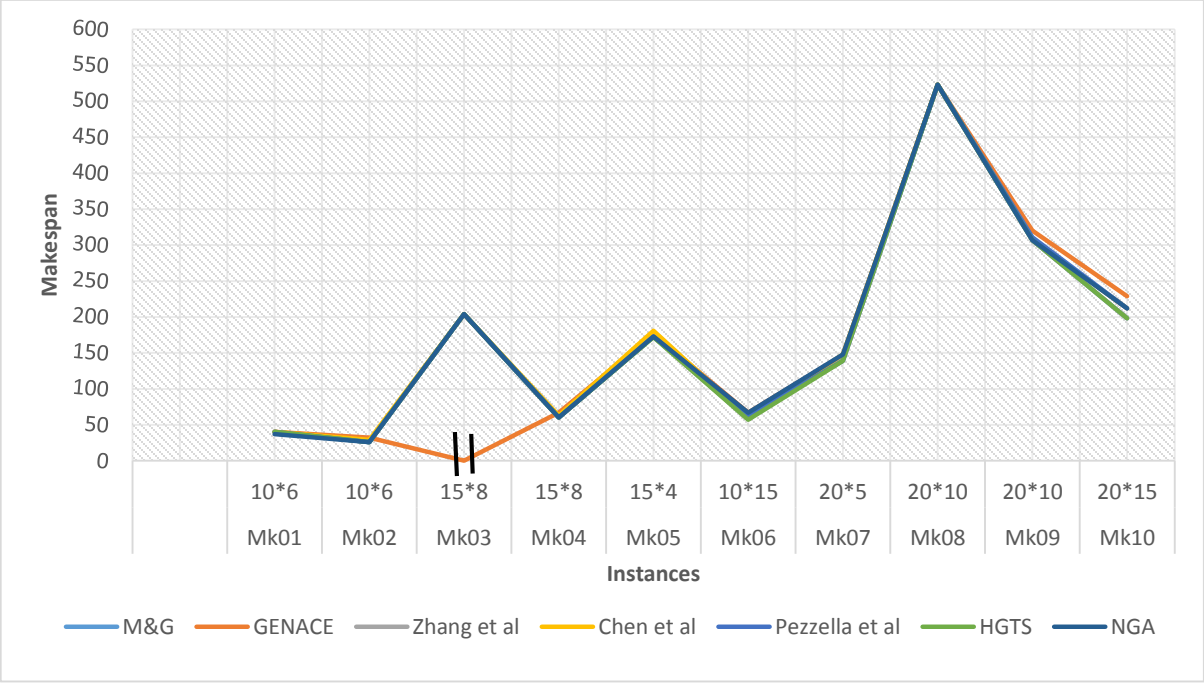
Problème	n*m	M&G	GENACE	Zhang et al	Chen et al	Pezzella et al	HGTS	AGP
		C_m	C_m	C_m	C_m	C_m	C_m	C_m
Mk01	10*6	40	40	40	40	40	40	37
Mk02	10*6	26	32	26	29	26	26	26
Mk03	15*8	204	-	204	204	204	204	204
Mk04	15*8	60	67	60	63	60	60	60
Mk05	15*4	173	176	173	181	173	172	173
Mk06	10*15	58	67	58	60	63	57	67
Mk07	20*5	144	147	144	148	139	139	148
Mk08	20*10	523	523	523	523	523	523	523
Mk09	20*10	307	320	307	308	311	307	307
Mk10	20*15	198	229	198	212	212	198	212

Tableau 4.2. Makespan obtenu par application de différent AG pour les instances de Brandimarte

A partir du tableau 4.2 donnant les résultats du calcul du makespan pour les instances de Brandimarte, avec différents AG (dont le AGP) nous remarquons

- AGP améliore le C_m pour le problème MK01 par rapport aux autres AG
- AGP donne la même valeur du C_m que les autre AG pour les problèmes MK03 et MK08
- Pour les quatre problèmes (MK02 MK04 MK05 MK09) AGP donne les même bons résultats que M&G.
- Nous retrouvons la même valeur du C_m pour MK06 que GENACE et enfin pour MK07 et MK10 les mêmes valeurs que Chen et al

Pour apprécier encore mieux les validations de notre AGP, nous présentons sur la figure 4.18 la comparaison des valeurs du C_m obtenu par l'utilisation de notre AGP, pour les benchmarks de Brandimarte.



N/A : aucune information donnée

Figure 4.18. makespan obtenu par application de différents AG pour les instances de Brandimarte

Sur la figure 4.19a, nous présentons la régression de makespan pour le problème Mk01 calculé avec algorithme génétique proposé. Cette courbe montre l'existence de temps de recherche qui se caractérise par une décroissance du makespan initial (78) en fonction des

générations, jusqu'à arriver à 37 à la 100 éme génération. Cela signifie que la recherche est efficace et l'amélioration de la solution se fait très lentement jusqu'à stabilité de la recherche pour C_m= 37. À partir du programme que nous avons développé il est possible dessiner le diagramme de Gantt pour voir clairement le traitement des jobs. La figure 4.19b : montre la succession des opérations des différents jobs sur toutes les machines.

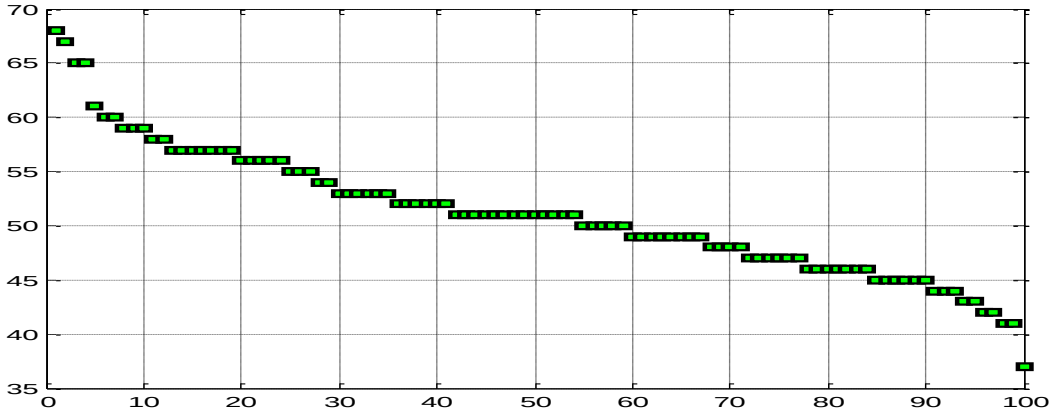


Figure 4.19.a .Régression de makespan pour MK01

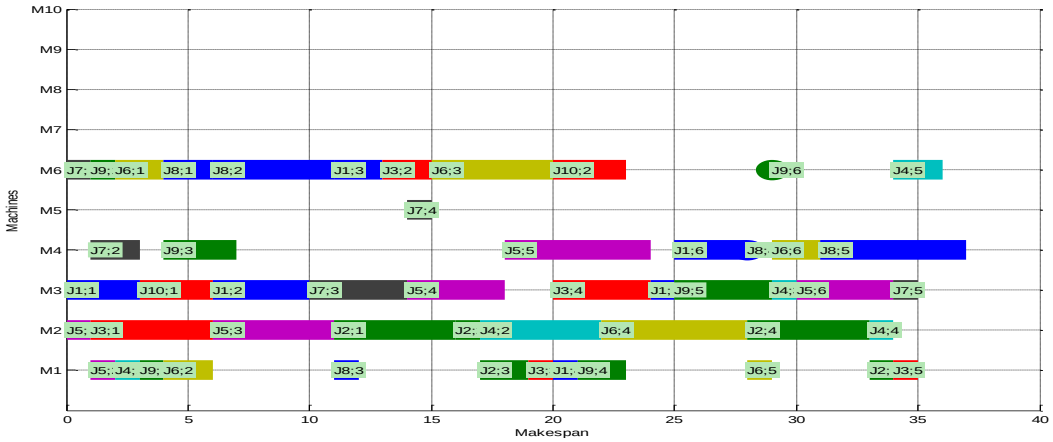


Figure 4.19 .b Diagramme de Gantt pour MK01

Figure 4.19. Régression de makespan et le diagramme de Gantt de (MK01)

4.2 Comparaison de l'AGP avec différents AGs pour les instances de Chambers et Barnes

D'une manière analogue que pour les instances de Brandimarte, nous validons notre AGP par comparaison de la valeur du C_m et du temps de calcul, calculé par l'AGP et autres AG pour les instances de Chambers et Barnes. Le tableau 4.3 donne les résultats obtenus pour les 21 problèmes.

Problème	n*m	(LB, UB)	Zhang et al		AGP	
			C_m	t	C_m	t
mt10c1	10*11	655,927	927	23.25	927	13.15
mt10cc	10*12	655,914	910	19.27	908	13.27
mt10x	10*11	655,929	918	21.45	918	15.45
mt10xx	10*12	655,929	918	20.38	918	15.68
mt10xxx	10*13	655,936	918	25.39	918	18.31
mt10xy	10*12	655,913	905	24.37	906	18.04.
mt10xyz	10*13	655,849	847	30.24	849	19.23
setb4c9	15*11	857,924	914	12.81	914	9.08
setb4cc	15*12	846,909	909	20.16	909	15.13
setb4x	15*11	847,937	925	8.92	925	6.92
setb4xx	15*12	846,930	925	54.06	925	30.16
setb4xxx	15*13	845,925	925	62.81	925	42.31
setb4xy	15*12	845,924	916	27.78	916	19.38
setb4xyz	15*13	838,914	905	40.26	905	32.33
seti5c12	15*16	1027,1185	1174	70.69	1175	55.36
seti5cc	15*17	955,1136	1136	69.53	1136	55.53
seti5x	15*16	955,1218	1209	67.56	1199	56.63
seti5xx	15*17	955,1204	1204	78.29	1204	63.29
seti5xxx	15*18	955,1213	1204	105.25	1202	90.23
seti5xy	15*17	955,1148	1136	70.47	1136	45.12
seti5xyz	15*18	955,1127	1125	70.56	1128	69.23

Tableau 4.3. valeurs du C_{max} et du temps de calcul en utilisant l'AGP et d'autreS AG pour les instances de Chambers et Barnes

Nous remarquons que les valeurs obtenues du C_m sont en moyenne égale cependant il existe nette amélioration du temps de calcul total avec l'utilisation de notre AGP

Pour mieux illustrer la diminution du temps de calcul nous présentons sur la figure 4.20 le temps de calcul pour les deux AG (AGP et Zhang et al) pour toutes les instances de Chambers et Barnes.

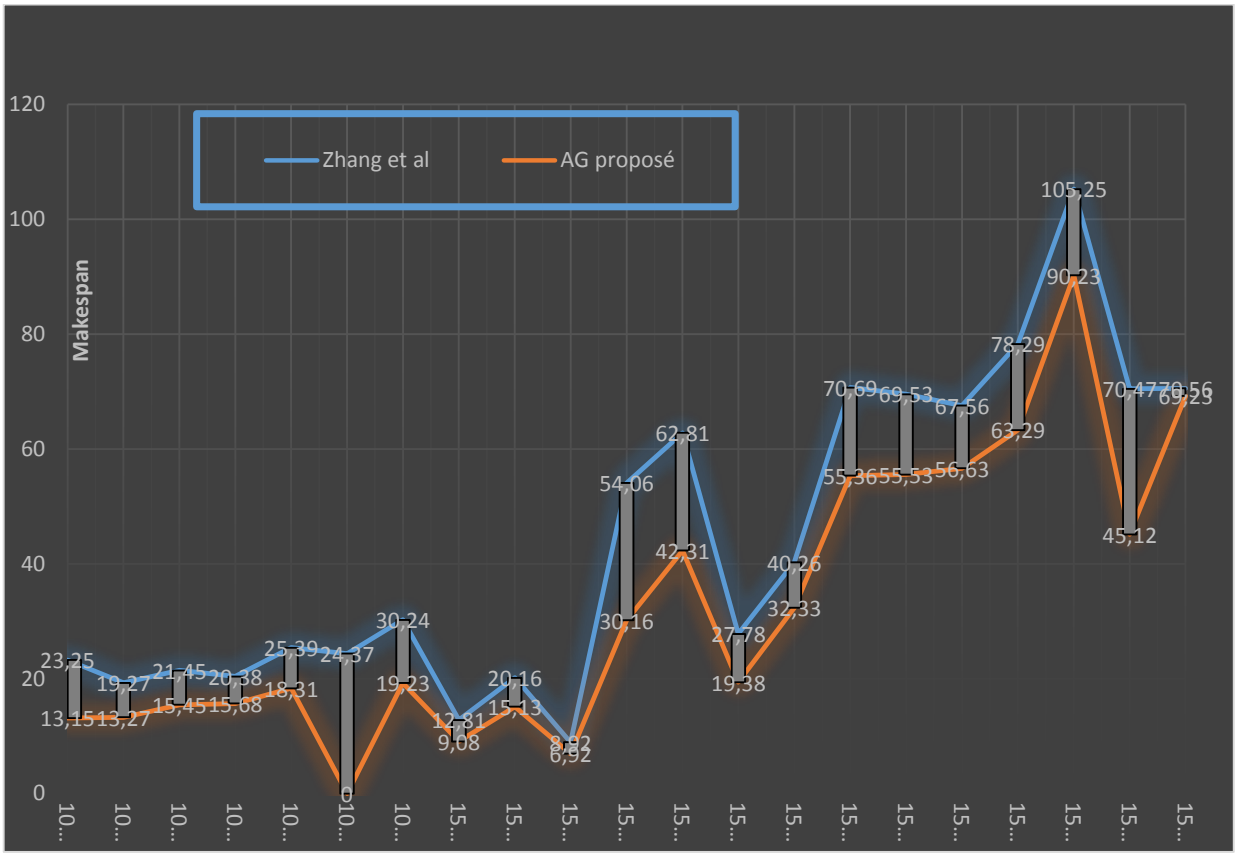


Figure 4.20 le temps de calcul de deux approches en fonction des instances Chambers et Barnes

5. Conclusion

L'optimisation du problème d'ordonnancement job shop flexible avec les AG fait l'objet de ce chapitre. L'objet de notre travail été de minimiser le makespan et le temps de calcul , pour le AG que nous avons proposé le chromosome représenter les permutations des jobs avec un codage par valeur. La validation de cet algorithme est obtenue par sa comparaison avec d'autres AG pour le calcul du C_m et le temps de calcul pour les benchmarks tirés de la littérature les résultats obtenu sont très satisfaisant.

Le chapitre suivant présente notre seconde contribution à savoir l'utilisation d'un nouvelle méta- heuristique appelée l'algorithme de la recherche coucou pour la résolution d'un problème de job shop.

CHAPITRE 5

RESOLUTION DU PROBLEME PAR LA RECHERCHE COUCOU

Ce chapitre présente l'utilisation de la recherche coucou pour la résolution d'un problème de job shop . Après une présentation de cette nouvelle méta-heuristique, nous présentons l'algorithme que nous avons développé .la validation de notre approche se fait par comparaison de valeurs obtenu avec notre algorithme avec ceux de six approches tirées dans la littérature.

"Tout ce que je sais, c'est que je ne sais rien, tandis que les autres croient savoir ce qu'ils ne savent pas."

Socrate

1. Introduction

Dans ce chapitre nous présentons une métaheuristique « les algorithmes de recherche coucou (Cuckoo Search algorithm) pour résoudre le problème d'ordonnancement d'un système de production de type Job-Shop et job shop flexible afin de minimiser le makespan.

Ce chapitre contient deux parties. Dans la première partie, nous présentons les algorithmes de recherche coucou et ses principes. Dans la deuxième partie nous appliquons cette nouvelle métaheuristique pour la résolution des problèmes d'ordonnancement job shop classique et job shop flexible, nous présentons les résultats expérimentaux et nous les interprétons.

2. Présentation de l'algorithme de la recherche coucou

La recherche coucou est inspirée du comportement de reproduction et d'une espèce spéciale d'oiseau appelé coucou. Les coucous ont un caractère parasitaire dans la ponte, l'incubation et la nourriture de leurs oisillons. En effet certains coucous ne construisent jamais de nids pour pondre leurs œufs, ils parasitent les nids d'autres oiseaux d'espèces différentes.

Après la ponte, les femelles coucous se sauvent en s'affranchissant de leurs responsabilités parentales envers leurs œufs. Ainsi leurs œufs sont confiés aux oiseaux hôtes qui seront leurs parents adoptifs. Les œufs coucous risquent d'être tués par leurs parents adoptifs si leur degré de similitude avec les œufs de l'oiseau hôte est faible.

Figure 6.1. *Oiseau de coucou*

La première méta-heuristique inspirée du comportement des coucous, dans leurs reproductions, a été proposée en 2009 par Xin-she yang et Suash deb [RAJ11].

Xin-she yang et Suash deb se sont basée sur trois principe pour proposer leur nouvelle méta heuristique [MAR14] :

1. Chaque coucou pond un œuf à la fois, il le dépose dans un nid qu'il choisit aléatoirement
2. Les meilleurs nids (œuf) seront reportés à la prochaine génération.
3. Le nombre d'hôtes disponibles (nids) est fixé. L'oiseau hôte découvre l'œuf avec une probabilité $P_a \in [0\ 1]$.

2.1 Les variantes de la recherche coucou

Depuis la première apparition de la recherche coucou en 2009, de nombreuses variantes de l'algorithme coucou ont été développés par beaucoup de chercheures. Nous présentons quelques-uns sur la figure 5.2 et le tableau 5.1.

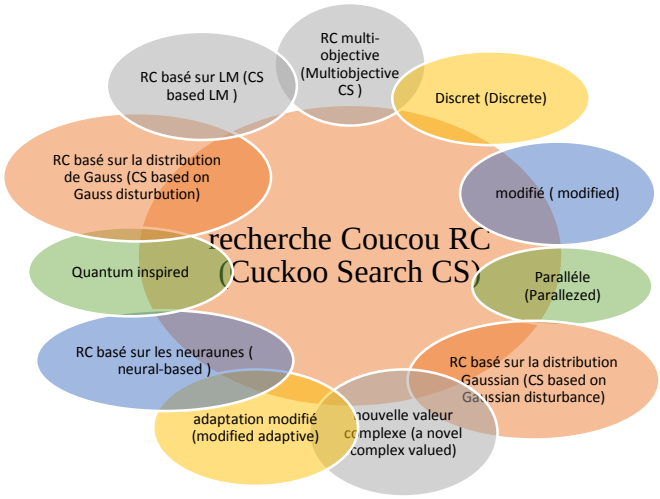


Figure 5.2. les variantes de la recherche couco

Nom de la variant	*Auteur*
RC Discrete (Discrete CS)	Jati and Manurung (2012) [JAT 12]
RC Discrete et binaire (Discrete Binary CS)	Gherboudj et al (2013) [GHE 13]
RC Discrete pour le voyageur de commerce (Discrete CS for TSP)	Ouaarab et al [OUA 13]

RC basé sur les neuraunes(neural-based)	Khan and Sahai (2013) [KHA 13]
Quantum inspired	Layeb (2010) [LAB 10]
modifié (modified)	Tuba et al. (2011)[TUB 11]
Paralléle (Parallezed)	Subotic et al (2012) [SUB 12]
RC basée sur la distribution Gaussian (CS based on Gaussian disturbance)	Wang et al (2014)
RC multi-objective (Multiobjective CS)	Yang and Deb (2013) [YAN 13]
nouvelle valeur complexe (a novel complex valued)	Zhou and Zheng (2013)
RC basé sur la distribution de Gauss (CS based on Gauss disturbution)	Zheng and Zhou (2014)

Tableau 5.1. Les variantes de la recherche Coucou

2.2 Algorithme de la recherche coucou

L'algorithme originale de la recherche Coucou est présenté dans l'algorithme 5.1

Algorithme 5.1 .algorithme de la recherche coucou

Entrée : Population de nid $x_i=(x_{i1}, \ldots\ldots x_{iD})^T$pour i= 1,…….$N_p$
Sortie : Meilleur solution (x_{best})
1:Générer population initiale des nids hotes ()
2 : eval=0 ;
3 :**tant que** (condition arrêt non satisfaite)**faire**
4 :**pour** i-1 **:**N_p**faire**
5 : x_i= nouvelle solution de génération (x_i) & correspondant à la valeur $f_{min} = min(f(x))$
6 : f_i = evaluation de nouvelle solution (x_i)
7 : eval=eval+1 ;
8 :j=[rand (0,1)*$N_p + 1$] ;
9 :**Si**$f_j < f_i$**Alors**
10 : $x_i = x_j$; $f_i = f_j$;
11:**FinSi**
12 :**Si** (rand (0,1) $<$ p_a**Alors**
13:Init _ nid (x_{Worst}) ;
14 :**FinSin**
15 :**Si** ($f_i < f_{min}$) **Alors**
16 : $x_{best} = x_i$; $f_{min} = f_i$;
17 :**FinSin**
18 :**Finpour**
19 :**Fin Tant que**
20 : Fin

3. La résolution de problème d'ordonnancement de job shop par la recherche coucou

3.1 La recherche coucou appliqué à un problème d'ordonnancement job shop classique

Comme la recherche coucou est un algorithme évolutionnaire nous devons tout d'abords définir la population. Cette population est formé d'un ensemble de nids, ce dernier est un vecteur ligne qui représente les permutations des jobs : nous utilisons le codage par valeur.

Pour illustré notre exemple, nous considérons un système de job shop classique de 3 job et a machines tableau 5.2.

Numéro de job	Type de machine	Temps d'exécution
1	1	8
	2	7
	3	14
2	4	6
	1	13
	2	10
3	3	18
	2	16

Tableau.5.2 Problème job shop 3*4.

Codage du nid :

Le nid représente la permutation des des jobs (figure 5.3)

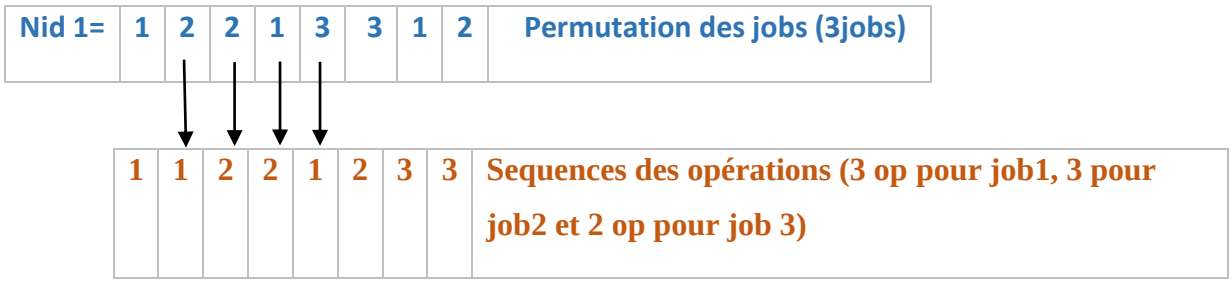


Figure 5.3. Codage du nid.

Population initiale :

1	**2**	**2**	**1**	**3**	**3**	**1**	**2**
2	2	1	3	1	3	2	1
1	3	1	2	2	2	1	3

Figure 5.4. Population initiale.

Décodage du nid :

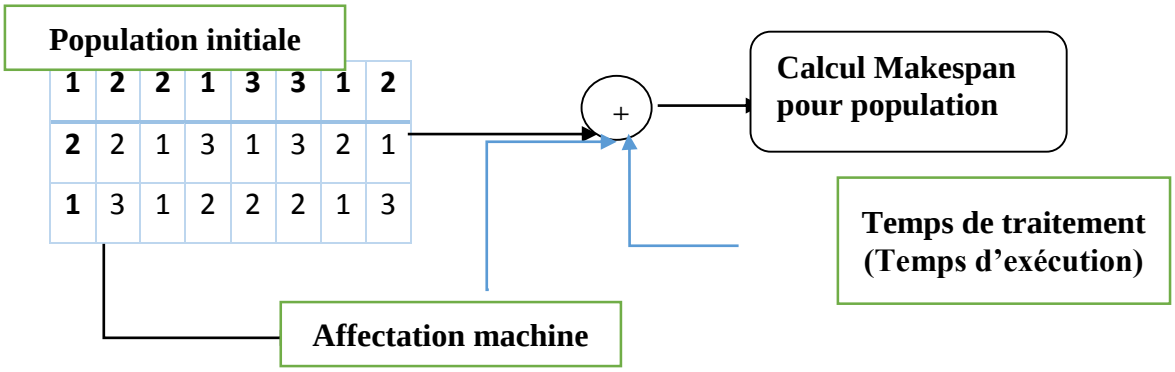


Figure 5.5 Décodage du nid

L'algorithme 5.2 présente l'algorithme développé dans le cadre de ce travail. La recherche coucou est basée sur la population initiale constituée des permutations des nombre des jobs. Cette population est générée aléatoirement puis on calcule le makespan pour chaque individu, on garde la meilleure valeur comme valeur initiale, et les meilleurs individus. Pour le reste des individus nous devons les reproduire par régénération des coucous. Si les coucou présente de bonne valeurs, on remplace le nid par le coucou ; si non pas de changement. Ainsi de suite jusqu'à la dernière génération.

Algorithme 5.2. Algorithme de recherche coucou pour un job shop classique

Début
Créer un nid %les Permutations des jobs est le codage d'un nid n_id'une population
Générer aléatoirement la population initiale
Remplir le tableau opération
Affecter les machines
Affecter le temps de traitement
Calculer makespan %La valeur de la fonction objectif
Garder la meilleure valeur
P_a : Probabilité de découverte des œufs dans les nids

Tant que critère d'arrêt (nombre de générations)**Faire**
Choisir un nid aléatoire avec fitness F_n
Pour l=1: nombre d'individus découvert **Faire**
Générer un coucou aléatoirement avec fitness F_C % coucou = un individu
Si$F_c < F_n$**Alors**
Remplacer le nid par coucou %Garder le meilleur individu
Fin Si
Trier les solutions et les nids
Fin Pour %Garder un pourcentage Pa de la population
Fin Tant que %Aller à la prochaine génération
Fin

3.2. Validation de l'approche pour un job shop classique.

Pour valider l'algorithme de la recherche coucou que nous avons développé, nous avons suivi la mémé approche que pour l'AG.

Ainsi nous appliquons notre algorithme de RC pour le calcul du makespan pour deux classes de benchemarks : les instances de Fischer and Thompson (1963) (FT06, FT10, FT20) et ceux de Lawrence (1984) (LA01 jusqu'à LA40). Les valeurs du makespan obtenu pour les mêmes 43 instances mais avec d'autres algorithmes évolutionnaires qui sont ceux développé par (Storer et al (1992), Croce et al (1995), Conçalves et Beirao (1999), Binato et al (2002), Wang et zheng (2001) [MPY 14] [JM 05]

Problème n*m			**Approches**
FT06 6*6	FT10 10*10	FT20 20*20	
55	965	1215	Nakano et Yamada (1991)
-	954	1180	Storer et wu (1993)
55	949	1189	Frang et al (1993)
-	960	1249	Dorndorf et pesch (1995)
55	977	1215	Fang et Xi (1997)
55	930	1165	Wang et Zheng (2001)
55	972	1207	Jia et al (2003)

55	930	1165	Concalves et al (2005)
55	997	1196	Asadeh et Kamran (2010)
57	999	1249	RCP :Recherche coucou proposé

Tableau 5.3. Validation de l'algorithme de recherche coucou par les benchmarks Fisher et Thompson.

Problème	n*m	Storer et al (1992)	Croce et al (1995)	Concalves et beirao(1999)	Binato et al (2002)	Wang et zheng (2001)	RCP
LA06	10x5	-	926	926	926	926	926
LA11	20x5	-	1222	1222	1222	1222	1225
LA16	10x10	981	979	977	946	945	981
LA21	15x10	-	1097	1047	1091	1058	1047
LA36	15x15	1305	1305	1305	1334	1292	1305

Tableau 5.4. Validation de l'algorithme de la recherche coucou par les benchmarks Lawrence.

Sur la figure 5.6 nous représentons la régression du makespan de génération en génération commençant par 130 jusqu'à 57. Nous traçons le diagramme de Gantt pour voir et vérifier l'ordonnancement des jobs comme donné dans la figure 5.7

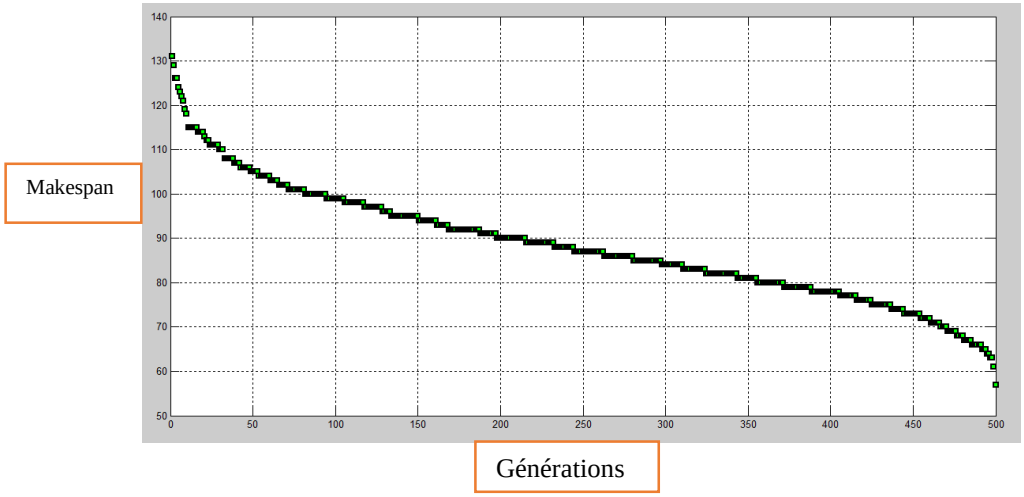


Figure 5.6. Régression du Makespan par l'algorithme de recherche coucou pour l'instance FT06.

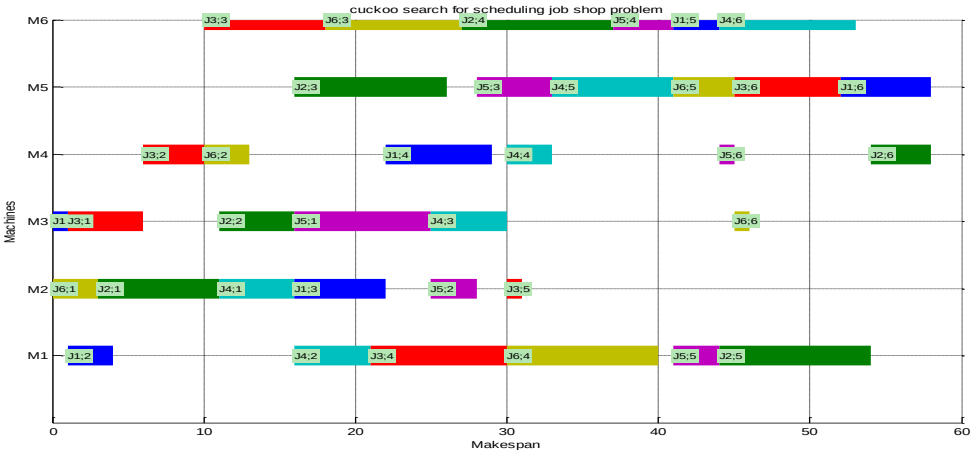


Figure 5.7.Diagramme de Gantt FT06.

4. La recherche coucou appliquée à un problème d'ordonnancement job shop flexible

On utilise simultanément le même principe de codage qui a trouvé dans [DRI 14] pour notre Recherche Coucou Proposé (RCP). Le tableau 5.5, donne un exemple du problème job shop flexible partielle (P-FJSP) avec cinq machines et deux jobs, chaque cellule de tableau représente le temps d'exécution d'opération O_{ij} sur la machine M_k.la figure 5.8 suivante désigne le codage de notre nid dans cet exemple.

Job	Opérations	M_1	M_2	M_3	M_4	M_5
J_1	O_{11}	2	6	5	3	4
	O_{21}	-	8	-	4	-
J_2	O_{21}	3	-	6	-	5
	O_{22}	4	6	5	-	-
	O_{23}	-	7	1	5	8

Tableau 5.5. Temps d'exécution du problème job shop flexible partiel (P-FJSP)

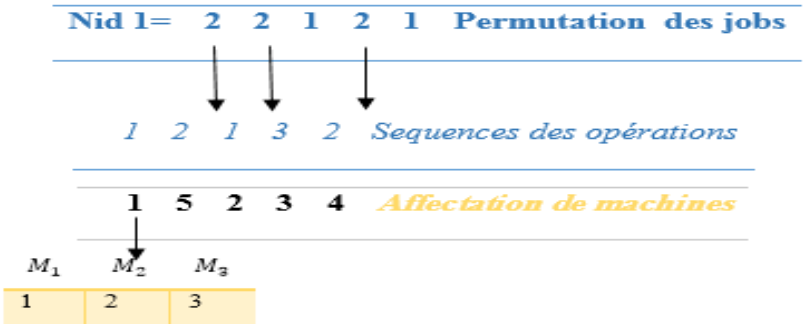


Figure 5.8.codage du nid

L'approche de la recherche coucou proposé (RCP) dans le cas du problème job shop flexible est donné par l'algorithme 5.3 suivant.

Algorithme 5.3 Algorithme de Recherche Coucou Proposé (RCP)

Début

Créer un nid{les Permutations des jobs est le codage d'un nid $\boldsymbol{n_i}$d'une population}

Permutation des jobs

Séquence des opérations

Affectation des machines % pas de traitement . lecture directe du fichier sources

Générer aléatoirement la population initiale

Calculer makespan

Garder la meilleure valeur

$\boldsymbol{P_a}$: Probabilité de découverte des œufs dans les nids

Tant que critère d'arrêt (nombre de générations)

Choisir un nid aléatoire avec fitness $\boldsymbol{F_n}$

Générer un coucou aléatoirement avec fitness $\boldsymbol{F_C}$

Si $\boldsymbol{F_c} < \boldsymbol{F_n}$alors

Remplacer le nid par coucou {Garder le meilleur individu }

Fin si

Trier les solutions et les nids

Garder un pourcentage Pa de la population

Aller à la prochaine génération

Fin Tant que

Fin

4.1 .Validation de l'approche pour un job shop flexible

Pour le cas du job shop flexible partiel ou total validons l'algorithme proposé comme pour le job shop classique, autrement dit nous comparons les valeurs obtenu du makespan avec notre algorithmes pour les instances tiré de la littérature avec ceux obtenu (bien sûr pour les même instances) avec d'autres algorithmes évolutionnaires.

Les instances utilisées dans ce cas sont ceux de Kacem et al [KAC 02]

❖ Pour un job shop flexible partiel (P-FJSP) sont les suivantes :

Instance 1 (I1) : (taille moyenne 8 jobs / 27 opérations / 8 machines)

❖ Pour un job shop flexible total (T-JSF) sont les suivantes

Instance 2 (I2) : (taille petite 4 jobs / 12 opérations / 5 machines)

Instance 3 (I3) : (taille moyenne 10 jobs / 29 opérations / 7 machines)

Instance 4 (I4) : (taille moyenne 10 jobs / 30 opérations / 10 machines)

Instance 5 (I5) : (taille grande15 jobs / 30 opérations / 10 machines)

L'algorithme de recherche coucou proposé est comparé à des algorithmes suivants [KHB02] [LEI10] [ZEN11]:

- **Hybridation des algorithmes évolutionnaire approche d'optimalité de Pareto** : l'approche proposée par Kacem, Hammadi et Borne (2002)
- **PSO+SA** : l'approche proposée par Lei (2004)
- **MOGA**: algorithme génétique multi objective (multi objectif Genetic Algorithm) l'approche proposée par Zhang (2011).
- **MOPSO+LS** : l'approche proposée par Zheng (1999).
- **EPABC**: l'approche proposée par (2008) .
- **PGDHS** : l'approche proposée par Thammano (2013).

L'implémentation est faite en le langage MATLAB version 9 (2011), sur une machine avec un processeur 2.3 GH et 4 GO de RAM.

Le tableau 5.2 : résume les résultats expérimentaux avec les données de Kacem . Il énumère les noms de problèmes, dimension du problème (nombre job × nombre de machines), la

meilleure solution de temps total d'achèvement makespan (C_m), la solution obtenue par notre algorithme proposé (RCP) et la solution obtenue par chacun des autres algorithmes qui mentionnés ci-dessus.

A partir du tableau 5.6, on constate que notre algorithme de RC donne de meilleurs résultats dans les problèmes I2 et I4, et les même résultats dans les deux problèmes I1et I5 que l'approche PSO+SA

Problème	n*m	AL+CGA	PSO+SA	MOGA	MOPSO+LS	EPABC	PGDHS	RCP
I1	8*8	16	15	14	14	14	14	15
I2	4*5	16	-	11	11	11	11	11
I3	10*7	15	-	11	11	11	11	13
I4	10*10	7	7	7	7	7	7	7
I5	15*10	23	12	11	11	11	11	12

Tableau 5.6. Comparaison entre les makespan dans les différentes approches en fonction des instances de KACEM

La figure 5.9 présentent le régression de makespan et Diagramme de Gantt pour le problème I2 (4*5)

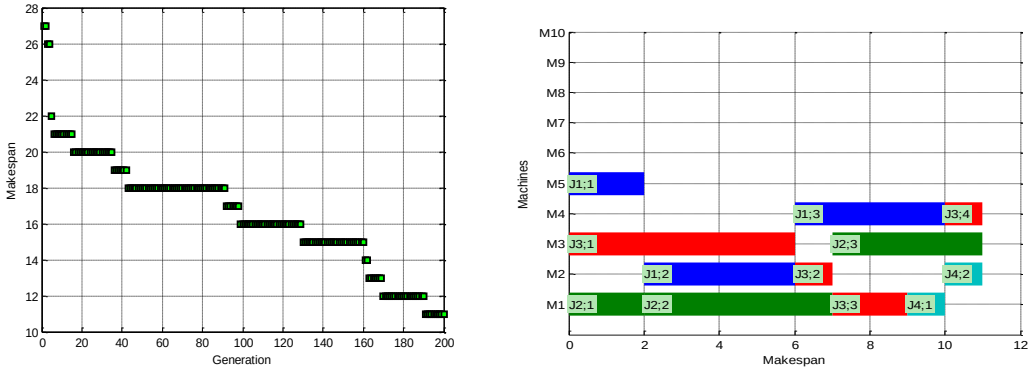


Figure 5.9 Régression de Makespan et le diagramme de Gantt (4*5)

La figure 5.10 présentent le régression de makespan et Diagramme de Gantt pour le problème I1 (8*8).

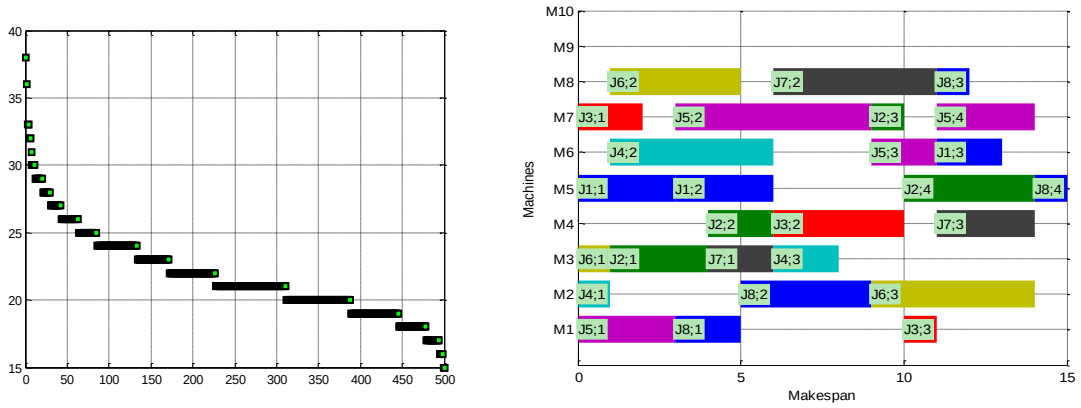


Figure 5.10. Régression de makespan et le Diagramme de Gantt (8*8)

Dans la figure 5.11 présentent la régression de makespan et Diagramme de Gantt pour le problème I4 (10* 10).

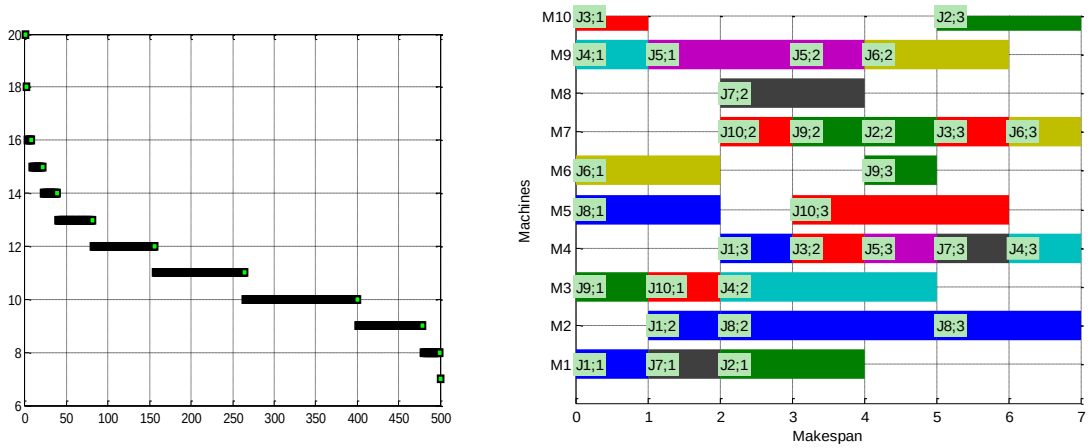


Figure.5.11 Régression de Makespan et le Diagramme de Gantt (10*10).

4.2 Comparaison de l'RCP et de l'AGP pour les instances de Chambers et Barnes

Le tableau suivant illustre les valeurs du makespan et du temps d'exécution pour les deux approches développés et cela les instances de chambre et Barnes.

Problème	n*m	AGP		RCP	
		C_m	t	C_m	t
mt10c1	10*11	927	13.15	928	7.23
mt10cc	10*12	908	13.27	910	7.08
mt10xxx	10*13	918	18.31	919	14.67
mt10xy	10*12	906	18.04.	906	12.23
setb4cc	15*12	909	15.13	909	10.56
setb4xy	15*12	916	19.38	916	17.63
seti5x	15*16	1199	56.63	1209	41.03
seti5xy	15*17	1136	45.12	1136	36.08
seti5xyz	15*18	1128	69.23	1125	59.58

Tableau 5.7. Comparaison entre les makespan et le temps de calcul dans les deux approches proposée en fonction des instances de Chambers et Barnes.

5. Conclusion

Notre seconde contribution dans le cadre de ce travail est l'application de la recherche coucou pour la résolution d'un problème d'ordonnancement job shop classique et flexible

D'un maniéré similaire que pour les AG la validation de l'algorithme développé à consister à comparer les valeurs du makespan obtenu avec cet algorithme pour des instances tiré de la littérature avec ceux (pour les mêmes instances obtenu avec d'autre algorithme évolutionnaire). Les résultats obtenus sont très satisfaisant

Ce chapitre se termine par la comparaison des C_{max} et t pour les deux approches développes dans le cadre de cette thèse.

CONCLUSION GENERALE

ET

PERSPECTIVES

CONCLUSION GENERALE ET PERSPECTIVES

Dans cette thèse, nous nous sommes intéressés aux problèmes d'ordonnancement des ateliers de production. Plus particulièrement les problèmes de job-shop classique et job-shop flexible. Dans le cas non-préemptif, c'est-à-dire qu'une fois commencée, l'exécution d'une opération ne peut en aucun cas être interrompue ni pour des raisons de maintenance, ni pour l'exécution d'une autre opération sur la même machine. Ainsi et pour donner une chronologie permettant au lecteur de suivre le travail, notre travail a été structurée en cinq chapitres.

Nous avons abordé dans le premier chapitre quelques concepts de base d'ordonnancement, des problèmes d'ordonnancement dans les ateliers de production, des critères d'optimisation ainsi que la notion de complexité des problèmes d'optimisation combinatoire .Nous avons terminé ce chapitre par la description de certaines méthodes exactes et approchées de résolution de problème d'ordonnancement.

Dans le deuxième chapitre, nous avons abordé la description du problème d'ordonnancement job shop classique et job shop flexible. Nous décrivons dans un premier temps la formulation mathématique du problème. Nous traitons ensuite la représentation du problème d'ordonnancement job shop par les réseaux de Petri.

Dans le troisième chapitre, nous avons présenté un état de l'art sur des problèmes d'ordonnancement d'une maniéré générale, mais plus spécialement relatif aux ateliers de type job shop classiques et job shop flexible résolu par les algorithmes Génétiques.

Dans le quatrième chapitre on a développé une approche basée sur les algorithmes génétiques pour la résolution du problème job shop afin de minimiser le makespan. Dans notre approche nous proposons une nouvelle représentation du chromosome et différentes stratégie de croissement et mutation. On a vérifié les performances de notre approche par son application à plusieurs benchmarks et la comparaison des valeurs du makespan avec ceux obtenu par comparons notre approche différents algorithmes trouvé dans la littérature.

Dans le cinquième chapitre, nous avons développé une approche basée sur la recherche coucou pour la résolution du problème job shop flexible et job shop classique afin de minimiser le makespan .Dans notre approche nous proposons une nouvelle représentation du nid et différentes stratégies de trie. Nous avons vérifié les performances de notre approche par plusieurs benchmarks et nous comparons notre approche par différents algorithmes évolutionnaire trouvé dans la littérature.

Perspectives

Pour le problème étudié, l'idée que nous prévoyons de développer, dans un premier temps, et de reprendre les deux algorithmes proposés mais avec d'autre contraintes. En ce qui concerne les problèmes de job-shop classique, continuer de développer notre modélisation par les réseaux de Petri est développé des heuristiques basées sur la recherche tabou avec la contrainte de blocage Rcb.

Pour le problème de job-shop flexible, nous proposons un axe de recherche qui consiste à ajouter une nouvelle contrainte pour le modèle sera plus applicable dans le cas réel.

Dans les deux cas, il serait peut être intéressant de développer une nouvelle méthode exacte basée sur une procédure par séparation et évaluation pour essayer de résoudre de façon optimale des problèmes de tailles plus importantes.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[ABD 10] F.T.Abdelmaguid . Representation in Genetic Algorithm For the Job Shop scheduling problem . A Computational Study Journal Software Engineering And Application 3(2010) 1155-1162.

[ABD 14] S.Abdullah . M .A. Nezhad . Fuzzy job shop scheduling problems : A review. Information Sciences 278 (2014) 380-407.

[AKE 56] S. B. Akers . A graphical approach to production scheduling problems. Operations Research. 4 (1956) 244-245.

[ALA 07] K .Alaykyran,O.Engin,A.Doyen .Using ant colony optimization to solve the hybrid flow shop scheduling problems.The International Journal of Advanced Manufacturing Technology 35 (2007)541–550.

[BAS 05] M. Basseur . Conception d'algorithmes coopératifs pour l'optimisation multi-objectif : application aux problèmes d'ordonnancement de type flow-shop. Thèse de Doctorat, Université des sciences et technologies de Lille 2005.

[BEA 93] J. Beasley. OR-Library.http://people.brunel.ac.uk/~mastjjb/jeb/info.Html.2011

[BEL 86] R.E. Bellman.The Bellman continuum . Editions Robert S. Roth, 1986.

[BER 01] G.Berthiau P.Siarry.Etat de l'art des méthodes d'optimisation globale .Operations Research 35 (2001) 329-365.

[BIE 95] P. Bierwirth . Generalized permutation approach to job shop scheduling with genetic algorithmes . Operations Research 17(3)(1995) 87-92.

[BIR 03] S.I. Birbil, S.Fang . An Electromagnetism-like Mechanism for global Optimization . Journal of global Optimization 25 (2003) 263- 282.

[BLU 05] C.Blum . Beam-ACO hybridizing ant colony optimization with beam search: an application to open shop scheduling .Computers and Operations Research 32 (2005) 1565–1591.

[BOU 06] H. Boukef, F. Tangour et M. Benrejeb. Sur la formulation d'un problème d'ordonnancement de type flow-shop d'ateliers de production en industries pharmaceutique . Journées Tunisiennes d'Electrotechnique et d'Automatique, Hammamet, 2006.

[BRA 91] S.A. Brah, J.L. Hunsuchker .Branch and bound algorithm for the flow shop with multiple processors .Eur. J. Oper. Res. 51 (1991) 88–99.

BIBLIOGRAPHIE

[BRA 93] P.Brandimarte. Routing and scheduling in a flexible job shop by tabu search . Annals of Operations Research 41(1993) 157–183.

[BRU 90] P.Brucker, R.Schile . Job-shop scheduling with multi-purpose machines. Computing 45(4) (1990) 369–375.

[BRU 93] P. Brucker, B. Jurisch . A new lower bound for the job-shop scheduling problem. European J. Oper. Res 64 (1993)156-167.

[BRU 94] P.Brucker . A polynomial algorithm for the two-machine job-shop scheduling problem with a fixed number of jobs. Operations Research Spektrum 16 (1994) 5-7.

[BUR 12] S.Burnwal, S.Deb .Scheduling optimization of flexible manufacturing systemusing a cuckoo search based Approach . International Journal of Advanced Man-ufacturing Technology (2012).

[CAR 82] J. Carlier .The one machine sequencing problem . European J. Operations Research 11 (1982) 42-47.

[CAR 84] J. Carlier, P. Chrétienne , C. Girault . Modelling scheduling problems with Petri nets Advanced studies in Petri nets .Lecture notes in Computer Science. Springer Verlag, Paris, 1984.

[CAR 88] J. Carlier , P. Chrétienne . Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes . Edition Masson . Paris 1988.

[CHA 96] J. B. Chambers . J. W. Barnes. Flexible Job Shop Scheduling by Tabu Search. The University of Texas, Austin, TX, Technical Report Series ORP96-09, Graduate Program in Operations Research and Industrial Engineering, 1996.

[CHA 12a] I. A. Chaudhry . A Genetic Algorithm Approach for Process Planning and Scheduling in Job Shop Environment . Proceedings of the World Congress on Engineering 3(2012).

[CHA 12b] K. Chandrasekaran, S.P. Simon . Multi-objective scheduling problem , hybrid approach using fuzzy assisted cuckoo search algorithm .Swarm And Evolution-Ary Computation 5 (2012) 1–16.

[CHE 96] R. Cheng, M. Gen, Y. Tsujimura . A tutorial survey of job-shop scheduling problems using genetic algorithms .Comput. Ind. Eng 30 (1996) 983–997.

BIBLIOGRAPHIE

[CHE 99] H. Chen, J. Ihlow,and C.A. Lehmann . Genetic algorithm for flexible Job shop scheduling .IEEE International Conferenceon Robotics and Automation 2 (1999) 1120- 1128.

[CHE 12] J.C.Chen,C.C.Wu,C.W.Chen .Flexible job shop with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm . Expert Systems with Application .39 (2012)10016-10021.

[CHR 83] P. Chrétienne. Les réseaux de pétri temporises .Thèse de Doctorat, Université de Paris VI, Paris, 1983.

[CHU 96] C. Chu, J. Proth. L'ordonnancement et ses applications. Sciences de l'Ingénieur, Edition Masson, Paris, 1996.

[CLE 02] M. Clerc , J.Kennedy . The Particle Swarm. Explosion, stability, and convergence in a multidimensional complex space . IEEE Transactions on Evolutionary Computation, 6 (2002) 58-73.

[COL 94] A. Colorni, M. Dorigo, V. Maniezzo . Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL) 34 (1) (1994), 39–53.

[CRA 01] D. Craciun .Equivalents dynamiques des réseaux de puissance : Comparaison de méthodes évolutionnaires 2001.

[CRO 95] F.Croce, R. Tadei, and G.A .Volta . Genetic Algorithm for the Job Shop Problem, Computer and Operations Research .22 (1995) 15-24.

[DAV 85] L. Davis . Job shop scheduling with genetic algorithms. Proceedings of the first international conference on genetic algorithms . 5 (1985) 136–140.

[DAV 91] L. Davis .Handbook of genetic algorithm .New York: Van Nostrand Reinhold, 1991.

[DEF 09] F.N.Defersha, M.Chen .A Coase-Grain Parallel Genetic Algorithm for flexible job shop scheduling with lot streaming . In IEEE International Conference on Computational Science and Engineering. (2009).

[DEN 00] M.Den Besten,T.Stutzle,M.Dorigo. Ant colony optimization for the total weighted tardiness problem. Computer Science Springer,Berlin,1917 (2002) 611–620.

BIBLIOGRAPHIE

[DEM 12] Y.Demir, S.Kursat . Evalution of mathematical models for flexible job shop scheduling problems .Applied Mathematical Modelling DOI.10.1016/J.apm 2012

[DHA 05] C. Dhaenens. Optimisation combinatoire multi objectif : apport des méthodes coopératives et contribution à l'extraction de connaissances. Thèse d'habilitation à diriger des recherches .Université des sciences et technologies de Lille 2005.

[DOR 97] M. Dorigo et L.M. Gambardella. Ant colony system: a cooperative learning approach to the travelling salesman problem. IEEE Transactions on Evolutionary Computation . 1(1997). 53-66.

[DRE 03] J. Dréo,A. Pétrowski, P. Siarry, E. Taillard . Métaheuristiques pour l'optimisation difficile . Editions Eyrolles. Paris France .2003.

[DRI 13] I.Driss, K.N.Mouss .A genetic algorithm for job shop scheduling problems . CPI Telemcen –Algeria 21-23 october 2013

[DRI 14] I. Driss, K. N. Mouss, A. Laggoun , Modélisation du problème d'ordonnancement job shop avec contrainte de blocage par les réseaux de Petri . Conférence sur les Signaux et Système . JSS Gulma - Algeria 19-20 Novembre 2014.

[DRI 15] I.Driss, K.N.Mouss,A.Laggoun . A New genetic algorithm for flexible job shop scheduling problem . Journal of Mechanical Science and Technology . 29 (3) (2015) 1273-1281 .

[DUG 07] F. Yalaoui . Hybrid Job Shop and parallel machine scheduling problems: minimization of total tardiness criterion: Multiprocessor Scheduling: Theory and Applications Book edited by Eugene Levner. December 2007 ISBN 978-3-902613-02-8.

[ESQ 99] P.Esquirol , P.Lopez . L'Ordonnancement . Edition Parie Economica ISBN 1999 2-7178-3798-1.

[FAL 91] E.Falkenauer , S.Bouffouix . A Genetic Algorithm for job shop .proceeding of IEEE international conference on robotics and Automation . Sacramento california (1991)824-829.

[FAT 07] P.Fattahi , M.Saidi Mehrabad, F .Joli .Mathematical Modeling and Heuristic Approaches to Flexible Job Shop Scheduling Problems . Journal of Inteligent Manufacturing .18 (2007) 331-342.

BIBLIOGRAPHIE

[FAT 14] P. Fattahi, S. M. H. Hosseini, F. Jolai . A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations . Applied Mathematical Modelling 38 (2014) 119–134.

[FAY 05] C. Fayad, S. Petrovic . A fuzzy genetic algorithm for real-world job shop scheduling . Innov. Appl. Artif. Intell (2005) 76–83.

[FER 05] L. Ferro, S. Khin, N. Salman .Résolution pratique de problèmes NP complets. 2005.

[FIS 14] I.J. Fister, I.Fister, and X.Yang .A short discussion about economic optimization design of shell and tube heart ex changers by a cuckoo search algorithm. Applied Thermal Engineering (2014) 1-3.

[GAG 02] C.Gagne , M.L. Price , M.Gravel . Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence dependent setup times. Journal of the Operational Research Society 53(2002). 895–906.

[JAI 99] A.S. Jain et S. Meeran . Deterministic job-shop scheduling : past, present and future . European Journal of Operational Research. 113(1999) 390-434.

[GAJ 06] Y.Gajpal, C. Rajendran, H. Ziegler .An ant colony algorithm for scheduling in flow shops with sequence-dependent setup times of jobs. International Journal of Advanced Manufacturing Technology 30 (2006) 416–424.

[GAR 76] M.R.Garey,D.S.Johmson ,and R.Sethi .The complexity of flow shop and job shop scheduling . Mathematics of Operational Research 1 (1976) 117-129 .

[GEN 99] M. Gen, R. Cheng . Genetic Algorithms and Engineering Optimization. WileyInterscience .1999.

[GHE 13] A. Gherboudj . Méthodes de résolution de problèmes difficiles académique. Thèse de doctorat .Université de Constantine 2. Algérie .2013.

[GHR 00a] O. Ghrayeb . An efficient genetic algorithm for JSSP with fuzzy durations. 2000

[GHR 00b] O.A. Ghrayeb. Solving Job-Shop Scheduling Problems with Fuzzy Durations Using Genetic Algorithms. New Mexico State University 2000.

[GHR 03] O. Ghrayeb. A bi-criteria optimization: minimizing the integral value and spread of fuzzy makspan of job shop scheduling problems. Appl. Soft Comput. (2003) 197–210.

BIBLIOGRAPHIE

[GIR 08] B.S. Girish , N.Jawahar . Scheduling job shops associated with multiple routings with genetic and ant colony heuristics (2008).

[GLO 89] F. Glover .Tabu search, part I. ORSA. Journal of Computing .1(1989)190-206.

[GLO 90] F. Glover .Tabu search, part II. ORSA. Journal of Computing .2(1990) 4-32 .

[GOL 89] E.D. Goldberg. Genetic Algorithms in Search, Optimisation and Machine Learning . Addison-Wesley Longman Publishing Boston 1989.

[GOL00] D. E. Goldberg .The design of innovation: lessons from genetic algorithms. lessons for the real world. Tech. Forecasting and Social Change 64 (1) (2000)7–12.

[GON 78] T. Gonzalez, S. Sahni .Flow shop and job shop schedules: Complexity and approximation Operations Research. 20 (1978) 36-52.

[GON05] J.F Gonc-alves, J.J.M Mendes, M.G.C Resende . A hybrid Genetic algorithm for the job shop scheduling problem .European Journal of Operational Research 167 (2005)77–95.

[GOR 08] H.G. Goren . A review of applications of genetic algorithms in lot sizing .Springer 2008.

[GOR 11] A.Gorine. Ordonnancement de systèmes flexibles avec contrainte de blocage, Thèse doctorat université de paulverlaine Metz 2011.

[GOT 93] Gotha . Les problpèmes d'ordonnancements. RAIRO-Recherche Opérationnelle, 27 (1993) 77-150.

[GUP 93] M. C. Gupta, A. Gupta, A. Kumar .Minimizing flow time variance in a single machine system using genetic algorithms.European Journal of Operational Research, 70 (1993) 289–303.

[GZA 01] M. Gzara .Méthode coopérative d'aide multicritère à l'ordonnancement flou .Thèse de Doctorat . Université des Sciences et Technologies de Lille 1, 2001.

[HAR 05] E. Hart , P. Ross .Evolutionary Scheduling: A Review. Spring Genetic Programming and Evolvable Machines, 6 (2005)191–220.

[HEF 82] N. Hefetz, I. Adiri .An efficient optimal algorithm for the two-machines unittime job-shop schedule length problem . Math. Oper. Res. 1 (1982) 354-360.

BIBLIOGRAPHIE

[HO 04] N.B .Ho, J.C .Tay . GENACE : An Efficient Cultural Algorithm for solving the flexible job shop problem . Proceeding of IEEE Congress on Evolutionary Computation,1(2004) 1759-1766.

[HO 07] N.B Ho , J.C Tay, M. Edmund, K.Lai . An effective architecture for learning and evolving flexible job shop schedules . European journal of Operational Research 179 (2007) 316-333.

[HOL 73] J.H. Holland. Genetic Algorithms and the optimal allocation of trials, SIAM Journal of Computing. Vol. 2, N° 2, pp. 88-105, 1973.

[HOU 11] Y .Houbad . Modélisation et ordonnancement temps réel d'un job shop à l'aide des méthaheuristique . Mémoire de Magistére en automatique université de Telemcen 2011

[HOU 12] H . Houari. Planification et ordonnancement en temps réel d'un job shop en utilisant l'intélligence artificielle . Mémoire de Magistére en automatique . Université de Telemcen 2012

[HWA 14] S.F.Hwang,Y.Hsu,and Y.Chen .A genetic algorithm for the optimization of fiber angles in composite laminates . Journal of Mechanical Science and Technology. 28(8) (2014) 3163-3169.

[JAC 56] J. R. Jackson. An extension of Johnson's results on job lot scheduling. Naval Res. Logist, Quart. 3(1956) 201-203

[JAI 98a] A.S. Jain , S.Meeran . A-state –of-the –art review of job shop . Electronic and Mecanicang ineering Dundee Scotland.UK1998.

[JAI 98b] A.S. Jain, S. Meeran . Job-shop scheduling using neural networks . Int. J. Prod. Res. 36 (1998) 1249–1272.

[JAT 12] G.K.Jati, H.M.Manurung, S.Suyanto .Discrete cuckoo search for traveling salesman problem. In: 7th International Conference On Computing And Convergence Technology (2012) 993–997.

[JOH 54] S. M. Johnson .Optimal two-and three-stage production schedules with setup time included. Naval Research Logistic Quarterly . 1(1954) 61-81.

BIBLIOGRAPHIE

[JOL 13] F.Jolai,H.Asefi,M.Rabiee,P.Rameza.Biobjective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem . Scientia Iranica, 20 (2013) 861-872.

[JON 05] J.F. Gonc-alves , J.J.Mendes , M.C.G. Resende . A hybrid Genetic algorithm for the job shop scheduling problem . European Journal of Operational Research 2005 (167) 77–95

I. Kacem. Genetic algorithm for the flexible job-shop scheduling problem . IEEE International conference on Systems, Man and Cybernetics .4 (2002)3464–3469.

[KAC 02b] I.Kacem, S. Hammadi,and P.Borne .Pareto-optimality approach for flexible job shop scheduling problems Hybridization of evolutionary algorithms and fuzzy logic Matimatic and Computers in Simulation .60 (2002) 245-276.

[KAC 02c] I.Kacem, S.Hammadi,and P. Borne .Approche By Localization And Multiobjective Evolutionary And Optimization For Flexible Job Shop Scheduling Problems . IEEE Transations Man and Cybernetrics .32 (1) (2002) 1-13.

[KAC 03] I. Kacem .Ordonnancement multicritère des job-shops flexibles : formulation, bornes inférieures et approche évolutionniste coopérative. Thèse de Doctorat, Université de Lille 1.2003.

[KAL 12] S.Kalantari .A Multi-Population Based Frog-Memetic AlgorithmFor Job Shop Scheduling Problem . Advanced Computing: An International Journal 3 (2012) .

[KEB 08] M.Kebabela . Utilisation des stratégies méta heuristique pour l'ordonnancement de type job shop . Mémoire de magistère département de GI. Uuniversité de Batna 2008.

[KHA 08a] S. Khalouli,F.Ghedjati,A.Hamzaoui .Method based on ant colony system for solving the hybrid flow shop scheduling problem . In: 7th International Conference on Modelling . Optimization and SIMulation Systems (MOSIM'08), 2(2008)1407–1416.

[KHA 10] S.Khalouli . Métaheuristique à base de modéle : application à l'ordonnancement d'atelier flow shop hybride monocritére . Thése de doctorat . Université de Reims 2010

[KIM 14] N.Kim ,H.Kim ,and J.Lee .Damage detection of truss structures using two stage optimization based on micro genetic algorithm . Journal Of Mechanical Science And Technology 28(9) (2014) 3687-3695 .

BIBLIOGRAPHIE

[KIR 83] S. Kirkpatrick, C. D. Jr, M. P. Vecchi . Optimization by simulated annealing. Science . 220 (1983) 671-680 .

[KUB 95] W. Kubiak, S. Sethi, C. Srish kandarajah .An efficient algorithm for a job shop problem . Math. Industrial System l (1995) 203-216.

[KUC 10] A.M. Kuczapski, M.V. Micea, L.A. Maniu, V.I. Cretu. Efficient generation of near optimal initial populations to enhance genetic algorithms for job-shop scheduling . Inf. Technol. Control 39 (2010) 32–37.

[KUM 96] N. Kumar, G. Srinivasan .A genetic algorithm for job shop scheduling a case study . Comput. Ind. 31 (1996) 155–160.

[LAC 05] P. Lacomme . Méthodes exactes et approché pour l'optimisation des systèmes à moyen de transport . Thèse HDR . Université Blaise Pascal 2005.

[LAG 13] A.Laggoun . Developpement d'une Approche Pour La Resolution d'un Probleme De Lot Sizing Avec Transport . Mémoire de Magister département de GI Université de Batna -Algérie 2013.

[LAW 84] S. Lawrence. Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques .Graduate School of Industrial Administration .Carnegie-Mellon University .1984

[LAY 10] A. Layeb .Utilisation des Approches d'Optimisation Combinatoire pour la Vérification des Applications Temps Réel .Thèse de doctorat. Université de Constantine 2 .Algérie 2010

[LAY 11] A.Layeb. A novel quantum inspired cuckoo search for knapsack problems .Int. J. Bio-Inspired Comput. 3(2011), 297–305 .

[LEI 10a] D. Lei .Solving fuzzy job shop scheduling problems using random key genetic algorithm .Int. J. Adv. Manuf. Technol. 49 (2010) 253–262.

[LEI 10b] D. Lei .Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling, Appl. Soft Comput. 12 (2012) 2237–2245

[LEN 77] J. K. Lenstra, R. H. G. Rinnooy Kan, P. Brucker .Complexity of machine scheduling problems .Ann. Discrete Math. 4 (1977) 121-140.

[LI 10] Y.Li ,Y.Chen .Genetic algorithm for job shop scheduling .Journal of Software .5 (2010) 269–74.

BIBLIOGRAPHIE

[LI 12] J. Li, S. Xie, T. Sun, Y. Wang, H. Yang .Solving fuzzy job-shop scheduling problem by genetic algorithm .24th Chinese Control and Decision Conference (CCDC) I.EEE .2012 3243–3247

[LI 14] X.Li , M.Yin .Modified cuckoo search algorithm with self adaptive parameter method .Information Siences http://dx.doi.org/10.1016/j.ins. 2014

[LIN 01] F.-T. Lin .A job-shop scheduling problem with fuzzy processing times .Springer-Verlag, Berlin Heidelberg, 2001,409–418.

[LIN 12] J.H .Lin, H.C. Lee .Emotional chaotic cuckoo search for the reconstruction of chaotic dynamics . Latest Advances In Systems Science & Computational Intelligence. WSEAS Press, Athens 2012.

[LIU 05] T. Liu, T.Tsai, J.H.Chou . Improved genetic algorithm for the job-shop scheduling problem .The International Journal of Advanced Manufacturing Technology 27(2007) 1021–1029.

[LOP 01] P.Lopez , F. Roubellat. Ordonnancement de la production . Hermès Sciences, IC2 Productique, 2001.

[MAR 95] J.Marie, X.Xie . Les réseaux de Petri pour la conception et la gestion des systèmes de production .Edition Masson 1995. ISSN : 0249-6992.

[MAR 12] M.K. Marichelvam .An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems . International Jour-Nal Of Bio-Inspired Computation 4 (2012) 200–205.

[MAR 14] M.K. Marichelvan , T. Prabaharan, X.S.Yang . Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan .Applied Soft Computing 19 (2014) 93-101

[MAS 00] M. Mastrolilli . Flexible Job Shop Problem.http:// www.idsia.ch/~monaldo/ fjsp.html.

[MAT 04] D.C Mattfeld, C.Bierwirth .An efficient genetic algorithm for job shop scheduling with tardiness objectives . European Journal of Operational Research 155 (2004) 616–630.

[MEL 04] K. Mellouli, A.El Kamel, P. Borne .Programmation linéaire et applications . Eléments de cours et exercices résolus .Editions Technip 2004.

BIBLIOGRAPHIE

[MES 98] K.Mesghani , S.Hammadi, and P. Borne .On modeling genetic algorithms for flexible job shop scheduling problems .1998.

[MET 53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller. Equation of State Calculations by Fast Computing Machines .J. Chem. Phys., 21(1953) 1087–1092 .

[MOS 89] P. Moscato .On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms .Technical Report C3P 826, Cal-teech Concurrent Computation Program.1989

[MOT 10] A. Motaghedi,K.L. Sabri-Laghare, and M.Heydari. Solving flexible job shop scheduling problem with multi objectives . International Journal of Industrial Engineering and Production Research 21(2010) 197-209.

[NAI 05] T.Nait , F.Yalaoui, L.Amodeo, C. Chu .An ant colony system minimizing total tardiness for hybrid job shop scheduling problem with sequence dependent setup times and release dates . International Conference on Industrial Engineering and Systems Management, Marrakech, Morocco, (2005)10.

[NAK 91] R. Nakano, T. Yamada .Conventional genetic algorithm for job shop problems. Proceedings of the Fourth International Conference on Genetic Algorithms .San Mateo, (1991) 474–479.

[NEW 80] Newella . The heuristic of George Polya and its relation to artificial intelligence. The International Symposium on the Methods of Heuristic. University of Bern, Switzerland (1980)195-244.

[OSM 96] I.H. Osman, G. Laporte .Metaheuristics : A bibliography. Operations Research. 63(1996) 513-623.

[OUA 13] A .Ouaarab,B. Ahiod, and X.S.Yang . Discrete cuckoo search algorithm for the travelling salesman problem . Neural Comput Appl Springer (2013) 1–11.

[PAL 14] J.J.Palacios. ,A. González ,and C.R.González . Genetic tabu search for the fuzzy flexible job shop problem . Computers & Operations Research,(2014) 5474–89.

[PAN 09] S.G .Pannanbalam ,N. Jawahar, and B.S.Girish .Giffler and Thampson Procedure based genetic algorithms for scheduling job shops. Springer-Verbag Berlin Heidelberg (2009)229-259.

[PAR 03] B.J.Park ,.H.R Choi , H.S.Kim .A hybrid genetic algorithm for the job shop scheduling problems . Computers & Industrial Engineering .14 (2003) 597–613.

BIBLIOGRAPHIE

[PAR 09] Y.K.Park ,J.M.Yang . Optimization of mixed casting processes considering discrete ingot sizes . Journal Of Mechanical Science and Technology 23 (2009) 1899-1910.

[PAU 95] J.A.Paulli . Hierarchical approach for the FMS scheduling problem .European Journal of Operational Research. 86 (1995) 32- 42.

[PEA 84] J. Pearl. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Publ. Co, London, 1984.

[PET 04] S. Petrovic, C. Fayad . A fuzzy shifting bottleneck hybridised with genetic algorithm for real-world job shop scheduling . In: Proceedings of Mini-EURO Conference . Managing Uncertainty in Decision Support Models .Coimbra, Portugal, Citeseer 2004 1–6.

[PEZ 07] F.Pezzela ,G.Margenti ,G.Ciaschetti .A genetic algorithm for flexible job shop scheduling problem . Computers and Operations Research, 35(10) (2007).3202-3212.

[RAA 00] W.Raaymakers,J. Hoogeveen . Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing . European Operations Research, 126 (2000) 131–151

[RAF 09] S. N. Rafinejad, F. Ramtin, A. B. Arabani . A New Approach to Generate Rules in Genetic Algorithm Solution to a Job Shop Schedule by Fuzzy Clustering. Proceedings of the World Congress on Engineering and Computer Science 3 (2009).

[RAJ 11] R .Rajabioun . Cuckoo optimization algorithm, Applied Soft Computing 11 (2011) 5508-5518

[RAM 12] R.Ramkumar and A.Tamilarasi .A Real Pratical Approach for multi-objective job shop scheduling Using Fuzzy Logic Approach . Journal of computer science 8(2012) 606-612 ISSB 1549-3636

[REE 10] C. R. Reeves . Chapter 5 : Genetic Algorithms' International Series in Operations Research & Management Science 146 . Springer Science 2010

[ROY 70] B. Roy . Algèbre moderne et théorie des graphes. Editions Dunod, Paris, 1970.

[SAK 84] M. Sakarovitch,. Graphes et Programmation Linéaire . Edition Hermann, Paris, 1984.

BIBLIOGRAPHIE

[SAK 99] M. Sakawa, T. Mori . An efficient genetic algorithm for job shop scheduling problems with fuzzy processing time and fuzzy due date . Comput. Ind. Eng. 36 (1999) 325–341.

[SAK 00] M. Sakawa, R. Kubota . Fuzzy programming for multi-objective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms . Eur. Operations Research. 120 (2000) 393–407.

[SAK 01] M. Sakawa, R. Kubota .Two-objective fuzzy job shop scheduling through genetic algorithm . Electron. Commun. Jpn (Part III: Fundam. Electron. Sci.) 84 (2001) 60–68.

[SAK 09] A.Sakly. Méthode arborescentes pour la résolution de problèmes d'ordonnancement flexible . Thèse doctorat .Université de Toulouse 2009

[SAN 05] S.Sankar, S.G.Ponnambalam, V. Rathinavel, M.SVisveshvaren,Scheduling in parallel machine shop: an ant colony optimization approach. In: Industrial Conference on Industrial Technology, ICIT, (2005) 276–280.

[SHI 08] S.O. Shim, Y.D. Kim .A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property .Computing. Operational Research. 35 (2008) 863–875.

[SHI 09] S.O.Shim .Generating sub problems in branch and bound algorithms for parallel machines scheduling problem . Computers and Industrial Engineering. 57 (2009) 1150–1153.

[SLA 71] J. R. Slagle. Artificial intelligence: The heuristic programming approach. McGraw-Hill. pp. 3. New York, 1971

[SOL 79] R. L. Solso. Cognitive psychology. Harcourt Brace Jovanovich, Inc., New York. 436, 1979.

[SOM 01] H.Someya, M.Yamamura . Genetic algorithm with search area adaptation for the function optimization and its experimental analysis . Proceedings of congress on evolutionary computation .(2001)933–940.

[SOT 95] Y. N. Sotskov, N. V. Shaklevich . NP-hardness of shop scheduling problems with three jobs. Discrete Appl. Math , 59 (1995) 237-266.

[SRI 94] M. Srinivas, L.M. Patnaik .Genetic Algorithms: A Survey . IEEE 1994

BIBLIOGRAPHIE

[SRI 09] N.R.Srinivasa Raghavan, M.Venkataramana . Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization. International Journal of Advanced Manufacturing Technology 41 (2009) 986–996.

[STA 06] Standard PSO2006: http://www.particleswarm.info/Programs.html. 2012

[STU 98] T.Stutzle . An ant approachto the flow shop problem.Proceedingsofthe 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98), Aachen,Germany, (1998)1560–1564.

[SUB 12] M.Subotic, M.Tuba, and N.Bacanin, D. Simian . Parallelized cuckoo search algorithm for unconstrained optimization .Proceedings of the 5th WSEAS Congress on Applied Computing Conference and Proceedings of the 1st International Conference on Biologically Inspired Computation,(2012)151–156.

[SUN 10] W.Sun,Y.Pan ,X.Lu, and Q.Ma . Research on flexible job shop scheduling problem based on a modified genetic algorithm .Journal of Mechanical Science And Technology 24(10) (2010) 2115-2119.

[SUR 10] P. Surekha , S.Sumathi .Solving Fuzzy based Job Shop Scheduling Problems using Ga and Aco . Journal of Emerging Trends in Computing and Information Sciences E-ISSN 1(2010) 2218-6301

[TAM 12] A.Tamimarasi , S. Jayasankari . Evolution on GA based Model For Solving JSSP. International journal of computer application 43(2012) .

[TAN 09] F. Tangour, I. Saad et P. Borne .Optimisation par colonie de fourmis .Revue de l'Electricité et de l'Electronique .REE 2009 39-44 .

[TAY 08] J.C. Tay , N.B. Ho. Evolving dispatching rules using genetic programming for solving multi- objective flexible jobshop problems . Computers and Industrial Engineering .54 (2008) 453-473.

[TSU 95] Y. Tsujimura, M. Gen, E. Kubota .Solving job-shop scheduling problem with fuzzy processing time using genetic algorithm . J. Jpn Soc. Fuzzy Theory Syst 7 (1995) 1073–1083.

[TUB 11] M.Tuba,M.Subotic,N.Stanarevic .Modified cuckoo search algorithm for unconstrained optimization problems .Proceedings of the 5th European Conference on European Computing Conference, (2011) 263–268.

[VEN 12] A. Ventura , S. Yoon . A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. Springer Science 24 (2012).

BIBLIOGRAPHIE

[VIL 07] G.Vilcot . Algorithme approchés pour des problèmes d'ordonnancement multicritère des type job shop flexible et job shop multicritére. Thése doctorat .Université Francois Rabelais 2007.

[VIL 10] G. Vilcot, J.C Billaut . A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem . European Journal of Operational Research ,190(2010) 398–411.

[WAL 11] S.Walton, O.Hassan, K.Morgan, M.R. Brown. Modified cuckoo search: A new gradient free optimisation algorithm. Chaos, Solitons Fractals 44(9) (2011) 710–718

[WAN 99] Y Wang , L. Kwei . Implementing a general real time scheduling framework in the RED-Linux real-time kernel. In IEEE Real-Time Systems Symposium. (1999)246–255 .

[WAN 01] L.Wang , D.Zheng .An effective hybrid optimisation strategy for job-shop scheduling problems . Computers & Operations Research , 28(6)(2001) 585–596.

[WAN 12] R.Q.Y.Wang . A new hybrid genetic algorithm for job shop scheduling problem. Computers and Operations Research , 39 (2012) 2291–2299.

[WAT 05] M.Watanabe,K.Idab , M. Gen . A genetic algorithm with modified crossover operator and search area adaptation for the job shop scheduling problem. Computers and Industrial Engineering .48 (2005) 743–752

[XIA 06] W. Xia, Z. Wu. A hybrid particle swarm optimization approach for the job-shop scheduling problem . International Journal of Advanced Manufacturing Technology. 29 (2006) 360–366.

[XIN 09] L.N.Xing, Y.U.Chen, K.W.Yang . Multi. Population interactive coevolutionnary algorithm for flexible job shop scheduling problems. Comput Optim Appl(2009)DOI 10.1007//S 10589-009-9244-7 .

[YAD 14] R.N.Yadar ,V.Yadar,and G.H.Singh . Application of non dominated sorting genetic algorithm for multi objective optimization of electrical discharge diamond face grinding process. Journal of Mechanical Science And Technology,28(6) (2014) 2299-2306.

[YAL 02] F. Yalaoui, C. Chu .Parallel machine scheduling to minimize total tardiness .Int. J. Prod. Econ. 76 (2002) 265–279.

BIBLIOGRAPHIE

[YAN 09] X-S Yang, S. Deb. Cuckoo search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (2009)210-214 .

[YAN 10] X-S Yang, S. Deb. Engineering optimisation by cuckoo search. Int. J. Mathematical Modelling and Numerical Optimisation . 1(2010)330-343.

[YAN 13] X.S.Yang,S. Deb . Multi objective cuckoo search for design optimization. Computers and Operations Research . 40(6) (2013) 1616–1624 .

[YIN 04] K.C.Ying, C.J.Liao. An ant colony system for permutation flow shop sequencing Computers and Operations Research 31 (5)(2004) 791–801.

[YIN 07] K.C.Ying, S.W.Lin . Multi-heuristic desirability ant colony system heuristic for non permutation flow shops scheduling problems . International Journal of Advanced Manufacturing Technology 33 (7–8)(2007) 793–802.

[ZHA 05] C.Zhang,P. Li P,Y. Rao, S. Li . A new hybrid GA/SA algorithm for the job shop scheduling problem . Evolutionary Computation in Combinatorial Optimization (2005)246–259.

[ZHA 06] X. Zhang, X.Hu, X .Tan, J.H. Zhong ,Q. Huang . Implementation of an ant colony optimization technique for job shop scheduling problem .Transactions of the Institute of Measurement and Control 28(2006), 93–108

[ZHA 11] G.Zhang ,L.Gao, and Y.Shi . An effective genetic algorithm for the flexible job shop scheduling problem . Expert System with Application .38 (2011) 3563-3573.

[ZHA 12a] Q.Zhang ,H Manier,and A.Manier . A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constants and bounded proceding times. Computers and operations Research 39 (2012) 17-13.

[ZHA 12b] Y.Zhang , L.Wang, L. Modified adaptive cuckoo search (macs) algorithm and formal description for global optimisation . Int. J. Comput. Appl. Technol. 44(2)(2012) 73–79 .

[ZHO 01] H.Zhou, Y.Feng . The hybrid heuristic genetic algorithm for job shop scheduling. Computers and Industrial Engineering, 40(3) (2001) 191–200.