Centre d'Etude Doctorales : Sciences et Techniques de l'Ingénieur

N° d'ordre : 12/2019

THÈSE DE DOCTORAT

Présentée par

Mme. Ghita LEBBAR

Discipline : Génie mécanique

Spécialité : Génie industriel

Thèse en codirection préparée au sein du Laboratoire de Génie Mécanique (LGM) de la Faculté des Sciences et Techniques de Fès et le Laboratoire de Recherche en Eco-innovation Industrielle et Energétique (LR2E) d'ECAM EPMI de Cergy Pontoise, France

Sujet de thèse

Contribution à l'étude des problèmes d'ordonnancement des systèmes de production de type flow-shop : Modélisation, optimisation et expérimentation

Thèse présentée et soutenue le 27 Avril 2019 devant un jury composé de :

Nom Prénom	Titre	Etablissement	
Pr. Ahmed EL BIYAALI	PES	Faculté des Sciences et Techniques, Fès	Président
Pr. Lionel AMODEO	PES	Université de Technologie de Troyes, France	Rapporteur
Pr. Jalil ABOUCHITA	PES	Faculté des Sciences et Techniques, Fès	Rapporteur
Pr. Mohammed SALLAOU	PES	Ecole Nationale des Arts et Métiers, Meknès	Rapporteur
Pr. Abdelouahhab JABRI	PH	Faculté des Sciences et Techniques, Fès	Examinateur
Pr. Abdellah EL BARKANY	PH	Faculté des Sciences et Techniques, Fès	Directeur de thèse
Pr. Ikram EL ABBASSI Pr. Moumen DARCHERIF	EC PES	ECAM-EPMI de Cergy Pontoise, France ECAM-EPMI de Cergy Pontoise, France	Codirecteurs de thèse

« Un objectif sans plan s'appelle un vœu. »

Antoine de Saint Exupéry

À mes chers parents

À mes sœurs

À ceux qui me sont chers

Remerciements

Ce mémoire est le fruit d'un travail fastidieux réalisé dans le cadre d'une collaboration entre le Laboratoire de Génie Mécanique (LGM) de la Faculté des Sciences et Techniques (FST) de Fès et le Laboratoire de Recherche en Eco-innovation Industrielle et Energétique (LR2E) d'ECAM-EPMI de Cergy pontoise en France. Communément ce si long travail de doctorat réalisé avec beaucoup de soin est considéré comme un investissement singulièrement personnel. Pourtant, il ne faudrait pas omettre que sans le soutien et la contribution de plusieurs personnes, ce résultat n'aurait pas été atteint. Il me fait donc énormément plaisir de débuter ce mémoire par des chaleureux remerciements destinés aux personnes et aux entités qui, de près ou de loin, ont contribué à l'aboutissement de cette thèse.

Je tiens tout d'abord à remercier Monsieur le Professeur Ahmed EL BIYAALI pour l'honneur qu'il me fait en présidant le jury de ma thèse, ainsi que Messieurs les Professeurs Mohamed SALLAOU, Amodeo LIONEL et Jalil ABOUCHITA d'avoir accepté la lourde tâche d'être rapporteurs de ce mémoire. Leurs remarques et suggestions me permettront d'apporter des améliorations à la qualité de cette contribution scientifique. Je remercie également Monsieur le professeur Abdelouahhab JABRI d'avoir bien voulu s'intéresser et participer à l'évaluation de ce travail en tant qu'examinateur.

Je tiens à exprimer toute ma gratitude à mon directeur de thèse Monsieur le Professeur Abdellah EL BARKANY, pour avoir su diriger mes travaux avec beaucoup de disponibilité, de tact et d'intérêt. J'ai particulièrement apprécié sa très grande ouverture face à mes conditions de travail et la confiance qu'il a sue garder en ma capacité à articuler et à rendre à terme tous mes projets. Je ne sais comment exprimer ma profonde gratitude à cette personne autrement qu'en lui promettant d'agir comme elle avec les chercheurs dans ma situation, si un jour l'occasion m'en est donnée.

Je voudrais également exprimer ma reconnaissance à Madame le Professeur Ikram EL ABBASSI et Monsieur le Professeur Moumen DARCHERIF, Co-directeurs de thèse, pour avoir dirigé mes travaux avec beaucoup de disponibilité et d'encouragements. J'ai particulièrement apprécié la pertinence de leurs conseils avisés et leurs écoutes qui ont été prépondérants pour la bonne réussite de mes travaux généralement et des trois stages scientifiques réalisés au sein de leur laboratoire particulièrement.

J'adresse particulièrement mes remerciements à Monsieur le Professeur Mustapha IJJAALI, Doyen de FST de Fès, pour m'avoir assuré un environnement favorable pour mes travaux de recherche. Je remercie encore Monsieur le Professeur El Mestafa EL HADRAMI, Vice-Doyen chargé de la recherche, de la formation continue et des relations extérieures pour son soutien permanent et sa disponibilité. Je saisis l'occasion pour exprimer mes plus sincères remerciements à tous les professeurs du département de Génie Mécanique de la FST de Fès pour leurs conseils, leurs qualités humaines et professionnelles et sans négliger leurs formations intelligentes, qualifiantes et pertinentes.

Au sein de l'Equipe Management et Optimisation des Systèmes de Production (MOSP) du laboratoire Génie Mécanique de la FST de Fès, j'ai trouvé un excellent cadre de travail et une écoute à laquelle je porte beaucoup de gratitude. Je remercie vivement le Professeur Ahmed EL KHALFI, Directeur du Laboratoire Génie Mécanique, pour m'avoir permis de travailler au sein de ce laboratoire.

C'est un agréable devoir d'exprimer également ma très vive reconnaissance à Monsieur le Professeur Jean Michel BRUCKER, Directeur Scientifique d'ECAM-EPMI pour avoir assuré la

bonne coordination de mes séjours scientifiques en France, dans le cadre de la convention de codirection établie entre mes deux laboratoires d'accueil.

Mes remerciements vont également à Monsieur le Professeur Abdelouahhab JABRI, Professeur Habilité à la FST de Fès pour son aide précieuse et sa grande disponibilité qui m'ont permis de surmonter les difficultés et de progresser.

Je remercie chaleureusement Messieurs les professeurs Karim LABADI et Samir HAMACI, Enseignants Chercheurs à ECAM-EPMI/ECS-Lab, pour la gentillesse et la patience qu'ils ont manifestées à mon égard durant cette thèse, pour tous les conseils et la documentation qu'ils ont bien voulu m'envoyer, pour l'hospitalité dont ils ont fait preuve envers moi lors des séjours que j'ai effectués dans le laboratoire dans lequel ils m'ont accueilli. Leur disponibilité permanente et leurs conseils scientifiques pertinents ont largement contribué au bon déroulement de cette thèse.

Je remercie vivement tous les chercheurs du laboratoire LGM et du laboratoire LR2E qui, par leur amicale présence, leur sympathie, leur grande générosité intellectuelle et leur soutien indéfectible ont rendu ce travail agréable. Je leur exprime ici toute ma gratitude.

Il me tient tout particulièrement à cœur de remercier mes parents pour m'avoir soutenu tout au long de mon parcours de recherche et pour toute la confiance qu'ils m'ont apportée. Je n'oublierai pas mes sœurs ainsi que tous les membres de la famille pour leurs soutiens incomparables, à la mesure de tout ce qu'ils m'ont donné depuis toujours. Mille Merci.

Je tiens également à témoigner toute ma sympathie et ma profonde amitié à tous mes chers amis pour l'aide précieuse qu'ils m'ont assurée le long de ces années de thèse et spécialement à ma chère amie Nadia HAMDANI pour son encouragement permanent, son soutien moral et sa relecture attentive.

Nombreux sont ceux qui m'ont encouragé et aidé pour la réalisation de cette thèse. Il m'a été très difficile de rédiger cette page par souci d'oublier quelques personnes ... ! Qu'elles soient toutes assurées par ma plus profonde reconnaissance et mes sincères remerciements même si leurs noms n'y figurent pas !

Résumé

Les travaux de recherche étant du ressort de la résolution des problèmes d'ordonnancement suscitent l'intérêt d'un nombre considérable de chercheurs. Cette rivalité abondante est singulièrement due au large pan de problématiques qui émergent dans les systèmes d'ordonnancement, parmi lesquelles le problème d'atelier à cheminement unique, trivialement appelé « flow-shop », occupe une place extrêment dominante si bien qu'il représente le système de production le plus général qui peut se présenter dans l'industrie.

Dans cette thèse, la résolution des problèmes d'ordonnancement de ces ateliers a pour finalités principales la minimisation des retards relatifs aux dates d'échéances, via le biais de la minimisation du maximum des retards absolus et la maximisation de la productivité à travers la minimisation de la date d'achèvement de la tâche la plus tardive nommée « makepsan ». Cette résolution englobe l'étude des problèmes flow-shop monocritère et multicritères visant globalement à trouver l'allocation optimale des tâches sur les ressources, en répondant au mieux aux critères de performance abordés et en respectant des contraintes réalistes auxquelles le système peut être soumis. Parmi les restrictions issues des milieux industriels réels qui, à notre connaissance nécessitent jusqu'à présent des efforts importants, la contrainte de blocage et de disponibilité des tâches sur la première machine tiennent le premier rang. De nombreux modèles et approches d'optimisation sont développés et largement exploités pour atteindre l'excellence de la productivité. Néanmoins, l'application de tous ces outils et formalismes sur des systèmes englobant les paramètres les plus en vue dans la littérature et les plus affrontés dans les situations pratiques reste limitée.

Dans cet esprit, l'objectif de notre contribution porte principalement sur la modélisation et la résolution des problèmes mono-objectif et multi-objectifs à l'étude, à travers des approches basées sur les algorithmes génétiques, des approches hybrides et des approches fondées sur la notion de dominance. Relativement aux problèmes monocritère, nous considérons le makespan comme critère d'optimisation et nous proposons dans un premier temps une résolution exacte à travers le solveur (CPLEX) suivie d'une résolution approchée à l'aide des algorithmes génétiques. Ces algorithmes sont améliorés par la suite à travers une hybridation séquentielle avec le recuit simulé. Concernant les problèmes multicritères, nous considérons le makepsan et le maximum des retards absolus comme critères de performance et nous suggérons également une résolution exacte et une autre approchée, dans laquelle une amélioration au niveau de la première phase du processus de recherche du bon compromis de l'approche NSGA-II est effectuée. Pour les différents types de problèmes d'ordonnancement considérés, l'analyse des études comparatives décèle que les approches proposées fournissent des résultats pertinents sur les différentes instances numériques générées, comparativement aux meilleurs résultats trouvés auparavant dans la littérature.

Finalement, une validation expérimentale est effectuée sur une ligne réelle de conditionnement des produits pharmaceutiques. Cette dernière a prouvé non seulement l'intérêt de traiter de plus près les problèmes d'ordonnancement considérés, mais également l'efficacité des modèles et des approches de résolution proposés en vue de maximiser implicitement le profit, la performance et la productivité des systèmes de production.

Mots clés : Ordonnancement ; flow-shop ; algorithme génétique ; hybridation séquentielle, algorithme génétique élitiste de tri non dominé ; blocage ; disponibilité des tâches ; modélisation ; optimisation, expérimentation.

Abstract

Research on scheduling problems has been arousing the interest of a considerable number of researchers. This abundant rivalry is singularly due to the large number of problems that emerge in scheduling systems, among them, the problem of tracking single workshop, trivially called "flow-shop", occupies an extremely dominant place, given that it represents the most general production system that can be presented in the industry.

In this thesis, the main objectives of solving the flow-shop scheduling problems are to minimize delays related to due dates, through minimizing maximum tardiness and maximizing productivity throughout minimizing the completion date of the latest task called "makepsan". This resolution is divided into the study of single-criterion and multi-criteria flow-shop problems with the overall aim of finding the optimal allocation of tasks on resources by optimizing the performance criteria addressed and respecting realistic constraints to which the system can be subjected. Among the realistic constraints that, to our knowledge required significant efforts, the constraint of blocking and availability of tasks on the first machine rank first. Many optimization models and approaches are developed and widely used to achieve productivity excellence. Nevertheless, the application of all these tools and formalism on systems that include the most prominent parameters in the literature and the most confronted in practical situations remains limited.

In this context, the target of our contribution focuses mainly on modelling and solving the mono-objective and multi-objective problems under study through several approaches based on genetic algorithms, hybrid approaches and approaches based on the notion of dominance. Regarding single-criteria problems, we consider the makespan as an optimization criterion and we propose first an exact resolution through the solver (CPLEX) followed by an approached resolution using genetic algorithms. These algorithms are improved later through sequential hybridization with simulated annealing. Concerning multi-criteria problems, we consider makepsan and maximum tardiness as performance criteria and we suggest with the same manner an exact and approached resolution, in which an improvement in the first phase of the process of seeking the right compromise of the NSGA-II approach is performed. For the different types of scheduling problems considered, the analysis of the comparative studies reveals that the proposed approaches provide relevant results on the different numerical instances compared to the best results previously found in the literature.

Finally, an experimental validation is carried out on a real pharmaceutical packaging line. It has proved not only the interest in dealing more closely the scheduling problems considered, but also the effectiveness of the models and approaches proposed in order to implicitly maximize the profit, performance and productivity of production systems

Keywords: Scheduling; flow-shop; genetic algorithm; sequential hybridization; non-dominated sorting genetic algorithm II; blocking; release date; modeling; optimization, experiment.

Table des matières

Remerciements **4**

Résumé **6**

Abstract **7**

Liste des figures **12**

Liste des tableaux **15**

Liste des algorithmes **17**

Acronymes **18**

Liste des notations **19**

Introduction générale **21**

Chapitre I : Contexte et état de l'art sur les problèmes d'ordonnancement des systèmes de production **24**

1.1. Introduction 25

1.2. Présentation des problèmes d'ordonnancement d'atelier 25

1.2.1. Définitions 25

1.2.2. Tâches et opérations 26

1.2.3. Ressources 26

1.2.4. Contraintes 27

1.2.5. Critères d'optimisation 27

1.3. Classification des problèmes d'ordonnancement d'atelier 27

1.3.1. Ateliers à cheminement uniques (flow-shop) 28

1.3.2. Ateliers à cheminement multiples (job-shop) 28

1.3.3. Ateliers à cheminement libres (open-shop) 28

1.4. Problèmes d'optimisation et méthodes de résolution 30

1.4.1. Problèmes d'optimisation mono-objectif 31

1.4.2. Problèmes d'optimisation multi-objectifs 31

1.4.3. Méthodes de résolution mono-objectif 32

1.4.3.1. Méthodes exactes 32

1.4.3.2. Méthodes approchées 33

1.4.4. Méthodes de résolution multi-objectifs 35

1.4.4.1. Méthodes Pareto 36

1.4.4.2. Méthodes non Pareto 36

1.4.4.3. Méthodes hybrides 37

1.5. Problématique et orientation de l'activité de recherche 37
1.5.1. Les systèmes de production de type flow-shop 37
1.5.2. Problèmes d'ordonnancement de type flow-shop : État de l'art 38
1.5.3. Problèmes d'ordonnancement de type flow-shop : Classification 41
1.5.4. Orientation de nos travaux et opportunités 42
1.5.5. Flow-shop avec capacité de stockage limitée 43
1.5.6. Flow-shop avec contrainte de disponibilité des jobs 44
1.6. Conclusions 45

Chapitre II : Résolution mono-objectif des problèmes d'ordonnancement d'ateliers flow-shop avec contrainte de blocage 47

2.1. Introduction 48
2.2. Éventail des approches appliquées sur les ateliers flow-shop avec contrainte de blocage .. 48
2.3. Formulation mathématique mono-objectif 49
2.4. Résolution exacte des ateliers flow-shop avec contrainte de blocage 50
2.4.1. Solveur CPLEX 12.7 : ILOG OPL 50
2.4.2. Résultats numériques 51
2.5. Résolution approchée à travers les algorithmes génétiques 53
2.5.1. Introduction aux algorithmes génétiques 53
2.5.2. Procédure des algorithmes génétiques 54
2.5.3. Genèse de la population 55
2.5.4. Évaluation de la fonction objectif 55
2.5.5. Opérateur de sélection 55
2.5.6. Opérateur de croisement 56
2.5.7. Opérateur de mutation 57
2.5.8. Critère d'arrêt 58
2.5.9. Calibrage des paramètres des AGs : Plan d'expériences de Taguchi 59
2.5.10. Résultats numériques 61
2.5.10.1. Résultats des algorithmes génétiques 61
2.5.10.2. Comparaisons des AGs avec d'autres méthodes issues de la littérature 62
2.5.10.3. Discussion et synthèse 65
2.6. Résolution approchée à travers une hybridation séquentielle (SAGA) 66
2.6.1. Classification des approches hybrides 66
2.6.2. Approche du recuit simulé 67
2.6.2.1. Introduction au recuit simulé 67
2.6.2.2. Procédure du recuit simulé 68
2.6.2.3. Algorithme général du recuit simulé 69
2.6.3. Hybridation séquentielle des algorithmes génétiques et du recuit simulé 69
2.6.4. Calibrage des paramètres de la SAGA : Plan d'expériences de Taguchi 70
2.6.5. Résultats numériques 71

2.6.5.1. Résultats de la SAGA 72
2.6.5.2. Comparaisons de la SAGA avec d'autres méthodes issues de la littérature 72
2.6.5.3. Discussion et synthèse 75
2.7. Conclusions 76

Chapitre III : Modélisation et résolution des problèmes d'ordonnancement des systèmes flow-shop multi-objectifs avec contrainte de blocage et de disponibilité des tâches....77

3.1. Introduction 78
3.2. Intérêt de la résolution du système multi-objectifs avec blocage 78
3.3. Formulation mathématique 80
3.3.1. Modèle mathématique du flow-shop multi-objectifs avec blocage 80
3.3.2. Programmation linéaire en nombres entiers pour le flow-shop multi-objectifs avec blocage et disponibilité des tâches 82
3.3.2.1. Indices 84
3.3.2.2. Paramètres 84
3.3.2.3. Variables de décision 85
3.3.2.4. Modèle (PLNE) 85
3.3.2.5. Signification des équations 86
3.4. Résolution exacte du flow-shop multi-objectifs avec blocage 86
3.5. Résolution approchée du flow-shop multi-objectifs avec blocage 91
3.5.1. Introduction à l'algorithme NSGA-II 91
3.5.2. Procédure de l'algorithme R-NSGA-II.ver.2 92
3.5.2.1. Initialisation de la population 92
3.5.2.2. Codage des individus 93
3.5.2.3. Opérateurs génétiques 93
3.5.2.4. Algorithme de tri non-dominé rapide 94
3.5.2.5. Garantie de diversité et procédure de sélection 95
3.5.2.6. Élitisme et algorithme général 96
3.5.3. Calibrage des paramètres de la R-NSGA-II.ver.2 : Plan d'expériences de Taguchi ... 97
3.6. Résultats numériques 99
3.6.1. Mesures de performance 100
3.6.1.1. La mesure de distance (DM) 100
3.6.1.2. Le nombre global de solutions non-dominées (ONSN) 100
3.6.2. Résultats de la RNSGA-II.ver.2 100
3.6.3. Comparaison de la RNSGA-II.ver.2 avec d'autres approches issues de la littérature103
3.6.4. Discussion et synthèse 104
3.7. Conclusions 105

Chapitre IV : Validation expèrimentale : Cas de l'industrie pharmaceutique ...107

4.1. Introduction ... 108

4.2. Problématique industrielle soulevée ... 108

4.3. Ligne de conditionnement des produits pharmaceutiques ... 111

4.4. Phase préparatoire de l'étude du système ... 112

4.4.1. Données préliminaires ... 112

4.4.2. Présentation du simulateur Witness Horizon ... 113

4.4.3. Modélisation de la ligne de conditionnement des produits pharmaceutiques ... 114

4.4.4. Résultats des simulations préliminaires ... 115

4.5. Couplage simulation/optimisation ... 118

4.5.1. Procédure générale ... 118

4.5.2. Coulage LCPP/ SAGA ... 119

4.5.3. Coulage LCPP/ RNSGA-II.ver.2 ... 120

4.6. Simulations et résultats expérimentaux ... 121

4.6.1. Résultats de simulation/optimisation via LCPP/SAGA (5*6) ... 121

4.6.2. Résultats de simulation/optimisation via LCPP/SAGA (7*6) ... 121

4.6.3. Résultats de simulation/optimisation via LCPP/RNSGA-II. Ver. 2 (5*6) ... 122

4.6.3.1. Résultats du Fm/block/Cmax, Tmax ... 122

4.6.3.2. Résultats du Fm/Block, ri/Cmax, Tmax ... 122

4.6.4. Résultats de simulation/optimisation via LCPP/RNSGA-II. Ver. 2 (7*6) ... 122

4.6.4.1. Résultats du Fm/block/Cmax, Tmax ... 122

4.6.4.2. Résultats du Fm/Block, ri/Cmax, Tmax ... 123

4.6.5. Analyse et discussions ... 123

4.7. Conclusions ... 123

Conclusion générale et perspectives ... 125

Bibliographie ... 129

Annexes ... 140

Liste des figures

Chapitre I

Figure 1.1 : Caractéristiques temporelles d'une tâche 26
Figure 1.2 : Représentation d'un atelier de type flow-shop 28
Figure 1.3 : Représentation d'un atelier de type job-shop 28
Figure 1.4 : Ordonnancement dans les différents types d'ateliers [MAC,93] 29
Figure 1.5 : Fonction à différent minima 31
Figure 1.6 : Classification des méthodes d'optimisation mono-objectif 32
Figure 1.7 : Classification des méthodes d'optimisation multi-objectifs 36
Figure 1.8 : Représentation d'un atelier de type flow-shop 38
Figure 1.9 : Evolution des publications des problèmes de type flow-shop 39
Figure 1.10 : Utilisation des machines dans la littérature 41
Figure 1.11 : Pourcentage de publications par type de méthodes de résolution adoptées pour les problèmes d'ordonnancement de type flow-shop 42
Figure 1.12 : Diagramme du Gantt pour le flow-shop sans aucune contrainte 44
Figure 1.13 : Diagramme de Gantt pour le flow-shop avec blocage 44
Figure 1.14 : Flow-shop avec contrainte de disponibilité des jobs 45

Chapitre II

Figure 2.1 : Les composantes d'optimisation ILOG (Annexe 1) 51
Figure 2.2 : Comparaison des temps moyens d'exécution pour les problèmes de petite taille .. 53
Figure 2.3 : Comparaison des temps moyens d'exécution pour les problèmes de grande taille 53
Figure 2.4 : Mécanisme général des algorithmes génétiques 55
Figure 2.5 : Exemple de chromosome avec codage entier 55
Figure 2.6 : Exemple illustratif du croisement standard en 1-point 56
Figure 2.7 : Exemple illustratif du croisement standard en 2-points 57
Figure 2.8 : Exemple illustratif de mutation d'échange réciproque 58
Figure 2.9 : Graphique des effets principaux du rapport (S/B) 60
Figure 2.10 : Courbe de convergence des AGs 61
Figure 2.11 : Benchmark des résultats d'ARPD des différentes méthodes de comparaison 63
Figure 2.12 : Courbes de convergence de l'instance Ta109 des différentes méthodes de comparaison 63
Figure 2.13 : Techniques d'hybridation selon [DUV, 2000] 67
Figure 2.14 : Processus général du recuit simulé 68
Figure 2.15 : Graphique des effets principaux du rapport (S/B) 71
Figure 2.16 : Courbe de convergence de la SAGA 72
Figure 2.17 : Benchmark des résultats d'ARPD des différentes méthodes de comparaison 73
Figure 2.18 : Courbes de convergence de l'instance Ta109 des différentes méthodes de comparaison 74

Chapitre III

Figure 3.1 : Exemple illustratif du Flow-shop multi-objectifs avec blocage 82
Figure 3.2 : Exemple illustratif du Flow-shop multi-objectifs avec blocage et la contrainte de disponibilitè des tâches 83

Figure 3.3 : Temps de calcul moyens pour les problèmes de petite taille..... 90
Figure 3.4 : Temps de calcul moyens pour les problèmes de grande taille..... 90
Figure 3.5 : Procédure générale de l'algorithme génétique élitiste de tri non-dominé révisé 92
Figure 3.6 : Exemple de chromosome avec codage entier..... 93
Figure 3.7 : Exemple d'un Front pareto obtenu à travers le tri non-dominé rapide..... 94
Figure 3.8 : Calcul de la distance d'encombrement 95
Figure 3.9 : Schéma général de la RNSGA-II.ver.2 97
Figure 3.10 : Graphique des effets principaux du rapport (S/B) $F_m|block|C_{max}, T_{max}$,..... 98
Figure 3.11 : Graphique des effets principaux du rapport (S/B) $F_m|block, r_k|C_{max}, T_{max}$ 99
Figure 3.12 : Variation de la mesure ONSN de la R-NSGA-II.ver.2 selon les différentes capacités de stockage $F_m|block|C_{max}, T_{max}$..... 101
Figure 3.13 : L'ensemble des solutions non-dominées produites sur l'instance 40*20 (Front - Pareto) $F_m|block|C_{max}, T_{max}$..... 102
Figure 3.14 : L'ensemble des solutions non-dominées produites sur l'instance 40*20 (Front - Pareto) pour le problème$F_m|block, r_k|C_{max}, T_{max}$ 103
Figure 3.15 : Comparaison d'ONSN obtenus à travers la HDE et la R-NSGA-II.ver.2..... 104

Chapitre IV

Figure 4.1 : Vue globale du système de production étudié..... 109
Figure 4.2 : Illustration d'une séquence des palettes 110
Figure 4.3 : Représentation du système de production étudié..... 112
Figure 4.4 : Modélisation de la ligne de conditionnement étudiée sur le simulateur Witness. 115
Figure 4.5 : Temps de blocage du $F_m|block|C_{max}$ (5*6) 116
Figure 4.6 : Temps de blocage du $F_m|block|C_{max}$ (7*6) 116
Figure 4.7 : Temps de blocage du $F_m|block|C_{max}, T_{max}$ (5*6)..... 116
Figure 4.8 : Temps de blocage du $F_m|block|C_{max}, T_{max}$ (7*6)..... 117
Figure 4.9 : Temps de blocage du $F_m|block, ri|C_{max}, T_{max}$ (5*6)..... 117
Figure 4.10 : Temps de blocage du $F_m|block, ri|C_{max}, T_{max}$ (7*6)..... 117
Figure 4.11 : Démarche générale du couplage simulation/optimisation..... 118
Figure 4.12 : Schéma général du couplage LCPP/ SAGA..... 119
Figure 4.13 : Schéma général du couplage LCPP/ RNSGA-II. Ver.2 120
Figure 4.14 : Temps de blocage après optimisation du $F_m|block|C_{max}$ (5*6) 121
Figure 4.15 : Temps de blocage après optimisation du $F_m|block|C_{max}$ (7*6) 121
Figure 4.16 : Temps de blocage après optimisation du $F_m|block|C_{max}, T_{max}$ (5*6)..... 122
Figure 4.17 : Temps de blocage après optimisation du $F_m|block, ri|C_{max}, T_{max}$ (5*6) 122
Figure 4.18 : Temps de blocage après optimisation du $F_m|block|C_{max}, T_{max}$ (7*6)..... 122
Figure 4.19 : Temps de blocage après optimisation du $F_m|block, ri|C_{max}, T_{max}$ (7*6) 123

Annexes

Figure A2.1 : Type de complexité des problèmes..... 142
Figure A3.1 : Graphe disjonctif classique……..143
Figure A4.1 : Vue globlale de la ligne de condistionnement des produits pharmaceutique….144
Figure A4.2 : Le poste superviseur 145
Figure A4.3 : Illustration du logiciel du poste superviseur..... 145
Figure A4.4 : Sèquence des palletes prêtes pour le lancement 146
Figure A4.5 : Illustration du poste de lancement 146

Figure A4.6 : Illustration du poste de remplissage des comprimés de fer 147
Figure A4.7 : Illustration du poste de dosage du B12..... 148
Figure A4. 8 : Ilustration du poste du bouchonage et etiquetage des flacons..... 148
Figure A4. 9 : Illustration du poste de dèchargement des pallettes 149
Figure A4.10 : Illustration du poste de conditionnement des flacons..... 150
Figure A4.11 : Illustration du poste de fabrication des comprimés 151
Figure A5. 1 : Gamme de production du job 1 de la configuration (5*6) ……...…152
Figure A5. 2 : Gamme de production du job 2 de la configuration (5*6) 152
Figure A5. 3 : Gamme de production du job 3 de la configuration (5*6) 153
Figure A5. 4 : Gamme de production du job 4 de la configuration (5*6) 153
Figure A5. 5 : Gamme de production du job 5 de la configuration (5*6) 154
Figure A5. 6 : Gamme de production du job 1 de la configuration (7*6) 154
Figure A5. 7 : Gamme de production du job 2 de la configuration (7*6) 155
Figure A5. 8 : Gamme de production du job 3 de la configuration (7*6) 155
Figure A5. 9 : Gamme de production du job 4 de la configuration (7*6) 156
Figure A5.10 : Gamme de production du job 5 de la configuration (7*6)..... 156
Figure A5.11 : Gamme de production du job 6 de la configuration (7*6)..... 157
Figure A5.12 : Gamme de production du job 7 de la configuration (7*6)..... 157

Liste des tableaux

Chapitre I

Tableau 1.1 : Abréviation des différentes approches de résolution 39
Tableau 1.2 : Synthèse des travaux traitant les problèmes de type flow-shop..... 39

Chapitre II

Tableau 2.1 : Temps moyens d'exécution par problème sans blocage (en secondes) 52
Tableau 2.2 : Temps moyens d'exécution par problème avec blocage RSB (en secondes) 52
Tableau 2.3 : Différents niveaux de chaque paramètre des algorithmes génétiques 60
Tableau 2.4 : Paramètres optimaux des AGs 60
Tableau 2.5 : Résultats d'ARDP fournis par les AGs 61
Tableau 2.6 : Résultats de comparaison de l'indicateur ARDP 63
Tableau 2.7 : Meilleures limites supérieures fournies sur les instances de [TAI, 1983] 64
Tableau 2.8 : Différents niveaux de chaque paramètre de la SAGA 71
Tableau 2.9 : Paramètres optimaux de la SAGA 71
Tableau 2.10 : Résultats d'ARDP fournis par la SAGA..... 72
Tableau 2.11 : Résultats de comparaison de l'indicateur ARDP..... 73
Tableau 2.12 : Meilleures limites supérieures fournies sur les instances de [TAI, 1983] 74

Chapitre III

Tableau 3.1 : Différents scénarios de paramétrage des facteurs TF et DR..... 87
Tableau 3.2 : Temps d'exécution moyens par problème multi-objectifs avec blocage RSB 87
Tableau 3.3 : Temps d'exécution moyens par problème multi-objectifs avec blocage et disponibilité des tâches (en secondes)..... 88
Tableau 3.4 : Différents niveaux de chaque paramètre de la R-NSGA-II.ver.2 98
Tableau 3.5 : Paramètres optimaux de la R-NSGA-II.ver.2 98
Tableau 3.6 : Paramétrage des capacités de stockage variables 99
Tableau 3.7 : Résultats d'ONSN et de DM fournis par la RNSGA-II.ver.2 $F_m|block|C_{max}, T_{max}$ 101
Tableau 3.8 : Résultats d'ONSN fournis par la RNSGA-II.ver.2 $F_m|block, r_k|C_{max}, T_{max}$..... 102
Tableau 3.9 : Résultats de comparaison des mesures de performance de la HDE et la R-NSGA-II.ver.2..... 103

Chapitre IV

Tableau 4.1 : Pourcentage du Fer et du B12 de la gamme de production 1 (5*6)..... 112
Tableau 4.2 : Pourcentage du Fer et du B12 de la gamme de production 2 (7*6)..... 112
Tableau 4.3 : Durées opératoires de la gamme de production 1 (5*6) 113
Tableau 4.4 : Durées opératoires de la gamme de production 2 (7*6) 113
Tableau 4.5 : Paramètres optimaux des trois différents problèmes 121

Liste des algorithmes

Chapitre II

Algorithme 2.1 : Procédure de croisement standard en 2-points 57
Algorithme 2.2 : Procédure de mutation d'échange réciproque 58
Algorithme 2.3 : Algorithme génétique pour le $F_m|block|C_{max}$ 58
Algorithme 2.4 : Algorithme SA 69
Algorithme 2.5 : Algorithme SAGA pour résoudre le $F_m|block|C_{max}$ 70

Chapitre III

Algorithme 3.1 : Procédure d'initialisation de la population à travers l'algorithme RNEH-WPT 93
Algorithme 3.2 : Procédure de tri non-dominé rapide 94
Algorithme 3.3 : Procédure de calcul de la distance d'encombrement 96
Algorithme 3.4 : Procédure générale de l'approche R-NSGA-II.ver.2 96

Acronymes

AA	Hybrid Simulated Annealing
AA	Hybrid Simulated Annealing
ACO	Ant Colony Optimization
ACO	Ant Colony Optimization
AG	Algorithme Génétique
AGs	Algorithmes Génétiques
ARDP	Average Percentage Relative Difference
B & B	Branch and Bound
B&B	Branch and Bound
CPSO	Combinatorial particle swarm
CPSO	Combinatorial particle swarm
DDE	Discret differential evolution
DDE	Discret differential evolution
DF	Facteur d'Echéances
DM	Distance Mesure
GA	Genetic Algorithm
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedures
GRASP	Greedy Randomized Adaptive Search Procedures
IBM	International Business Machines
ICG	Iterated cocktail greedy
ICG	Iterated cocktail greedy
IG	Iterated greedy
IG	Iterated greedy
LCPP	Packaging Line for Pharmaceutical Products
LS	Local Search
LS	Local Search
MOGA	Multiple Objectives Genetic Algorithm
Np-complet	Non déterministe Polynomial-complet
NP-difficile	Non déterministe Polynomial-difficile
ONSN	Overall Number of Non-Dominated Solutions
OPL	Optimization Programming Language
PAES	Pareto Archived Evolution Strategy .
PF	Profile Fitting
PSO	Particle Swarm Optimization
PSO	Particle Swarm Optimization
RAM	Random Access Memory
RNSGA-II.ver.2	Revised Non Dominated Sorting Genetic Algorithm- II
RSB	Release When Starting Blocking
SA	Simulated Annealing
SAGA	Simulated annealing genetic algorithm
SPEA	Strength Pareto Evolutionary Algorithm
TF	Facteur du retard
TS	Tabu Search
TS	Tabu Search
VEGA	Vector Evaluated Genetic Algorithm

Liste des notations

C_J	Date de fin du job j
T_{max}	Maximum des retards absolus
T_J	Retard du Job j
L_{max}	Maximum des retards absolus algébriques
L_J	Retard algébrique du job j
E^w	Retard algébrique pondéré
$\overline{T}$	Retard moyen
$\sum Ui$	Nombre des jobs en retard
C^w	Date de fin pondérée du job
$\sum Fi$	Temps total de séjour
F^w	Temps total de séjour pondéré
MIT	Temps total d'inactivité
Fi	Temps de séjour
$\overline{F}$	Temps moyen de séjour
F1	Flow-shop à une machine
F2	Flow-shop à deux machines
Fm	Flow-shop à m machines
$\mathbb{R}^n$	Espace euclidien
Sp	Ensemble des solutions dominées
NP	Conteur de dominance
I[i]. m	Valeur de la fonction objectif m de l'individu i dans l'ensemble I
f_m^{max}	Valeur maximale de la fonction objectif m
f_m^{min}	Valeur minimale de la fonction objectif m.
V	Vitesse du convoyeur
$\frac{S}{N}$	Rapport Signal/Bruit
$C_{l,k}^i$	Makespan de la ième instance fourni par le ième algorithme dans la kème exécution
C_l^R	Makespan obtenu par [RON, 2005]
$e^{\frac{-\Delta f}{T}}$	Facteur de Boltzmann-Gibbs
T_{n+1}	Tempèrature à l'instant n+1
T_n	Tempèrature à l'instant n
n	Nombre total des jobs
m	Nombre total des machines
α	Poids associé au makespan.
β	Poids associé au maximum des retards absolus.
Z(P)	Fonction objectif
k	Indice sur la machine
$\pi_{(j)}$	Indice du job disposé à la Jème position
$D_{(j),k}$	Date de départ du job j sur la machine k
$S_{(j),k}$	Date de début du job j sur la machine k
$p_{(j),k}$	Temps opératoire du job j sur la machine k
i	Indice Sur le Job
j	Indice sur la machine
π(i)	Position du job i dans la séquence ∏
$t_{\pi(i),j}$	Temps opératoire du job qui se trouve dans la position i dans la séquence π sur la machine j
$d_{\pi(i)}$	Date d'échéance du job qui se trouve dans la position i dans la séquence π
$T_{\pi(i)}$	Retard de chaque job qui se trouve dans la position i dans la séquence π dans la séquence

$C_{\pi(i),j}$	Date de fin de chaque job qui se trouve dans la position i dans la séquence π sur la machine j
$H_{\pi(i),j}$	Date de début du job qui se trouve dans la position i dans la séquence π sur la machine j
$C_{\pi(N),M}$	Temps d'achèvement du dernier job sur la dernière machine
k	Indice sur le job.
i	Indice sur la machine
j	Indice sur la position du job k dans la séquence P.
t_{ki}	Temps d'exécution du job k sur la machine i, k =1, 2…,n , i= 1,2,…,m.
D_k	Date souhaitée au plus tard du job k (la date d'échéance du job k) k =1,2,…,n.
r_k	Date de disponibilité du job k, k =1, 2,…,n.
H_{ji}	Date de début du job k à la position j sur machine i
C_{ji}	Date de fin du job k à la position j sur machine i

Introduction générale

Les innovations technologiques de la production, les attentes de la clientèle et les évolutions du marché renforcées par la mondialisation en particulier, ont accentué la nécessité d'assurer tous les moyens permettant d'améliorer sans cesse la productivité tout en réduisant les coûts.

En vue d'atteindre une productivité accrue et réduire considérablement les coûts de production, l'emploi des techniques d'ordonnancement s'impose comme un choix opérationnel primordial pour toute entreprise afin de revêtir un large éventail d'activités ayant pour objet de faire en sorte que les matières, les équipements et les ressources humaines soient disponibles au bon moment et à l'endroit requis. Cette prise de décision opérationnelle permettant en effet de sélectionner le meilleur routage en tenant compte du volume de production, la disponibilité des ressources et leurs capacités a suscité depuis des décennies une recherche exhaustive vu l'importance industrielle particulière qu'elle porte pour les chercheurs et les praticiens aux milieux industriels qui se heurtent quotidiennement à des problématiques d'ordonnancement de complexité grandissante variant selon les technologies employées et les contraintes auxquelles est soumis le système.

Dans le même contexte industriel qui cherche principalement à améliorer le rendement et minimiser le coût de production, la réduction ou la suppression des stocks intermédiaires est devenue le leitmotiv des industriels. En effet, une réduction assimilée a pour objet de prévenir la désuétude des articles stockés et de gagner de l'espace résultant des zones de stockages supprimées et donc de réduire significativement les coûts de stockage. Néanmoins, le contingentement des zones de stockage entre les postes de travail consécutifs donne lieu à des situations de blocage menant à une augmentation importante du temps improductifs du système. Toutefois, lorsque la zone de stockage entre deux machines et inexistante ou pleine, la tâche reste bloquée sur la machine en amont même si son opération est achevée jusqu'à ce que la machine en aval soit disponible pour le traitement. Cette restriction de blocage est désignée dans la littérature par Release When Starting Blocking (RSB) et elle décrit l'une des contraintes de précédence qui peut exister entre les différentes structures du système.

Dans cet ordre d'idées, la contrainte pour laquelle les tâches ont des dates de disponibilité différentes sur la première machine représente l'une des restrictions qui influe sur la précédence et la fluidité associée au bon déroulement des opérations d'ordonnancement. A ce jour, il existe peu de recherches théoriques exhaustives touchant de près cet aspect temporel qui influe potentiellement sur le lancement et le démarrage des tâches successives et engendre par la suite des temps d'attente très élevés.

Les problèmes d'ordonnancement sont formulés en problèmes d'optimisation combinatoire de très forte complexité théorique, ils sont souvent classés comme étant des problèmes d'optimisation NP-difficiles (Annexe 2) [RIN, 1976]. Selon la nature du problème d'ordonnancement qui peut être mono-objectif ou multi-objectifs, deux grandes catégories d'approches de résolution peuvent être distingués :

- Les méthodes exactes qui garantissent l'optimalité globale avec un effort calculatoire qui augmente de façon exponentielle avec la taille du problème ;
- Les méthodes approchées qui permettent de fournir une solution proche de l'optimum dans un temps de calcul plus ou moins raisonnable.

Que ce soit sur le plan pratique ou théorique, les problèmes d'ordonnancement d'atelier sont parmi les problèmes d'ordonnancement qui occupent une place importante en raison de la complexité supplémentaire qui en résultent. Dans ce cas, la complexité ne se localise pas dans le processus de production prédéterminé mais plutôt dans la combinatoire qui naît de la prise en

compte des restrictions de l'ensemble des ressources considérées. Dans la littérature, Il existe trois types de problèmes d'atelier, lorsque l'ordre de passage des tâches est fixé et commun à toutes les tâches, nous parlons de problèmes d'atelier à cheminement unique (flow-shop). Si cet ordre est fixé mais varie d'une tâche à une autre, il s'agit des problèmes d'atelier à cheminements multiples (job-shop). Enfin, si le routage des tâches est libre, le problème est dit d'atelier à cheminements libres (open-shop).

Dans le cadre de l'étude théorique de cette thèse, nous nous intéressons particulièrement à la résolution des problèmes d'ordonnancement d'atelier de type flow-shop dans les cas monocritère et multicritères avec les contraintes de blocage et de disponibilité des tâches précitées. Dans le cas monocritère, le critère d'optimisation consiste à minimiser la date d'achèvement de tous les travaux nommé makespan. L'optimisation de cette mesure mène bien évidemment à une utilisation maximale des ressources et une réduction importante du temps d'attente ou de non production. Tandis que relativement au problème d'ordonnancement d'atelier flow-shop multicritères, nous rajoutons le maximum des retards absolus comme critère de performance du fait qu'actuellement les entreprises se doivent de séduire des clients toujours plus exigeants non seulement sur la qualité et le prix mais également sur le délai de mise à disposition. En effet, dans les deux cas de figures, la minimisation du makespan et du retard peut se traduire par des gains significatifs, surtout dans des milieux où la concurrence est de plus en plus accrue. A cet égard, l'objectif central de cette thèse est d'enrichir les modèles mathématiques existants et de proposer des approches de résolution efficaces permettant de supporter la prise de décision opérationnelle des problèmes d'ordonnancement d'atelier flow-shop mono-objectif et multi-objectifs, en tenant compte des différents types de contraintes existantes entre les différentes machines.

Pratiquement, un cas d'application a été identifié dans le laboratoire de recherche de l'école ECAM-EPMI située à Cergy pontoise en France. Ce laboratoire est équipé d'une ligne de conditionnement de produits pharmaceutiques dans laquelle la minimisation des temps d'attente, la maximisation de l'utilisation des ressources et l'augmentation de la qualité et la vitesse de son service dès que la commande est prise jusqu'à la livraison de ses produits, représentent des défis de tous les jours. La cible principale de cette phase de recherche réside dans la validation expérimentale de tous les modèles et les approches proposés en vue de résoudre théoriquement les problèmes d'ordonnancement d'atelier flow-shop d'une part, et de répondre à la problématique d'ordonnancement soulevée au niveau de la ligne de production étudiée d'autre part.

Partant de ce constat le projet collaboratif qui est né entre le laboratoire de recherche en éco-innovation industrielle et énergétique (LR2E) d'ECAM EPMI de Cergy pontoise et le laboratoire génie mécanique (LGM) de la faculté des sciences et techniques de Fès a constitué un élément décisif pour la réalisation de ce projet de doctorat.

Ce rapport s'articule autour de quatre chapitres organisés de la manière suivante :

Dans le premier chapitre, une vue d'ensemble sur les problèmes d'ordonnancement des systèmes de production est exposée. L'objectif de ce chapitre est de rappeler les différents éléments qui interviennent dans un problème d'ordonnancement en détaillant les différents types d'ateliers et introduisant les différentes approches d'optimisation proposées dans la littérature pour la résolution des problèmes d'ordonnancement monocritère et multicritères. Cette introduction aux problèmes d'ordonnancement nous mène à réaliser une étude bibliographique suivie d'une examination profonde de l'état de l'art dressé afin de positionner notre problématique et montrer son intérêt scientifique et pratique.

Le deuxième chapitre est dédié à la résolution mono-objectif de l'atelier flow-shop qui vise à minimiser le makespan avec contrainte de blocage. Ce chapitre est composé de trois parties : Dans la première partie, une résolution exacte à travers le solveur CPLEX est effectuée afin d'évaluer la

performance du modèle mathématique présenté et de mettre en évidence la complexité du système étudié. La deuxième partie est consacrée à une résolution approchée à travers la métaheuristique qui manipule une population de solution la plus connue dans la branche de la recherche opérationnelle, étant les algorithmes génétiques. Dans la troisième partie, une hybridation séquentielle des algorithmes génétiques et du recuit simulé est proposée en vue de bénéficier des caractéristiques respectives des deux approches combinées.

Le troisième chapitre est consacré à la résolution de deux types de problèmes d'ordonnancement multi-objectifs. Le premier étant le problème d'ordonnancement d'atelier flow-shop visant à minimiser simultanément le makespan et le maximum des retards absolus avec contrainte de blocage. Tandis que le deuxième problème représente le même système avec une contrainte de disponibilité des tâches supplémentaire. Dans les deux types de problèmes, une résolution exacte à travers le même solveur CPLEX est effectuée, suivie d'une résolution approchée à travers une amélioration au niveau de la phase d'initialisation de la population du fameux algorithme génétique élitiste de tri non-dominé (NSGA-II).

Dans le quatrième chapitre, une validation expérimentale de l'ensemble des travaux développé est effectuée à travers une minimisation du temps d'attente dû aux lacunes d'ordonnancement existantes dans une ligne réelle de production et de conditionnement des produits pharmaceutiques dans laquelle les contraintes considérées dominent fortement. Cette optimisation est effectuée à travers des approches de simulation/optimisation basées sur le couplage du modèle LCPP de la ligne de conditionnement des produits pharmaceutiques modélisé à travers le logiciel Witness avec les deux approches d'optimisation proposées pour la résolution des deux types de problèmes d'ordonnancement mono-objectif et multi-objectifs présentés dans le troisième et le quatrième chapitres, respectivement.

Finalement, nous présentons une conclusion générale qui synthétise nos contributions et qui formule quelques perspectives de recherche qui peuvent découler de ces études.

Chapitre I

Contexte et état de l'art sur les problèmes d'ordonnancement des systèmes de production

Chapitre 1
Contexte et état de l'art sur les problèmes d'ordonnancement des systèmes de production

1.1. Introduction

En vue de relever le défi de la performance et de la compétitivité dans un environnement économique, les organisations sont de plus en plus amenées à s'adapter et accroître leur réactivité vis-à-vis du marché afin de garantir leur pérennité. Dans cette perspective de perfectionnement, la recherche d'un ordonnancement capable d'obtenir la production des biens dans un délai acceptable en tenant compte des différentes contraintes, s'avère décisive pour réduire le coût de production et augmenter le gain des entreprises.

En effet, un ordonnancement consiste à planifier opérationnellement un procédé par lequel des priorités sur les ressources successives sont données à des tâches à l'exécution, tout en améliorant au mieux un ou plusieurs critères d'optimisation donnés. Pratiquement, les unités de production utilisent rarement une ou deux ressources à chaque poste pour exécuter une opération ; la chaîne de production est généralement constituée d'une série de postes. Cette duplication des ressources à chaque poste a pour but d'accroître la flexibilité et l'efficacité de la chaîne de production [MAC, 1993], [Edi, 2007] et [AWA, 2012], et a engendré l'apparition des problèmes d'ordonnancement des ateliers qui sont justifiés par une complexité supplémentaire comparativement aux problèmes d'ordonnancement traditionnels.

Historiquement, les problèmes d'ordonnancement d'ateliers présentent un grand intérêt à la fois pour les différentes communautés scientifiques et pour les praticiens en milieu industriel qui font face à des problématiques rendues complexes par les contraintes des situations réelles. C'est pourquoi leurs applications ont fait l'objet d'un très grand nombre de travaux de recherche et d'ouvrages.

L'objectif de ce chapitre est de présenter une revue de littérature sur les problèmes d'ordonnancement des systèmes flexibles de production. Dans la première partie de ce chapitre, des concepts de bases et les principales différentes approches de résolution des problèmes d'ordonnancement d'ateliers sont introduits. La deuxième partie porte sur le positionnement de notre problématique en se basant sur les références ayant, abordées, analysées et développées les problèmes d'ordonnancement d'ateliers et plus particulièrement, des ateliers de type flow-shop, menant à déterminer des pistes de recherches futures.

1.2. Présentation des problèmes d'ordonnancement d'atelier

1.2.1. Définitions

La littérature sur les problèmes d'ordonnancement recouvre de nombreuses définitions de l'ordonnancement. Dans [ROD, 1988], ordonnancer le fonctionnement d'un système industriel de production consiste à gérer l'allocation des ressources au cours du temps, tout en optimisant au mieux un ensemble de critères. Une autre définition centrée sur un problème d'ordonnancement est donnée par [CAR, 1988] tel que « Un ordonnancement est une programmation de l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution ». Avec un aperçu plus opérationnel, [GOT, 1993] et [LOP, 2001] définissent un ordonnancement comme étant une organisation des tâches, en déterminant leurs dates de démarrage et en leurs attribuant des

ressources matérielles ou humaines nécessaires, de telle sorte que des contraintes soient respectées, afin d'optimiser un certain objectif préalablement défini. Autrement dit, un ordonnancement selon [PAR, 1985], est définit comme suit :

« Déterminer ce qui va être fait, quand, où et avec quels moyens ; étant donné un ensemble de tâches à accomplir, le problème d'ordonnancement consiste à déterminer quelles tâches doivent être exécutées et à assigner des dates et des ressources à ces tâches de façon à ce que les tâches soient, dans la mesure du possible, accomplies en temps utile, au moindre coût et dans les meilleures conditions ».

Dans un problème d'ordonnancement interviennent quatre notions fondamentales à savoir **:**

- Les tâches et opérations ;
- Les ressources ;
- Les contraintes ;
- Les critères d'optimisation.

1.2.2. Tâches et opérations

L'exécution d'un ordonnancement repose sur la définition de l'ensemble des tâches (Travaux/ Jobs) à réaliser. Dans le cadre de notre travail, nous nous intéressons plus particulièrement aux problèmes d'ordonnancement d'ateliers. Dans ces circonstances, une tâche signifie « opération ». Comme illustré sur la figure 1.1, une opération est une entité élémentaire repérée dans le temps par cinq paramètres nécessaires à l'identification de son comportement pendant le traitement du processus, soit :

- $\mathbf{r_i}$: la date de réveil dans laquelle l'opération peut commencer sa première réalisation ;
- $\mathbf{d_i}$: la date d'échéance dont le surpassement engendre un écart temporel ;
- $\mathbf{S_i}$: la date de début de l'opération ;
- $\mathbf{C_i}$: la date de fin de l'opération ;
- $\mathbf{t_i}$: le temps opératoire de l'opération.

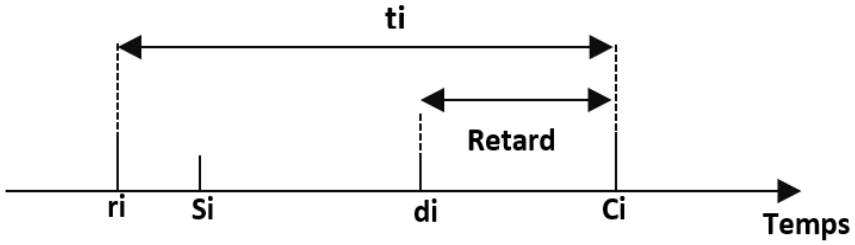


Figure 1.1 : Caractéristiques temporelles d'une tâche

Par ailleurs, la réalisation des opérations peut être arrêtée, il s'agit **d'opérations morcelables ou préemptibles**. Dans ces conditions, le traitement de l'opération peut être interrompu et relancé postérieurement soit totalement ou partiellement. Dans d'autre cas, l'exécution ne peut être interrompue qu'avant que le traitement ne soit complétement accompli, on parle alors **d'opérations non-morcelables ou non-préemptibles.**

1.2.3. Ressources

Les opérations nécessitent, pour leurs exécutions, un certain nombre de moyens matériels ou humains que l'on désigne « ressources ». Les ressources utilisées par les tâches peuvent être de natures diverses. On distingue **les ressources renouvelables** qui redeviennent disponibles en même quantité après avoir été affectées à une opération. Nous pouvons ainsi distinguer, **les ressources**

consommables qui s'épuisent globalement après avoir été affectées à une opération. Dans le cas des ressources restituables, deux types de ressources sont distingués soit :

- Les ressources disjonctives : qui ne peuvent traiter qu'une seule tâche à la fois ;
- Les ressources cumulatives : qui peuvent être utilisées par plusieurs opérations simultanément mais en nombre limité.

1.2.4. Contraintes

Une contrainte peut être définie comme un ensemble de conditions à satisfaire au cours de l'établissement de l'ordonnancement afin qu'il soit exécutable. Lorsque les contraintes sont relatives aux dates limites des opérations et décrivent des relations de précédences entre les différentes tâches, on parle de contraintes temporelles. Lorsqu'il s'agit de la disponibilité et la quantité des moyens utilisés, on parle plutôt des contraintes de ressources. Un recensement des différentes contraintes rencontrées dans la littérature pour la résolution des problèmes d'ordonnancement a été introduit par [T'KI, 2002].

1.2.5. Critères d'optimisation

Un critère correspond aux objectifs à optimiser afin de mesurer la performance de l'ordonnancement construit. Nous pouvons classer en critères réguliers et irréguliers [KAC, 2003].

- **Les critères réguliers :** ils sont dits réguliers car la valeur de la fonction objectif ne peut pas diminuer par l'insertion de temps mort entre deux tâches consécutives [HOO, 2005]. Ainsi, ils représentent des fonctions décroissantes des dates d'accomplissement des opérations. Parmi ces critères nous pouvons mentionner :

  - La minimisation des dates d'achèvement des actions ;
  - La minimisation du maximum des dates d'achèvement des actions ;
  - La minimisation de la moyenne des dates d'achèvement des actions ;
  - La minimisation des retards sur les dates d'achèvement des actions ;
  - La minimisation du maximum des retards absolus sur les dates d'achèvement des actions ;
  - La minimisation de la moyenne des retards sur les dates d'achèvement des actions ;
  - La minimisation du temps du cycle (dans le cas d'un ordonnancement cyclique).

- **Les critères irréguliers :** ils sont dits irréguliers car ils ne sont pas des fonctions monotones des dates de fin d'exécution des opérations. De plus, il est possible dans ce cas, de mettre au point le coût d'une solution en ajoutant un temps mort entre deux tâches consécutives, tels que :

  - La minimisation des encours ;
  - La minimisation du coût de stockage des matières premières ;
  - L'équilibrage des charges des machines ;
  - L'optimisation des changements d'outils.

1.3. Classification des problèmes d'ordonnancement d'atelier

Résoudre un problème d'ordonnancement d'un atelier consiste à trouver un ordonnancement optimal qui minimise une ou plusieurs mesures de performance données en respectant les contraintes techniques et temporelles prises en considération pour la résolution. Dans ces problèmes, l'ensemble des tâches qui doivent être traitées sur l'ensemble des ressources constitue un travail appelé '**job**'. Par ailleurs, le terme **machine** est utilisé désormais pour dénoter une ressource. Toutefois, en fonction de la nature de la ressource et l'ordre d'enchaînement des

opérations, plusieurs ateliers sont différenciés avec différentes extensions possibles pour chacun d'eux, à savoir :

- Ateliers à cheminement unique (flow-shop) ;
- Ateliers à cheminements multiples (job-shop) ;
- Ateliers à cheminements libres (open-shop).

1.3.1. Ateliers à cheminement uniques (flow-shop)

Le flow-shop est un système de traitement dans lequel les étapes de transformation sont identiques pour tous les produits fabriqués (voir figure 1.2). Dans ces ateliers appelés aussi ateliers à cheminements uniques, un ensemble de jobs doit être traité par un ensemble de machine disposé en série ; Tous les jobs passent sur toutes les machines dans le même ordre avec des durées opératoires différentes.

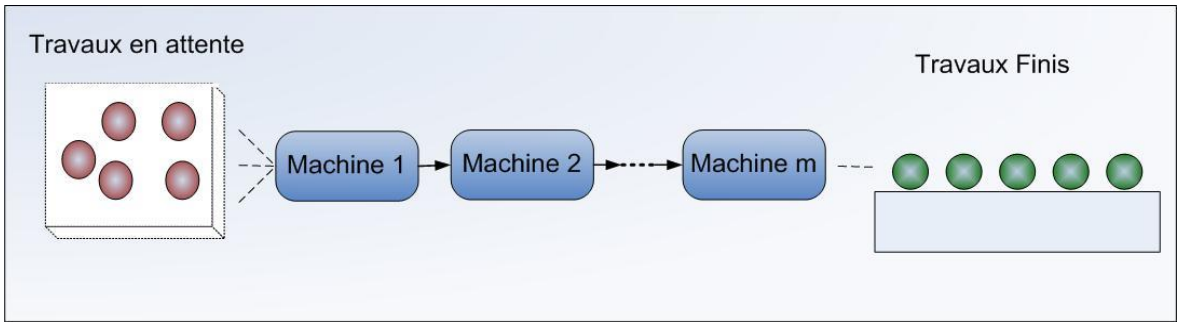


Figure 1.2 : Représentation d'un atelier de type flow-shop

1.3.2. Ateliers à cheminement multiples (job-shop)

Appelés aussi ateliers à cheminements multiples, dans ce type d'ateliers, chaque job possède son propre mode de passage sur les machines comme le montre la figure 1.3. Autrement dit, l'atelier consiste à ordonnancer un ensemble de jobs selon un ordre bien déterminé, variant selon la tâche à exécuter.

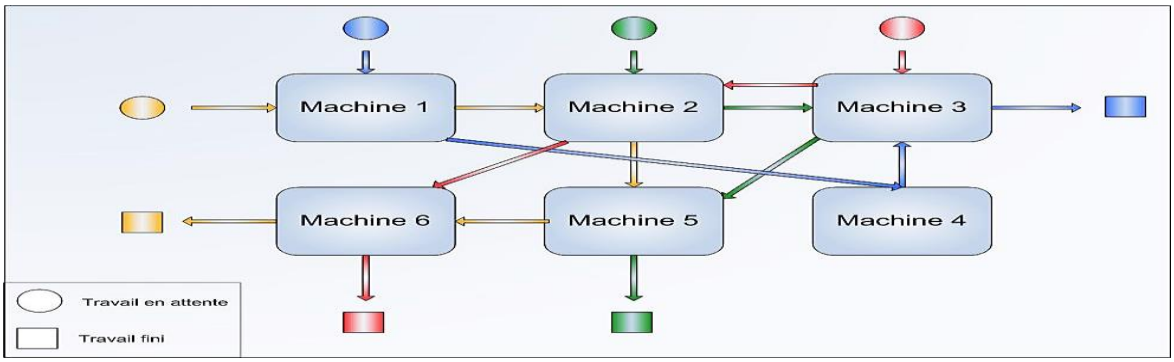


Figure 1.3 : Représentation d'un atelier de type job-shop

1.3.3. Ateliers à cheminement libres (open-shop)

Comparés aux autres types d'ateliers, l'open shop est le moins forcé, étant donné l'absence des contraintes d'enchaînement prédéfinies. Dans cet atelier, dit à cheminement libre, la gamme opératoire n'est pas fixée à l'avance et peut donc être différente d'une solution proposée du problème à une autre. Il convient de signaler que l'augmentation de la flexibilité des systèmes de production passe par la mise en place d'équipements multifonctions aisément adaptables. Dans cette optique, la flexibilité ou la modification du volume de fabrication s'attache éventuellement à

la duplication des machines disponibles pour la réalisation d'une tâche. Dans le cas d'un atelier de type Flow-shop, plusieurs machines identiques sont potentiellement disponibles sur un ou plusieurs étages pour traiter un ensemble d'opérations. Le modèle résultant est appelé dans la littérature flow-shop hybride. Également pour les ateliers de type Job-shop et Open-Shop donnant lieu aux job-shop flexible et à l'open-shop généralisé, respectivement.

Au motif que la machine destinée au traitement d'une opération n'est pas fixée préalablement, ces problèmes portent une complexité supplémentaire comparativement aux problèmes sans flexibilité des ressources. Le schéma synoptique de la typologie des problèmes d'ordonnancement d'atelier exposés par de [MAC, 1993] est représenté par la figure 1.4.

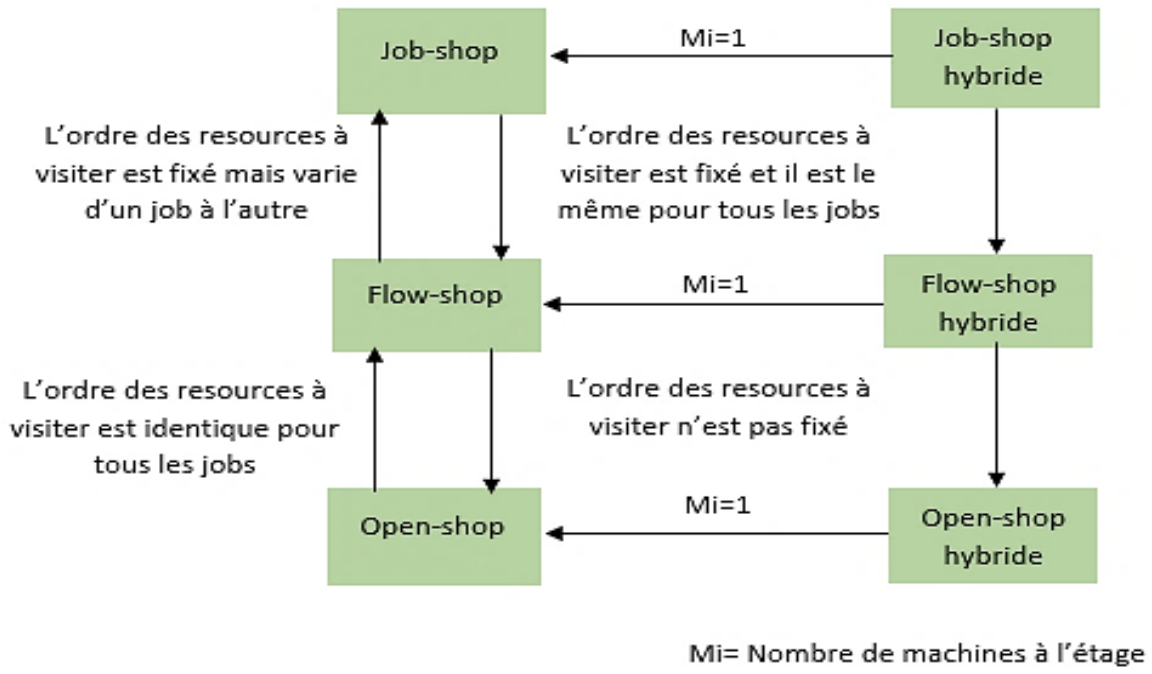


Figure 1.4 : Ordonnancement dans les différents types d'ateliers [MAC,93]

Une manière de simplifier les problèmes d'ordonnancement consiste à adopter une terminologie permettant d'identifier rapidement un problème d'ordonnancement donné. La notation la plus utilisée est celle suggérée par [GRA, 1979]. Elle est basée sur la concaténation de trois champs distincts **α|β|γ**.

Le champ α dépend de deux paramètres que nous pouvons représenter de la manière suivante : **α** = α1α2. Le premier paramètre α1 permet de décrire l'environnement manufacturier. Tandis que le deuxième α2 indique le nombre de machines. Le champ β possède une forme générique qui peut être notée ß=ß1ß2…ßn. Il permet de définir l'ensemble des contraintes concernant les jobs prises en compte. Les contraintes les plus répandues dans la littérature sont :

- **Perm** : Indique que les travaux sont traités dans chaque étape dans le même ordre. Elle est valable uniquement pour les problèmes de type Flow-shop où on ne considère que l'ensemble des ordonnancements de permutation ;
- **Prec** : Indique qu'il y a des contraintes de précédence entre les opérations ;
- $\mathbf{d_i}$: Signifie que les travaux ont des dates de fin au plus tard souhaitées ;
- $\mathbf{r_i}$: Indique que les travaux ont des dates de disponibilité différentes ;
- **Block** : Signifie que l'atelier dispose d'une zone de stockage de capacité limitée entre chaque machine, ce qui peut conduire à des blocages sur les machines ;

- **Snsd** : Indique qu'il y a un temps de réglage avant le traitement de chaque opération, ces temps dépendent de la séquence des opérations sur chaque machine ;
- **Batch** : Signifie que les opérations peuvent être regroupées en paquets lors de leurs exécutions.

Enfin, le champ **γ** correspond aux critères de performances à optimiser. Les critères les plus cruciaux permettant d'évaluer une solution d'un problème d'ordonnancement sont l'utilisation efficace des ressources, le délai total et le respect des dates d'échéances. Dans ce contexte, les variables qui interviennent le plus souvent dans l'expression de la fonction objectif en théorie d'ordonnancement sont :

- **La durée totale d'ordonnancement** : plus connue sous son appellation anglaise « Makespan ». Ce critère représente la date d'achèvement de la tâche la plus tardive C_J et il est défini par C_{max}= Max C_J ;
- **La somme des dates de fin d'exécution des travaux** : elle représente le temps total nécessaire pour la réalisation des tâches $\sum C_J$;
- **Le maximum des retards absolus** : défini par T_{max}= max T_J , tel que T_J= max (0, C_J-d_J) désigne le retard de la tâche j et d_J est la date de fin souhaitée ou encore la date d'échéance ;
- **Le maximum des retards absolus algébriques** : défini par L_{max}= Max L_J , tel que L_J= C_J-d_J représente le retard algébrique.

1.4. Problèmes d'optimisation et méthodes de résolution

Étant donné que les problèmes d'ordonnancement visent à trouver dans un ensemble discret une parmi les meilleures solutions réalisables, définie par une fonction objectif, en affectant des tâches aux ressources, ils sont considérés comme des problèmes d'optimisation combinatoire. Cependant, un problème d'optimisation combinatoire consiste à trouver un optimum d'une fonction objectif donnée en respectant les contraintes appliquées, faute de quoi la solution ne peut être réalisable. Cette nécessité d'optimiser se produit par l'importance d'assurer aux utilisateurs un système calibré répondant au mieux à leurs exigences. À la différence des problèmes d'optimisation qui cherchent à optimiser une seule fonction coût, ils existent des problèmes d'optimisation où plusieurs fonctions objectifs doivent être optimisées menant souvent à des situations dites contradictoires [ROY, 1993]. Un problème d'optimisation peut être définit comme suit :

$$\begin{cases} \boldsymbol{minimiser}\ \boldsymbol{f}(\vec{x}) \quad (\boldsymbol{fonction}\ \text{à}\ \boldsymbol{optimiser}) \\ \vec{g}(\vec{x}) \leq \mathbf{0} \quad (\boldsymbol{m\ contraintes\ d'in\acute{e}galit\acute{e}}) \\ \vec{h}(\vec{x}) = \mathbf{0} \quad\quad (\boldsymbol{p\ contraintes\ d'\acute{e}galit\acute{e}}) \end{cases} \tag{1}$$

Où $\vec{x} \in \mathbb{R}^n$, $\vec{g}(\vec{x}) \in \mathbb{R}^m$ et $\vec{h}(\vec{x}) \in \mathbb{R}^p$, $\vec{x}$ contient l'ensemble des variables de décision qui régissent la situation à modéliser, $\vec{g}(\vec{x})$ représente l'ensemble des m contraintes d'inégalité du problème et $\vec{h}(\vec{x})$ représente l'ensemble des p contraintes d'égalité du problème. A cet effet, trois types d'extrema peuvent être distingués : les extrema locaux faibles, forts et globaux. A titre d'illustration, un exemple d'une fonction à un minimum global : (M3), trois minima locaux faibles : (M2, M4 et M5) et trois minima locaux forts : (M1, M6 et M7), est présenté dans la figure 1.5.

- **Extrema global**

Un point $\vec{x^*}$ est un minimum global de la fonction f, si nous avons :

$$\boldsymbol{f}(\overrightarrow{\boldsymbol{x^*}}) < \boldsymbol{f}(\vec{\boldsymbol{x}}) \ \forall \ \vec{\boldsymbol{x}}, \text{ tel que } \overrightarrow{\boldsymbol{x^*}} \neq \vec{\boldsymbol{x}} \quad (2)$$

- **Extrema local faible**

Sachant que $V(\overrightarrow{x^*})$ représente un voisinage de $\overrightarrow{x^*}$, un point $\overrightarrow{x^*}$ est un minimum local de la fonction f, si nous avons :

$$\boldsymbol{f}(\overrightarrow{\boldsymbol{x^*}}) \leq \boldsymbol{f}(\vec{\boldsymbol{x}}) \ \forall \ \vec{\boldsymbol{x}} \in \boldsymbol{V}(\overrightarrow{\boldsymbol{x^*}}), \text{ tel que } \overrightarrow{\boldsymbol{x^*}} \neq \vec{\boldsymbol{x}} \quad (3)$$

- **Extrema local fort**

Sachant que $V(\overrightarrow{x^*})$ représente un voisinage de $\overrightarrow{x^*}$, un point $\overrightarrow{x^*}$ est un minimum local fort de la fonction f, si nous avons :

$$\boldsymbol{f}(\overrightarrow{\boldsymbol{x^*}}) < \boldsymbol{f}(\vec{\boldsymbol{x}}) \ \forall \ \vec{\boldsymbol{x}} \in \boldsymbol{V}(\overrightarrow{\boldsymbol{x^*}}), \text{ tel que } \overrightarrow{\boldsymbol{x^*}} \neq \vec{\boldsymbol{x}} \quad (4)$$

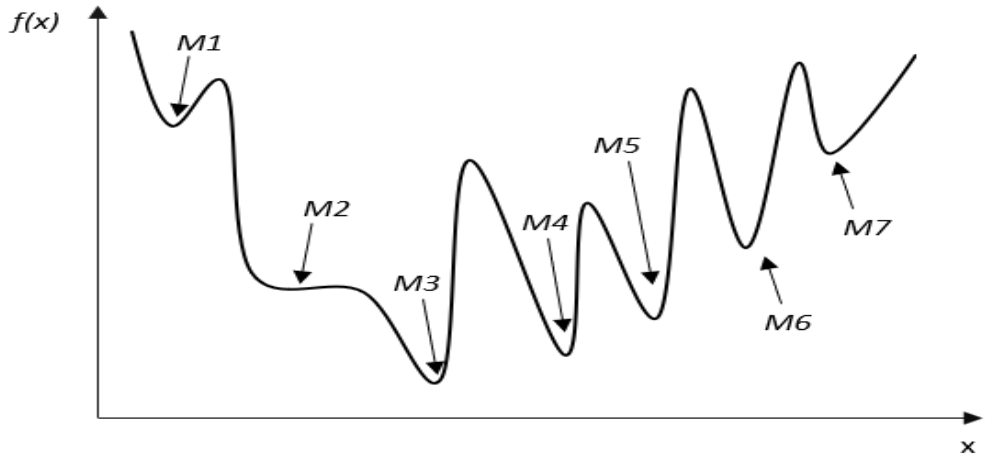


Figure 1.5 : Fonction à différent minima

1.4.1. Problèmes d'optimisation mono-objectif

Les problèmes mono-objectifs sont définis par une seule fonction économique. La formulation (1) correspond considérablement à la définition d'un problème d'optimisation mono-objectif.

1.4.2. Problèmes d'optimisation multi-objectifs

Les problèmes multi-objectifs existent quand un compromis entre plusieurs objectifs contradictoires est à rechercher. Mathématiquement, un problème d'optimisation multi-objectif peut être présenté de la manière suivante :

$$\begin{cases} \boldsymbol{minimiser}\ \vec{\boldsymbol{f}}(\vec{\boldsymbol{x}}) \ \begin{pmatrix} \boldsymbol{fonction} \text{ à } \boldsymbol{optimiser\ regroupant} \\ \boldsymbol{k\ fonctions\ objectifs} \end{pmatrix} \\ \vec{\boldsymbol{g}}(\vec{\boldsymbol{x}}) \leq \boldsymbol{0} \ \ (\boldsymbol{m\ contraintes\ d'in\acute{e}galit\acute{e}}) \\ \vec{\boldsymbol{h}}(\vec{\boldsymbol{x}}) = \boldsymbol{0} \quad (\boldsymbol{p\ contraintes\ d'\acute{e}galit\acute{e}}) \\ O\grave{u}\ \vec{\boldsymbol{x}} \in \mathbb{R}^{\boldsymbol{n}},\ \vec{\boldsymbol{f}}(\vec{\boldsymbol{x}}) \in \mathbb{R}^{\boldsymbol{k}}, \vec{\boldsymbol{g}}(\vec{\boldsymbol{x}}) \in \mathbb{R}^{\boldsymbol{m}}\ \boldsymbol{et}\ \vec{\boldsymbol{h}}(\vec{\boldsymbol{x}}) \in \mathbb{R}^{\boldsymbol{p}} \end{cases} \quad (5)$$

De la même manière, $\vec{\mathbf{f}}(\vec{\mathbf{x}})$ représente la fonction à optimiser qui regroupe l'ensemble des k fonctions objectifs, $\vec{\mathbf{g}}(\vec{\mathbf{x}})$ représente l'ensemble des m contraintes d'inégalité du problème et $\vec{\mathbf{h}}(\vec{\mathbf{x}})$ représente l'ensemble des p contraintes d'égalité du problème, respectivement. Contrairement à la résolution d'un problème d'optimisation mono-objectif, la solution d'un problème multi-objectif

n'est pas unique, elle rapporte un ensemble de solutions non dominées, connues comme l'ensemble des solutions Pareto optimal.

1.4.3. Méthodes de résolution mono-objectif

Du fait que les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoires qui se rencontrent dès lors qu'il est nécessaire de programmer dans le temps l'allocation des tâches aux ressources, nombreux sont les travaux de recherche qui ont été publiés sur leurs approches de résolution, compte tenu des différentes contraintes de réalisation. La figure 1.6 regroupe l'ensemble des approches d'optimisation les plus connues, dont nous distinguons principalement :

- **Les méthodes exactes :** qui permettent de fournir une solution exacte selon une stratégie connue [SMI ,1967] ;
- **Les méthodes approchées :** qui visent à obtenir des solutions approchées ayant un écart raisonnable par rapport à la solution optimale [OSM, 1989]. Généralement, l'efficacité des méthodes d'optimisation mono-objectif, utilisées pour la résolution de ce type de problèmes, se rapporte à deux facteurs principaux étant la qualité des solutions fournies et le temps d'exécution. Cependant, étant donné que ces problèmes sont justifiés par une forte complexité théorique, il n'existe à ce jour aucune approche de résolution universelle qui permet de fournir une solution optimale en un temps polynomial ou sans que le temps d'exécution n'augmente exponentiellement avec la taille du problème.

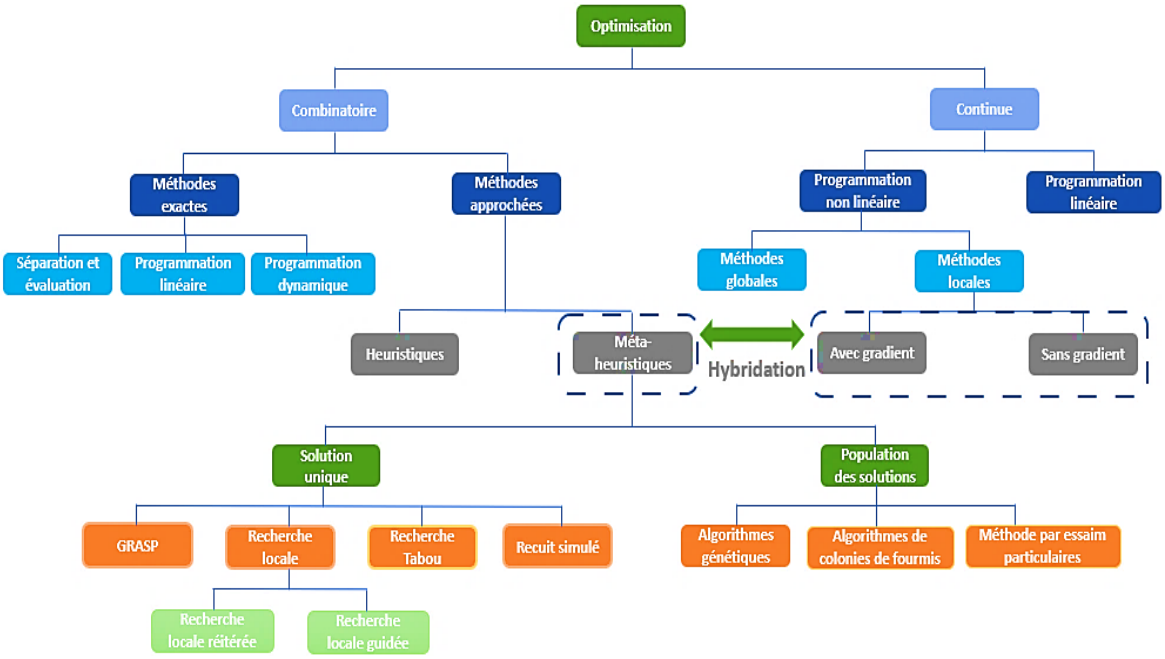


Figure 1.6 : Classification des méthodes d'optimisation mono-objectif

1.4.3.1. Méthodes exactes

Une méthode exacte peut être définie comme étant une application d'optimisation simple qui permet de fournir une solution optimale pour un problème d'optimisation combinatoire de taille finie. En examinant d'une manière implicite la totalité de l'espace de recherche, ces méthodes sont caractérisées par leur capacité de trouver l'optimum et de prouver son optimalité [PUC, 2005]. Parmi les méthodes exactes les plus utilisées dans la littérature, nous pouvons citer :

- **Méthode par séparation et évaluation (PSE)**

Connue sous son appellation anglaise « Branch and Bound » [LAN, 1960], cette méthode connait un très grand succès dans la littérature pour les problèmes d'optimisation combinatoires de petite taille. En explorant par énumération implicite l'ensemble de toutes les solutions, la méthode PSE repose sur une stratégie de diviser pour régner en se basant sur un arbre de recherche dans lequel les nœuds représentent les sous problèmes. Selon [COL, 2002], cet algorithme consiste à :

- Décomposer le problème en un ensemble de sous-problèmes de taille réduite ;
- Trouver une borne minimale (respectivement une borne maximale s'il s'agit d'un problème de maximisation) à chaque nœud de l'arborescence de recherche ;
- Eliminer les mauvais sous-problèmes ;
- Réeffectuer les étapes précédentes jusqu'à l'obtention de la solution optimale.

- **Programmation dynamique**

La programmation dynamique est basée sur le principe d'optimalité qui peut s'énoncer comme suit : « Si C'est un point qui appartient au chemin optimal entre A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C ». Développée par [BEL, 1954], cette approche d'optimisation se base sur des formules de récurrence visant à optimiser des critères quantitatifs représentant, à titre d'exemple, la longueur du chemin, le coût correspondant, le poids associé à ce chemin, etc. Pour les problèmes d'ordonnancement, cet outil est applicable uniquement si le problème est décomposable en étape et si le critère d'optimisation présente certaines propriétés particulières par exemple une forme additive ou multiplicative [LOP, 2001].

- **Programmation linéaire**

La programmation linéaire peut être définie comme l'outil d'optimisation le plus facile dans le monde de la recherche opérationnelle. Elle représente une technique mathématique d'optimisation de fonction à objectif linéaire sous des contraintes ayant la forme d'inéquations linéaires. Cet outil d'optimisation vise à sélectionner parmi différentes actions celle qui atteindra le plus probablement l'objectif visé. Parmi les algorithmes les plus importants pour le traitement d'un programme linéaire, nous citons la méthode du simplex. Ainsi, parmi les logiciels de résolution des problèmes de programmation linéaire, nous pouvons citer : CPLEX, LP-Solve et Mosel-Xpress.

1.4.3.2. Méthodes approchées

En dépit du progrès perpétuel de l'informatique, les praticiens dans les milieux industriels font toujours face à des problèmes de taille mémoire très importante, pour lesquelles les méthodes de résolution exactes peuvent garantir l'optimalité mais avec un temps de calcul qui risque d'augmenter exponentiellement avec la taille du problème. A cet effet, le but n'est plus alors d'atteindre méthodiquement la solution optimale mais plutôt d'aboutir à une solution de bonne qualité en un temps raisonnable. Partant de ce fait, les méthodes approchées, dites aussi d'approximation, représentent un dilemme intéressant pour résoudre les problèmes d'optimisation combinatoires de grande taille. Elles sont définies comme étant des implémentations permettant de trouver des solutions dont la qualité est garantie, en un temps de calcul aussi faible que possible. Et ce en exploitant généralement des processus aléatoires dans l'exploration de l'espace de recherche. Plusieurs approches de résolutions approchées ont vu le jour pour la résolution des problèmes d'ordonnancement [GOT, 1993] et [LOP, 2001]. Lorsqu'il s'agit des méthodes empiriques non fondées sur un modèle formel et qui sont développées pour un problème particulier, on les appelle méthodes constructives ou heuristiques. Ces algorithmes se basent sur des règles simplifiées pour optimiser un ou plusieurs critères de décisions déduits de la connaissance du problème. Parmi les heuristiques développées en théorie d'ordonnancement, nous pouvons distinguer selon [CHU, 1996] : FIFO (first in first out), SPT (Shortest Processing Time), LPT (Longest Processing Time), EDD (Earliest Due Date), SRPT (Shortest Remaining Processing Time) et ST (Slack Time).

Néanmoins, lorsque celles-ci sont adaptables à plusieurs types de problèmes d'optimisation combinatoires, on les appelle méthodes d'amélioration ou méta-heuristiques. Les métaheuristiques sont des algorithmes stochastiques qui permettent de générer une solution voisine en appliquant plusieurs fois et de façon différente, une petite transformation. Nous présentons dans ce qui suit, quelques méthodes métaheuristiques qui ont connu un grand succès en théorie d'ordonnancement.

- **Recherche locale**

Les métaheuristiques dites de recherche locale, ou de descente, sont très anciennes et doivent leur succès à leur rapidité et leur simplicité. A chaque pas de la recherche, ces méthodes progressent, à partir d'une solution courante, vers une solution voisine de meilleure qualité. Cette solution est acceptée si elle est meilleure que la solution courante et devient ainsi la solution courante. La recherche s'arrête lorsque toutes les solutions réalisables voisines sont moins bonnes que la solution courante. En d'autres termes, si aucune solution parmi les solutions voisines n'est strictement meilleure que la solution courante, nous arrêtons l'algorithme. Nous distinguons deux variantes de recherche locale soit :

- **La recherche locale réitérée** qui combine une procédure de recherche locale et des perturbations [LOU, 2003] ;

- **La recherche locale guidée** qui repose sur une modification dynamique de la fonction à optimiser en ajoutant une pénalité ou un ensemble de terme de pénalité, dans le but de rendre les minimas locaux déjà visités moins attractifs.

- **Recherche tabou**

La recherche tabou est une correction de la méthode de descente développée par [GLO,1998] pour s'échapper des minimas locaux. Grace à une liste taboue qui contient des mouvements ou des solutions qui sont temporairement interdits, cette approche repose sur un concept de mémoire de l'exploration des solutions pour garder les informations sur les solutions déjà visitées afin d'éviter le phénomène de cyclage entre plusieurs solutions. Ainsi, afin d'améliorer sa performance, la recherche tabou est souvent dotée d'un mécanisme d'échappement (appelé aussi mécanisme d'aspiration), de diversification et d'intensification.

- **Recuit simulé**

Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralisé à laquelle on introduit un paramètre de température qui est ajusté pendant la recherche [KIR, 1983]. Elle tire son origine des méthodes de simulation de Metropolis en mécanique statistique. De manière similaire, le principe de l'algorithme est itératif, il consiste à considérer une solution initiale, puis l'améliorer pas à pas en choisissant une nouvelle solution dans son voisinage. La nouvelle solution est acceptée selon la condition de Metropolis, qui se base sur une probabilité d'acceptation relative à la variable de température. Ce paramètre est fixé au début de l'algorithme et décroit tout au long du cycle d'optimisation.

- **Algorithmes génétiques**

Les algorithmes génétiques sont des outils d'optimisation stochastiques qui tirent leurs origines des concepts issus de la théorie de l'évolution de Darwin. Cette méthode dite évolutive, manipule un groupe de solutions admissibles à chacune des étapes du processus de recherche. L'ensemble de solutions est appelé population d'individus. Chaque individu ou chromosome représente une solution possible du problème donné. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles. Pendant le processus d'évolution, trois opérateurs fondamentaux interviennent : la recombinaison, la mutation et la sélection. La mutation et la recombinaison créent

de nouvelles solutions candidates, tandis que la sélection élimine les solutions candidates les moins prometteuses.

- **Algorithmes d'optimisation par essaim particulaires**

La méthode d'optimisation par essaim particulaire a été introduite en 1995 par [KEN, 1995], elle s'inspire des essaims d'insectes et leurs déplacements collectifs. Elle est basée sur une population d'individus, appelés particules, qui changent leur position avec le temps. Chaque particule est caractérisée par sa position courante et un vecteur de changement de position (appelé vélocité). Ainsi, chacune est dotée d'une mémoire qui permet de conserver la meilleure position rencontrée. En effet, le prochain mouvement est déterminé en fonction de la meilleure position stockée et du meilleur voisin au sens de l'essaim.

1.4.4. Méthodes de résolution multi-objectifs

La nature multi-objectifs des milieux industriels où les problèmes d'optimisation sont d'une complexité grandissante rend le développement des outils d'optimisation multi-objectifs crucial à la fois pour les chercheurs et les industriels impliqués dans ces systèmes. Cette importance académique, scientifique et industrielle résultante a fait naître un véritable axe de recherche. Partant de ce fait, les premiers travaux menés sur les problèmes multi-objectifs furent réalisés au 19^{éme} siècle sur des études en économie par Edgeworth et généralisés par Pareto. Leurs intérêts sont nés du fait que la plupart des problèmes d'optimisation réels sont décrits à l'aide de plusieurs critères souvent contradictoires devant être optimisés simultanément. Les problèmes multi-objectifs ont la particularité d'être beaucoup plus difficiles à traiter que leur équivalent mono-objectif. La difficulté réside dans l'absence d'une relation d'ordre total entre les solutions. En effet, la solution d'un problème multi-objectif est un ensemble de solutions compromis entre les différents objectifs à optimiser. L'ensemble des meilleures solutions est appelé le front Pareto. Cet ensemble de solutions constitue un équilibre, dans le sens qu'aucune amélioration ne peut être faite sur un objectif sans dégradation d'au moins un autre objectif. Résoudre un problème d'optimisation multi-objectif peut donc s'avérer long et fastidieux si des méthodes appropriées ne sont pas mises en œuvre. Comme illustré sur la figure 1.7, les approches de résolution des problèmes multi-objectifs peuvent être réparties en quatre classes suivantes :

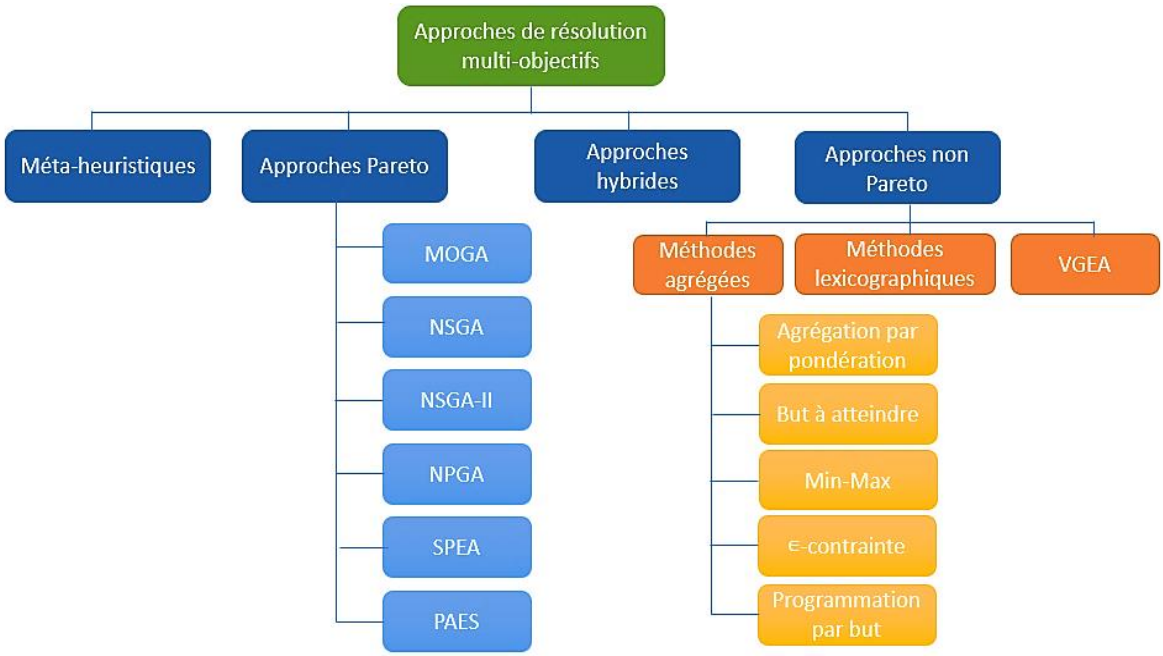


Figure 1.7 : Classification des méthodes d'optimisation multi-objectifs

1.4.4.1. Méthodes Pareto

Les approches Pareto sont des techniques d'optimisation basées sur la notion de la dominance au sens de Pareto. Ce concept d'optimalité développé par Goldberg pour résoudre les problèmes multi-objectifs permet de faire converger une population vers un ensemble de solutions efficaces. Selon ce mathématicien, la solution d'un problème économique multi-objectif peut être énoncé comme suit : « Dans un problème multi objectif, il existe un équilibre tel que l'on ne peut améliorer un critère sans détériorer au moins un des autres ». Cet équilibre est appelé optimum Pareto. Donc, une solution x est dite Pareto optimale si elle n'est dominée par aucune autre solution appartenant à l'espace réalisable X. Ces solutions sont appelées solutions non dominées ou non inférieures. Parmi les approches d'optimisation Pareto les plus répandues dans la littérature, nous citons : Algorithmes Génétiques à plusieurs objectifs (Multiple Objectives Genetic Algorithm (MOGA)), Algorithmes Génétiques de Tri non-Dominé (Non Dominated Sorting Genetic Algorithm (NSGA)), Algorithmes Génétiques de Tri non -Dominé-II (Non dominated Sorting Genetic Algorithm-II (NSGA-II)), (Strength Pareto Evolutionary Algorithm (SPEA)) et (Pareto Archived Evolution Strategy (PAES)).

1.4.4.2. Méthodes non Pareto

Cette famille de méthodes cherche à ramener le problème initial à un ou plusieurs problèmes mono-objectifs. Parmi ces approches nous citons :

- **Les méthodes agrégées** : Ces méthodes transforment un problème multi-objectif en un problème mono-objectif en utilisant soit un modèle additif ou un modèle multiplicatif. Les méthodes les plus répandues dans cette classe sont : la méthode de pondération (Weighting method), la méthode de programmation par but (Goal programming), la méthode du min-max, la méthode du but à atteindre (Goal Attainment) et la méthode du ε-contrainte.
- **Les méthodes non agrégées :** En général, les méthodes dites non agrégées et non Pareto possèdent un processus de recherche qui traite séparément les objectifs. [FOU,1985] ont proposé la méthode lexicographique dans laquelle les objectifs sont préalablement rangés par ordre d'importance par le décideur. Ensuite, l'optimum est

obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite. [SCH, 1985] propose une extension d'un algorithme génétique simple pour la résolution d'un problème multi-objectif dénommée Vector Evaluated Genetic Algorithm (VEGA). La seule différence avec un algorithme génétique simple est la manière dont la sélection s'effectue.

1.4.4.3. Méthodes hybrides

Afin d'améliorer l'efficacité d'un algorithme d'optimisation et palier à ses inconvénients, les chercheurs ont tendance, de nos jours, à combiner plusieurs heuristiques et méta-heuristiques. Un cas particulier de l'hybridation entre deux méthodes consiste à combiner un algorithme génétique avec une méthode de recherche locale. Dans une telle hybridation, on substitue souvent la mutation par une méthode de recherche locale. Dans le cas des problèmes multi-objectifs, nous pouvons citer les méthodes hybrides suivantes :

- La méthode MOTS combinant une population et une recherche tabou ;
- La méthode PSA combinant un algorithme génétique et le recuit simulé ;
- La méthode M-PAES intégrant un schéma généralisant l'implémentation d'un grand nombre d'algorithmes hybrides pour l'optimisation multi-objectifs.

Il convient d'ajouter que les métaheuristiques peuvent aussi résoudre les problèmes d'optimisation de nature multi-objectifs.

1.5. Problématique et orientation de l'activité de recherche

Étant donné que le choix des pistes de recherche à considérer est souvent dicté par des éventualités industrielles et académiques, la fonction ordonnancement s'approprie évidement un rôle impressionnant dans de nombreuses problématiques d'actualités, particulièrement en gestion de production des organisations de biens ou de services. L'enjeu économique considérable de ces problèmes ainsi que leurs champs d'applications qui n'ont pas cessé de s'améliorer et de se transformer, ont suscité depuis des décennies une étude intensive et approfondie. En d'autres termes, malgré que la résolution des problèmes d'ordonnancement a retenu l'attention de beaucoup de chercheurs et de praticiens dans les milieux industriels, beaucoup de questions restent à présent en suspens, à savoir :

Comment pouvons-nous parvenir à produire le plus possible ? Comment pouvons-nous parvenir à produire le plus rapidement possible avec le moins de coût possible ?, Comment pouvons-nous respecter les dates d'échéances imposées par le client ?, Comment pouvons-nous atteindre un niveau de gain considérable en tenant compte des contraintes réelles issues du monde industriel ? et finalement, laquelle des méthodes de résolution est adaptée à la résolution de ce type de problème ?, etc.

Ces questions et autres qui ont un sens très considérable dans le fonctionnement des systèmes de production et plus particulièrement des systèmes d'ordonnancement trouvent des solutions dans nos travaux de recherche. Dans ce contexte, dans le cadre de cette thèse, nous nous intéressons à la résolution du problème d'ordonnancement le plus général qui peut se présenter dans l'industrie. Il s'agit des problèmes d'ordonnancement de type flow-shop, caractérisés par un processus d'élaboration de produits linéaire. Toutefois, ce type d'industrie qui concerne de nombreuses activités notamment le textile, les industries pharmaceutiques et la fabrication de circuits imprimés.

1.5.1. Les systèmes de production de type flow-shop

Le flow-shop est un système de traitement dans lequel la séquence des tâches de chaque job $(J_1, \dots, J_n)$ est entièrement spécifiée [VAI, 2012]. Dans cet atelier, un ensemble de jobs doit être

traité par un ensemble de $(M_1, \ldots, M_m)$ machines disposées en série. Tous les jobs passent sur toutes les machines dans le même ordre. Ainsi, chaque job est composé de M opérations $(O_{k1}, \ldots, O_{km})$, chaque opération O_{ki} a besoin d'un temps d'exécution t_{ki}. Il convient d'ajouter ainsi que la préemption des tâches n'est pas autorisée, chaque job ne peut avoir qu'une seule opération en cours d'exécution et chaque machine ne peut effectuer qu'une seule opération à la fois. La figure 1.8 schématise le processus d'élaboration des produits dans un atelier de production de type flow-shop :

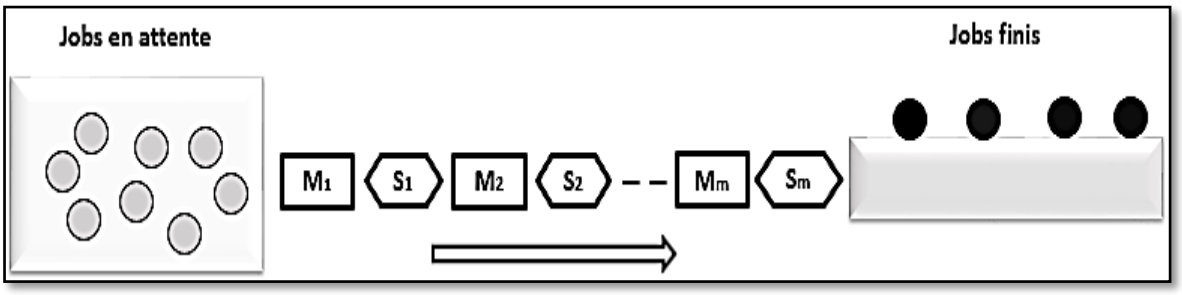


Figure 1.8 : Représentation d'un atelier de type flow-shop

1.5.2. Problèmes d'ordonnancement de type flow-shop : État de l'art

Mener à bien une recherche réclame la capacité de jongler avec plusieurs tâches allant de la découverte et l'évaluation des documents pertinents à la synthèse de l'information.

Dans la perspective d'établir une bibliographie générale et complète sur la résolution des problèmes d'ordonnancement des ateliers de type Flow-shop soumis simultanément à différentes contraintes, plusieurs bases de données ont été examinées notamment : DOAJ, Open J–Gate, IEEE Xplore, Science Direct, Web of Knowledge, Taylor et Francis Online, Springer, Elsevier, IOP science et Scopus. Cependant, comme toutes les bases de données bibliographiques reposent sur un langage documentaire qui fonctionne en mots-clés, le choix de ces mots-clés ou des descripteurs correspondants aux problèmes considérés reste une des étapes essentielles de la recherche. En effet, les mots-clés qui ont été utilisés sont : Flow-shop, ordonnancement, monocritère, multicritère, optimisation, mono-objectif, multi-objectif et contraintes. Outre que, plusieurs combinaisons des descripteurs précités ont également été élaborées afin de rendre les résultats plus pertinents.

La littérature retenue (357 documents) est répartie dans une bonne proportion en articles à caractère scientifique, en articles provenant de professionnels de l'industrie et en quelques thèses de doctorat. Une examination soigneuse nous a permis de trouver 73 documents. La figure 1.9 illustre l'évolution des publications des problèmes en question.

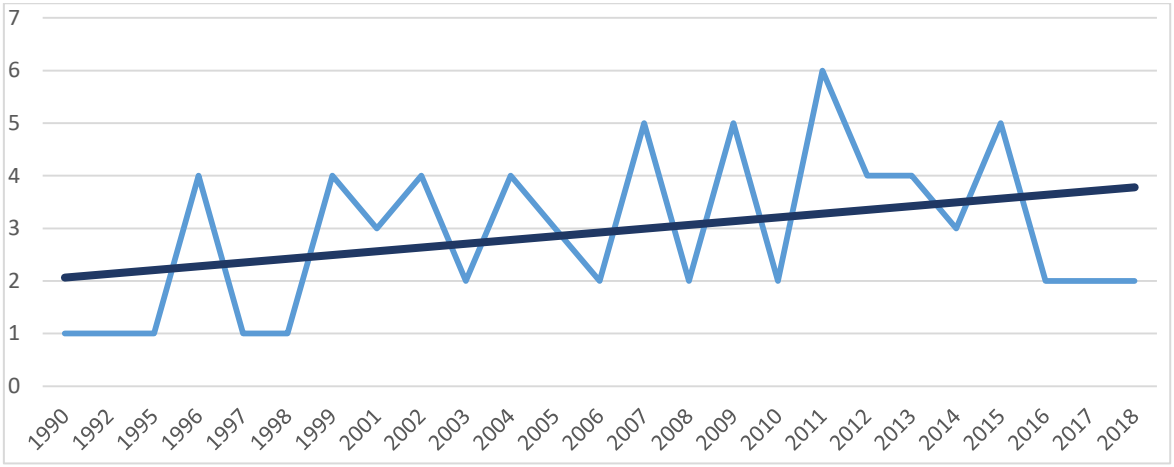


Figure 1.9 : Evolution des publications des problèmes de type flow-shop

La progression exponentielle des publications dans les dernières années révèle que la pertinence de la recherche dans l'optimisation des systèmes de production de type Flow-shop a gagné plus d'attention au fil des années. La prescience que les praticiens et les chercheurs de la théorie d'ordonnancement doivent revoir leurs approches de résolution a été fortifiée. Dans la section qui suit, nous présentons clairement cette principale évolution à travers un tableau (tableau 1.1) regroupant l'ensemble des recherches menées sur les problèmes d'ordonnancent d'atelier de type Flow-shop et des approches adoptées pour sa résolution dans une forme facilement accessible. Il convient d'ajouter que le tableau 1.2 regroupe l'ensemble des abréviations des méthodes exactes et approchées qui ont été utilisées.

Tableau 1.1 : Abréviation des différentes approches de résolution

Abbrèviations	Approches
B&B	Branch and Bound (Séparation et évaluation)
SA	Simulated Annealing (Recuit simulé)
GRASP	Greedy Randomized Adaptive Search Procedures (Algorithme glouton de recherche adaptative aléatoire)
TS	Tabu Search (Recherche tabou)
LS	Local Search (Recherche locale)
GA	Genetic Algorithm (Algorithme génétique)
ACO	Ant Colony Optimization (Colonies de fourmis)
PSO	Particle Swarm Optimization (Essaims particulaires)
IG	Iterated greedy (Algorithme glouton iteratif)
AA	(Recuit simulé hybride)
DDE	Discret differential evolution (Evolution différentielle discrete)
CPSO	Combinatorial particle swarm (Essaims particulaires hybrides)
ICG	Iterated cocktail greedy

Tableau 1. 2 : Synthèse des travaux traitant les problèmes de type flow-shop

Approches	Problèmes	Références
B&B	F2|permu, di|Cmax, Tmax	[DAN, 1990]
	F2||Cmax, ∑ Fi	[RAJ, 1992]
	F2||Cmax, Fi	[NAG, 1995]
	F2||(Cmax, ∑ Ui], Cmax	[LIA,1997]

	F2‖Cmax, Fi	[YEH, 1999]
	F2‖Cmax, $\sum$ Ci	[SAY, 1999]
	F2‖Cmax, F^w	[YEH, 2001]
	F2‖$\sum$ Fi, Ti	[LEE, 2001]
	F2 |Prec|$\sum$ Ti	[PAN, 2002]
	F2‖Cmax, $\sum$ Ci	[T'KI,2003]
	F2|Permu|Cmax	[HOU, 2003]
	F2|Permu|Cmax, Emax	[TOK, 2004]
	F2‖$\overline{T}$	[CHE, 2005]
	F2‖$\sum$ Ci , Cmax	[LIN, 2006]
	Fm|Permu|F^w, T^w, (F^w, T^w), (F^w, T^w, E^w)|	[MAD, 2009]
	F1|ri|C^w	[NES, 2012]
	F2|ri|$\sum$ Ui	[ARD, 2013]
	Fm|block|$\sum$ Ci	[MOS ,2013]
	F2‖ Cmax	[LIU, 2013]
	F2|Permu|Cmax	[MOR, 2014]
	F2 |Snsd|$\sum$ Ci	[DET, 2015]
SA	Fm|Permu|Cmax, Tmax	[CHA, 1999]
	Fm|Permu|Cmax, Fi, $\sum$ Fi	[SUR, 2004]
	Fm|Permu|Cmax, Fi	[VAR, 2005]
	F2‖F^w, T^w	[MES, 2010]
	Fm|Permu|Cmax, Tmax	[KHA, 2011]
	Fm|Snsd|Cmax	[LIN,2011]
GRASP	Fm|Permu|Cmax, Tmax	[KHA, 2007]
	Fm|Permu|(Cmax, Tmax), (Cmax, Tmax, Fi)	[ARR, 2011]
	Fm|permu|T^w	[MOL, 2015]
TS	F2‖$\sum$ Ci , Cmax	[GUP, 1999]
	Fm|Permu|(Cmax, Tmax), (Cmax, $\sum$ Ti)	[ARM, 2004]
	Fm‖Cmax	[LIN, 2009]
LS	F2|Permu|(Cmax, Fi), (Cmax, F^w), (Cmax, T^w)	[GUP, 2002]
	Fm|Permu|Cmax, Tmax, Ci, Ti	[GEI, 2007]
	Fm|permu|C^w, T^w	[GEI, 2011]
	Fm|Permu|Cmax, $\sum$ Ci , (T^w, $\sum$ Ti)	[DUB, 2011]
	Fm‖$\sum$ Fi	[PAN, 2012]
	Fm|ri|Cmax	[REN, 2015]
	Fm|block|Cmax	[TAS, 2015]
GA	F2|Permu|Cmax, $\overline{F}$	[NAG, 1996]
	F2‖(Cmax, $\sum$ Fi)	[NEP, 1996]
	Fm‖Cmax, $\sum$ Fi, I	[SRI, 1996]
	Fm‖(Cmax, Tmax), (Cmax, Tmax, Fi)	[MUR, 1996]
	Fm‖(Cmax, Ti), (Cmax, Ti, Fi)	[ISH, 1998]
	Fm|Permu|Cmax, Fi, Ti	[MUR, 2001]
	Fm‖Cmax, $\sum$ Fi, $\sum$ Ti, Tmax	[CHA, 2003]
	Fm‖(Cmax, Tmax), (Cmax, $\sum$ Ti)	[ARR, 2005]
	Fm‖Cmax, $\sum$ Fi	[PAS, 2006]
	Fm‖Cmax, $\sum$ Fi	[YAN, 2007]
	Fm|permu|Cmax	[RAJ, 2009]
	Fm|permu|Cmax	[FRA, 2009]
	Fm|permu|Cmax	[PUG, 2014]
	F2|Permu|$\sum$ Ci	[SHI, 2015]
CPSO	Fm|block|Cmax	[MAN, 2016]
ICG	Fm|No_wait|Cmax	[LIN, 2016]
ACO	F2‖Cmax, $\sum$ Ci	[T'KI, 2002]

	Fm|Permu|Cmax, $\sum$ Fi	[RAJ, 2004]
	Fm||Cmax, $\sum$ Fi, MIT	[YAG, 2008]
	Fm|Permu|Cmax, (Cmax, $\sum$ Ti)	[LIN, 2008]
	F2||$\sum$ Ci	[HUA, 2009]
	Fm||Cmax, $\sum$ Fi	[YAG, 2010]
	Fm|Permu|Cmax, $\sum$ Fi	[RAB, 2011]
	Fm|Permu|Cmax	[AHM, 2012]
PSO	Fm|Permu|$\sum$ Fi	[LIA,2007]
	Fm|permu|C^w, T^w	[RAH, 2007]
	Fm|Permu, batch|Cmax	[DAM, 2012]
	Fm|Permu|Cmax	[MAR, 2013]
	Fm|Permu|Cmax, Tmax	[TSA, 2014]
	Fm|block|Cmax	[MAT, 2017]
AA	Fm|No_wait|Cmax, $\sum$ Ti	[ALI, 2017]
IG	Fm|Permu|Cmax	[RUB, 2019]
DDE	Fm|block|Cmax	[GUA, 2018]

1.5.3. Problèmes d'ordonnancement de type flow-shop : Classification

En examinant l'état de l'art relatif à l'environnement manufacturier , nous avons constaté que 63,01 % des travaux traitent le problème d'atelier flow-shop à m-machines, 35,61% établissent la résolution des problèmes flow-shop à 2-machines et seulement un article qui correspond à 1,36 % aborde le problème tenant compte d'une seule machine. La figure 1.10 montre l'utilisation des machines dans la littérature.

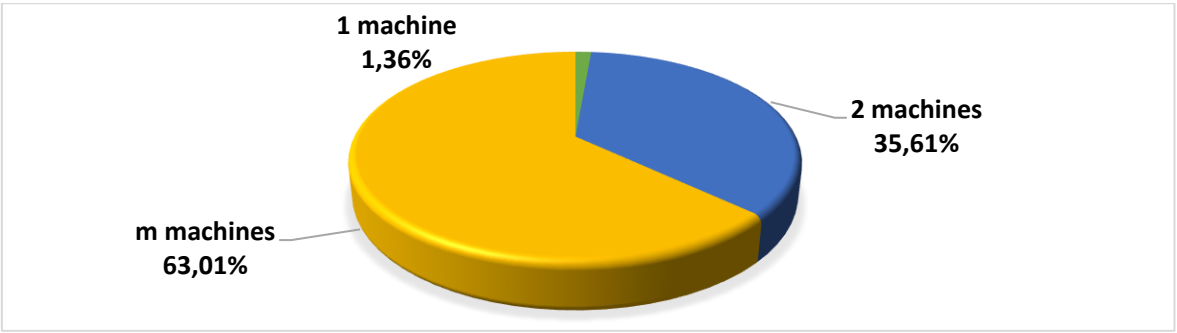


Figure 1.10 : Utilisation des machines dans la littérature

Relativement aux critères de performance adoptés dans la littérature, les critères liés au retard des travaux apparaissent dans trente-sept articles à savoir : le maximum des retards absolus Tmax se manifeste dans treize articles, le retard du job **Ti** apparait dans cinq documents et le total des retards $\sum$ **Ti** apparait dans sept articles. Le maximum des retards absolus algébriques **Lmax**, le retard algébrique pondéré $\mathbf{L^w}$ et le retard moyen $\overline{\mathbf{T}}$ ont été abordés chacun dans un seul article. Le retard total pondéré $\mathbf{T^w}$est aperçu dans neuf documents et le nombre des jobs en retard $\sum$ **Ui** dans deux travaux. Vient ensuite, les critères liés au temps de réalisation du job, notamment : le makespan **Cmax** qui représente l'un des critères de performance le plus souvent adopté, il se manifeste dans soixante-six documents. Le temps de réalisation du travail **Ci** a été développé dans un seul article, le temps total de réalisation du travail $\sum$ **Ci** est perçu dans onze articles et le temps total de réalisation du travail pondéré $\mathbf{C^w}$ a été repéré dans trois travaux. Alors que, le temps de séjour **Fi** est présent dans une bonne proportion d'articles de la littérature. Il a été ainsi développé dans huit articles. Le temps total de séjour $\sum$ **Fi** est apparu dans quinze, le temps moyen de séjour $\overline{\mathbf{F}}$ dans un seul article et le temps total de séjour pondéré $\mathbf{F^w}$ dans six travaux. Enfin, nous trouvons

le temps d'inactivité I et le temps total d'inactivité MIT qui sont présents chacun dans un seul article.

A propos des contraintes considérées dans la littérature, celle de la permutation est la plus retenue, Ce qui lui permet d'être remarquable dans trente-deux documents. De plus, nous pouvons citer celle de blocage qui a été perçue dans deux articles. La contrainte de précédence a été développée dans un seul article. Trois travaux ont traité le problème d'ordonnancement des ateliers flow-shop avec des dates de disponibilité des jobs différentes. En définitive, nous constatons que pour la contrainte batch, la contrainte du temps de réglage et celle qui concerne les dates de fin au plus tard souhaitées des travaux di, chacune est présente dans un seul article.

Au sujet des méthodes introduites pour la résolution des problèmes d'atelier de type flow-shop, il a été constaté que les méthodes exactes sont présentes dans 31,34% de la documentation trouvée et les méthodes approchées apparaissent dans 68,66%. La figure 1.11 montre le pourcentage de publications par type de méthodes de résolution adoptées pour les problèmes de type flow-shop.

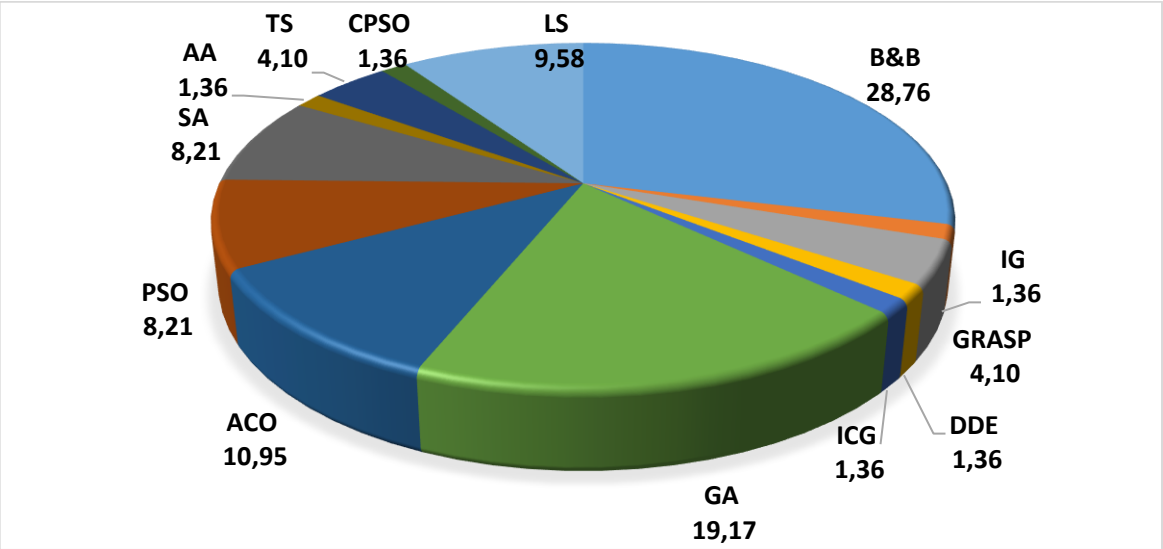


Figure 1.11 : Pourcentage de publications par type de méthodes de résolution adoptées pour les problèmes d'ordonnancement de type flow-shop

Pour les méthodes exactes, la méthode la plus répandue est celle de B&B ; notre analyse a permis de déceler qu'elle représente la quasi-totalité 28,76% du pourcentage global des méthodes précitées. Quant aux méthodes approchées, les métaheuristiques sont les plus développées et plus particulièrement la méthode des algorithmes génétiques qui représente une ration de 19,17%. Vient ensuite, la méthode des algorithmes de colonies de fourmis qui s'approprie la part de 10,95% et celle de la recherche locale 8,21%. Sans omettre de signaler que la méthode de recherche tabou et celle de descente, ainsi que la méthode du recuit simulé et celle des essaims particulaires requièrent le même pourcentage équivaut à 4,10%. 8,21 %, respectivement. Les deux quatre fragments des méthodes approchées se présentent comme suit : 1,36 % pour l'algorithme glouton itérative, le recuit simulé hybride, l'évolution différentielle discrète, l'essaim particulaire hybride et l'algorithme glouton itérative, respectivement.

1.5.4. Orientation de nos travaux et opportunités

Les résultats de l'analyse de la littérature confirment que, concernant l'environnement manufacturier, la majorité des travaux ont traité le problème à m-machines, étant donné qu'ils représentent les ateliers les plus répandus dans les situations industrielles rèelles. Ainsi, au sujet des critères de performances, le makespan Cmax et le maximum des retards absolus Tmax occupent la

grande part des critères abordés dans le développement des méthodes de résolution des problèmes précités. Également, la majorité des documents trouvés prennent en considération des contraintes de réalisation des jobs. Pour les problèmes de type flow-shop, la contrainte de permutation, la contrainte pour laquelle les travaux ont des dates de disponibilité différentes et celle de blocage tiennent les premiers rangs.

Il reste à savoir que les résultats enregistrés reflètent la réalité des problèmes d'ordonnancement rencontrés dans le monde industriel. Ainsi, la combinaison des différentes configurations précitées, qui tiennent une place historique importante, permettra de cerner la difficulté du problème au calcul d'une solution optimale. Dans cette perspective, notre premier axe de recherche consiste à développer des techniques de résolution plus efficaces pour les problèmes d'ordonnancement des ateliers de type flow-shop avec contrainte de blocage et de disponibilité des jobs. Notre contribution dans ce sens s'attache, d'une part, à une résolution mono-objectif qui minimise uniquement la date d'achèvement de tous les jobs et d'autre part, à une résolution multi-objectif qui minimise simultanément le makespan et le maximum des retards absolus. En se basant sur la codification à trois champs, notre intérêt portera sur la résolution du $F_m|block|C_{max}$ et $F_m|block|C_{max}, T_{max}$ et $F_m|block, r_k|C_{max}, T_{max}$ respectivement.

Dans un contexte particulier, le large pan de la littérature examinée pour résoudre les problèmes précités admet implicitement que la capacité de stockage entre les machines consécutives est infinie ; c'est le cas où il y a une préservation de l'ordre des jobs sur les machines initiales et terminales. Réellement, cette hypothèse est très rarement vérifiée en pratique. En effet, le flow-shop devient un flow-shop de blocage dans lequel l'ordre de passage séquentiel des opérations sur les ressources risque d'être bloqué si celles-ci n'admettent pas un espace de stockage suffisant pour conserver les opérations traitées.

Dans la même optique, notre second axe de recherche consiste à tenir en compte des dates de disponibilité des jobs. Ce dernier axe de recherche représente une initiative cruciale nécessitant des efforts considérables vu que pratiquement une machine ne peut pas exécuter une tâche à temps 0 avant l'instant T de sa disponibilité. Historiquement, la plupart des études menées dans ce sens négligent ce cas de figure, ce qui engendre lors du calcul de l'ordonnancement des lacunes et incohérences au moment de la mise en œuvre des séquences d'ordonnancement dans un atelier réel.

1.5.5. Flow-shop avec capacité de stockage limitée

Dans la perception traditionnelle des milieux industriels, la capacité de stockage entre les machines consécutives est prétendue être illimitée. Réellement, assurer une capacité de stockage infinie sur les terrains industriels engendre l'augmentation des coûts. Dans ces circonstances, il est important de chercher à les réduire voire possiblement les supprimer. D'ailleurs, pour faire face à ce problème, la majorité des praticiens trouve comme solution la restriction de leurs espaces de stockage afin d'accroitre la productivité en réduisant les coûts.

Limiter les espaces de stockage entre les machines consécutives représente un dilemme en théorie d'ordonnancement étant donné les blocages importants qui en résultent. En effet, ces blocages surviennent du fait que le job reste bloqué sur la machine tant que la machine suivante n'est pas disponible pour exécuter son opération. Ces temps de blocage engendrent une augmentation potentielle de la date d'achèvement finale des jobs et affectent par la suite la performance des processus de production.

Afin d'éclairer l'impact de la contrainte de blocage sur un système de production de type flow-shop, nous présentons sur les figures 1.12 et 1.13 un cas simple d'un flow-shop à 5 jobs et 4 machines avec contrainte de blocage et sans aucune contrainte. Les temps opératoires des quatre

jobs sont donnés dans la matrice suivante $t_{5,4}$. Nous signalons ainsi que l'absence de contrainte n'impliquent pas l'absence du temps mort.

$$t_{5,4}=\begin{pmatrix}5 & 4 & 4 & 3\\ 5 & 4 & 4 & 6\\ 3 & 2 & 3 & 3\\ 6 & 4 & 4 & 2\\ 3 & 4 & 1 & 5\end{pmatrix}$$

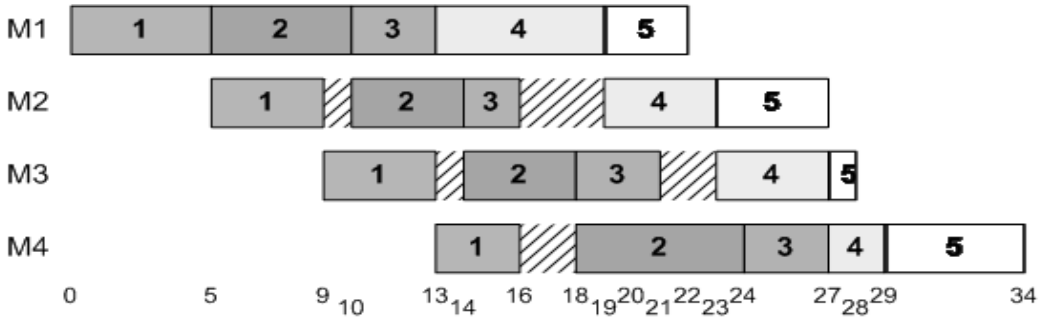


Figure 1.12 : Diagramme du Gantt pour le flow-shop sans aucune contrainte

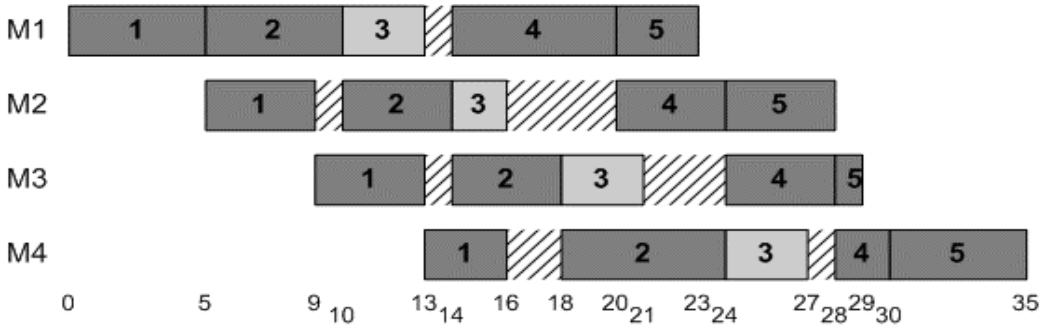


Figure 1.13 : Diagramme de Gantt pour le flow-shop avec blocage

Dans l'exemple de la figure 1.12 qui représente le diagramme de Gantt d'un flow-shop sans blocage, la machine M1 est disponible pour exécuter l'opération du job J4 dès que le job J3 termine son opération sur la machine M1. Le makespan correspondant est égal à 34 (unité temporelle)

Dans le cas où nous appliquons une contrainte de blocage entre les machines, pour le même exemple, le temps total d'exécution augmente d'une unité de temps. Cela est dû au ralentissement de l'exécution de la première opération du job J4 sur la machine M1. Dans ces conditions, le job J3 reste bloqué sur la machine M1 tant que la machine M2 n'est pas disponible pour traiter son opération. De plus, le job J4 ne peut pas débuter son opération sur la machine M1 avant que le job J3 passe sur la machine M2. Ce qui engendre un retard au niveau de la production.

1.5.6. Flow-shop avec contrainte de disponibilité des jobs

Sur la présentation du problème d'atelier flow-shop peuvent venir se greffer de nombreuses notions qui ont pratiquement des aspects très importants portant sur les applications des systèmes étudiés. La contrainte de disponibilité des tâches représente l'une des contraintes usuelles représentatives du monde industriel. Il s'agit de prendre en considération un temps avant lequel la tâche ne peut pas commencer son traitement sur la première machine. Ce temps contient plusieurs facteurs à savoir : le temps de transports ou de latence, le temps de réglage, le temps de montage, etc.

De la même maniéré, pour illustrer l'impact de la contrainte de disponibilité des jobs sur le système étudié, nous considérons l'exemple de la figure 1.14 en rajoutant uniquement les dates disponibilité de chaque job sur la première machine comme suit : $r_k = \{1, 3, 11, 8, 20\}$ pour k= 1, 2, 3, 4, 5.

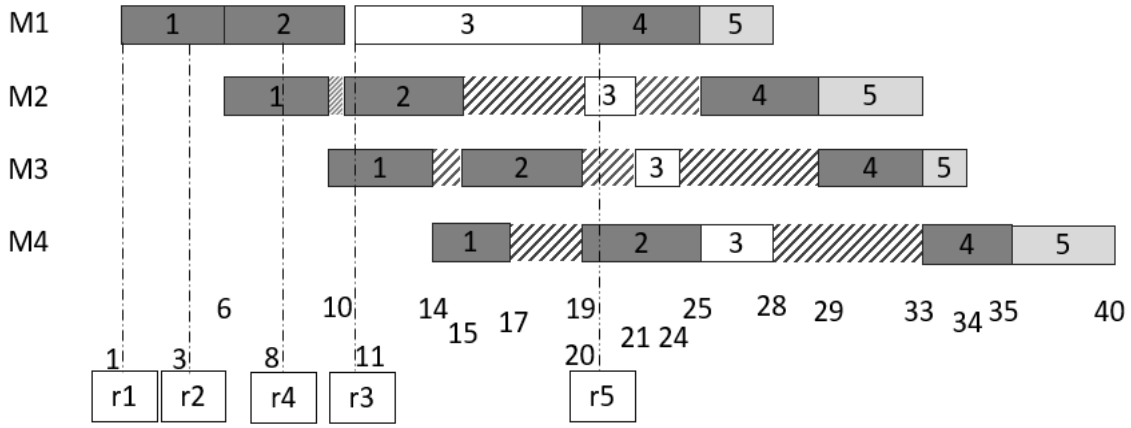


Figure 1.14 : Flow-shop avec contrainte de disponibilité des jobs

Il est bien clair que le temps total d'exécution a augmenté de 6 unités de temps. Cela est dû au retard de l'exécution des jobs J1 et J3 sur la première machine. En effet, ces derniers ne peuvent pas commencer leurs traitements sur la machine M1 avant leurs dates de disponibilité fixées préalablement.

1.6. Conclusions

Dans ce chapitre, nous avons rappelé les principales caractéristiques des systèmes d'ordonnancement d'ateliers, en passant en revue l'ensemble des critères, contraintes et éléments essentiels à la résolution introduits dans la littérature à ce jour. Nous nous sommes focalisés par la suite sur une classification des problèmes d'ordonnancement d'ateliers en fonction de l'ordre de passage et de l'organisation des systèmes d'une manière générale. Cette classification nous a mené à fixer notre atelier de recherche, dans cette optique, nous allons nous intéresser plus particulièrement à l'atelier de type flow-shop comme étant le plus général et intuitivement le plus répandu dans les secteurs industriels. En effet, cet atelier a suscité l'attention de plusieurs chercheurs en théorie d'ordonnancement, mais il n'existe à ce jour aucune étude exhaustive des différents problèmes posés par cette classe de systèmes.

En outre, nous avons abordé dans ce chapitre un large pan de méthodes de résolution mono et multi-objectifs mises au point en vue de résoudre les problèmes des systèmes précités. Deux classes sont distinguées pour les problèmes d'ordonnancement mono-objectif, la première permet de fournir une solution optimale dans un temps prohibitif, tandis que la deuxième qui connait un très grand succès dans la littérature permet d'obtenir une solution proche de l'optimum dans un temps d'exécution qui est raisonnable. De même pour les approches dites multi-objectifs, elles sont réparties en plusieurs classes, notamment les approches Pareto, les approches hybrides et les approches non-Pareto. Cette multiplicité d'outils de résolution peut s'expliquer par le grand gain scientifique porté pour ces systèmes, par la diversité des problèmes qu'ils suscitent ou encore par leur complexité et les défis qu'ils ne cessent de soulever.

Dans la dernière partie de ce chapitre, nous avons effectué une étude bibliographique générale menée d'une analyse comparative afin de fixer les perspectives de nos travaux de recherche. L'analyse des résultats de la littérature révèle particulièrement que la majorité des travaux ignorent

des aspects très importants dans le fonctionnement de cette classe de systèmes de production. Il s'agit en particulier de se placer dans le contexte où la modélisation et la résolution de l'ordonnancement sont soumises uniquement à un seul critère de performance et à une ou deux contraintes ; ce qui en réalité n'est pas toujours le cas. Nous nous plaçons dans un contexte industriel où un compromis entre plusieurs objectifs doit être trouvé et les opérations des jobs sont interrompues tout à la fois par un ensemble de contraintes. Dans cette optique, en se basant sur la codification à trois champs, notre intérêt de recherche portera sur la résolution des problèmes suivants : $F_m|block|C_{max}$, $F_m|block|C_{max}, T_{max}$ et le $F_m|block, r_k|C_{max}, T_{max}$ t, et ce en se basant sur plusieurs approches que nous allons développer dans les chapitres suivants .

Dans ce contexte, le chapitre suivant entamera la première résolution mono-objectif du système flow-shop qui vise à minimiser le makespan avec contrainte de blocage. Dans cette partie de recherche, plusieurs types d'approches de résolution seront développés pour résoudre efficacement le système mono-objectif étudié, notamment, des approches de résolution exactes, approchées et hybrides.

Chapitre II

Résolution mono-objectif des problèmes d'ordonnancement d'ateliers flow-shop avec contrainte de blocage

Chapitre 2
Résolution mono-objectif des problèmes d'ordonnancement d'ateliers flow-shop avec contrainte de blocage

2.1. Introduction

Dans le chapitre précédent, structuré en une étude bibliographique complète, nous avons dressé un état de l'art et tiré quelques perspectives qui nous semblait pertinent à suivre. Le présent chapitre est dédié, en grande partie, à la résolution du problème d'ordonnancement d'atelier flow-shop mono-objectif avec contrainte de blocage et ayant pour objectif central la minimisation du makespan. Ce chapitre s'articule autour de trois grandes parties consacrées à trois types de résolution, notamment, la résolution exacte à travers un logiciel de résolution des problèmes de programmation linéaire nommé CPLEX, la résolution approchée à travers la recherche génétique et la résolution approchée à travers l'hybridation séquentielle de la recherche génétique et le recuit simulé.

- Dans la première partie, une résolution exacte du modèle mono-objectif avec contrainte de blocage est effectuée afin d'évaluer la performance du modèle et d'étudier de plus près la complexité du système considéré ;
- Dans la deuxième partie, un algorithme génétique est proposé et calibré à travers les plans d'expériences de Taguchi. Le choix de cette métaheuristique évolutionnaire est dicté par son succès prouvé dans la résolution des problèmes d'optimisation combinatoire. Les résultats obtenus par cette même approche dans le traitement des problèmes d'ordonnancement de type flow-shop avec contraintes de blocage mixtes [TRA, 2013] nous ont également encouragé à l'utiliser pour la résolution du même problème avec la contrainte RSB.

La troisième partie est dédiée principalement à la combinaison des constituants de différentes métaheuristiques comme étant actuellement l'une des tendances les plus fructueuses dans le domaine de l'optimisation. La principale incitation à l'hybridation de ces algorithmes est de combiner les qualités et les avantages des différentes approches d'optimisation. Dans cette étude, en combinant la capacité d'exploration des algorithmes génétiques et le pouvoir d'échapper à l'optimum local du recuit simulé, un algorithme nommé Simulated Annealing Genetic Algorithm (SAGA) est suggéré afin de résoudre le problème d'ordonnancement de type flow-shop avec capacité de stockage limitée en minimisant potentiellement le makespan.

2.2. Éventail des approches appliquées sur les ateliers flow-shop avec contrainte de blocage

La résolution des problèmes d'ordonnancement a connu une impulsion très importante ces dernières décennies, tant par l'utilisation des heuristiques d'amélioration que par l'emploi des heuristiques constructives. Parmi les heuristiques constructives développées pour résoudre les problèmes d'ordonnancement d'atelier de type flow-shop avec contrainte de blocage, [MCC, 1989] ont proposé une heuristique innovante, connue sous le nom de Profile Fitting (PF) permettant de minimiser le temps de cycle dans une chaîne de montage constituée de m-machines disposées en série dans lesquelles la contrainte de blocage domine fortement. [LEI, 1990] ont suggéré une autre

technique constructive pour aborder le même problème avec et sans permutation, afin d'augmenter l'exploitation des stocks et réduire les temps de blocage entre les machines.

Or, il a été conclu par la suite que l'heuristique développée ne permet pas de fournir des solutions meilleures que celles fournies par l'heuristique proposée initialement par [NAW, 1983] pour le flow-shop traditionnel. [RON, 2004] ont proposé trois heuristiques constructives, connues sous les noms MinMax (MM), (MME) qui représente la combinaison de MM avec NEH, et (PFE) qui combine respectivement l'heuristique PF avec NEH. Leurs résultats ont montré que les trois algorithmes surpassent considérablement l'heuristique NEH dans la résolution des problèmes qui dépassent 500 jobs et 20 machines. [Pan, 2010] ont proposé deux heuristiques constructives, à savoir : (wPF) et l'algorithme Pan-Wang (Pw) reposant sur la méthode PF. Ensuite, ils ont combiné la procédure NEH avec PF, wPF et Pw, donnant lieu aux PF-NEH (x), wPF-NEH(x) et Pw-NEH(x), où $x \in [1, n]$ représente le nombre de séquences à générer. Leurs résultats de calcul ont décelé que le Pw-NEH apporte des solutions prometteuses au problème considéré, par rapport aux algorithmes existants.

Relativement aux métaheuristiques [CAR, 2001] ont proposé un algorithme génétique (AG) en utilisant l'idée du ralentissement délibéré du traitement de certains opérateurs. Les solutions fournies ont indiqué que l'algorithme génétique donne des résultats très satisfaisants comparativement aux heuristiques d'amélioration présentées par [ABA, 2000]. [GRA, 2000] ont développé deux algorithmes de recherche tabou, appelés (TS) et (TS+M), et ils les ont comparés par la suite aux bornes supérieures rapportées par [RON, 2005]. Finalement, ils ont conclu que les algorithmes de recherche tabou surpassent l'approche Branch and Bound et l'algorithme génétique de Carrafa. [LIU, 2001] ont proposé un algorithme d'essaim de particules hybrides (HPSO) pour résoudre le problème d'atelier flow-shop avec une capacité de stockage entre les machines limitée. Un algorithme hybride d'évolution différentielle discrète (HDDE) a été suggéré par [WAN, 2010]. [RIB, 2011] ont introduit un algorithme itéré gourmand (IG) dont les résultats ont surpassé le HDDE et ont produit la meilleure solution innovante pour la suite de référence de [TAI, 1983]. [HAN, 2012] ont proposé un algorithme amélioré de colonie d'abeilles artificielles discrètes (IABC) pour résoudre efficacement le problème d'atelier flow-shop avec contrainte de blocage. Un algorithme discret de colonie d'abeilles artificielles combinant l'évolution différentielle (DE-ABC) a été présenté par [HAN, 2015]. [SAD, 2015] ont suggéré une recherche prioritaire d'une métaheuristique randomisée (Meta-RAPS) sur la base de l'algorithme NEH. Récemment, [HAN, 2016] ont proposé un algorithme modifié d'optimisation des mouches des fruits (MFFO) basé sur l'odeur et la vision. Les résultats de calcul du MFFO indiquent que l'algorithme proposé est supérieur aux algorithmes suggérés dans la littérature.

2.3. Formulation mathématique mono-objectif

Dans un problème $F_m|block|C_{max}$, un ensemble $(j_1, \dots, j_n)$ de n tâches indépendantes doit être exécuté séquentiellement sur un ensemble de m machines $(M_1, \dots, M_m)$ dans le même ordre en admettant que la capacité de stockage entre les machines est finie. Chaque tâche est composée de $(O_{j1}, \dots, O_{jm})$ opérations, où chaque opération O_{jk} a un temps de traitement entier positif. On suppose que les temps de réglage sont inclus dans les temps de traitement des opérations. Tous les jobs sont prêts à être traités au moment de la mise à zéro. Chaque machine ne peut effectuer qu'une seule opération à la fois et chaque job ne peut avoir qu'une seule opération en cours de réalisation simultanément. De plus, en raison de l'absence des stocks intermédiaires entre les machines, des situations connues dans la littérature et les applications industrielles par le blocage des systèmes de production apparaissent. Le blocage survient plus précisément quand la zone tampon de stockage

en aval d'une machine est pleine. Par conséquent, le job reste bloqué sur la machine en amont jusqu'à ce que de l'espace y soit libéré.

Cette restriction de blocage décrit l'une des contraintes de précédence, qui peut exister entre les différentes structures du système. Or, cette contrainte a une énorme importance tant au niveau

pratique qu'au niveau scientifique vue que la majorité des travaux de recherche se focalisent principalement sur la résolution des problèmes d'atelier flow-shop avec une capacité de stockage intermédiaire entre les machines non limitée.

Étant donné que la cible principale est de trouver la séquence Π optimale qui minimise la date d'achèvement finale nommée makespan en prenant en considèration la contrainte RSB, la date de départ $D_{(j),k}$ de chaque job sur chaque machine peut être calculée mathèmatiquement en utilisant les expressions récursives suivantes ou graphiquement en s'appuyant les graphes disjonctifs (Annexe 3):

$$S_{(j),0} = 0 \tag{6}$$

$$D_{(1),k} = D_{(1),k-1} + p_{(1),k} \qquad k = 1,2,.+..,m-1 \tag{7}$$

$$S_{(j),0} = D_{(j)-1,1} \quad j = 2,3,\dots,n \tag{8}$$

$$D_{(j),k} = max\ \left(D_{(j),k-1} + p_{(j),k}, D_{(j)-1,k+1}\right)\ k = 1,\dots,m\ j = 2,\dots,n \tag{9}$$

$$D_{(j),m} = D_{(j),m-1} + p_{(j),m} \qquad j = 1,2,\dots,n \tag{10}$$

Dans les équations récursives précitées, Π= (π1,π2,π3,..,πn) représente un ensemble de permutations des jobs. Où, $\pi_{(j)}$ est l'indice du job disposé à la J^{ème} position de Π, $D_{(1),k}$ représente la date de départ du job π(1) sur la machine k, $p_{(j),k}$ représente le temps opératoire de la tâche $\pi_{(1)}$ sur la machine k, $S_{(j),0}$indique la date de début du job $\pi_{(j)}$ sur la première machine. L'équation (7) calcule la date de départ du premier job sur la machine k. L'équation (9) calcule la date de départ du job $\pi_{(j)}$sur la machine k, et l'équation (10) calcule l'heure de départ du travail $\pi_{(j)}$ sur la dernière machine m. Selon [RON, 2001], la valeur du makespan peut alors être calculée comme suit :

$$C_{max}(\pi) = D_{(n),m} \tag{11}$$

Il convient d'ajouter que l'objectif principal de la résolution du problème $F_m|block|C_{max}$, est de trouver la meilleure séquence π^* dans l'ensemble de toutes les permutations possibles, de sorte que le date de réalisation du dernier travail dans la permutation soit optimale, soit :

$$C_{max}(\pi^*) \leq C_{max}(\pi) \quad \pi^*, \pi \in \Pi \tag{12}$$

2.4. Résolution exacte des ateliers flow-shop avec contrainte de blocage

2.4.1. Solveur CPLEX 12.7 : ILOG OPL

Cette partie est consacrée à l'analyse exacte de la performance du modèle linéaire correspondant au système étudié. Cependant, le modèle mathématique présenté dans la section précédente représente une programmation linéaire pour laquelle une résolution exacte à travers un logiciel commercial de programmation linéaire est nécessaire pour qualifier la complexité et l'état d'optimalité du problème considéré. En effet, plusieurs logiciels ont vu le jour pour résoudre les problèmes d'optimisation linéaires dont le plus performant disponible est le solveur CPLEX.

Développé initialement par l'équipe de Robert Bixby pour la résolution du problème de voyageurs de commerce de grande taille, le succès prouvé du CPLEX a engendré sa commercialisation par la société IBM depuis son acquisition de l'entreprise française ILOG en 2009 avec la version 6.0. La version la plus récente, à ce jour, est la version 12.8 dans laquelle un système permettant de paramétrer finement le fonctionnement du solveur dénommé « système générique » est développé. Aujourd'hui, IBM dispose de plusieurs plateformes configurables et extensibles d'optimisation parmi lesquelles une plateforme robuste dédiée uniquement aux applications d'ordonnancement a été créée.

L'objectif principal de cette plateforme réside dans l'aide à la décision des industriels grâce à des fonctions d'analyse par hypothèse, de gestion de scénarios et de collaboration. La figure 2.1 illustre les différents composants de la suite d'optimisation ILOG. Ce solveur de programmation linéaire repose sur deux stratégies d'utilisation, la première consiste à invoquer un interpréteur de commande dédié d'une manière interactive **(**CPLEX interactif), tandis que la seconde consiste à amener directement les fonctionnalités du moteur depuis son propre code, que ce soit du C, du C++ ou du Java.

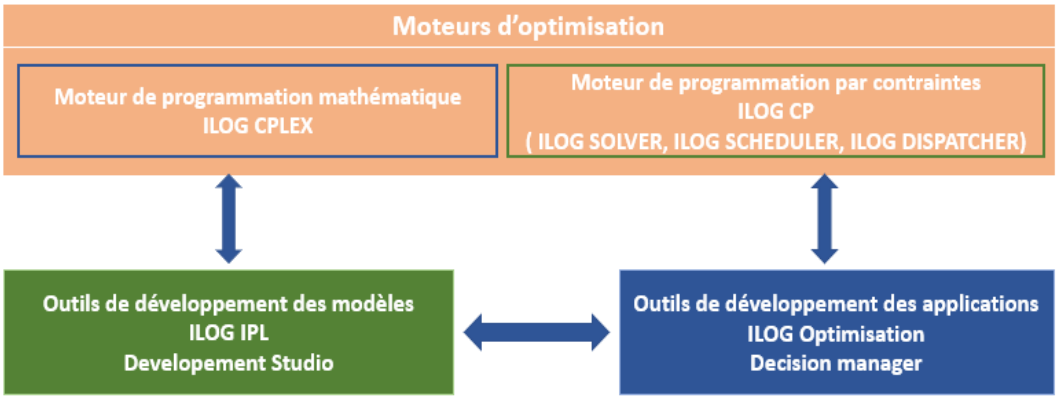


Figure 2.1 : Les composantes d'optimisation ILOG (Annexe 1)

Le solveur résout principalement les programmes linéaires et linéaires mixtes, les programmes quadratiques et quadratiques mixtes, les programmes avec contraintes quadratiques et avec contraintes quadratiques mixtes. Ces programmes sont éventuellement traités en se basant sur un large éventail d'implémentations, notamment, le simplexe primal, le simplexe dual, le simplexe de réseau, et finalement le simplexe combiné avec le Branch and Bound et la génération de coupes.

2.4.2. Résultats numériques

Afin d'évaluer la puissance de calcul du modèle proposé, il convient d'examiner le problème dans ses cas les plus difficiles ; en effet, 100 instances différentes pour chaque dimension fixée de n jobs et m machines ont été générées.

De surcroit, dans le plan d'expériences adopté pour le problème considéré, les temps opératoires ont été générés uniformément dans l'intervalle [1, 100]. Selon [BAK, 2010], la génération des durées opératoires entre 1 et 100 est basée sur deux raisons :

- La première raison est liée à l'uniformité historique, c'est-à-dire que la majorité des travaux de recherche portant sur les essais de calcul des modèles linéaires, y compris les problèmes d'ordonnancement, génèrent les temps opératoires à partir d'une distribution uniforme dans l'intervalle [1, 100] ;

- La deuxième raison est liée au fait qu'il est préférable d'utiliser des données représentatives des problèmes réels d'ordonnancement vu que la génération des petits intervalles permettra certainement d'aboutir très facilement à des solutions optimales. Or, ces solutions ne seront pas nécessairement réalistes en raison des conclusions inadéquates qui pourraient en résulter.

Les tableaux 2.1 et 2.2 rapportent les temps de calcul moyens obtenus en utilisant le logiciel CPLEX 12.6 pour trouver le makespan optimal des problèmes d'ordonnancement de type

flow-shop avec et sans contrainte de blocage RSB. Pour avoir plus de clairvoyance sur la capabilité du modèle précité, des études comparatives des temps de calcul moyens entre les problèmes de petite taille et de grande taille avec et sans blocage sont effectuées comme illustré sur les figures 2.2 et 2.3.

Le modèle a été codé dans IBM ILOG AMPL 12.6.0 et résolu à l'aide du solveur CPLEX 12.6.0 implémenté dans un ordinateur équipé d'un microprocesseur ayant 4,0 Go de mémoire vive (RAM). De plus, une limite de 3600 secondes a été imposée dans laquelle une solution optimale doit être fournie sinon le processus doit s'arrêter).

Tableau 2.1 : Temps moyens d'exécution par problème sans blocage (en secondes)

Jobs	**Machines**							
	5	**6**	**7**	**10**	**15**	**20**	**50**	**100**
5	0.03	0.04	0.06	0.11	0.13	0.19	0.33	0.52
6	0.05	0.12	0.10	0.17	0.21	0.23	0.45	1.12
7	0.11	0.15	0.16	0.28	0.33	0.49	0.68	2.14
8	0.13	0.18	0.19	0.30	0.73	1.05	3.62	9.36
9	0.20	0.20	0.56	0.67	2.07	3.15	11.58	48.03
10	0.37	0.51	0.98	1.99	5.14	7.65	52.27	292.12
11	0.57	1.02	1.87	4.63	19.01	30.27	305.47	1019.56
12	0.89	1.92	5.16	9.37	48.17	107.34	1594.12	4932.10

Tableau 2.2 : Temps moyens d'exécution par problème avec blocage RSB (en secondes)

Jobs	**Machines**							
	5	**6**	**7**	**10**	**15**	**20**	**50**	**100**
5	0.06	0.08	0.09	0.13	0.16	0.21	0.41	0.57
6	0.08	0.10	0.13	0.19	0.28	0.39	0.63	1.22
7	0.09	0.13	0.17	0.28	0.39	0.53	0.92	3.74
8	0.15	0.19	0.23	0.34	0.87	1.19	4.16	11.57
9	0.25	0.26	0.98	0.83	2.21	4.35	15.43	60.63
10	0.65	0.84	1.11	2.91	5.82	8.95	72.12	292.12
11	1.38	1.42	4.46	9.63	22.73	42.47	335.17	1227.93
12	6.84	9.12	18.38	49.37	81.17	123.34	1624.12	6127.04

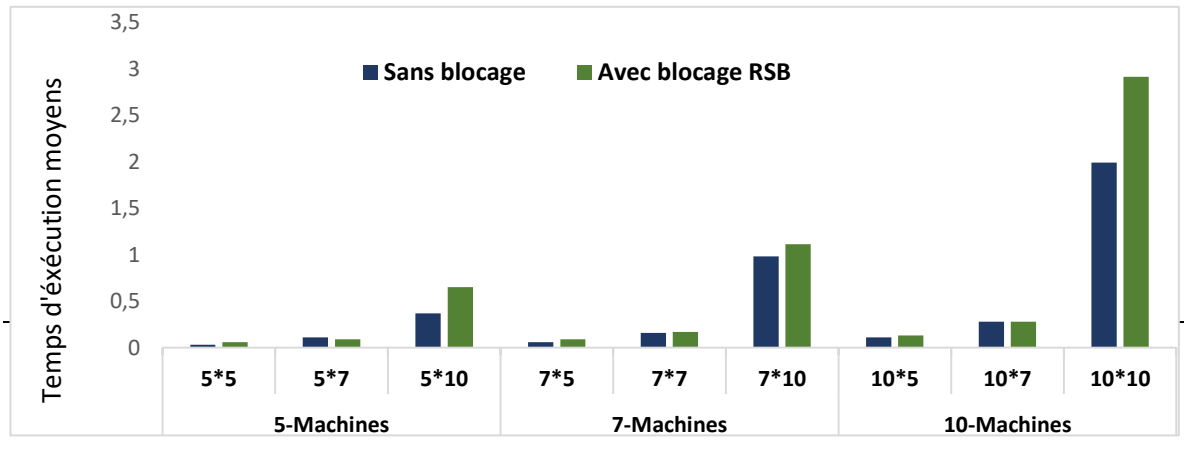


Figure 2.2 : Comparaison des temps moyens d'exécution pour les problèmes de petite taille

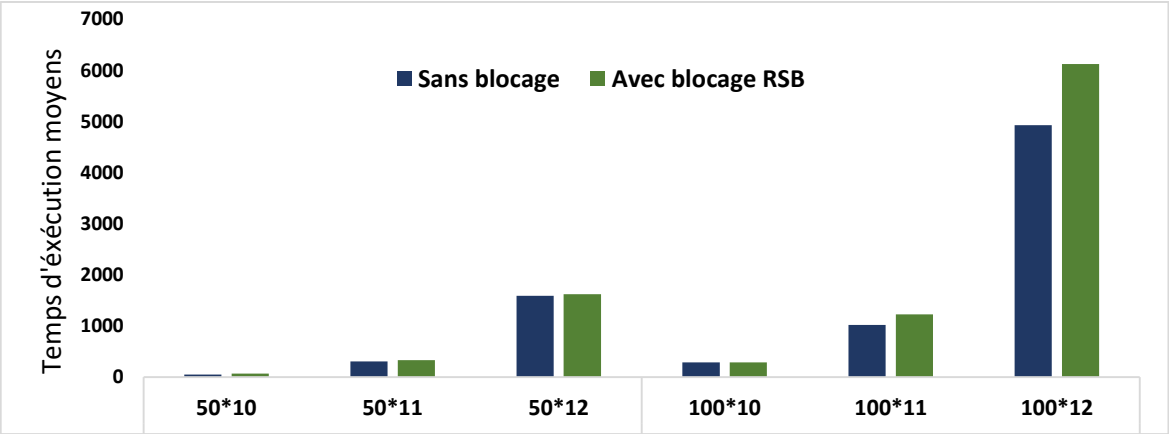


Figure 2.3 : Comparaison des temps moyens d'exécution pour les problèmes de grande taille

Relativement à tous les problèmes considérés, les temps moyens de calcul sont relativement raisonnables (moins de 50 secondes) du moment où la taille du problème est inférieure à 10*50 (10 jobs et 50 machines). Dès que la taille est supérieure à 10*100, les temps de calculs deviennent très importants. En plus, les temps moyens de calcul des problèmes avec blocage RSB sont relativement plus élevées comparativement aux temps d'exécution moyens sans contrainte de blocage. A titre d'exemple, pour le problème à 12 jobs et 100 machines, le temps moyen de calcul pour le cas sans blocage est estimé à une heure et demi alors qu'il est de plus de deux heures pour le cas du blocage RSB. Ce constat peut être expliqué par la complexité algorithmique des problèmes d'ordonnancement d'atelier flow-shop avec blocage prouvée par [HAL, 1996]. Ce dernier a montré que $F_m|block|C_{max}$ avec un nombre de machines m ≥3 est fortement NP-difficile (Annexe 2) en s'appuyant sur les résultats obtenus par [PAP, 1980]. Plus tard, [ALL, 2008] a montré que ce système est NP-difficile (Annexe 2) même avec m ≥2. Ces résultats montrent que l'usage des méthodes exactes ne pourra pas fournir des solutions optimales en un temps polynomial pour des problèmes de grandes taille issus des situations réelles. D'où l'intérêt de l'utilisation des méthodes approchées qui sont généralement employées lorsque les méthodes exactes échouent.

Les métaheuristiques sont parmi les méthodes approchées qui ont suscité depuis longtemps beaucoup d'attention et occupent jusqu'à présent une place considérable dans la littérature. Ces approches de haut niveau dites d'amélioration ont été conçues pour résoudre les problèmes de complexité grandissante (Annexe 2) dans lesquels ces outils d'optimisation ont largement prouvé leur efficacité. Elles ont certainement toujours été comptées parmi les approches les plus populaires pour traiter les problèmes d'optimisation combinatoires. Néanmoins, elles comportent un large éventail d'obstacles qui handicape considérablement leurs succès. Nous expliquerons dans la section suivante les différentes méthodes approchées adoptées en vue de minimiser le makespan dans un flow-shop avec contrainte de blocage.

2.5. Résolution approchée à travers les algorithmes génétiques

2.5.1. Introduction aux algorithmes génétiques

Initiés dans les années 1970 par John Holland [HOL, 1970], les algorithmes génétiques (AGs) sont des méthodes de recherche stochastiques qui permettent de résoudre un large éventail des problèmes d'optimisation combinatoire. Ils sont inspirés du mécanisme biologique de reproduction et de sélection naturelle. Fondés sur la durabilité de l'intuition la plus prometteuse, les AGs permettent de trouver efficacement une nouvelle génération avec une fonction coût meilleure. Ils

ont connu un très grand succès dans différents champs d'application, en finance [PER, 2000], en théorie du contrôle optimal [KRI, 1992], [MIC, 1992] et [MAR, 1996], et encore en théorie des jeux répétés

[AXE, 1987]. Ce grand succès s'explique par leur simplicité, leur performance connue en exploration et leur efficacité même pour les problèmes de complexité grandissante.

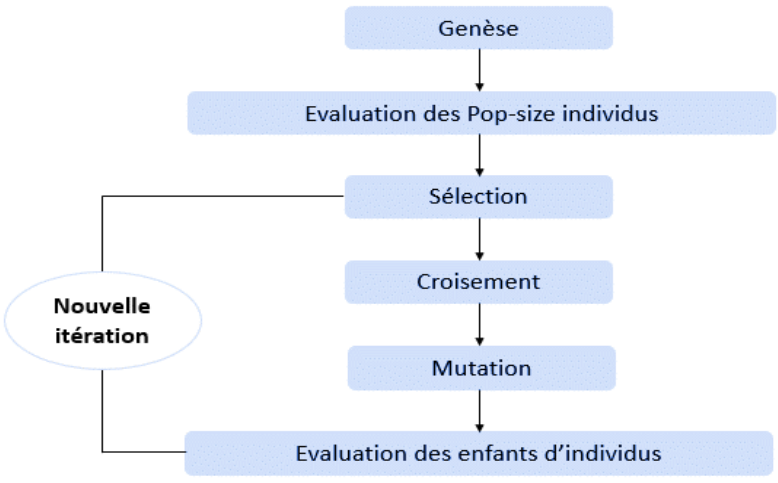
Partant du principe de base des AGs, le processus d'évolution est simulé à travers une population de solutions représentant des individus, chaque individu est doté d'un génotype constitué d'un ou plusieurs chromosomes. Ces derniers sont constitués d'un ensemble d'éléments, appelés "gènes", qui peuvent prendre plusieurs valeurs, appelées "allèles". Dans ces algorithmes dits évolutionnaires, trois opérateurs fondamentaux interviennent allant de la sélection qui éradique les solutions les moins prometteuses aux croisement et mutation qui font naître de nouvelles solutions concurrentes en explorant l'espace d'état. En outre, pour mettre en œuvre un algorithme génétique, il est nécessaire de disposer de quatre données qui correspondent pratiquement à la taille de la population, la probabilité de croisement, la probabilité de mutation et le nombre total de générations.

La force des algorithmes génétiques réside dans cinq caractéristiques principales faisant d'eux une approche d'optimisation très robuste relativement aux autres outils d'optimisation, à savoir :

- Le codage des variables de décision ;
- La possibilité de faire entrer en jeu un grand nombre de paramètres ;
- La manipulation d'une population de solutions à la fois au lieu d'une seule solution ;
- La capacité de faire évoluer l'espèce en héritant les caractéristiques avantageuses de ses ascendants ;
- L'évaluation directe sur la fonction objectif et non pas sur sa dérivée.

2.5.2. Procédure des algorithmes génétiques

L'algorithme génétique commence par une étape appelée genèse dans laquelle une génération de population initiale de taille Pop_Size d'individus est générée. Pour chaque individu généré, une fonction coût est calculée en vue de définir le score d'adaptation des individus lors du processus de sélection. Ces individus évoluent à travers l'application du croisement selon une probabilité Pc. Par la suite, les enfants obtenus subissent une inversion au niveau des gènes avec une probabilité de mutation Pm. Ces trois phases d'évolutions permettent avec une grande chance de produire une



nouvelle population meilleure que celle de la génération précédente.

Figure 2.4 : Mécanisme général des algorithmes génétiques

A chaque nouvelle génération, les nouvelles populations se renforcent et une boucle est effectuée tant que l'évaluation estime que la solution n'est pas encore optimale. La figure 2.4 présente le mécanisme général de la recherche génétique. Les principaux opérateurs seront présentés, en détail, dans les sections suivantes.

2.5.3. Genèse de la population

La genèse représente la première étape de la recherche génétique dans laquelle la génération aléatoire de la population Pop_Size d'individus initiaux est effectuée. Chaque individu est codé par un chromosome qui représente classiquement un point de l'espace de recherche. Habituellement, un chromosome est représenté sous forme de chaine de bits 0-1. Pour les problèmes d'ordonnancement, l'AG considère les séquences des jobs comme des solutions candidates ou des chromosomes. Chaque solution dans ce cas, est représentée par des nombres entiers égaux au nombre total de jobs à traiter [NAE, 2004]. Au final, le code génétique d'une solution attendue est composé de N job. La figure ci-dessous illustre un exemple de chromosome avec codage entier par permutation.

1	8	2	3	5	6	9	7	4

Figure 2.5 : Exemple de chromosome avec codage entier

2.5.4. Évaluation de la fonction objectif

Afin de mesurer la performance de chaque individu, la fonction objectif correspondante est calculée. Cette phase d'évaluation permet de déterminer la capacité d'un individu à persister en lui attribuant un poids nommé fitness. Le potentiel de chaque solution candidate donne une vision sur les plus prometteuses méritant d'être retenues dans la phase de sélection, et combinées et mutées dans la phase de la reproduction. Dans le cas de la résolution mono-objectif du problème d'ordonnancement d'atelier considéré, la fonction fitness consiste à minimiser le makespan.

$$C_{max}(\pi) = D_{(n),m} \tag{13}$$

2.5.5. Opérateur de sélection

Appelé aussi opérateur d'élimination, il constitue une heuristique décisive dans l'implémentation du mécanisme génétique permettant de diriger l'évolution et guider la recherche en déterminant les solutions candidates supposées être les plus prometteuses pour la génération des descendants. Différentes techniques de sélection ont vu le jour pour éliminer les chromosomes menant à des enfants de piètre qualité. Les plus adoptées entre elles sont :

- **Sélection par rang :** elle consiste à trier la population dans un ordre croissant ou décroissant, en fonction de la qualité (fonction fitness) des individus puis attribuer à chacun un rang ;
- **Sélection par roulette :** elle s'agit de former une roue de loterie biaisée pour laquelle on associe à chaque individu de la population un secteur d'une roue, dont l'angle est proportionnel à la qualité de l'individu qu'il représente ;

- **Sélection par tournoi :** elle consiste à choisir aléatoirement deux individus au hasard et sélectionner le meilleur en termes de performance en comparant leur fonction d'adaptation. Les individus qui participent à un tournoi sont remis ou sont retirés de la population, selon le choix de l'utilisateur. Cette technique a l'avantage de permettre à l'utilisateur de créer des tournois avec beaucoup de participants ou bien mettre en avant ceux qui gagnent les tournois, ce qui favorisera la pérennité de leurs gènes.

Étant donnée sa facilité d'implémentation et sa flexibilité, la sélection par tournoi est la technique de sélection utilisée dans la résolution envisagée.

2.5.6. Opérateur de croisement

Le croisement tend à augmenter la force de la population courante en répartissant les individus aléatoirement en couples hermaphrodites. Cet opérateur favorise l'exploration de l'espace de recherche. En termes plus concrets, il permet de créer de nouvelles combinaisons des paramètres des composants de façon à former deux descendants possédant des caractéristiques issues des deux parents. Cet opérateur est exécuté avec une probabilité Pc, appelée probabilité de croisement. Après la sélection de deux individus de la population courante, un nombre aléatoire $\alpha \in [0, 1]$ est généré. Si $\alpha \leq Pc$, l'opération de combinaison des parents est effectuée. Les techniques de croisement les plus souvent utilisées dans la littérature sont :

- **Croisement standard en 1-point :** il consiste à choisir au hasard un point de coupure pour chaque parent, et à partir de ce point, il faut remplir les cases manquantes des enfants en échangeant la deuxième partie des parents dans l'ordre des gènes et en ne reprenant que les gènes non encore transmis. Un exemple d'illustration de ce type de croisement est présenté dans la figure 2.6.

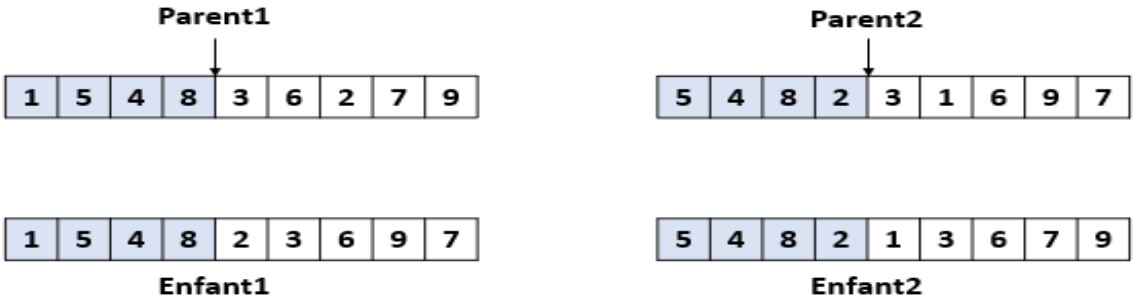


Figure 2.6 : Exemple illustratif du croisement standard en 1-point

- **Croisement standard en 2-points :** Dans ce cas, deux points de coupure sont choisis aléatoirement pour chaque ascendant. Les gènes en extrémités sont directement hérités des parents aux enfants. La partie manquante est complétée par la suite en recopiant de gauche à droite et dans le même ordre des gènes, les gènes résiduels du parent2 à l'enfant1 et ceux du parent1 à l'endant2, respectivement. La figure 2.7 présente un exemple illustratif de ce type de combinaison.

- En effet, nous avons utilisé le croisement standard en 2-points car il est généralement considéré comme le plus efficace. Ce dernier se mettra en exécution de la manière suivante (Algorithme 2.1) :

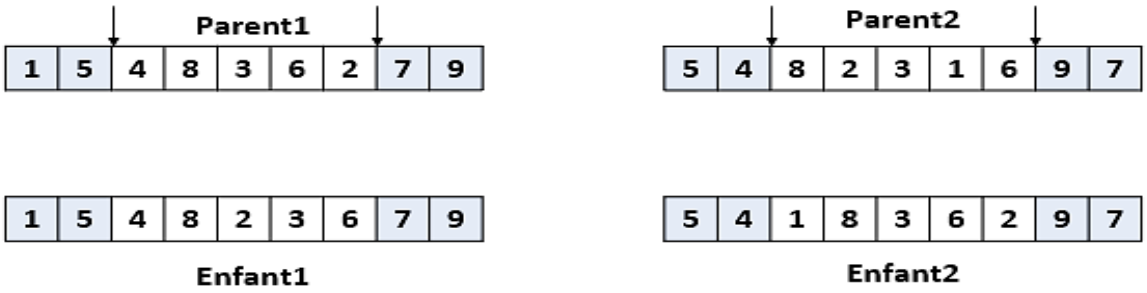


Figure 2.7 : Exemple illustratif du croisement standard en 2-points

Algorithme 2.1 : Procédure de croisement standard en 2-points

Étape 1. Choisir deux parents (P1et P2) de la population pour produire deux enfants (E1 et E2) ;

Étape 2. Choisir aléatoirement deux points de coupure C1 ∈ [1, ..., N] et C2 ∈ [1, ..., N] avec C1 < C2 ;

Étape 3. Générer α et définir la valeur de la probabilité de croisement Pc ;

Étape 4. Si α ≤ Pc :

- **Étape 4.1.** Les gènes en extrémités sont directement hérités des parents aux enfants ;
- **Étape 4.2.** La partie manquante est complétée par la suite en recopiant de gauche à droite et dans le même ordre des gènes, les gènes résiduels du P2 à E1 et ceux du P1 à E2, respectivement ;

Étape 5. Si α > Pc, hériter les gènes du parent P1 à l'enfant E1 et du parent P2 à l'enfant E2.

2.5.7. Opérateur de mutation

La mutation est le troisième opérateur utilisé dans le processus de la recherche génétique, elle est appliquée pour préserver la diversité génétique d'une génération à l'autre en permettant de générer des points dans des régions a priori sans intérêt. La mutation s'effectue avec une probabilité appelée probabilité de mutation Pm. Cet opérateur joue un rôle très important dans l'orientation de la recherche vers un optimum aussi bien global que local. Comme pour les autres opérateurs génétiques, plusieurs techniques de mutation ont été introduites dans la littérature [NAE, 2004]. Nous citons ici les plus utilisées :

- **Opérateur d'échange réciproque :** Il consiste à choisir deux gènes et de les inverser ;
- **Opérateur d'inversion simple :** Il s'agit de choisir au hasard deux points de coupure et permuter les positions des gènes placés au milieu ;

- **Opérateur d'insertion :** Il consiste à sélectionner aléatoirement un gène et une position dans l'individu à inverser, puis à insérer le gène sélectionné dans la position sélectionnée.

Dans ce travail de recherche, nous utilisons une mutation d'échange réciproque dans laquelle, un gène à une position est retiré et implanté à la position du deuxième gène choisit avec une probabilité de mutation Pm comme illustré sur la figure 2.8. En outre, les principales étapes du

Étape 3. Générer β et définir la valeur de la probabilité de mutation Pm ;

Étape 4. Si β ≤ Pm, un gène à une position est retiré et implanté à la position du deuxième gène choisi ;

Étape 5. Si α > Pm, conserver le même chromosome.

mécanisme de mutation utilisé sont décrites comme suit (Algorithme 2.2) :

Figure 2.8 : Exemple illustratif de mutation d'échange réciproque

2.5.8. Critère d'arrêt

Le critère d'arrêt de la recherche génétique occupe une place importante dans l'évaluation de la qualité des individus obtenus. Généralement, ils sont de deux types : le nombre de générations à exécuter, fixé a priori, et l'algorithme qui s'arrête lorsque la population cesse d'évoluer ou n'évolue plus suffisamment. Dans le cas du problème considéré, le critère d'arrêt adopté est le nombre de générations égal à Max_iter. Après Max_iter générations, l'algorithme génétique s'arrête et donne la meilleure séquence (chromosome) qui correspond à un makespan optimal.

Finalement, l'algorithme suivant décrit clairement les principales étapes de la recherche génétique pour la résolution du problème d'ordonnancement d'un atelier de type flow-shop avec contrainte de blocage. Cette recherche est initialisée par une taille de population (Pop_Size), une probabilité de croisement (Pc), une probabilité de mutation (Pm), et par un critère d'arrêt (Max_iter). Sachant que α et β sont deux valeurs générées aléatoirement dans l'intervalle [0,1]. Les étapes de cet algorithme sont présentées dans l'algorithme 2.3.

Algorithme 2.3 : Algorithme génétique pour le $F_m|block|C_{max}$

Étape 1. Initialiser : Pop_Size, Pc, Pm, Max_iter ;

Étape 2. Choisir aléatoirement deux points de coupure C1 ∈ [1, ..., N] et C2 ∈ [1, ..., N] avec C1 < C2 ;

Étape 3. Calculer la fonction fitness 'Cmax' pour chaque séquence de la population courante ;

Étape 4. Sélectionner par tournoi la meilleure fitness g_oldbest de la population courante ;

Étape 5. **Tant que le nombre d'itérations maximal n'est pas atteint :**

Étape 6. i=1

Étape 7. Pour chaque séquence : de 1 à Pop_Size

Sélectionner aléatoirement deux séquences de P(N) aléatoirement

Si $\alpha \leq Pc$:

Appliquer le croisement standard en 2-points et produire deux enfants

Si $\beta \leq Pm$:

Appliquer la mutation réciproque

Copier les ascendants obtenus dans la nouvelle population P (n+1)

Evaluer le makespan de tous les chromosomes de la nouvelle population g_i (P (n+1))

Étape 8. i =i+1

Étape 9. **Jusqu'à** i = Max-iter

Étape 10. **Fin tant que**

2.5.9. Calibrage des paramètres des AGs : Plan d'expériences de Taguchi

La force et l'efficacité des approches d'optimisation dépendent en grande partie de la sélection adéquate des paramètres. Dans cette perspective, une méthodologie de choix nommée 'les plans d'expériences' a été utilisée en vue d'optimiser les paramètres du processus génétique. Généralement, cette technique vise à identifier et à instaurer les liens existants entre des variables d'entrée du processus et une grandeur d'intérêt, appelée réponse. Plusieurs types de plans d'expériences ont vu le jour pour paramétrer et gérer l'impact des variables sur la variation de la réponse, ils diffèrent cependant par la manière d'aborder les facteurs considérés, leurs niveaux correspondants et l'interférence existante entre eux. La méthode Taguchi est l'une des méthodes de mise en œuvre des plans d'expériences la plus adoptée dans la littérature. Développée par Genichi Taguchi au Japon dans les années 1960 [TAG,1962], cette approche se base sur des plans optimaux qui permettent de fournir le maximum d'informations avec le minimum de tests, en explorant les facteurs qui influencent la moyenne et la variance de la performance du processus. En outre, cet outil, connu par sa robustesse, permet de minimiser l'impact du facteur de bruit qui ne peut être contrôlé par les concepteurs et de trouver le meilleur niveau des facteurs contrôlables, et ce, en utilisant un tas de notions dont le facteur contrôlé, la table orthogonale, le facteur bruit, le ratio signal-bruit, etc. Partant de ce fait, la démarche de mise en œuvre d'une expérimentation est schématisée comme suit :

- Définir les caractéristiques à mesurer et les modalités de leur mesure ;
- Sélectionner les facteurs et leurs valeurs à tester ;
- Sélectionner la matrice d'expériences à utiliser ;
- Réaliser les essais et mesurer les résultats ;
- Analyser les résultats et définir la configuration devant optimiser les performances du système envisagé ;
- Faire un essai de validation des résultats finals.

Dans un plan de Taguchi, le rapport signal bruit (S/B) représente un indicateur de variation utilisé pour mesurer la variation existante dans la réponse en question afin d'identifier les facteurs de contrôle qui réduisent la variabilité. Il existe trois catégories de caractéristiques de performance pour étudier le rapport signal/bruit : Préférer plus petit- Préférer plus nominal- Préférer plus grand. Comme la présente étude porte sur un problème de minimisation, le rapport signal/bruit doit être mesuré à l'aide de la valeur la plus faible, comme indiqué ci-dessous :

$$\frac{S}{N} = -10\, log_{10}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{y_i^2}\right) \quad (14)$$

Où, n et y représentent le nombre d'observations et la variable réponse, respectivement. Relativement à l'approche d'optimisation que l'on souhaite paramétrer, quatre facteurs, dont la taille de la population, le nombre maximum d'itérations, la probabilité de croisement et la probabilité de mutation sont définis comme étant des facteurs de contrôle critiques. Cette phase de recherche a été élaborée sur Minitab représentant la référence mondiale et incontournable pour les praticiens visant à optimiser et contrôler leurs processus. Dans cet ordre d'idées, 27 séries d'expériences ont été réalisées en s'appuyant sur la distribution de la table orthogonale, et trois niveaux ont été attribués à chaque facteur, comme l'illustre le tableau 2.3.

Tableau 2.3 : Différents niveaux de chaque paramètre des algorithmes génétiques

Facteurs	Pop_Size	Max-iter	PC	Pm
Niveau 1	100	200	40 %	10 %
Niveau 2	200	400	60 %	40 %
Niveau 3	400	600	90 %	70 %

La figure 2.9 représente un graphique des effets principaux crée sur Minitab en traçant la moyenne des caractéristiques pour chaque niveau de facteur. Par ailleurs, quant au calibrage de l'approche d'optimisation en question, les quatre facteurs considérés ci-dessus doivent être ajustés tels que : la taille de la population = 400, le nombre d'itérations = 400, la probabilité de mutation = 0,1 et la probabilité de croisement = 0,4. Le tableau 2.4 regroupe le niveau de chaque facteur sélectionné en vue de contrôler et optimiser la méthode d'optimisation adoptée pour la résolution du système $F_m|block|C_{max}$.

Tableau 2.4 : Paramètres optimaux des AGs

Méthode	Pop_Size	Max-iter	PC	Pm
AG	400	400	40 %	10 %

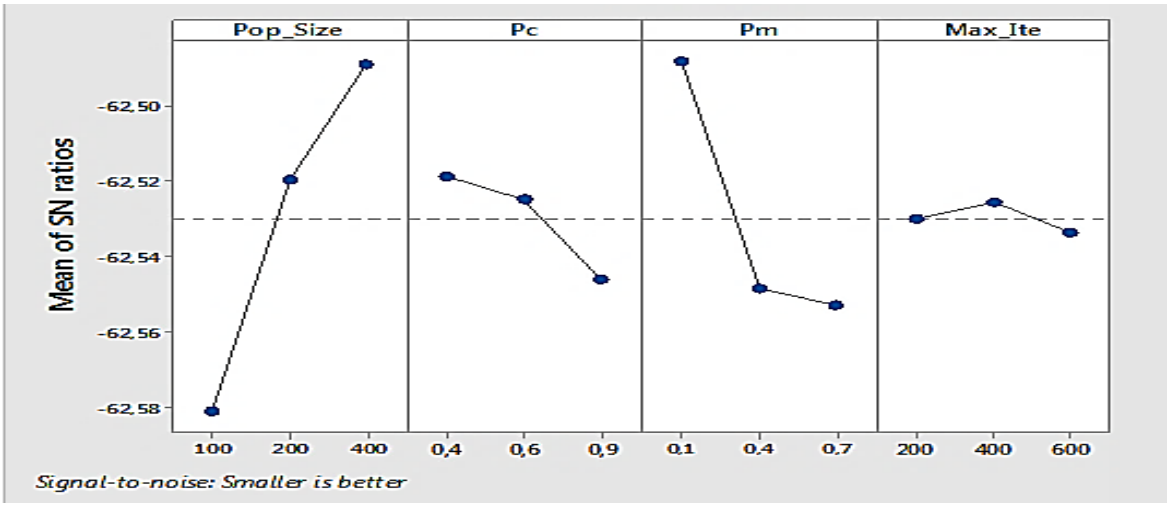


Figure 2.9 : Graphique des effets principaux du rapport (S/B)

2.5.10. Résultats numériques

Dans cette section, nous présentons l'ensemble des tests expérimentaux que nous avons établi pour la validation de l'efficacité de la méthode proposée précédemment en vue de résoudre le problème d'ordonnancement flow-shop avec contrainte de blocage. Toutefois, pour évaluer la performance relative à l'algorithme proposé, des tests ont été effectués sur la suite de référence de [TAI, 1983]. Le benchmark est constitué de 120 problèmes d'atelier flow-shop de différentes tailles, allant de 20 tâches et 5 machines à 500 tâches et 20 machines sachant que chaque problème comprend 10 instances. L'algorithme est programmé sur python 3.5 et implémenté dans un ordinateur équipé d'un microprocesseur Core i4, ayant 4,0 Go de mémoire vive (RAM).

En outre, l'algorithme proposé est comparé à d'autres métaheuristiques existantes dans la littérature, à savoir, IG [RIB, 2011], IABC [HAN, 2012], DE-ABC [HAN, 2015] et MFFO [HAN, 2016]. Pour chaque cas, l'algorithme est exécuté cinq réplications indépendantes. Au fur et à mesure, un indicateur de qualité dénommé ARDP est adopté pour mesurer la performance de l'AG proposé. Cette mesure calcule la différence relative moyenne entre le pourcentage du meilleur makespan retenu après chaque exécution et celui de référence obtenue par la méthode exacte Branch and Bound proposée par [RON, 2005]. En somme, l'ARDP peut être formulé de la manière suivante :

$$ARDP_i = \frac{1}{50}\sum_{l=1}^{10}\sum_{k=1}^{5}\frac{C_l^R - C_{l,k}^i}{C_l^R} * 100\% \qquad (15)$$

Où, $C_{l,k}^i$ et C_l^R désignent le makespan de la i^{ème} instance fourni par le i^{ème} algorithme dans la k^{ème} exécution et celui obtenu par [RON, 2005], respectivement. Évidemment, plus la valeur de l'ARDP est élevée, plus le résultat fourni par l'algorithme est meilleur.

2.5.10.1. Résultats des algorithmes génétiques

Étant donné que la performance d'une méthode d'optimisation se rapporte à deux facteurs principaux, notamment, la convergence et la vitesse de convergence. La figure 2.10 et le tableau 2.5 correspondent à la courbe de convergence des AGs proposés et aux résultats d'ARDP fournis.

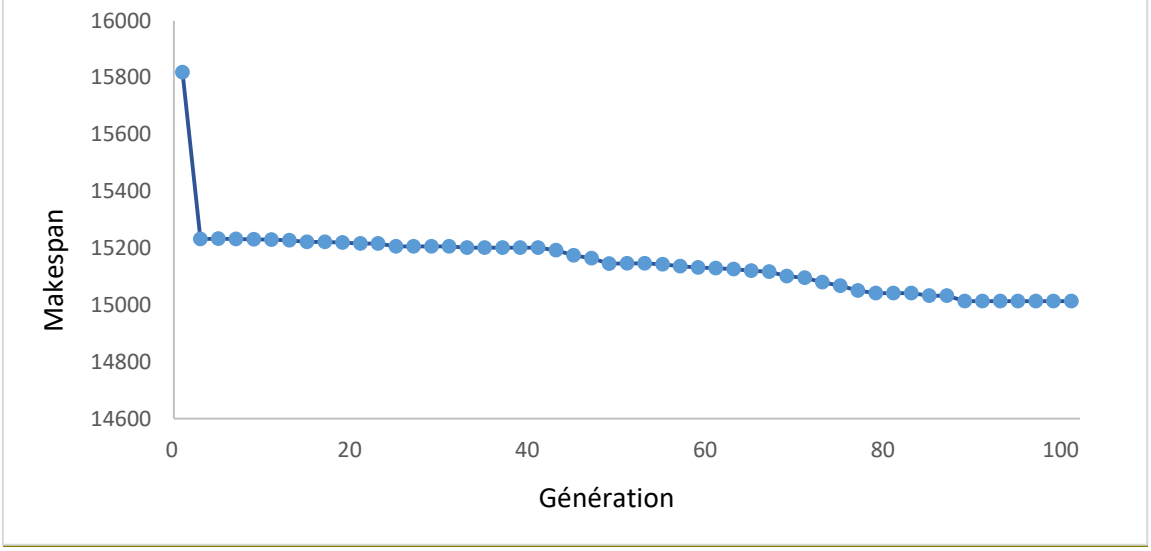


Figure 2.10 : Courbe de convergence des AGs

Tableau 2.5 : Résultats d'ARDP fournis par les AGs

Problèmes	ARDP
20*5	0.4
20*10	2.39
20*20	3.30
50*5	4.96
50*10	6.26
50*20	6.3
100*5	3.01
100*10	6.07
100*20	5.61
200*10	4.57
200*20	4.7
500*20	3.6
Moyenne	**4.26**

2.5.10.2. Comparaisons des AGs avec d'autres méthodes issues de la littérature

Dans cette partie, nous comparons les performances de notre approche avec celles des algorithmes IABC, DE-ABC, IG et MFFO existants dans la littérature. Le tableau 2.5, les figures 2. 11 et 2.12 donnent les résultats de comparaison des ARDP fournis et du temps de calcul nécessaire à la convergence de chacune des approches de comparaison. À partir des résultats obtenus, nous constatons que :

- Pour les instances, 20*10, 20*20, 50*5, 50*10, notre méthode atteint les meilleurs résultats avec des pourcentages de déviations relatives égales à 2.39, 3.30, 4.96, 6.26, respectivement ;
- Pour l'instance, 20*20, le pourcentage de déviation relative des AGs est égale à celui obtenu par l'algorithme IABC ;
- Pour les instances qui restent, la méthode MMFO est la plus performante en ce qui a trait aux pourcentages de déviations relatives moyens ;
- Concernant la vitesse de convergence, au fur et à mesure que le temps de calcul augmente, la courbe de convergence de l'algorithme MMFO atteint le niveau le plus bas comparativement à L'IG et l'AG. Par contre, sur ce point l'AG est légèrement supérieure que l'IG à partir de la 80ème génération.

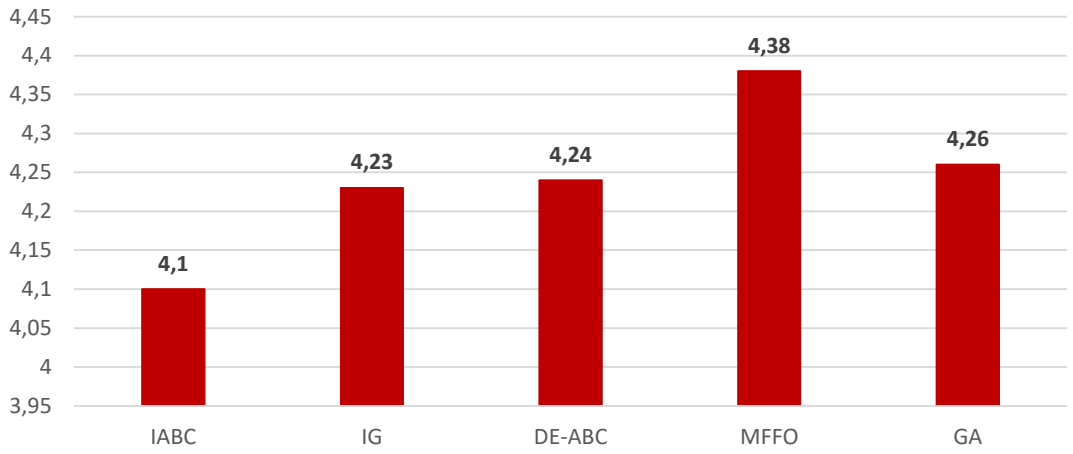


Figure 2.11 : Benchmark des résultats d'ARPD des différentes méthodes de comparaison

Tableau 2.6 : Résultats de comparaison de l'indicateur ARDP

Problèmes	IABC	DE-ABC	IG	MFFO	GA
20*5	0.41	0.34	**0.43**	0.36	0.4
20*10	2.38	2.34	2.37	2.38	**2.39**
20*20	3.30	3.29	3.30	3.28	**3.30**
50*5	4.68	4.89	4.91	4.94	**4.96**
50*10	6.15	6.25	6.33	6.23	**6.26**
50*20	6.29	6.45	6.38	**6.33**	6.3
100*5	2.31	2.67	2.89	**3.27**	3.01
100*10	5.97	6.27	6.10	**6.63**	6.07
100*20	5.39	5.53	5.48	**5.79**	5.61
200*10	4.14	4.37	4.25	**4.68**	4.57
200*20	4.46	4.7	4.7	**4.88**	4.7
500*20	3.7	3.72	3.7	**3.77**	3.6
Moyenne	**4.10**	**4.24**	**4.23**	**4.38**	**4.26**

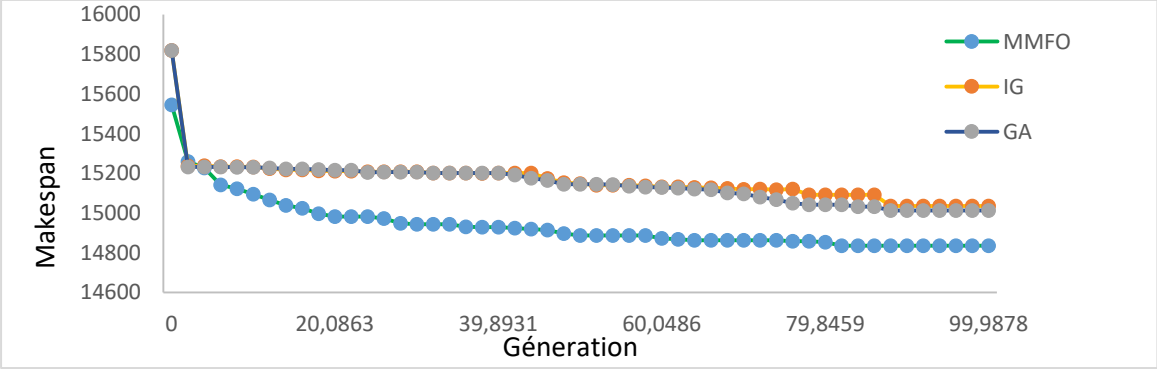


Figure 2.12 : Courbes de convergence de l'instance Ta109 des différentes méthodes de comparaison

Afin d'évaluer en profondeur et obtenir plus de clairvoyance sur la performance de l'algorithme proposé, des limites supérieures ont été générées sur les 120 instances introduites par [TAI, 1983] pour résoudre le problème de l'ordonnancement des flow-shop avec blocage. Le tableau 2.6

rapporte les meilleures limites supérieurs (mises en évidence en gras) atteintes par les AGs et les différentes méthodes de résolution proposées dans la littérature, notamment, la B&B développée par [RON, 2005] et la MMFO exposée dans les travaux de [HAN, 2016]. A partir de ce tableau, nous pouvons y voir que :

- 38 instances atteignent des résultats plus pertinents comparativement à ceux obtenus par la Branch and Bound ;
- La majorité des instances sur lesquelles les AGs atteignent de bons résultats sont de petites tailles (maximum TA030).

Les AGs excellent uniquement sur 6 instances comparativement à la MFFO. Or, la MFFO reste la plus performante dans la majorité des tailles des problèmes instances.

Tableau 2.7 : Meilleures limites supérieures fournies sur les instances de [TAI, 1983]

Instance	B&B	MMFO	GA	Instance	B&B	MMFO	GA
Ta001	1384	**1374**	**1374**	**Ta031**	3151	**2997**	3005
Ta002	1411	**1408**	**1408**	**Ta032**	3395	**3189**	3207
Ta003	1294	**1280**	**1280**	**Ta033**	3184	**3007**	3014
Ta004	1448	**1448**	**1448**	**Ta034**	3303	**3120**	3127
Ta005	1366	**1341**	**1341**	**Ta035**	3272	**3159**	3169
Ta006	1363	**1363**	**1363**	**Ta036**	3400	**3161**	3178
Ta007	1381	**1381**	**1381**	**Ta037**	3228	**3013**	3115
Ta008	1384	**1379**	**1379**	**Ta038**	2360	3054	**3052**
Ta009	1378	**1373**	**1373**	**Ta039**	3104	**2908**	**2908**
Ta010	1283	**1283**	**1283**	**Ta040**	3264	**3116**	3117
Ta011	1736	**1698**	**1698**	**Ta041**	3913	**3638**	3642
Ta012	1897	**1833**	**1833**	**Ta042**	3798	**3487**	3497
Ta013	1677	**1659**	**1659**	**Ta043**	3723	**3481**	3491
Ta014	1622	**1535**	**1535**	**Ta044**	3885	**6665**	3670
Ta015	1658	**1617**	**1617**	**Ta045**	3934	**3628**	6332
Ta016	1640	**1590**	**1590**	**Ta046**	3831	**3611**	3615
Ta017	1634	**1622**	**1622**	**Ta047**	3957	**3681**	3685
Ta018	1741	**1731**	**1731**	**Ta048**	3774	**3563**	3569
Ta019	1777	**1749**	**1749**	**Ta049**	3784	**3532**	**3531**
Ta020	1847	**1782**	**1782**	**Ta050**	3928	**3624**	**3623**
Ta021	2530	**2436**	**2436**	**Ta051**	4886	**4500**	4503
Ta022	2297	**2234**	**2234**	**Ta052**	4668	**4276**	4278
Ta023	2560	**2479**	**2479**	**Ta053**	4666	**4266**	4274
Ta024	2399	**2348**	**2348**	**Ta054**	4650	**4344**	4365
Ta025	2538	**2435**	**2435**	**Ta055**	4475	**4268**	4269
Ta026	2467	**2383**	**2383**	**Ta056**	4521	**4280**	4282
Ta027	2502	**2390**	**2390**	**Ta057**	4576	**4308**	**4308**
Ta028	2411	**2328**	**2328**	**Ta058**	4688	**4310**	4321

Ta029	2421	**2363**	**2363**	**Ta059**	4532	4310	4322
Ta030	2407	**2323**	**2323**	**Ta060**	4846	**4415**	4417
Ta061	6455	**6130**	**6141**	**Ta091**	14113	**13384**	13424
Ta062	6214	**5992**	**6008**	**Ta092**	14127	**13294**	13311
Ta063	6124	**5913**	**5921**	**Ta093**	14416	**13416**	13463
Ta064	5976	**5710**	**5723**	**Ta094**	14435	**13337**	13344
Ta065	6173	**5938**	**5949**	**Ta095**	14113	**13340**	13354
Ta066	6094	**5808**	**5811**	**Ta096**	13309	**13081**	13085
Ta067	6262	**5959**	**6002**	**Ta097**	14563	**13561**	13607
Ta068	6061	**5865**	**5881**	**Ta098**	14329	**13504**	13644
Ta069	6474	**6106**	**6102**	**Ta099**	13923	**13217**	13308
Ta070	6366	**6112**	**6141**	**Ta100**	14435	**13400**	13439
Ta071	7496	**6988**	**6998**	**Ta101**	15579	**14192**	14908
Ta072	7281	**6714**	**6729**	**Ta102**	15728	**14963**	14962
Ta073	7400	**6834**	**6892**	**Ta103**	15915	**15043**	15107
Ta074	7670	**7102**	**7125**	**Ta104**	16039	**14949**	14987
Ta075	7317	**6804**	**6935**	**Ta105**	15938	**14857**	14899
Ta076	7301	**6615**	**6639**	**Ta106**	15911	**14909**	14929
Ta077	7247	**6770**	**6811**	**Ta107**	15898	**14980**	15009
Ta078	7315	**6806**	**6859**	**Ta108**	16022	**15005**	15021
Ta079	7631	**6997**	**7055**	**Ta109**	15817	**14834**	15013
Ta080	7411	**6910**	**6928**	**Ta110**	15969	**14954**	14954
Ta081	8347	**7774**	**7799**	**Ta111**	38334	**35838**	36609
Ta082	8372	**7838**	**7892**	**Ta112**	38642	**35970**	36108
Ta083	8265	**7777**	**7802**	**Ta113**	38163	**35757**	36778
Ta084	8365	**7818**	**7836**	**Ta114**	38625	**36070**	36628
Ta085	304	**7799**	**7836**	**Ta115**	38492	**35869**	35869
Ta086	8450	**7841**	**7851**	**Ta116**	38551	**36099**	37006
Ta087	8507	**7929**	**7979**	**Ta117**	38179	**35760**	36805
Ta088	8584	**7976**	**7993**	**Ta118**	38664	**35921**	36792
Ta089	8341	**7894**	**7889**	**Ta119**	38339	**35739**	36885
Ta090	8489	**7929**	**7918**	**Ta120**	38540	**36120**	36184

2.5.10.3. Discussion et synthèse

En dernière analyse, les résultats démontrent que l'approche adoptée ne fournit pas des résultats satisfaisants malgré son efficacité connue dans la résolution des problèmes d'optimisation combinatoires qui sont justifiés par une forte complexité théorique. En effet, le problème rencontré survient du fait que les AGs se lancent à une recherche locale d'un minimum qui n'est pas forcément l'optimum attendu. Ce phénomène génétique appelé convergence prématurée est dû au fait qu'un bon individu se met en l'espace de plusieurs générations, par la suite il risque d'envahir toute la population et empêcher une évolution stable et équilibrée.

Pour aboutir à une recherche efficace, il faut préserver la diversité génétique en maintenant un équilibre entre l'exploitation des bonnes solutions rencontrées et l'exploration des zones d'intérêt. Typiquement, pour arriver à une convergence globale, on assiste à une tendance vers les algorithmes hybrides permettant de combiner les atouts de deux types de méthodes locales et globales. Cette combinaison mène évidemment vers un compromis entre l'exploitation et l'exploration, ce qui permet d'éviter le piégeage de la fonction objectif dans des optimums locaux.

Dans cette perspective, pour palier à l'inconvénient majeur des AGs, éliminer ses faiblesses et améliorer la qualité des solutions fournies, une méthode hybride constituée de la méthode utilisée et celle du recuit simulé sera proposée dans la partie qui suit pour la résolution mono-objectif du problème d'atelier flow-shop sous contrainte de blocage visant à minimiser le makespan. Cette incorporation consiste à exploiter les avantages respectifs des algorithmes génétiques et du recuit simulé en combinant judicieusement leurs algorithmes suivant une approche synergétique. Le bénéfice majeur du recuit simulé réside dans le fait qu'il permet au système de se déplacer à la baisse des états d'énergie, tout en sautant hors des minima locaux (en particulier à des températures plus élevées) en raison de l'acceptation probabiliste de quelques mouvements à la hausse. Ce qui permet par la suite de compenser l'inconvénient critique des algorithmes génétiques étant la convergence prématurée.

2.6. Résolution approchée à travers une hybridation séquentielle (SAGA)

Étant convaincus qu'un compromis entre l'exploration de l'espace d'état (diversification) et l'exploitation des meilleures zones de recherche (intensification) doit être trouvé, la recherche d'un outil d'optimisation combinant cette dualité s'avère être primordial pour améliorer le comportement global des métaheuristiques et éviter la stagnation de la fonction coût dans les optimums locaux. Dans cet esprit, on assite de nos jours à une tendance vers les algorithmes hybrides permettant de tirer profit des atouts de chacune des métaheuristiques utilisées, allant des méthodes locales qui assurent l'intensification aux celles globales qui excellent en exploration. Cette technique d'hybridation semble constituer une alternative très cruciale quand elle est fondée sur une vraie maîtrise des points forts et faibles de chacune des méthodes candidates.

2.6.1. Classification des approches hybrides

Plusieurs algorithmes hybrides sous forme de plusieurs types d'hybridations possibles ont été développés afin d'étendre de façon commode le spectre d'application et hausser l'efficacité des approches combinées [DUV, 2000]. Selon la classification proposée par cet auteur, trois catégories d'hybridation sont distinguées en fonction de leur conception (Figure 2.13).

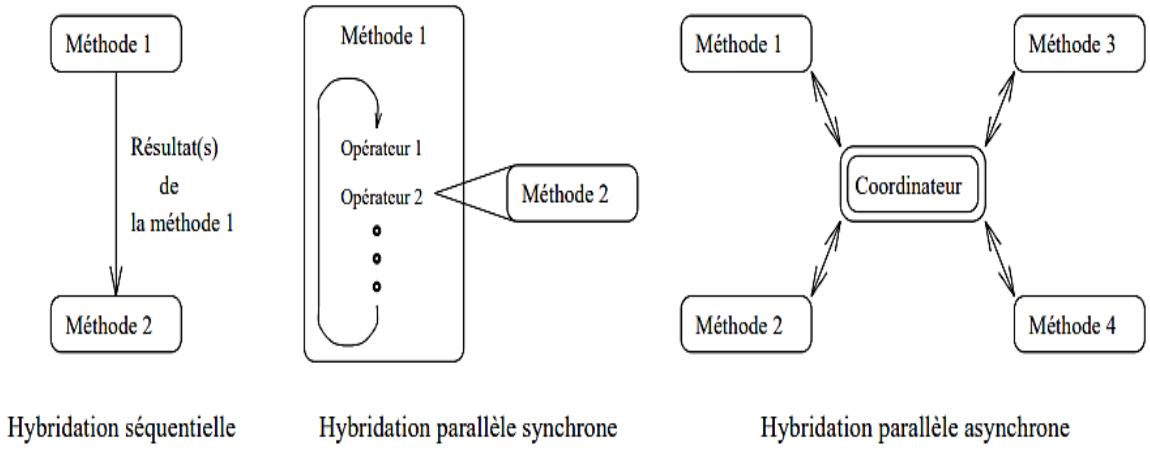


Figure 2.13 : Techniques d'hybridation selon [DUV, 2000]

La première technique d'hybridation dite séquentielle consiste à exécuter séquentiellement différentes méthodes de sorte que les résultats fournis par l'une servent de solutions initiales à l'autre. Vient ensuite, l'hybridation parallèle synchrone qui est réalisée par l'incorporation d'une méthode de recherche particulière dans un opérateur. Cette technique est plus complexe à mettre en œuvre que la précédente. Son objectif central réside dans la combinaison d'une recherche locale avec une recherche globale dans le but d'améliorer la convergence. La dernière technique étant l'hybridation parallèle asynchrone, elle consiste à impliquer deux algorithmes de telle façon que l'un ajuste l'autre en faisant évoluer en parallèle différentes méthodes de recherche.

Historiquement, plusieurs méthodes hybrides ont été proposées pour résoudre une grande variété de problèmes d'optimisation combinatoire. Cette idée de combiner est née non seulement suite au piégeage de la fonction objectif mais également par rapport au coût temporel qui augmente exponentiellement avec la complexité du problème.

De ce fait, pour palier efficacement à ces restrictions flagrantes et améliorer les résultats obtenus par les algorithmes génétiques présentés précédemment, nous proposons dans nos travaux de recherche une hybridation séquentielle des algorithmes génétiques avec le recuit simulé (Simulated Annealing (SA)). Cette hybridation consiste à appliquer le recuit simulé pour sélectionner la nouvelle population de chaque itération. L'algorithme final SAGA se compose de deux phases qui alternent régulièrement le progrès génétique.

2.6.2. Approche du recuit simulé

2.6.2.1. Introduction au recuit simulé

Le recuit simulé est une méthode d'optimisation empirique de type Monte-Carlo dont l'origine est tirée de l'analogie avec le recuit des métaux en métallurgie. Développée par les physiciens Kirk Patrick, Gellat, Vecchi en 1983 [KIR, 1983], cette approche stochastique permet de résoudre les problèmes d'optimisation combinatoires de façon quasi-optimale. Depuis son invention, elle doit son succès à sa rapidité et son efficacité dans diverses problématiques d'optimisation de grande taille [DRE, 2003]. En outre, cette technique d'optimisation itérative est basée sur les simulations de Metropolis en mécanique statique. L'originalité de cette méthode réside dans sa capacité d'échapper aux optima locaux en se dirigeant dans la simulation vers des solutions voisines de moins bonne qualité avec une probabilité non nulle. Au long du processus de la recherche, un

refroidissement lent d'un paramètre primordial dans le progrès local est effectué, ce paramètre correspond à la température T qui est initialisée au début avec une configuration élevée et varie au cours de la recherche pour tendre pratiquement vers 0. Autrement dit, l'algorithme commence par une configuration initiale du système et subit par la suite une modification élémentaire en diminuant légèrement la température, quand l'algorithme atteint les très basses températures, les états les plus probables constituent en principe d'excellentes solutions au problème d'optimisation.

2.6.2.2. Procédure du recuit simulé

Essentiellement, l'idée de base consiste à effectuer un mouvement selon une distribution de probabilité qui dépend de la qualité des différents voisins en se basant sur l'algorithme de Metropolis. Dans le schéma original de Metropolis, la probabilité pour un système physique de posséder une énergie E, lorsque l'équilibre thermodynamique est atteint à une température T, est proportionnelle au facteur de Boltzmann-Gibbs $e^{\frac{-\Delta f}{T}}$ où, K représente la constante de Boltzmann. Dans ces conditions, au niveau de la mise en œuvre du processus, une solution primaire S est générée aléatoirement, le passage de la solution courante S à une solution voisine S* dépend de la probabilité P de la distribution de Boltzman-Gibbs précitée. En effet, cette transition est acceptée si la condition suivante est remplie : (Voir Figure 2.14)

$$P = min\left\{1, e^{\frac{-f}{T}}\right\} \geq \Omega \qquad (16)$$

Où, f=f(S)-f(S*) désigne la différence entre la fonction objectif des deux états, Ω est un nombre aléatoire généré dans l'intervalle [0, 1] et T est le paramètre de contrôle courant dans le processus.

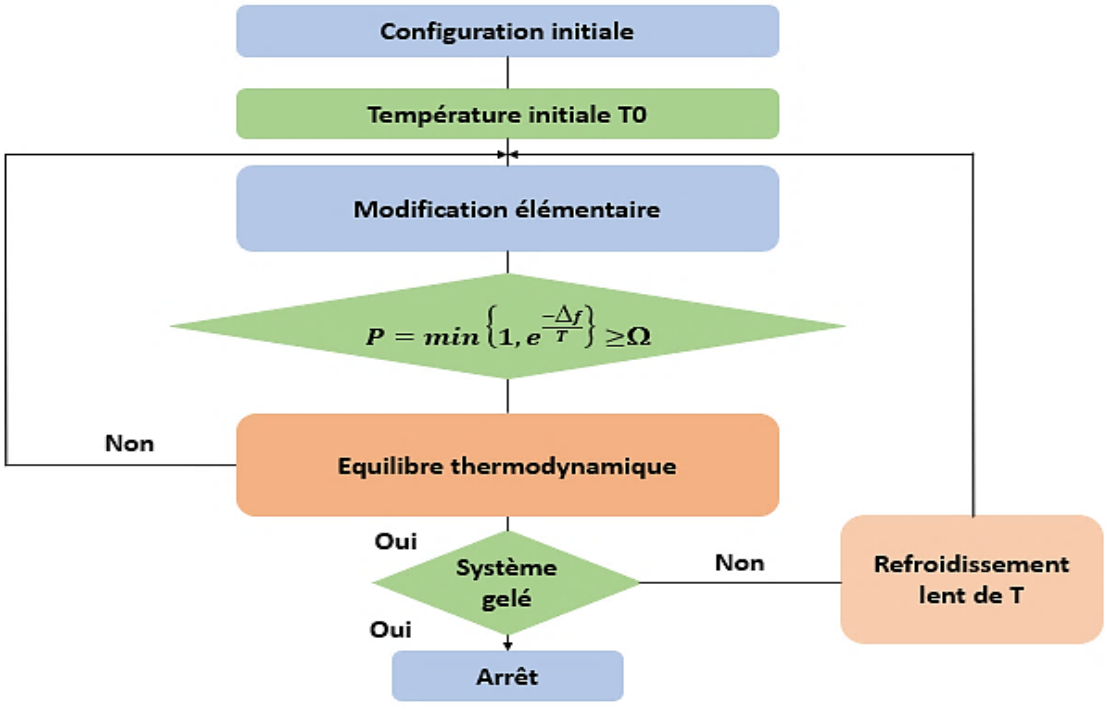


Figure 2.14 : Processus général du recuit simulé

A partir d'une valeur élevée de température T, une recherche d'état est effectuée à chaque niveau avec un certain nombre d'itérations désigné comme longueur de chaîne de Markov. Cette longueur représente le nombre de mouvements effectués pour une valeur fixe de la température qui décroit logarithmiquement avec le temps. Étant donné que $0<\alpha<1$ est le paramètre de vitesse de refroidissement, le programme de recuit est donc défini comme suit :

$$T_{n+1} = \alpha T_n \quad (17)$$

2.6.2.3. Algorithme général du recuit simulé

En se basant sur les étapes schématisées en dessus dans la figure 2.14, les principales étapes du recuit simulé sont expliquées en détail dans l'algorithme 2.4 suivant :

Algorithme 2.4 : Algorithme SA

1. Choisir aléatoirement une solution du système à optimiser et évaluer sa fonction objectif f=f(S) ;
2. Initialiser la température T (Température élevée) ;
3. Perturber cette solution pour obtenir une nouvelle solution S*=S+S ;
4. Calculer f=f(S)-f(S*);
5. Accepter ou refuser la solution S* en se basant sur la condition de Metropolis ;
6. Sauver la meilleure solution rencontrée ;
7. Si l'équilibre thermodynamique est atteint ==> refroidissement lent de la température

   Sinon reprendre l'étape 3 ;
8. Si la température T est inférieure à une température seuil voisine de 0 ==> meilleure configuration est trouvée==> arrêt du programme

   Sinon reprendre l'étape 3.

2.6.3. Hybridation séquentielle des algorithmes génétiques et du recuit simulé

Une méthode hybride peut être efficace ou inopérante selon la sélection de ses composants. Pour concevoir une méthode hybride performante, il faut savoir caractériser les atouts et les défauts de chaque méthode candidate. Pour notre cas, le choix de l'hybridation séquentielle des algorithmes génétiques et du recuit simulé est dicté bien entendu par une grande maîtrise des atouts et défauts de chacun de ces outils d'optimisation. De ce fait, malgré la force prouvée de la méthode locale précitée pour la résolution d'une grande variété des problèmes d'optimisation combinatoires, son principal inconvénient réside dans son arrêt lorsqu'aucun point voisin ne peut améliorer la solution courante. En effet, cet inconvénient ne peut être pallié que si on arrive à répéter le processus à partir de plusieurs points de départ générés aléatoirement à travers une mesure couramment utilisée, ou si les solutions de départ sont fournies par un algorithme génétique. Ce constat de complémentarité des approches génétiques et du recuit simulé sera exploité dans cette section pour résoudre le problème d'ordonnancement des ateliers de type flow-shop avec contrainte de blocage considérée. Cette hybridation nommée SAGA se déroule en deux étapes : Dans la première, un ensemble de descendants est produit en appliquant les différents opérateurs de recherche génétique. Tandis que dans la deuxième, les descendants obtenus sont acceptés ou refusés pour passer à la génération suivante selon le critère Metropolis formulé précédemment, étant donné que f dans le cas du problème d'ordonnancement de l'atelier flow-shop avec capacité de stockage limitée, représente la différence du makespan entre la nouvelle population P(n+1) et l'ancienne P(n), nommées $g_{\_newbest}$ et $g_{\_oldbest}$ respectivement. Sans omettre de signaler que le remplacement de l'ancienne population s'effectue si $g_{\_newbest}$ est meilleur que $g_{\_oldbest}$ à travers la probabilité de Boltzmann. Ce processus est répété itérativement jusqu'à ce que le nombre maximum d'itérations soit atteint et que la valeur optimale globale du makespan soit obtenue.

Sachant que Cool_rate désigne la vitesse de refroidissement et Temp représente la température, les étapes spécifiques de l'approche coopérative SAGA proposées sont illustrées dans l'algorithme 2.5 suivant.

Algorithme 2.5 : Algorithme SAGA pour résoudre le $F_m|block|C_{max}$

1. Initialiser : Pop_Size, Pc, Pm, Max_iter, Cool-rate, Temp=T0 ;
2. Générer aléatoirement la population initiale Pop_Size (n) composée des séquences aléatoires des jobs ;
3. Calculer la fonction fitness 'Cmax' pour chaque séquence de la population courante ;
4. Sélectionner par tournoi la meilleure fitness $g\_{oldbest}$ de la population courante ;
5. **Tant que le nombre d'itérations maximal n'est pas atteint :**
6. i=1
7. Pour chaque séquence : de 1 à Pop_Size
   - 7.1. Sélectionner aléatoirement deux séquences de P(N) aléatoirement
   - 7.2. Si $\alpha \leq Pc$:
   - 7.3. Appliquer le croisement standard en 2-points et produire deux enfants
   - 7.4. Si $\beta \leq Pm$:
   - 7.5. Appliquer la mutation réciproque
   - 7.6. Copier les ascendants obtenus dans la nouvelle population P(n+1)
   - 7.7. Evaluer le makespan de tous les chromosomes de la nouvelle population $g_i(P(n+1))$
   - 7.8. Sauvegarder la meilleure et la mauvaise solutions rencontrées :

     $g\_{newbest} = \min [g]$

     $g\_{newworst} = \max [g]$
   - 7.9. Si $g\_{newbest} < g\_{oldbest}$:

     Population_Temporaire = Nouvelle_Population [$g\_{newbest}$:]
   - 7.10. Sinon si $g\_{newbest} == g\_{oldbest}$:

     i=i+1
   - 7.11. Sinon si $g\_{newbest} > g\_{oldbest}$:

     Calculer X= exp [- ($g\_{newbest} - g\_{oldbest}$) / Temp]
     Si X > :
     Population_Temporaire=Nouvelle_Population [$g\_{newbest}$:]

   Nouvelle_Population [: $g\_{newworst}$]= Population_Temporaire
   PS (n+1) = Nouvelle_Population
   Refroidissement lent de la temperature Temp:
   Temp = cool_rate *Temp
8. Si le critère d'arrêt est satisfait, arrêter la procédure de la SAGA et retourner la meilleure configuration du makespan Cmax trouvée, sinon aller à l'étape 5.
9. **Fin tant que**

2.6.4. Calibrage des paramètres de la SAGA : Plan d'expériences de Taguchi

Le calibrage des paramètres de la SAGA est effectué en suivant la technique de Taguchi utilisée dans le cas du paramétrage des algorithmes génétiques présenté précédemment. De la même

manière, 27 séries d'expériences ont été réalisées en s'appuyant sur la distribution de la table orthogonale, et trois niveaux ont été affectés à chaque facteur, comme l'illustre le tableau 2.8.

Tableau 2.8 : Différents niveaux de chaque paramètre de la SAGA

Facteurs	Pop_Size	Max-iter	Pc	Pm	Cool_rate	Temp
Niveau 1	100	200	40 %	10 %	0.3	500
Niveau 2	200	400	60 %	40 %	0.5	700
Niveau 3	400	600	90 %	70 %	0.7	1000

La figure 2.15 représente le graphique des effets principaux crée sur Minitab en traçant la moyenne des caractéristiques pour chaque niveau de facteur. Les six facteurs considérés ci-dessus doivent être ajustés tels que : la taille de la population = 400, le nombre d'itérations = 600, la probabilité de mutation = 0,1 la probabilité de croisement = 0,9, la vitesse de refroidissement=0.5 et la température=500. Le tableau 2.9 regroupe le niveau sélectionné pour chaque facteur en vue de contrôler et optimiser la méthode SAGA proposée pour la résolution du système $F_m|block|C_{max}$.

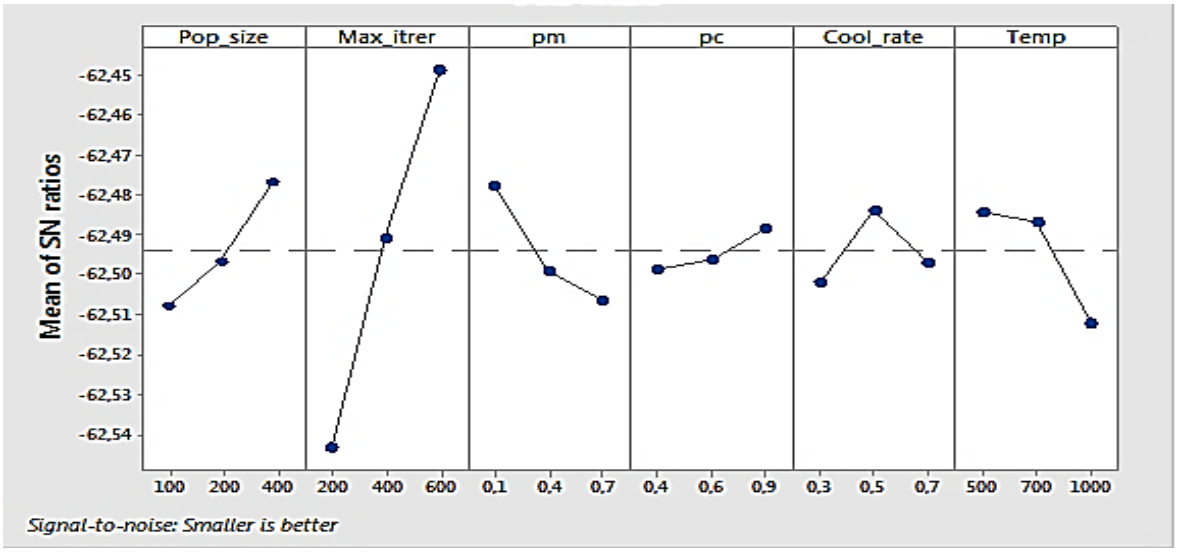


Figure 2.15 : Graphique des effets principaux du rapport (S/B)

Tableau 2.9 : Paramètres optimaux de la SAGA

Méthode	Pop_Size	Max-iter	Pc	Pm	Cool_rate	Temp
SAGA	400	600	90 %	10 %	0.5	500

2.6.5. Résultats numériques

Dans cette section, nous présentons l'ensemble des tests expérimentaux que nous avons établi pour la validation de l'efficacité de la SAGA proposée en vue de résoudre le problème d'ordonnancement flow-shop avec contrainte de blocage. La performance de l'algorithme proposé a été évaluée aussi en se basant sur les instances de [TAI, 1983]. L'algorithme SAGA est programmé avec les mêmes configurations que celles des algorithmes génétiques (python 3.5 et implémentés dans un ordinateur équipé d'un microprocesseur Core i4, ayant 4,0 Go de mémoire vive (RAM)).

L'algorithme hybride proposé est comparé aux métaheuristiques abordés au niveau de l'étude comparative de la performance des AGs, notamment, IABC [HAN, 2012], DE-ABC [HAN, 2015], IG [RIB, 2011] ET MFFO [HAN, 2016]. Ainsi, pour chaque taille de problème, l'indicateur ARDP est semblablement calculé.

2.6.5.1. Résultats de la SAGA

La figure 2.16 et le tableau 2.10 correspondent à la courbe de convergence de la SAGA proposée et aux résultats d'ARDP fournis.

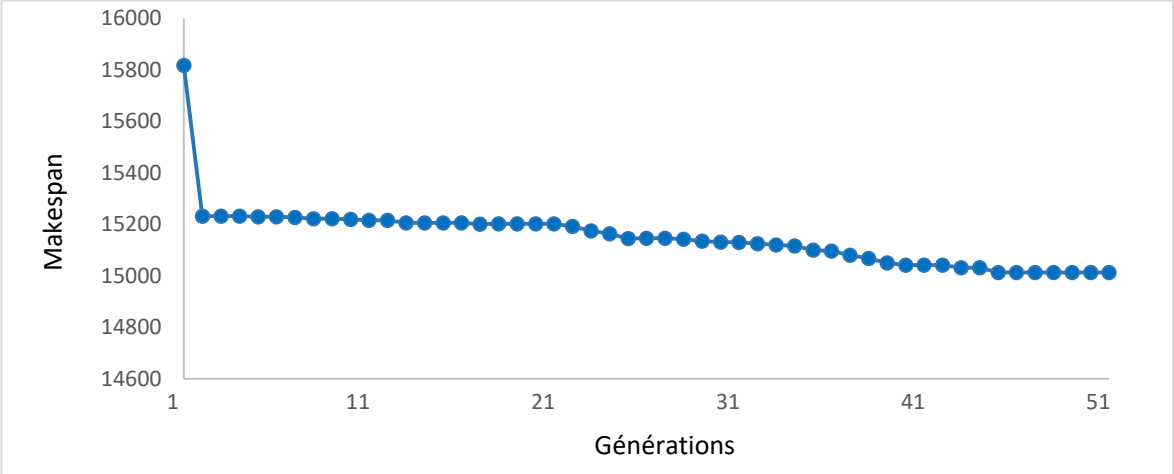


Figure 2.16 : Courbe de convergence de la SAGA

Tableau 2.10 : Résultats d'ARDP fournis par la SAGA

Problème	ARDP
20*5	0.48
20*10	2.41
20*20	3.31
50*5	4.94
50*10	6.42
50*20	6.54
100*5	3.26
100*10	6.71
100*20	5.83
200*10	4.81
200*20	4.97
500*20	3.72
Moyenne	**4.45**

2.6.5.2. Comparaisons de la SAGA avec d'autres méthodes issues de la littérature

Dans cette partie, nous comparons la performance de la SAGA avec celles des algorithmes IABC, DE-ABC, IG, MFFO et des AGs que nous avons proposé dans nos travaux de recherche. Le tableau 2.11 et les figures 2.17 et 2.18 présentent respectivement les résultats de comparaison des ARDP fournis et du temps de calcul nécessaire à la convergence de chacune des approches de comparaison pour l'instance sélectionnée Ta109 de la suite de benchmarks de [TAI, 1983]. À partir des résultats obtenus, nous constatons que :

- Concernant la vitesse de convergence, au fur et à mesure que le temps de calcul augmente, la courbe de convergence de l'algorithme SAGA atteint le niveau le plus bas comparativement aux autres approches de comparaison ;

- Comme il montre le tableau et la figure ci-dessous, la moyenne totale de l'ARDP obtenue par l'algorithme SAGA proposé est supérieure aux valeurs respectives de 4,10, 4,24, 4,23, 4,38 et 4,26 obtenues par IABC, DE-ABC, IG, MMFO et GA. Il est clair que l'algorithme SAGA proposé dépasse les algorithmes IABC, DE-ABC, IG, MFFO et GA pour la plupart des 12 tailles de problèmes d'ordonnancement d'atelier de blocage (20*5, 20*10, 20*20, 50*10, 50*20, 100*10,100*20,200*10 et 200*20) ;
- Pour l'instance, 50*5, le pourcentage de déviation relative des AGs reste le meilleur comparativement à tous les algorithmes de comparaison.

Pour les instances qui restent, la méthode MMFO est la plus performante en ce qui a trait aux pourcentages de déviations relatives moyens.

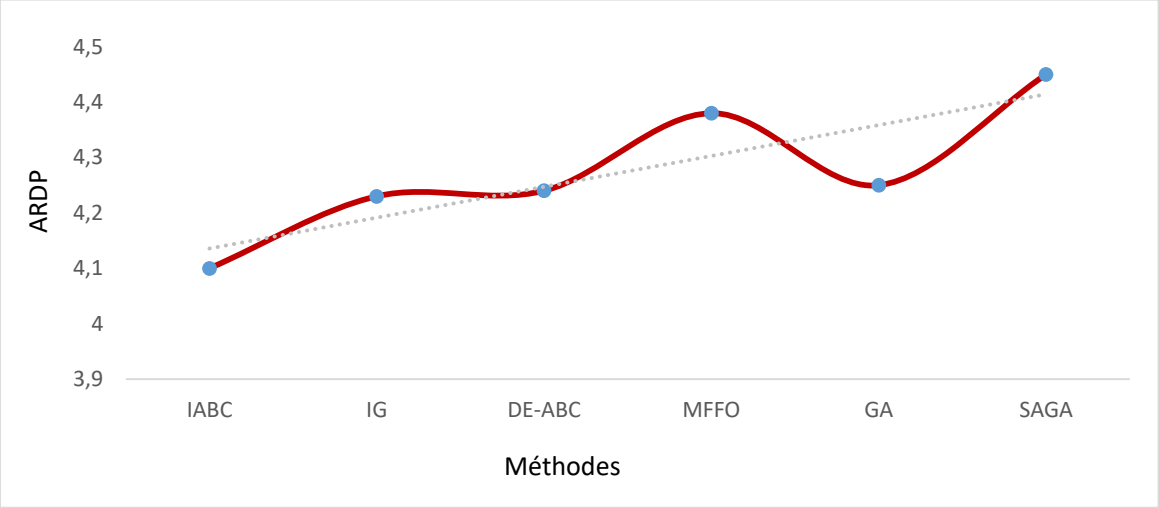


Figure 2.17 : Benchmark des résultats d'ARPD des différentes méthodes de comparaison

Tableau 2. 11 : Résultats de comparaison de l'indicateur ARDP

Problèmes	IABC	DE-ABC	IG	MFFO	GA	SAGA
20*5	0.41	0.34	0.43	0.36	0.4	**0.48**
20*10	2.38	2.34	2.37	2.38	2.39	**2.41**
20*20	3.30	3.29	3.30	3.28	3.30	**3.31**
50*5	4.68	4.89	4.91	4.94	**4.96**	4.94
50*10	6.15	6.25	6.33	6.23	6.26	**6.42**
50*20	6.29	6.45	6.38	6.33	6.3	**6.54**
100*5	2.31	2.67	2.89	**3.27**	3.01	3.26
100*10	5.97	6.27	6.10	6.63	6.07	**6.71**
100*20	5.39	5.53	5.48	5.79	5.61	**5.83**
200*10	4.14	4.37	4.25	4.68	4.57	**4.81**
200*20	4.46	4.7	4.7	4.88	4.7	**4.97**
500*20	3.7	3.72	3.7	**3.77**	3.6	3.72
Moyenne	**4.10**	**4.24**	**4.23**	**4.38**	**4.26**	**4.45**

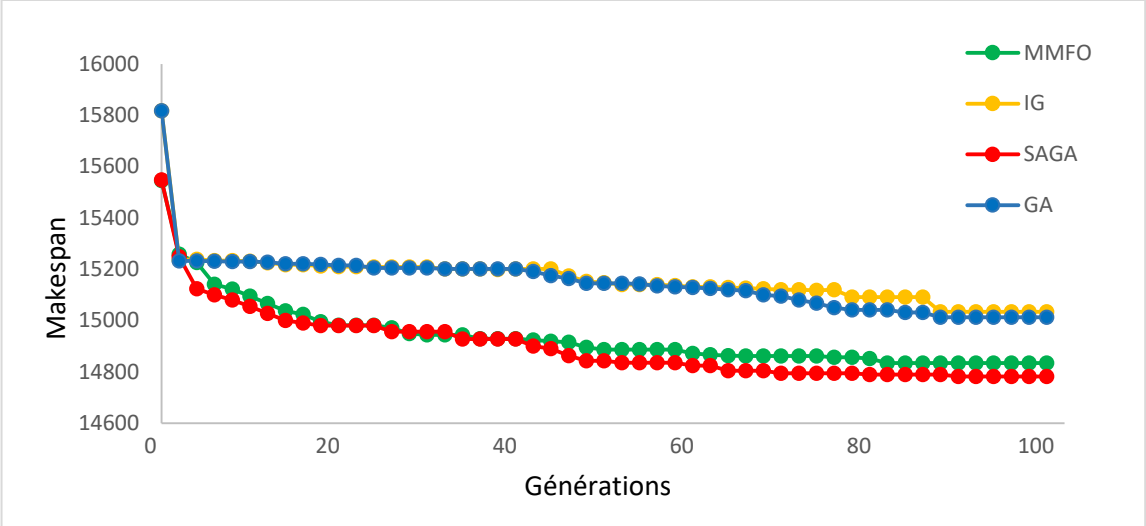


Figure 2.18 : Courbes de convergence de l'instance Ta109 des différentes méthodes de comparaison

En outre, des limites supérieures ont été générées sur les 120 instances introduites par [TAI, 1983] pour résoudre le problème de l'ordonnancement des flow-shop avec blocage. Le tableau 2.12 rapporte les meilleures limites supérieurs (mises en évidence en gras) atteintes par la SAGA et les différentes méthodes de résolution proposées dans la littérature, notamment, la B&B développée par [RON, 2005] les AGs développés précédemment et la MMFO présentée par [HAN, 2016]. À partir de ce tableau, on peut y voir que :

- 83 instances atteignent des résultats plus pertinents comparativement à ceux fournis par la Branch and Bound, la MFFO, et les AGS ;
- La SAGA fournit de bons résultats même pour les problèmes d'ordonnancement de grande taille ;
- Pour les problèmes de petite taille allant de Ta001 à Ta030, les résultats, les résultats fournis pour la SAGA sont pratiquement similaires à ceux obtenus par la MFFO ;
- À partir de l'instance TA031, les valeurs fournies ne cessent de s'améliorer à part quelques instances, notamment, Ta032, Ta044, Ta061, et la Ta097 ou la MFFO reste toujours performante.

Tableau 2. 12 : Meilleures limites supérieures fournies sur les instances de [TAI, 1983]

Instance	B&B	MMFO	SAGA	GA	Instance	B&B	MMFO	SAGA	GA
Ta001	1384	1374	1374	1374	**Ta031**	3151	2997	**2989**	3005
Ta002	1411	1408	1408	1408	**Ta032**	3395	**3189**	3192	3207
Ta003	1294	1280	1280	1280	**Ta033**	3184	3007	**2998**	3014
Ta004	1448	1448	1448	1448	**Ta034**	3303	3120	**3117**	3127
Ta005	1366	1341	1341	1341	**Ta035**	3272	3159	**3158**	3169
Ta006	1363	1363	1363	1363	**Ta036**	3400	3161	**3156**	3178
Ta007	1381	1381	1381	1381	**Ta037**	3228	**3013**	**3013**	3115
Ta008	1384	1379	1379	1379	**Ta038**	2360	3054	**3050**	3052
Ta009	1378	1373	1373	1373	**Ta039**	3104	2908	**2897**	2908
Ta010	1283	1283	1283	1283	**Ta040**	3264	3116	**3094**	3117
Ta011	1736	1698	1698	1698	**Ta041**	3913	3638	**3624**	3642
Ta012	1897	1833	1833	1833	**Ta042**	3798	3487	**3475**	3497
Ta013	1677	1659	1659	1659	**Ta043**	3723	3481	**3480**	3491

Ta014	1622	1535	1535	1535	**Ta044**	3885	**3665**	3666	3670
Ta015	1658	1617	1617	1617	**Ta045**	3934	3628	**3620**	6332
Ta016	1640	1590	1590	1590	**Ta046**	3831	3611	**3607**	3615
Ta017	1634	1622	1622	1622	**Ta047**	3957	3681	**3671**	3685
Ta018	1741	1731	1731	1731	**Ta048**	3774	3563	**3562**	3569
Ta019	1777	1749	1749	1749	**Ta049**	3784	3532	**3529**	3531
Ta020	1847	1782	1782	1782	**Ta050**	3928	3624	**3619**	3623
Ta021	2530	2436	2436	2436	**Ta051**	4886	4500	**4496**	4503
Ta022	2297	2234	2234	2234	**Ta052**	4668	4276	**4276**	4278
Ta023	2560	2479	**2465**	2479	**Ta053**	4666	4266	**4257**	4274
Ta024	2399	2348	2348	2348	**Ta054**	4650	4344	**4338**	4365
Ta025	2538	2435	2435	2435	**Ta055**	4475	4268	**4261**	4269
Ta026	2467	2383	**2372**	2383	**Ta056**	4521	4280	**4275**	4282
Ta027	2502	2390	2390	2390	**Ta057**	4576	4308	**4305**	4308
Ta028	2411	2328	2328	2328	**Ta058**	4688	**4310**	**4310**	4321
Ta029	2421	2363	2363	2363	**Ta059**	4532	4310	**4308**	4322
Ta030	2407	2323	2323	2323	**Ta060**	4846	**4415**	**4415**	4417
Ta061	6455	6130	6135	6141	**Ta091**	14113	**13384**	**13357**	13424
Ta062	6214	5992	5985	6008	**Ta092**	14127	**13294**	**13280**	13311
Ta063	6124	5913	5910	5921	**Ta093**	14416	**13416**	**13416**	13463
Ta064	5976	5710	5702	5723	**Ta094**	14435	**13337**	**13335**	13344
Ta065	6173	5938	5931	5949	**Ta095**	14113	**13340**	**13311**	13354
Ta066	6094	5808	5799	5811	**Ta096**	13309	**13081**	**13026**	13085
Ta067	6262	5959	5941	6002	**Ta097**	14563	**13561**	**13571**	13607
Ta068	6061	5865	5862	5881	**Ta098**	14329	**13504**	**13502**	13644
Ta069	6474	6106	6093	6102	**Ta099**	13923	**13217**	**13201**	13308
Ta070	6366	6112	6105	6141	**Ta100**	14435	**13400**	**13389**	13439
Ta071	7496	6988	6971	6998	**Ta101**	15579	**14192**	**14889**	14908
Ta072	7281	6714	6709	6729	**Ta102**	15728	**14963**	**14952**	14962
Ta073	7400	6834	6820	6892	**Ta103**	15915	**15043**	**15026**	15107
Ta074	7670	7102	7989	7125	**Ta104**	16039	**14949**	**14930**	14987
Ta075	7317	6804	6799	6935	**Ta105**	15938	**14857**	**14857**	14899
Ta076	7301	6615	6607	6639	**Ta106**	15911	**14909**	**14925**	14929
Ta077	7247	6770	6770	6811	**Ta107**	15898	**14980**	**14972**	15009
Ta078	7315	6806	6801	6859	**Ta108**	16022	**15005**	**15000**	15021
Ta079	7631	6997	6964	7055	**Ta109**	15817	**14834**	**14782**	15013
Ta080	7411	6910	6895	6928	**Ta110**	15969	**14954**	**14921**	14954
Ta081	8347	7774	7751	7799	**Ta111**	38334	**35838**	**36504**	36609
Ta082	8372	7838	7883	7892	**Ta112**	38642	**35970**	**35946**	36108
Ta083	8265	7777	7769	7802	**Ta113**	38163	**35757**	**35615**	36778
Ta084	8365	7818	7803	7836	**Ta114**	38625	**36070**	**36045**	36628
Ta085	304	7799	7769	7836	**Ta115**	38492	**35869**	**35762**	35869
Ta086	8450	7841	7870	7851	**Ta116**	38551	**36099**	**36099**	37006
Ta087	8507	7929	7973	7979	**Ta117**	38179	**35760**	**35760**	36805
Ta088	8584	7976	7975	7993	**Ta118**	38664	**35921**	**35865**	36792
Ta089	8341	7894	7832	7889	**Ta119**	38339	**35739**	**35695**	36885
Ta090	8489	7929	7906	7918	**Ta120**	38540	**36120**	**35971**	36184

2.6.5.3. Discussion et synthèse

En dernière analyse, les résultats de calcul démontrent l'efficacité et la performance de l'approche SAGA proposée pour résoudre le problème Fm |block| Cmax. La principale raison pour laquelle le SAGA peut être plus performant que l'AG traditionnel et les autres métaheuristiques est due à la combinaison des avantages de l'AG et le processus de refroidissement de l'algorithme SA en échappant à l'optimum local dans lequel, l'AG peut être piégé.

À partir des résultats, nous avons pu conclure que combinées ensemble, ces deux structures d'optimisation améliorent la qualité des solutions produites et compensent bien entendu l'inconvénient majeur de la méthode des algorithmes génétiques constatés auparavant, ce qui permet de confirmer le besoin de considérer implicitement l'algorithme local dans la recherche génétique.

2.7. Conclusions

Au cours de ce chapitre, une résolution exacte permettant d'évaluer la performance du modèle établi a été effectuée, cette résolution nous a permis de confirmer la conclusion décelée auparavant concernant la complexité algorithmique du problème. Dans le même ordre d'idées, cette dernière a fait l'objet d'un tremplin pour l'ouverture sur une résolution adéquate du problème considéré vu qu'en s'appuyant sur une méthode exacte, le temps moyen de calcul augmente considérablement avec la taille du problème.

Dans cet esprit, des méthodes approchées ont été proposées pour la résolution du problème d'optimisation mono-objectif avec contrainte de blocage. La première approche de résolution a porté sur l'application des algorithmes génétiques dans lesquels les résultats obtenus ont mis en lumière leurs faiblesses qui réside dans leurs convergences prématurées. Ce constat nous a laissé entrevoir de réelle possibilité de combinaisons approchées. Par conséquent, une résolution à travers une hybridation séquentielle a été suggérée afin de pallier à l'inconvénient majeur des AGs, cette combinaison a donné lieu à un algorithme nommé SAGA. De plus, étant donné que les paramètres du problème et des méthodes utilisées nécessitent un calibrage empirique, la méthode de Taguchi a été utilisée permettant de sélectionner les paramètres menant à une résolution optimale. Au final, une étude comparative a été menée sur les résultats numériques obtenus par l'hybridation des algorithmes génétiques avec le recuit simulé suggérée et ceux obtenus par les algorithmes robustes existants dans la littérature. L'analyse de l'étude comparative établie a décelé que, l'approche que nous avons établi fourni des résultats pertinents sur les différents tests effectués comparativement aux meilleurs résultats trouvés auparavant dans la littérature pour la minimisation du makespan dans un atelier flow-shop soumis à des contraintes de blocage.

A ce stade de notre travail, nous disposons d'un nouvel outil d'optimisation pour les problèmes d'ordonnancement mono-objectif. Le chapitre qui suit portera sur le développement d'un modèle mathématique permettant de prouver l'état d'optimalité à partir de l'évaluation numérique du problème d'optimisation multi-objectif fixé en analyse bibliographique. Il s'agit du problème d'ordonnancement d'atelier flow-shop avec capacité de stockage limitée et dont l'objectif principal est la minimisation du makespan et du maximum des retards absolus, respectivement.

Chapitre III

Résolution des problèmes d'ordonnancemen des systèmes flow-shop multi-objectifs avec contrainte de blocage et de disponibilité des tâches

Chapitre 3
Résolution des problèmes d'ordonnancement des systèmes flow-shop multi-objectifs avec contrainte de blocage et de disponibilité des tâches

3.1. Introduction

Dans un environnement industriel, les praticiens sont confrontés quotidiennement à des problèmes d'ordonnancement de complexité grandissante, qui surgissent dans des domaines très variés. Toutefois, les problèmes à résoudre sont fréquemment de nature multi-objectifs où ils sont amenés à tenir compte simultanément de deux ou plusieurs critères d'optimisation contradictoires.

Dans cet esprit, la qualité des solutions fournies à travers la résolution mono-objectif décrite dans le chapitre précédent nous laisse entrevoir de réelles possibilités d'optimisation multi-objectifs des problèmes d'ordonnancement des ateliers flow-shop vu la difficulté pratique et théorique qu'ils soulèvent.

Cependant, la résolution des problèmes d'ordonnancement de type flow-shop mono-objectif a fait l'objet de nombreuses études approfondies. Néanmoins, d'après l'examen des processus industriels réels, une seule mesure de performance est considérée comme insuffisante pour l'amélioration des performances des systèmes industriels. Plus concrètement, la majorité des études menées dans la littérature pour le traitement de l'environnement d'ordonnancement étudié minimisent uniquement le makespan qui représente le temps de réalisation maximal des jobs. Bien que la minimisation du makespan génère sans aucun doute, la maximisation de l'utilisation des ressources, la réduction du temps et du coût de production, l'augmentation du débit, l'utilisation modérée de l'électricité etc.…, mais le respect du délai de livraison des tâches n'est pas garanti par l'optimisation de cette mesure de performance. Dans ces circonstances, pour s'assurer que chaque job est consigné à temps et garantir un niveau de service élevé, la minimisation du maximum des retards absolus est une autre mesure de performance à laquelle il faut également prêter beaucoup d'attention et fournir énormément d'efforts.

Dans cet ordre d'idées, ce chapitre s'articule principalement autour de la résolution de deux types de problèmes d'ordonnancement multi-objectifs, le premier étant le $(F_m|block|C_{max}, T_{max})$ tandis que le deuxième problème représente le même système avec une contrainte de disponibilité des tâches supplémentaire $(F_m|block, r_k|C_{max}, T_{max})$. Pour ce faire, deux types de résolutions sont effectuées pour résoudre les systèmes précités, la première résolution est exacte, elle sera effectuée en vue de prouver l'état d'optimalité et le dégrée de complexité du système étudié, cette étape est mise en œuvre à travers le solveur CPLEX décrit précédemment. Tandis que la deuxième résolution gravite autour d'une résolution approchée à travers une amélioration du fameux algorithme génétique élitiste de tri non-dominé (NSGA-II). Cette amélioration réside dans la première phase du mécanisme génétique afin d'améliorer précocement la convergence et la qualité des solutions produites.

3.2. Intérêt de la résolution du système multi-objectifs avec blocage

Au cours de la dernière décennie, l'optimisation combinatoire multi-objectifs a donné un formidable élan à la recherche opérationnelle, non seulement en raison du caractère multi-objectifs

de la majorité des applications réelles, mais aussi parce qu'il existe encore de nombreuses interrogations théoriques dans ce domaine.

De grands efforts ont été consacrés depuis plusieurs années, à l'étude des problèmes d'ordonnancement multi-objectifs d'atelier de type flow-shop avec les contraintes considérées. La première étude étant de [DAN, 1990], ces chercheurs ont proposé des procédures constructives permettant d'évaluer le compromis entre le makespan et le maximum des retards absolus pour la résolution du problème d'ordonnancement d'atelier de type flow-shop de permutation. En outre, [MUR, 1996] ont proposé un cadre d'algorithme génétique (MOGA) pour le problème d'ordonnancement d'atelier flow-shop avec la minimisation du makespan et du retard total d'une part, et du makespan, des pénalités du retard total et du temps d'écoulement total d'autre part; ils étaient axés sur la mise en œuvre de deux caractéristiques : l'une concerne la procédure de sélection qui a été basée sur des pondérations variables attribuées à chaque objectif, l'autre se focalise sur une stratégie de préservation de l'élitisme qui utilise plusieurs solutions au lieu d'une solution élite unique. [ISH, 1998] ont suggéré une approche hybride de recherche génétique locale (IMMOGLS) en appliquant la recherche locale à chaque individu généré par la recherche génétique. [CHA, 1999] ont proposé une technique de recuit simulé pour le même problème et ont comparé les résultats obtenus avec ceux de [DAN, 1990]. [ZIT, 1999] ont proposé un algorithme d'évolution de Pareto (SPEA) caractérisé par des solutions de tri externes non dominées et l'intégration d'une procédure de regroupement permettant de réduire l'ensemble non dominé sans détruire ses caractéristiques. [ISH, 2003] ont introduit un IMMOGLS2 en utilisant quelques variantes du MOGLS ; leurs techniques ne sélectionnent que les bonnes solutions comme solutions initiales en se basant sur la recherche locale. Dans [TOK, 2004], les auteurs ont manipulé le problème du flow-shop de permutation à deux machines afin de minimiser le makespan et la précocité maximale. Pour ce faire, ils ont développé une Branch and Bound et une heuristique pour former toutes les solutions efficaces en fonction de deux critères. [ALL, 2004] a proposé une nouvelle heuristique pour traiter le même problème abordé par [DAN, 1990] et [CHA, 1999]. [ARM, 2004] ont présenté un algorithme de recherche tabou révisé pour un problème d'atelier flow-shop bi-critères. [PON, 2004] ont introduit une procédure de recherche multi-objectifs TSP-GA pour un problème de permutation basé sur la somme pondérée du makespan, des temps d'écoulement moyens et des temps d'arrêt machine ; où la population initiale a été formée par la solution obtenue initialement par l'algorithme du voyageur de commerce. [LOU, 2005] ont introduit une approche multi-objectifs basée sur le recuit simulé pour trouver des solutions potentiellement efficaces pour le flow-shop de permutation en incorporant un ensemble de solutions non dominées. Un problème d'ordonnancement en considérant le makespan et le temps d'exécution total comme critères de performance a été traité par [RAV, 2005]. Dans [RAH, 2007], un nouvel algorithme MOSFLA (multi-objectif shuffled frog-leaping algorithm) a été présenté pour rechercher localement des solutions optimales de Pareto-optimal en vue de minimiser la date d'achèvement moyenne pondérée et le retard moyen pondéré de traitement. [TAV, 2007] ont suggéré un algorithme immunitaire multi-objectifs efficace (MOIA) pour résoudre le même problème d'ordonnancement. Les résultats obtenus ont été comparés avec l'algorithme d'évolution de Pareto (SPEA-II). [QIA, 2009] ont proposé un algorithme hybride HDE pour traiter le problème de planification des ateliers flow-shop de permutation multi-objectifs avec des tampons limités entre machines consécutives. Ils ont d'abord appliqué une règle de la valeur de l'ordre la plus élevée (VVO) pour convertir les valeurs continues des individus dans DE en séquences des jobs. Ensuite, ils ont mis l'accent sur l'exploitation par l'utilisation de la procédure de recherche locale. [WEI, 2011] ont suggéré une procédure de recherche locale multi-objectifs (MOLS) incorporée à la NSGA-II pour minimiser simultanément le makespan et le retard maximal du problème d'ordonnancement d'atelier flow-shop. Ils ont appliqué des structures de voisinage de MOINS (insertion multi-objectif) et MOEX

(échange multi-objectif) pour améliorer l'efficacité de la permutation. [JOL, 2013] ont proposé trois approches bi-objectives basées sur le recuit simulé, connues sous le nom de recuit simulé pondéré classique (CWSA), recuit simulé pondéré normalisé (NWSA) et recuit simulé flou (FSA) pour réduire au maximum le makespan et les retards. Dans les travaux de [HAN, 2014], une NSGA-II améliorée pour résoudre le problème de l'ordonnancement des ateliers flow-shop avec quatre critères est proposée. Premièrement, ils ont utilisé quatre variantes de l'heuristique NEH pour former les individus initiaux, puis ils ont incorporé l'estimation de l'algorithme de distribution (EDA) et un opérateur de mutation basé sur l'insertion et l'échange dans NSGA-II pour remplacer les opérateurs traditionnels de croisement et de mutation. [SMU, 2015] ont proposé une nouvelle version de l'algorithme de recuit simulé (VESA) pour résoudre le problème en considérant un nuage de solutions candidates au lieu d'une seule solution, en d'autres termes, un vecteur de traitement parallèle et des permutations évaluées en parallèle étaient a été considéré comme approximatifs à la frontière de Pareto. [KUM, 2016] ont proposé une nouvelle approche méta-heuristique de l'algorithme de recherche locale d'émulation gravitationnelle modifiée (MGOELS) pour résoudre les problèmes d'ordonnancement d'atelier de permutation avec la minimisation du makespan et du temps de découlement total. Récemment, [NOU, 2017] ont développé une nouvelle technique d'algorithme génétique basée sur NSGA-II pour minimiser simultanément le makespan et le temps total d'exécution, cette technique est appelée MGBA, elle incorpore une variante spécifique de l'heuristique NEH dans la génération de l'étape initiale des solutions pour rechercher localement la frontière Pareto-optimal pour l'atelier flow-shop avec contrainte de blocage.

Comme ce qui précède ressort, différentes combinaisons de critères et de contraintes ont été considérées pour entamer une résolution multi-objectifs en théorie d'ordonnancement. Or, les problèmes $F_m|block|, C_{max}, T_{max}$ et $F_m|block, r_k|, C_{max}, T_{max}$ n'ont pas encore été abordé d'une manière exhaustive dans la littérature. En effet, notre contribution vise particulièrement à proposer une approche de résolution efficace pour résoudre les problèmes d'ordonnancement qui minimise simultanément le makespan et le maximum des retards absolus dans un atelier flow-shop avec capacité de stockage limitée entre les machines consécutives et avec des dates de disponibilité différentes sur la première machine.

3.3. Formulation mathématique

3.3.1. Modèle mathématique du flow-shop multi-objectifs avec blocage

En se basant sur les mêmes hypothèses décrites dans le chapitre précèdent pour modéliser le système mono-objectif qui minimise le makespan sous contrainte de blocage. Nous présentons dans cette section la formulation mathématique permettant de modéliser cette fois-ci le système multi-objectifs qui minimise simultanément le makespan et le maximum des retards absolus sous contrainte de blocage.

Pour ce faire, soit $\prod$= (π(1), π(2), …, π(n)) une solution optimale du problème d'ordonnancement de l'atelier flow-shop avec contrainte de blocage. Où, π(i) correspond à la position du job i dans la séquence $\prod$, N représente le nombre de jobs, M désigne le nombre de machines, $t_{\pi(i),j}$ correspond au temps opératoire du job qui se trouve dans la position i sur la machine j, $d_{\pi(i)}$ est la date d'échéance du job qui se trouve dans la position i dans la séquence , $T_{\pi(i)}$ désigne le retard de chaque job dans la séquence, $C_{\pi(i),j}$ es la date de fin de chaque job sur la machine j et $H_{\pi(i),j}$ représente la date de début du job i sur la machine j. Dans ces circonstances, le makespan et le maximum des retards absolus peuvent être récursivement calculés comme suit :

$$H_{\pi(i),j} = Max(C_{\pi(i-1),j}, H_{\pi(i-1),j+1} + C_{\pi(i),j-1}) \quad (18)$$

$$C_{\pi(i-1),j}=H_{\pi(i-1),j}+t_{\pi(i-1),j} \quad (19)$$

$$C_{\pi(i),j-1}=H_{\pi(i),j-1}+t_{\pi(i),j-1} \quad (20)$$

$$T_{\pi(i)}=Max\ (C_{\pi(i),M}-d_{\pi(i)},0) \quad (21)$$

$$C_{max}=\ C_{\pi(N),M} \quad (22)$$

$$T_{max}=\underset{1<i<n}{max}\left(T_{\pi(i)}\right) \quad (23)$$

Où, i=1…n and j =1…m, notablement, Sachant que $C_{\pi(N),M}$ représente le temps d'achèvement du dernier job sur la dernière machine, le temps total requis pour accomplir le traitement de tous les jobs nommés makespan C_{max} est exprimé par l'équation suivante :

$$C_{max}=\ C_{\pi(N),M} \quad (24)$$

De plus, le retard du job en position i dans la permutation est obtenu par l'équation suivante :

$$T_{\pi(i)}=Max\ (C_{\pi(i),M}-d_{\pi(i)},0) \quad (25)$$

Au final, le retard maximal T_{max}est exprimé par l'équation suivante :

$$T_{max}=\underset{1<i<n}{max}\left(T_{\pi(i)}\right) \quad (26)$$

Pour l'optimisation du problème G (π) = ($g_1(\pi), g_2(\pi)$) bi-critères considéré, l'objectif est de trouver un ensemble de solutions optimales à savoir Pareto optimal qui optimise simultanément l'ensemble des mesures de performance considérées qui sont souvent en conflit avec elles-mêmes. En d'autres termes, la minimisation du makespan et du retard maximum G (π) = (Cmax (π), Tmax(π)) consiste à déterminer toutes les permutations propres au front de Pareto comprenant toutes les solutions non dominées. En fait, une solution π domine π* si et seulement si :

$$Cmax\ (\pi) \leq Cmax\ (\pi^*) \quad \text{et}\ \ Tmax(\pi) \leq Tmax(\pi^*) \quad (27)$$

$$Cmax\ (\pi) < Cmax\ (\pi^*) \quad \text{ou}\ \ Tmax(\pi) < Tmax(\pi^*) \quad (28)$$

En effet, comme l'objectif final de la résolution multi-objectif réside dans la fixation de toutes les solutions non-dominées qui dominent les autres mais qui ne dominent pas elles-mêmes, il est nécessaire de trouver toutes les solutions qui ne conduisent pas à l'amélioration d'une fonction objectif au détriment de la détérioration d'une autre.

Pour illustrer l'impact de la contrainte de blocage sur le makespan et le maximum des retards absolus du système étudié, nous considérons dans la figure 3.1 un exemple d'un flow-shop à 4 jobs et 4 machines. Les temps opératoires de chaque job sont donnés dans la matrice suivante $[t]_{4,4}$.

$$[t]_{4,4}=\begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix}=\begin{bmatrix} 1 & 1 & 2 & 1 \\ 2 & 1 & 2 & 2 \\ 3 & 2 & 1 & 2 \\ 1 & 1 & 2 & 3 \end{bmatrix}$$

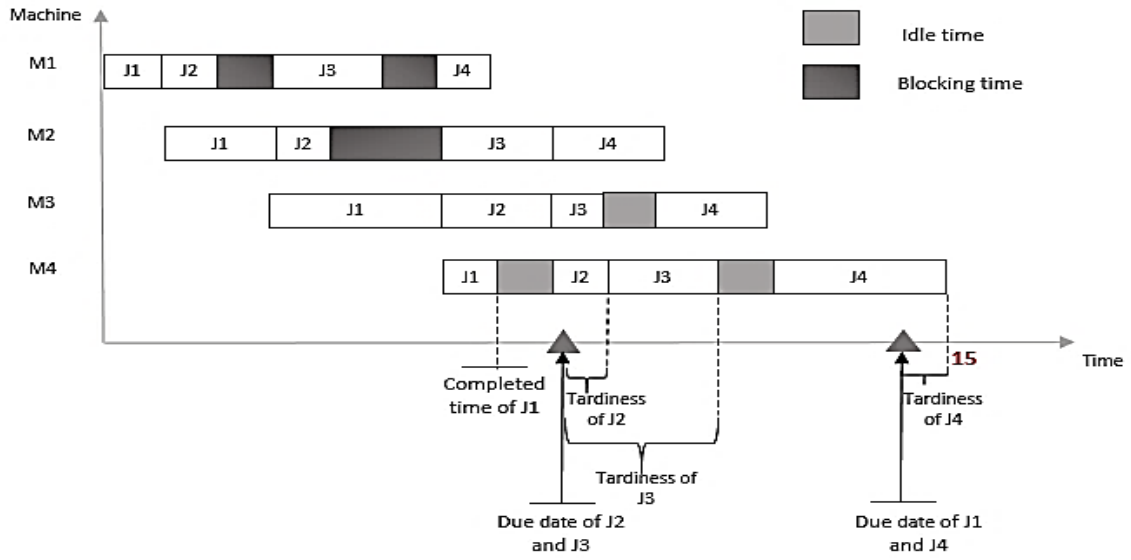


Figure 3.1: Exemple illustratif du flow-shop multi-objectifs avec blocage

La figure 3.1 représente un diagramme de Gantt du problème considéré. À partir de cette figure, nous pouvons constater que le job J3 ne peut pas commencer son opération sur la machine M1 tant que le job J2 n'a pas encore achevé son opération sur la même machine. Cependant, le job J2 ne peut pas quitter la première machine M1 en raison de l'absence de stocks intermédiaires d'une part et d'autre part la machine en aval n'est pas disponible pour traiter son opération. Le temps de blocage généré par le job J2 est égal à 1 ut, ce qui impacte la date de début du job J3 sur la machine M1 et par conséquent, la date de début de toutes les tâches successives. Le contexte de blocage empêche ainsi le traitement entier des jobs sur la machine restreinte, ce qui justifie l'augmentation respective du makespan et du retard des jobs.

3.3.2. Programmation linéaire en nombres entiers pour le flow-shop multi-objectifs avec blocage et disponibilité des tâches

Le système que nous allons traiter dans cette section regroupe l'ensemble des contraintes et des critères abordés précédemment en rajoutant la contrainte de disponibilité étant donnée la place importante qu'elle occupe dans la littérature. Dans ce contexte, le problème $F_m|block, r_k|C_{max}, T_{max}$ consiste en un ensemble $(J_1, \dots, J_n)$ de tâches qui doivent être exécutées séquentiellement sur un ensemble $(M_1, \dots, M_m)$ de machines dans le même acheminement du processus. Sachant que, chaque tâche possède une date d'échéance d_k et une date de disponibilité r_k pour laquelle elle est présumée être prête pour le traitement de sa première opération sur la première machine. En outre, une tâche est composée de O_{ki} opérations constructives $(O_{k1}, \dots, O_{km})$, où, chaque opération a un temps d'exécution entier positif égal à t_{ki}. Nous supposons ainsi que :

- Il n'y a pas de zones de stockage entre les machines et que le job peut être bloqué jusqu'à ce que la machine en aval soit libre ;
- Chaque opération ne peut être effectuée que sur une seule machine et chaque machine ne peut traiter qu'une seule opération à la fois ;
- Les temps de préparations et de réglages sont inclus dans les temps opératoires des opérations ;
- Le processus ne peut être interrompu à aucun moment, pour être repris ultérieurement sur une autre machine ;

- Il est présumé qu'il n'y a pas de temps mort sur la première machine et qu'il n'y a pas de temps de blocage sur la dernière machine.

Afin d'illustrer l'impact de la contrainte de disponibilité des tâches sur les problèmes d'ateliers flow-shop avec blocage, un exemple est fourni dans la figure 3.2, il illustre un cas simple du problème mentionné ci-dessus, où trois machines et quatre jobs sont considérés. Étant donné que la séquence de traitement est la suivante : P = (J_1, J_2, J_3, J_4), les dates de disponibilité des tâches sur la première machine sont considérées comme suit r_k = [2, 3, 8, 4] et les temps d'exécution des travaux sur les machines sont regroupés dans la matrice suivante.

$$t_{k,i}=\begin{pmatrix}1 & 3 & 6 & 2\\ 4 & 2 & 2 & 3\\ 2 & 1 & 3 & 3\end{pmatrix} \qquad k \in \{1,..,3\},\ i \in \{1,..,4\}$$

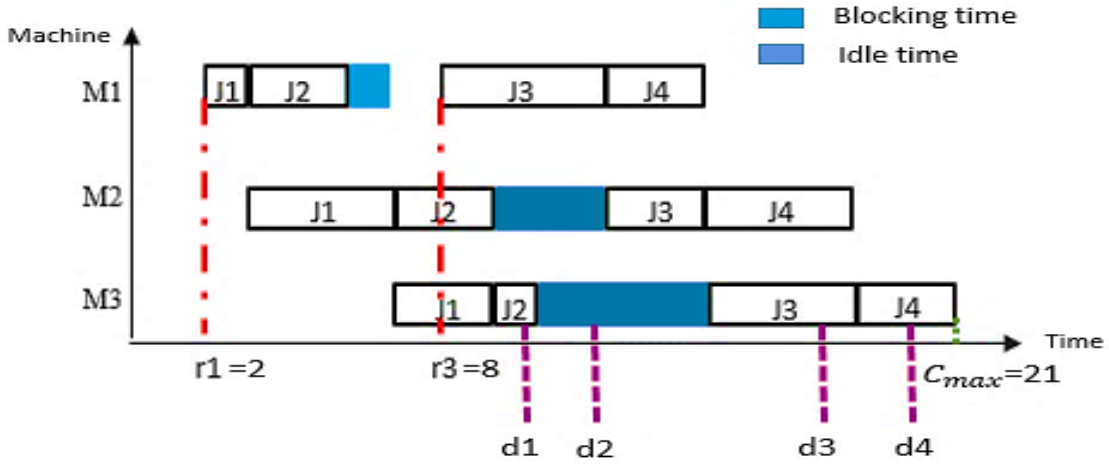


Figure 3.2: Exemple illustratif du flow-shop multi-objectifs avec blocage et la contrainte de disponibilité des tâches

À partir de l'exemple illustré sur la figure 3.2, nous pouvons constater la présence de la contrainte de blocage entre les machines M1 et M2. Ce blocage survient du fait que le job J2 reste bloqué sur la machine M1 jusqu'à ce que la machine M2 libère l'opération du job J1 comme l'espace de stockage entre les machines consécutives est nulle. Nous remarquons ainsi qu'il existe un temps non productif pendant lequel les jobs ne sont pas prêts pour commencer leurs opérations sur la première machine. Ainsi, même si la première machine M1 se libère après le blocage pour commencer le traitement de la première opération du Job J3, elle reste improductive jusqu'à ce que le job J3 soit prêt pour le traitement. Ce qui permet de mettre en lumière l'impact de la contrainte de disponibilité des tâches sur le système et les critères à optimiser.

Vu que la cible de cette étude est d'obtenir la meilleure séquence, qui minimise la somme pondérée du maximum des retards absolus et du makespan, respectivement. Selon [DAN, 1990], ces critères de performance peuvent être formulés comme suit :

$$C_{max} = \max_{j\in[1....n]} C_{jm} \tag{29}$$

$$T_{max} = \max_{j\in[1....n]} T_{jm} \tag{30}$$

Où, $j = 1, \dots, n$, m représente la dernière machine du système de production. C_{jm} désigne la date d'achèvement de chaque job j sur la dernière machine Mm et $T_{jm} = \max(C_{jm} - d_k, 0)$ représente le retard du job j.

Il convient de signaler que la somme pondérée est l'une des approches scalaires types les plus couramment utilisées, compte tenu de sa maîtrise et son efficacité dans la résolution des problèmes d'optimisation convexes. Cette technique intuitive consiste à transformer un problème multi-objectifs en un problème qui combine les différentes fonctions objectif du problème en une seule fonction objectif de façon linéaire.

La façon la plus simple de cette technique est d'attribuer à chaque fonction objectif un facteur de pondération w_i approprié fourni par l'utilisateur en fonction de l'importance relative à chaque objectif, afin d'obtenir une fonction d'objectif scalaire agrégée. Le produit qui fournit l'objectif scalaire $F(x, w)$ peut être posé comme suit :

$$F(x, w) = \sum_{i=1}^{p} w_i f_i(x) \quad (31)$$

Où les valeurs w_i sont comprises dans l'intervalle [0,1]. Ces poids doivent être normalisés si les échelles et les unités des objectifs sont différentes [RON, 2012], ainsi ils doivent vérifier $\sum_{i=1}^{p} w_i = 1$.

Vu que les échelles et les unités des critères considérés dans notre problématique sont similaires, la fonction objective du problème à l'étude peut être calculée comme suit :

$$Z = \alpha\, C_{max} + \beta\, T_{max} \quad (32)$$

Où, α et β représentent les poids du makespan et du maximum des retards absolus. Tant que $0<\alpha, \beta<1$ et $\alpha+\beta=1$, la fonction objective peut alors être exprimée de la manière suivante :

$$Z = \alpha\left(\max_{j\in[1\dots n]} C_{jm}\right) + (1-\alpha)\max\left(\max_{j\in[1\dots n]}\left(C_{jm} - d_k, 0\right)\right) \quad (33)$$

En se basant sur ce qui a été précité, un modèle mathématique de programmation linéaire en nombre entier (PLNE) est proposé dans la section suivante pour résoudre le problème $F_m|block, r_k|C_{max}, T_{max}$. En considérant la contrainte r_k, ce modèle est inspiré du modèle présenté dans les travaux de [TRA,2012] pour minimiser le makespan dans un flow-shop avec un blocage mixte entre les machines consécutives. Nous avons adapté notre modèle pour tenir compte à la fois des différentes contraintes et l'ensemble des critères d'optimisation considérés. Les sous-sections suivantes présentent l'ensemble des indices, des paramètres et des variables utilisés pour le développement du PLNE en question.

3.3.2.1. Indices

k Indice sur le job.
i Indice sur la machine.
j Indice sur la position du job k dans la séquence P.

3.3.2.2. Paramètres

n Nombre total des jobs, k =1,2,…,n.
m Nombre total des machines, i= 1,2,…,m.
t_{ki} Temps d'exécution du job k sur la machine i, k =1, 2…,n , i= 1,2,…,m.
D_k Date souhaitée au plus tard du job k (la date d'échéance du job k) k =1,2,…,n.
r_k Date de disponibilité du job k, k =1, 2,…,n.

α Le poids associé au makespan.
β Le poids associé au maximum des retards absolus.
Z(p) La fonction objectif.

θ_k: $\begin{cases} 1 & \text{S'il s'agit d'un probléme sans contrainte de blocage et sans temps mort} \\ 0 & \text{Sinon} \end{cases}$

ε_k : $\begin{cases} 1 & \text{S'il s'agit d'un probléme avec contrainte de blocage entre la machine } M_i \text{ et } M_{i+1} \\ 0 & \text{Sinon} \end{cases}$

μ_k : $\begin{cases} 1 & \text{S'il s'agit d'un probléme avec temps mort entre la machine } M_i \text{ et } M_{i+1} \\ 0 & \text{Sinon} \end{cases}$

3.3.2.3. Variables de décision

H_{ji} : Date de début du job k à la position j sur machine i
C_{ji} : Date de fin du job k à la position j sur machine i

x_{kj} : $\begin{cases} 1 & \text{Si le job se trouve à la position j de la séquence P} \\ 0 & \text{Sinon} \end{cases}$

3.3.2.4. Modèle (PLNE)

La fonction objectif :

$$\text{Min} \quad Z(p)= \alpha\, C_{max} + (1-\alpha)\, T_{max} \quad \alpha > 0 \tag{34}$$

Sous les contraintes :

$$\sum_{k=1}^{n} x_{kj} = 1 \quad \forall j \in \{1 \dots . n\} \tag{35}$$

$$\sum_{j=1}^{m} x_{kj} = 1 \quad \forall k \in \{1 \dots . n\} \tag{36}$$

$$C_{max} \geq C_{jm} \quad \forall j \in \{1 \dots . n\} \tag{37}$$

$$T_{max} \geq C_{jm} - \sum_{k=1}^{n} x_{kj}\, D_k \quad \forall j \in \{1 \dots . n\} \tag{38}$$

$$H_{j1} \geq \sum_{k=1}^{n} r_k\, x_{kj} \quad \forall j \in \{1 \dots . n\} \tag{39}$$

$$H_{ji} \geq H_{j-1,i} + \sum_{k=1}^{n} t_{ki}\, x_{kj} \quad \forall j \in \{2 \dots . n\} \quad \forall i \in \{1 \dots . m\} \tag{40}$$

$$H_{ji} \geq H_{j,i-1} + \sum_{k=1}^{n} t_{ki-1}\, x_{kj} \quad \forall j \in \{1 \dots . n\} \quad \forall i \in \{2 \dots . m\} \tag{41}$$

$$H_{ji} \geq H_{j-1,i+1} \quad \forall j \in \{2 \dots . n\} \quad \forall i \in \{2 \dots . m\} \tag{42}$$

$$H_{ji} \geq C_{j-1,i}\theta_k + H_{j-1,i+1}\, \varepsilon_k + C_{j,i-1}\, \mu_k \quad \forall j \in \{2 \dots . n\} \quad \forall i \in \{2 \dots . m\} \tag{43}$$

$$C_{j1} \geq \sum_{k=1}^{n} r_k\, x_{kj} + \sum_{k=1}^{n} t_{k1}\, x_{kj} \quad \forall j \in \{1 \dots . n\} \tag{44}$$

$$C_{ji} \geq H_{ji} + \sum_{k=1}^{n} t_{ki}\, x_{kj} \quad \forall j \in \{1 \dots . n\} \tag{45}$$

$$C_{jm} \geq H_{jm} + \sum_{k=1}^{n} t_{ki}\, x_{kj} \quad \forall j \in \{1 \dots . n\} \tag{46}$$

$$x_{kj} \epsilon \{0,1\} \quad \forall k \in \{1 \dots . n\} \; \forall j \in \{1 \dots . n\} \tag{47}$$

$$H_{ji} \geq 0 \quad \forall j \in \{1 \dots . n\} \quad \forall i \in \{1 \dots . m\} \tag{48}$$

$$C_{ji} \geq 0 \quad \forall j \in \{1 \ldots . n\} \quad \forall i \in \{1 \ldots . m\} \tag{49}$$

3.3.2.5. Signification des équations

- L'équation 34 calcule la fonction objectif qui porte sur la minimisation de la somme pondérée du makespan eu maximum des retards absolus ;
- L'équation 35 garantit que chaque position de la séquence n'est affectée qu'un seul job ;
- L'équation 36 montre que chaque job n'occupe qu'une seule position dans la séquence ;
- L'équation 37 indique que la valeur du makespan doit être supérieure ou égale à la date d'achèvement de tous les jobs sur la dernière machine. Le maximum des retards absolus est calculé dans L'équation 38 ;
- L'équation 39 force les jobs à commencer leurs traitements après leurs dates de disponibilité ;
- L'équation 40 représente la contrainte de précédence entre deux opérations qui doivent être traitées par la même machine, pour laquelle la machine ne peut commencer l'opération suivante, que lorsque l'opération précédente dans le processus est terminée ;
- L'équation 41 modélise la contrainte de précédence entre deux opérations successives d'une même tâche pour laquelle la tâche ne peut pas commencer son opération dans la machine en aval tant qu'elle n'a encore terminé pas son opération dans la machine en amont ;
- La contrainte RSB entre la machine $M_i$et la machine M_{i+1} est modélisée dans la contraintes L'équation 42 ;
- L'équation 43 désigne que, pour chaque date de début H_{ji}, une seule contrainte est active ;
- L'équation 44 calcule le temps de réalisation des jobs sur la première machine en tenant compte de la date de disponibilité de chaque job ;
- Les équations 45 et 46 correspondent respectivement au calcul du temps de réalisation des jobs dans les machines $M_i$et dans la dernière machine M_m ;
- L'équation 47 indique que x_{kj} est une variable booléenne, elle est égale à 1 si le job j_k est affecté à la j-ème position de séquence et 0 sinon ;
- Les équations 48 et 49 représentent les contraintes logiques qui énoncent la non-négativité des dates de début et de fin des jobs.

3.4. Résolution exacte du flow-shop multi-objectifs avec blocage

La performance des modèles présentés dans les sections 3.3.1 et 3.3.2 ont été évalués en se basant sur la méthode exacte Branch and Bound (B&B) intégrée sur le logiciel CPLEX décrit dans la section précédente. Cette section rapporte les résultats numériques obtenus.

Dans cette optique, nous signalons que cent instances différentes pour chaque dimension fixée de N jobs et M machines sont générées, les temps opératoires sont générés uniformément dans l'intervalle [1, 100] pour les mêmes raisons introduites auparavant. Ainsi, l'ampleur du problème varie selon les instances mentionnées ci-dessous.

(n, m) = {(10,7), (10,10), (15,3), (15,7), (20,3), (20,7)}.

Selon la conclusion décelée de plusieurs travaux de recherche qui ont été menés sur les problèmes d'ordonnancement avec le maximum des retards absolus comme mesure de performance, la difficulté du problème dépend de deux paramètres, notamment TF et DR [POT,

1982]. Le premier paramètre représente le facteur de retard qui influe sur le nombre moyen des jobs en retard et le second correspond pratiquement à la plage de dispersion des dates d'échéance qui contrôle leur variance, respectivement. Les paramètres précités constituent l'intervalle à partir duquel les dates d'échéance des jobs doivent être générées. Dans cet esprit, étant donné que P représente la somme des temps opératoires de tous les jobs, les dates d'échéance des jobs $\mathbf{d}_{\pi(\mathbf{i})}$ ou $D_k$sont générées aléatoirement en utilisant une distribution uniforme sur l'intervalle suivant :

$$d_{\boldsymbol{\pi}(\boldsymbol{i})} \; / \;\; D_k \, \epsilon \left[P\left(1 - TF - \frac{DR}{2}\right), P\left(1 - TF + \frac{DR}{2}\right)\right] \tag{50}$$

Les valeurs de TF et DR sont fixées en fonction des différents scénarios présentés dans le tableau 3.1. Ces scénarios représentent les différentes configurations des paramètres TF et DR adoptées par [POT, 1982].

Tableau 3.1 : Différents scénarios de paramétrage des facteurs TF et DR

Scénarios	Facteur du retard (TF)	Facteur d'échéances (DR)
Scénario 1	0.2	0.6
Scénario 2	0.2	1.2
Scénario 3	0.4	0.6
Scénario 4	0.4	1.2

Les dates de disponibilité des tâches r_k est un autre paramètre entier généré à partir d'une distribution uniforme (0, QP) pour la résolution du PLNE avec disponibilité des tâches, comme indiqué par [PAR, 2015]. Où, Q est un facteur qui détermine la plage de distribution, il a été fixé à 0,4 et P représente la somme des temps opératoires de tous les jobs. De plus, le poids du facteur de pondération prend trois valeurs différentes de 0,25, 0,5 et 0,75, respectivement pour chaque taille du problème.

Les modèles ont été codés dans IBM ILOG AMPL 12.6.0 et résolus à l'aide du solveur CPLEX 12.6.0 implémenté dans un ordinateur équipé d'un microprocesseur ayant 4,0 Go de mémoire vive (RAM). De plus, une limite de 3600 secondes a été imposée dans laquelle une solution optimale doit être fournie sinon le processus doit).

Les temps de calcul moyens obtenus en utilisant le logiciel CPLEX 12.6 pour résoudre le problème $F_m|block|, C_{max}, T_{max}$ sont tracés sur le tableau 3.1, Ainsi que ceux fournis pour résoudre le problème $F_m|block, r_k|, C_{max}, T_{max}$ sont regroupés dans le tableau 3.2.

Tableau 3.2 : Temps d'exécution moyens par problème multi-objectifs avec blocage RSB

n	m	TF	DR	Temps moyen (Secondes)
10	**7**	0.2	0.6	1.27
			1.2	1.09
		0.4	0.6	0.57
			1.2	0.63
10	**10**	0.2	0.6	2.94
			1.2	1.17
		0.4	0.6	2. 12
			1.2	1.93
15	**3**	0.2	0.6	14.56
			1.2	12.98
		0.4	0.6	12.72
			1.2	12.58

15	**7**	0.2	0.6	116.42
			1.2	81.49
		0.4	0.6	62.03
			1.2	40.18
20	**3**	0.2	0.6	141.29
			1.2	97.23
		0.4	0.6	123.22
			1.2	101.16
20	**7**	0.2	0.6	375.67
			1.2	313.79
		0.4	0.6	364.21
			1.2	223.79

Tableau 3.3 : Temps d'exécution moyens par problème multi-objectifs avec blocage et disponibilité des tâches (en secondes)

n	m	α	TF	DR	Temps moyen (Secondes)	Nœuds explorés (B & B)
10	**7**	0.25	0.2	0.6	1.78	4284
				1.2	1.88	3723
			0.4	0.6	0.89	4197
				1.2	0.85	1143
		0.5	0.2	0.6	1.09	3860
				1.2	0.67	2794
			0.4	0.6	1.04	1617
				1.2	0.95	1302
		0.75	0.2	0.6	1.94	2374
				1.2	1.93	2119
			0.4	0.6	1.25	1229
				1.2	1.00	1068
10	**10**	0.25	0.2	0.6	3.56	5784
				1.2	1.72	4604
			0.4	0.6	2. 46	3666
				1.2	2.01	3537
		0.5	0.2	0.6	5.58	8684
				1.2	3.65	4244
			0.4	0.6	2.20	5505
				1.2	1.56	5185
		0.75	0.2	0.6	4.60	5343
				1.2	3.58	4044
			0.4	0.6	3.67	4957
				1.2	2.01	3205
15	**3**	0.25	0.2	0.6	15.94	17636
				1.2	13.91	14763
			0.4	0.6	13.03	13417
				1.2	12.84	11639
		0.5	0.2	0.6	16.87	20997
				1.2	12.08	20642
			0.4	0.6	14.98	15364
				1.2	11.88	15014
		0.75	0.2	0.6	15.08	19100
				1.2	13.14	18445
			0.4	0.6	14.03	15885
				1.2	6.05	14167

15	**7**	0.25	0.2	0.6	65.74	40326
				1.2	43.91	25475
			0.4	0.6	93.03	57471
				1.2	79.25	40577
		0.5	0.2	0.6	125.08	65007
				1.2	98.11	31942
			0.4	0.6	74.08	54228
				1.2	30.11	36061
		0.75	0.2	0.6	95.11	63159
				1.2	83.13	54137
			0.4	0.6	89.18	51775
				1.2	45.09	49062
20	**3**	0.25	0.2	0.6	150.87	170900
				1.2	101.03	100749
			0.4	0.6	135.73	149889
				1.2	109.76	99500
		0.5	0.2	0.6	268.69	198901
				1.2	198.21	104379
			0.4	0.6	233.67	130612
				1.2	100.85	99794
		0.75	0.2	0.6	210.8	164491
				1.2	199.61	110355
			0.4	0.6	254.03	146881
				1.2	216.00	122471
20	**7**	0.25	0.2	0.6	399.82	233430
				1.2	345.63	230743
			0.4	0.6	397.29	221390
				1.2	296.36	212214
		0.5	0.2	0.6	419.40	323640
				1.2	401.20	283696
			0.4	0.6	399.70	303735
				1.2	321.56	232126
		0.75	0.2	0.6	419. 94	383687
				1.2	212.36	155537
			0.4	0.6	385.88	218405
				1.2	369.55	205890

Comme les tableaux 3.2 et 3.3 montrent, plusieurs facteurs liés au plan d'expérience peuvent clarifier la dissimilitude importante de la complexité des problèmes considérés. Les plus cruciaux tournent autour du facteur du retard TF et du facteur d'échéance DR. Cependant, en fixant DR et en se concentrant uniquement sur l'augmentation de la valeur du TF, il peut être perceptible que le temps de calcul moyen devient plus petit. Cela s'explique par la tendance à l'attribution d'une date d'échéance anticipée à un grand nombre de jobs lorsque le facteur de retard augmente, ce qui génère par la suite un nombre considérable de jobs en retard. D'un autre côté, quand la valeur du DR augmente, nous remarquons peu de variation temporelle dans la résolution du problème. Ce résultat peut s'expliquer par la faiblesse des limites résultantes de l'ordonnancement réalisable.

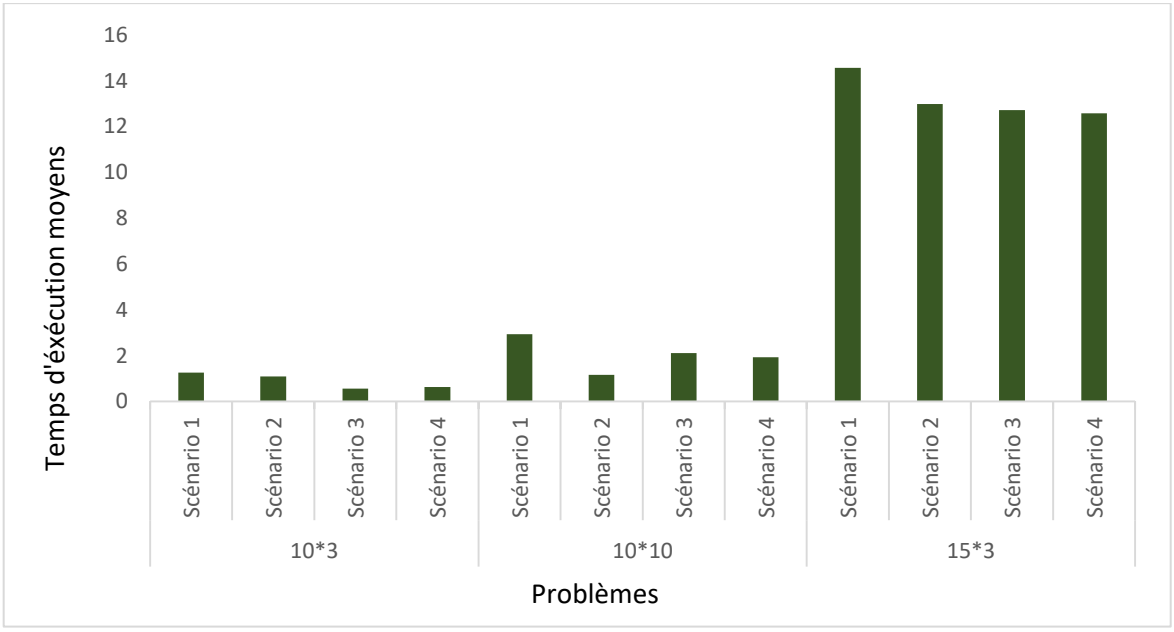


Figure 3.3 : Temps de calcul moyens pour les problèmes de petite taille

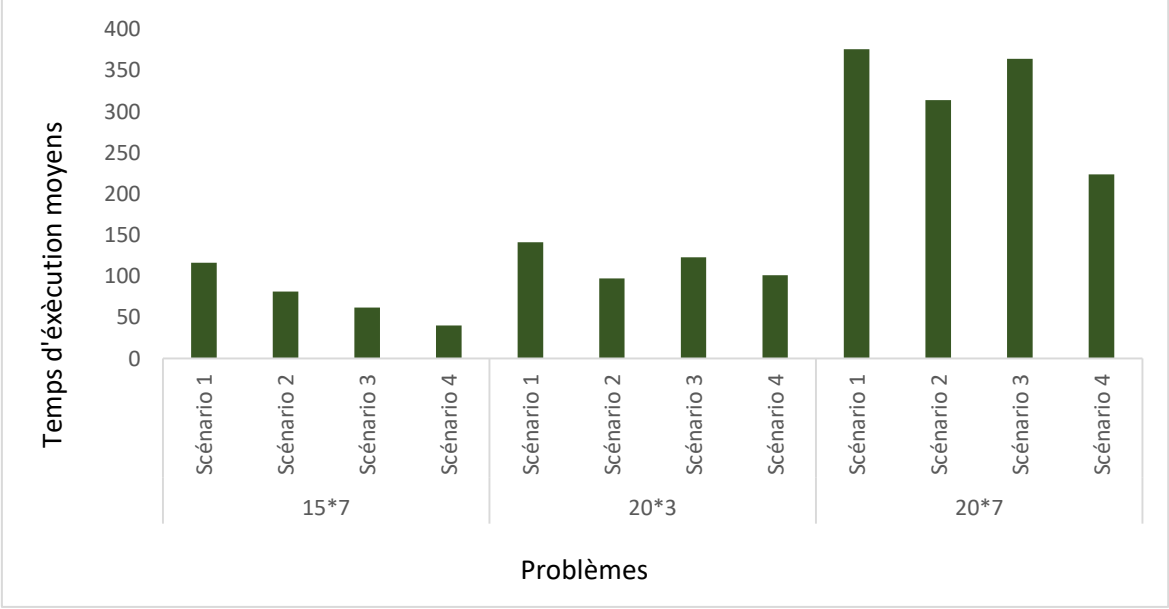


Figure 3.4 : Temps de calcul moyens pour les problèmes de grande taille

Le constat ressorti est valable pour les deux types de problèmes étudiés. Nous rajoutons à ceci, pour le problème $F_m|block, r_k|C_{max}, T_{max}$, une foi que la taille du problème dépasse 15 jobs, les temps de calcul et le nombre de nœuds sont relativement importants, ce qui est dû aux données prises en compte dans le plan d'expérience qui sont éventuellement difficiles à évaluer. Par exemple, lorsque le facteur de pondération, le facteur de retard et le facteur d'échéance sont de 0,25, 0,2 et 0,6, les temps de calcul et le nombre de nœuds explorés pour un problème de taille n=20 représentent 99,55 % et 98,16 % du temps moyens et du nombre de nœuds explorés pour un problème de taille n= 10.

Lorsqu'il s'agit d'un ordonnancement de grande taille, les temps de calcul augmentent considérablement comme illustré sur les figures 3.3 et 3.4. Au final, cette conclusion confirme ce

qui a été montré dans le chapitre précédent. Il convient de conclure que même pour les problèmes d'ordonnancement multi-objectifs avec blocage et pour les problèmes d'ordonnancement multi-objectifs avec blocage et une contrainte de disponibilité des tâches à considérer, l'usage des méthodes exactes s'avère inadéquat pour la résolution des problèmes de grande taille provenant des cas industriels réels. Il parait par contre opportun de nous pencher particulièrement vers les méthodes approchées qui sont généralement employées lorsque les problèmes d'optimisation sont justifiés par une complexité grandissante.

3.5. Résolution approchée du flow-shop multi-objectifs avec blocage

3.5.1. Introduction à l'algorithme NSGA-II

L'algorithme génétique élitiste de tri non-dominé II (NSGA-II) est une version évolutive du célèbre algorithme génétique élitiste de tri non-dominé (NSGA) proposé par [DEB., 2002] pour résoudre les problèmes d'optimisation multi-objectifs non convexes. La différence entre les deux versions réside dans l'attribution de la valeur de classement de chaque chromosome de la population et dans la technique adoptée pour maintenir la diversité de l'ensemble des solutions Pareto-optimal.

La figure 3.5 illustre clairement la procédure de la NSGA-II. De plus, cette approche élitiste actualisée qui a attiré beaucoup d'attention de la part des équipes de recherche en théorie d'ordonnancement, en particulier [WEI, 2012], [HAN, 2014] et [NOU, 2017], possède quatre caractéristiques principales, à savoir :

- Elle utilise une procédure de tri rapide et non dominée pour classer tous les individus en fonction de leur niveau de non-domination ;
- Elle intègre l'élitisme qui préserve toutes les meilleures solutions (non- dominées) de la population des parents et des enfants ;
- Elle soulève les difficultés de la fonction de partage en mettant en œuvre la technique de la distance d'encombrement de l'espace afin de maintenir la diversité et la propagation des solutions.

Elle intègre un opérateur de comparaison du degré d'encombrement de l'espace ($\prec n$) afin d'orienter la sélection aux différentes étapes de l'approche.

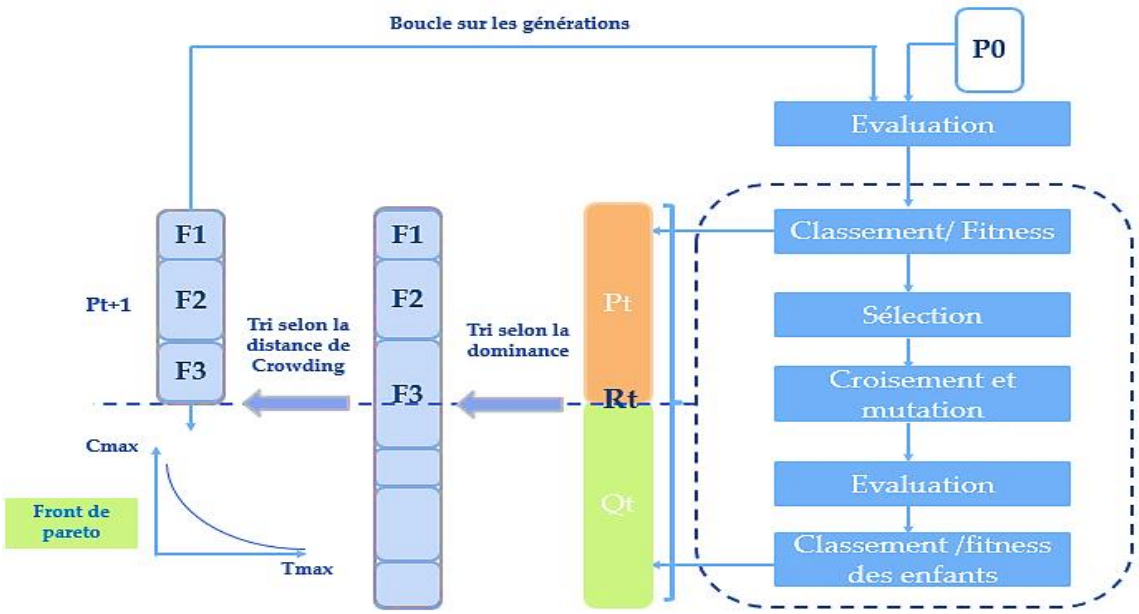


Figure 3.5 : Procédure générale de l'algorithme génétique élitiste de tri non-dominé révisé

3.5.2. Procédure de l'algorithme R-NSGA-II.ver.2

À chaque itération, de nouveaux descendants sont créés par les opérateurs classiques de croisement et de mutation appliqués à la population actuelle. Par la suite, les différents opérateurs de l'algorithme, y compris la distance d'encombrement, le tri non dominé rapide et la sélection en fonction du degré d'encombrement, sont effectués sur les deux parents et les descendants générés. Bien que les opérateurs de la NSGA-II aient démontré leur efficacité dans plusieurs domaines problématiques variés, l'initialisation de la population reste le moteur central qui assure la convergence rapide et améliore l'efficacité de l'algorithme entier. Par conséquent, une version de l'algorithme génétique de tri non dominé II révisé (R-NSGA-II), notées R-NSGA-II.ver.2 est proposée pour résoudre les deux problèmes d'ordonnancement multi-objectifs en question. Dans cette approche révisée, une version spécifique de l'heuristique NEH est appliquée lors de la phase d'initialisation de la population afin d'améliorer les caractéristiques de convergence et trouver efficacement un ensemble Pareto-optimal. Les détails des étapes les plus cruciales de la RNSGA-II.ver.2 proposée pour résoudre le problème considéré sont expliquées dans les sous-sections suivantes.

3.5.2.1. Initialisation de la population

L'heuristique NEH proposée par [NAW, 1983] a pris une importance significative en théorie de l'ordonnancement depuis son apparition. La prémisse de base de cette heuristique est que les tâches qui ont un temps opératoire total plus important doivent avoir plus de priorité comparativement aux tâches qui ont des temps opératoires moins important. En revanche, dans le cas des problèmes d'ordonnancement où la contrainte de blocage domine fortement, les jobs qui nécessitent un temps opératoire total plus élevé peuvent bloquer leurs jobs successifs et générer par la suite un temps de blocage beaucoup plus important. Par conséquent, La prémisse raisonnable que nous proposons est que la priorité doit être accordée dans ce cas aux jobs qui ont un délai de traitement plus court. Cette variante NEH appelée NEH-WPT a déjà été proposée par [WAN, 2010] pour résoudre le problème d'ordonnancement avec blocage visant à minimiser le temps total d'écoulement. Dans le présent travail de recherche, une version révisée du NEH-WPT, appelée RNEH-WPT, est déployée pour générer 4 solutions initiales visant à minimiser le makespan et le maximum des retards

absolus, respectivement. Le reste Pop_Size-4 individu sont générés aléatoirement dans l'espace de recherche pour garantir la diversité des solutions. L'algorithme suivant détaille les étapes principales de l'algorithme d'initialisation de la population RNEH-WPT proposée.

Algorithme 3.1 : Procédure d'initialisation de la population à travers l'algorithme RNEH-WPT

Étape 1. Initialiser l=1 et i=0 ;

Début

Étape 2. Calculer la somme des temps opératoires de chaque job ;

Étape 3. Fixer une séquence partielle qui arrange les jobs dans l'ordre décroissant en fonction de leurs temps totaux de traitement ;

Étape 4. i=2, extraire les deux premiers jobs de la séquence partielle, calculer la somme pondérée du makespan et du maximum des retards absolus et définir une nouvelle séquence courante ayant la meilleure fitness obtenue à partir du calcul de la somme pondérée ;

Étape 5. i=i+1, procéder au ième job dans la séquence partielle, l'insérer par la suite dans k possibles positions dans la séquence courante et sélectionner la meilleure séquence ayant la plus petite somme pondérée comme séquence courante ;

Étape 6. Si i =N, soit $\pi=\pi^*$ et aller à l'étape 7, sinon retourner à l'étape 5 ;

Étape 7. Soit l=l+1, si l=4, obtenir 4 solutions initiales et passer à l'étape 8, sinon, retourner à l'étape 1 ;

Étape 8. Le reste Popo_size-4 individus sont générés aléatoirement dans l'espace de recherche.

Fin

3.5.2.2. Codage des individus

La solution est présentée sous forme d'un vecteur de valeurs allant de 1 jusqu'au nombre des jobs N. Cette représentation de la solution est connue sous le nom permutation de jobs qui indique l'ordre de routage des jobs dans la séquence. La figure 3.6 illustre clairement le codage de la solution avec neuf jobs. Où, la première valeur indique quel job doit être exécuté en premier, la seconde indique lequel doit être traité en second et ainsi de suite.

5	4	2	3	8	7	6	1	9

Figure 3.6 : Exemple de chromosome avec codage entier

3.5.2.3. Opérateurs génétiques

La population sélectionnée produit des descendants par le biais d'opérateurs de croisement et de mutation. Dans notre travail de recherche, un croisement standard en deux points avec une probabilité de croisement Pc est appliqué, dans lequel deux points sont choisis au hasard, les jobs en dehors des deux points sont hérités des parents aux descendants, les jobs restants sont délocalisés vers le deuxième parent dans le même ordre de gauche vers la droite. Par conséquent, deux descendants sont générés. Pour chaque deux descendants générés par la technique du

croisement standard en deux points, une mutation d'échange réciproque est effectuée. Dans cet opérateur génétique, un job est retiré de sa position et mis à une autre position aléatoire avec une probabilité de mutation Pm. Il en résulte la génération d'une population descendante de taille N. Les sections 2.5.6 et 2.5.7 présentées dans le chapitre 2 sont recommandées aux lecteurs pour plus de clarification.

3.5.2.4. Algorithme de tri non-dominé rapide

La caractéristique la plus importante de l'approche NSGA-II réside dans sa technique de tri non-dominé rapide. Dans cette étape, à l'aide de deux facteurs, à savoir, le nombre de domination np qui énumère le nombre de solutions dominées et Sp qui représente l'ensemble des solutions qu'une solution domine, $O(MN^2)$ des comparaisons sont effectuées pour classer les solutions dans différents fronts non-dominés. La figure 3.7 représente un exemple une de tri rapide non dominée. Le premier front comprend toutes les solutions qui ont un nombre de domination égal à zéro. Par la suite, pour trouver les individus du second front, les particules de premier rang sont soustraites momentanément et la procédure de comparaison précitée est établie sur son ensemble de solutions dominées Sp. Ce processus est répété jusqu'à ce que chaque solution soit associée à un front. Pour plus de clarification, la procédure de tri non-dominé rapide est principalement présentée dans l'algorithme 3.2.

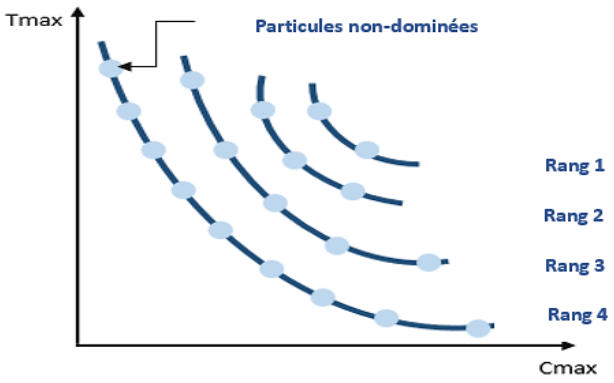


Figure 3.7 : Exemple d'un Front Pareto obtenu à travers le tri non-dominé rapide

Algorithme 3.2 : Procédure de tri non-dominé rapide

```
Pour chaque p ∈ P :
    Sp= Ø
    np=0
    Pour chaque q ∈ P :
        Si p domine q (p < q) :
            Ajouter q à l'ensemble des solutions dominées par p (Sp= Sp U {q})
        Sinon si q domine p (q < P) :
            Incrémenter le conteur de dominance (np= np+1)
    Si np=0 :
        Prank=1
        Front1= Front1 U {p}
```

```
Initialiser le conteur des fronts i=1
Tant que Fronti ≠Ø :
    Q= Ø
    Pour chaque p∈ Front1
        Pour chaque q∈ Sp
            nq=nq-1
            if nq=0 ( q appartient au prochain front)
            qrank=i+1
            Q= Q U {q}
    i=i+1
Fi=Q
```

3.5.2.5. Garantie de diversité et procédure de sélection

Une fois que la technique de tri rapide non-dominé est effectuée, la sélection du tournoi binaire est appliquée. Selon l'étude originale de [DEB, 2002], entre deux solutions de rang différent, on choisit la solution de rang inférieur. Sinon, si les deux solutions représentées appartiennent au même front ou au même rang, la solution ayant une valeur de distance d'encombrement plus élevée est privilégiée. Cette mesure de distance qui améliore la diversité des solutions permet d'obtenir la densité des solutions entourant une solution particulière dans la population, en calculant la distance moyenne entre de deux points de chaque côté de ce point et pour chacun des objectifs à optimiser. Cette mesure sert à estimer le périmètre du cuboïde formé en utilisant les voisins les plus proches comme sommets (ce qu'on appelle la distance d'encombrement). La figure 3.8 illustre le calcul de la distance d'encombrement, les points marqués en cercles remplis sont des solutions du même front non dominé.

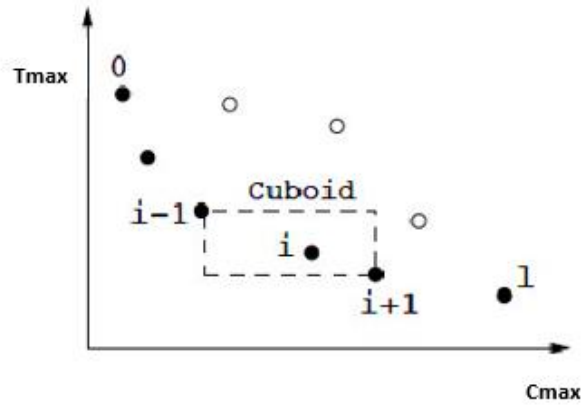


Figure 3.8 : Calcul de la distance d'encombrement

Le calcul de la distance d'encombrement nécessite le classement par ordre croissant de la population en fonction de chaque valeur des fonctions objectifs. Par la suite, pour chaque fonction objectif, une valeur de distance infinie est attribuée aux solutions situées aux bornes dont les valeurs de la fonction objectif sont la plus petite et la plus grande valeur, respectivement. Toutes les autres solutions intermédiaires s'approprient à une valeur de distance égale à la différence normalisée absolue des valeurs de deux solutions adjacentes. Ce calcul se poursuit avec les deux fonctions objectifs, la valeur globale de la distance d'encombrement est calculée comme la somme

des valeurs de distance individuelles correspondant à chaque objectif. Chaque fonction de l'objectif est normalisée avant de calculer la distance d'encombrement. L'algorithme 3.3 décrit la procédure de calcul de la distance d'encombrement de toutes les solutions d'un ensemble non-dominé. Sachant que I[i]. m se réfère à la valeur de la fonction objectif m de l'individu i dans l'ensemble I et les paramètres f_m^{max} et f_m^{min} représentent les valeurs maximales et minimales de la fonction objectif m.

Algorithme 3.3 : Procédure de calcul de la distance d'encombrement

L= |I| (le nombre de solution dans l'ensemble non-dominé I)

Pour chaque i, initialiser la distance $I[i]_{distance} = 0$

Pour chaque fonction objectif m

Classer les fitness dans un ordre croissant

$I[1]_{distance} = I[l]_{distance} = \infty$

Pour i =2 à (l-1)

$$I[i]_{distance} = +I[i]_{distance} + \frac{(I[i+1].m - I[i-1].m)}{(f_m^{max} - f_m^{min})}$$

En s'appuyant sur les rangs obtenus à travers la procédure de tri non-dominé rapide et la distance d'encombrement calculée pour chaque individu, un opérateur de comparaison encombré (<n) est appliqué pour orienter le processus de sélection au niveau de toutes les étapes de l'algorithme vers un front de Pareto optimal uniformément réparti. Dans cet opérateur, nous préférons, entre deux solutions ayant des rangs différents, la solution ayant le rang le plus bas. Sinon, si les deux solutions appartiennent au même front, nous préférons la solution qui est située dans une région moins encombrée.

3.5.2.6. Élitisme et algorithme général

À ce stade, les descendants générés sont combinés avec la population actuelle, puis la population des 2N est triée en fonction de la dominance de Pareto. À partir de là, le processus réitère pour générer les générations ultérieures. Bien que toutes les meilleures solutions précédentes et actuelles soient ajoutées dans la population, l'élitisme est bien garanti. En s'appuyant sur les techniques précitées, les étapes les plus importantes de l'approche génétique élitiste R-NSGA-II.ver.2 de tri non-dominé proposée peuvent être résumées dans l'algorithme 3.4 suivant et la figure 3.9.

Algorithme 3.4 : Procédure générale de l'approche R-NSGA-II.ver.2

Étape 1. Initialiser les paramètres ;

Étape 2. Initialiser une population de taille N :

Étape 2.1. Appliquer RNEH_WPT pour générer 4 individus ;

Étape 2.2. Générer le reste Pop_size- 4 aléatoirement ;

Étape 3. Évaluer chaque individu de la population et attribuer un rang selon la procédure de tri rapide non dominé :

Étape 4. Générer une population d'enfants de taille N :

Étape 4.1. Appliquer la sélection de tournoi binaire ;

Étape 4.2. Appliquer le croisement standard en 2-points avec une probabilité Pc ;

Étape 4.3. Appliquer la mutation d'échange réciproque une probabilité Pm ;

Étape 5. Pour i=1 jusqu'à Max_iter :

Étape 5.1. Pour chaque individu de la population combinée :

Étape 5.1.1. Attribuer un rang basé sur le tri rapide non-dominé ;

Étape 5.1.2. Déterminer la distance d'encombrement sur chaque front ;

Étape 5.2. Sélectionner les meilleures N solutions en se basant sur l'opérateur de comparaison d'encombrement d'espace ;

Étape 5.3. Pour chaque individu de la population combinée :

Étape 5.3.1. Appliquer la sélection de tournoi binaire ;

Étape 5.3.2. Appliquer le croisement standard en 2-points avec une probabilité Pc ;

Étape 5.3.3. Appliquer la mutation d'échange réciproque une probabilité Pm ;

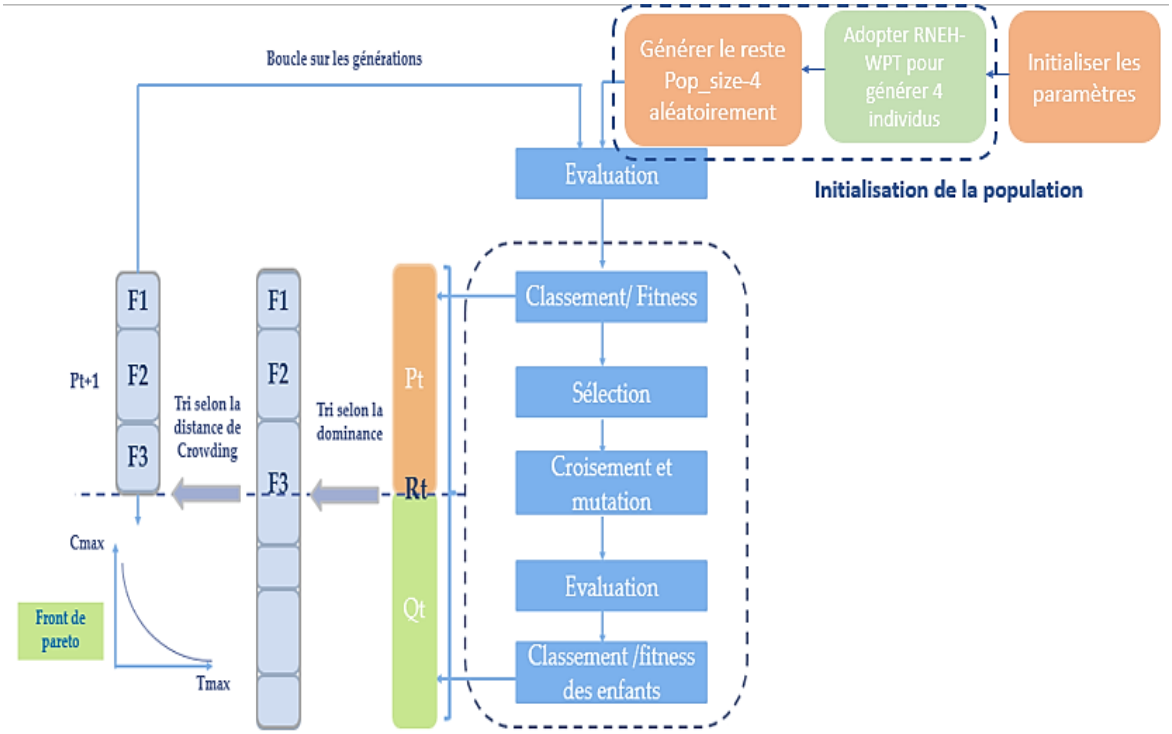


Figure 3.9 : Schéma général de la RNSGA-II.ver.2

3.5.3. Calibrage des paramètres de la R-NSGA-II.ver.2 : Plan d'expériences de Taguchi

Afin d'optimiser efficacement les deux systèmes ($F_m|block|C_{max}, T_{max}$, $F_m|block, r_k|C_{max}, T_{max}$), des tests ont été établis à travers les mesures de performances utilisées dans les plans dynamiques de Taguchi utilisés précédemment dans le calibrage des paramètres des AGs et de la SAGA. Dans le même ordre d'idées, 27 séries d'expériences ont été réalisées en s'appuyant sur la distribution de

la table orthogonale, et trois niveaux ont été affectés à chaque facteur, comme l'illustre le tableau 3.4

Tableau 3.4 : Différents niveaux de chaque paramètre de la R-NSGA-II.ver.2

Facteurs	Pop_Size	Max-iter	Pc	Pm
Niveau 1	50	50	20 %	10 %
Niveau 2	100	100	40 %	40 %
Niveau 3	200	300	60 %	70 %

Les figures 3.10 et 3.11 représentent les graphiques des effets principaux crées sur Minitab en traçant la moyenne des caractéristiques pour chaque niveau de facteur. Par ailleurs, quant au calibrage de l'approche d'optimisation en question, les quatre facteurs considérés ci-dessus doivent être ajustés tels que :

Pour le problème $F_m|block|C_{max}, T_{max}$: la taille de la population = 50, le nombre d'itérations = 100, la probabilité de mutation = 0,1 et la probabilité de croisement = 0,9.

Pour le problème $F_m|block, r_k|C_{max}, T_{max}$: la taille de la population = 200, le nombre d'itérations = 300, la probabilité de mutation = 0,4 et la probabilité de croisement = 0,4. Le tableau 3.5 regroupe le niveau de chaque facteur sélectionné en vue de contrôler et optimiser la méthode d'optimisation adoptée pour la résolution des systèmes $F_m|block|C_{max}, T_{max}$ et $F_m|block, r_k|C_{max}, T_{max}$, respectivement.

Tableau 3.5 : Paramètres optimaux de la R-NSGA-II.ver.2

Méthode	Pop_Size	Max-iter	Pc	Pm
$F_m\|block\|C_{max}, T_{max}$	50	100	60 %	10%
$F_m\|block, r_k\|C_{max}, T_{max}$	200	300	40%	40%

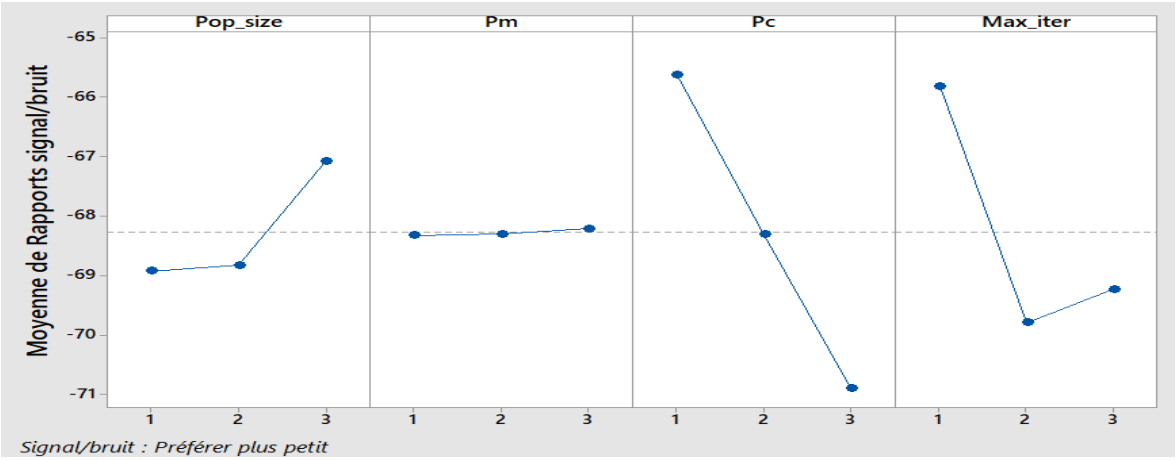


Figure 3.10 : Graphique des effets principaux du rapport (S/B) $F_m|block|C_{max}, T_{max},$

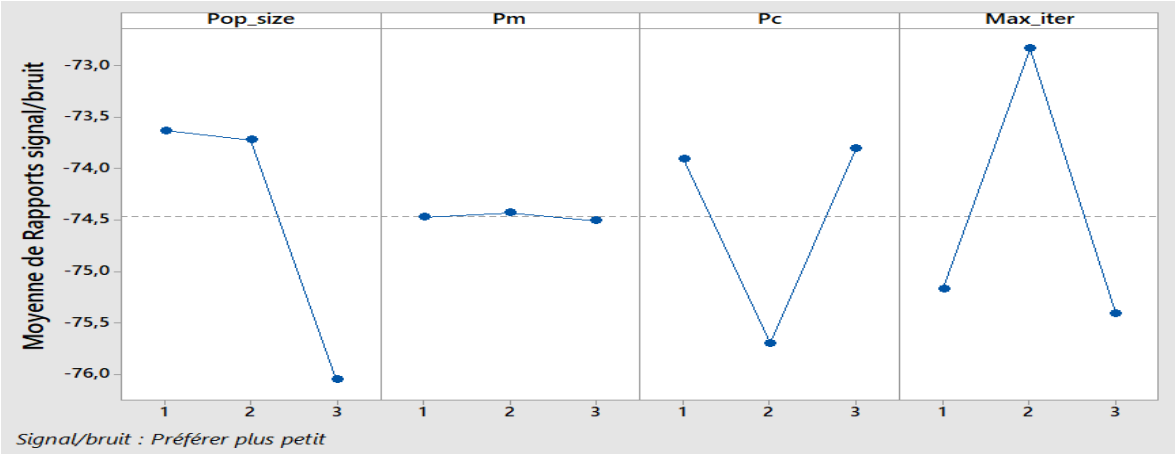


Figure 3.11 : Graphique des effets principaux du rapport (S/B) $F_m|block, r_k|C_{max}, T_{max}$

3.6. Résultats numériques

En vue d'évaluer l'efficacité de la RNSGA-II.ver.2 proposée pour résoudre les problèmes d'ordonnancement multi-objectifs étudiés, des tests ont été effectués sur six problèmes de Taillard, à savoir, (Ta031, Ta051, Ta071, Ta081 et Ta101) [TAI, 1983]. De plus, en supposant que pour chaque problème, trois situations sont étudiées, notamment :

- Première situation : capacité de stockage =0 ;
- Deuxième situation : capacité de stockage = 2 ;
- Troisième situation : les capacités de stockage entre les machines consécutives sont variables et fixées comme celles de notre référence de comparaison (Qian et al., 2009).

En somme, le réglage de ces capacités variables est représenté dans le tableau 3.6.

Tableau 3.6 : Paramétrage des capacités de stockage variables

Problème	Capacité de stockage																	
Ta031	3	2	1															
Ta051	1	3	1	1	2	3	1	3	2	3	3	1	3	3	2	1	3	2
Ta071	2	2	1	1	1	2	4	1										
Ta081	2	1	4	4	2	3	2	3	2	2	1	3	1	3	2	3	2	3
Ta101	2	2	3	2	1	1	1	2	3	2	2	2	2	2	2	3	2	4

De plus, pour avoir des comparaisons équitables, la date d'échéance de chaque job est calculée comme suit :

$$d_{\pi(i)} = C_{\pi(i)} + random\left[-\frac{C^*}{10}, \frac{C^*}{10}\right] \quad (51)$$

Où, $d_{\pi(i)}$ est la date d'échéance du job i dans la séquence π, $C_{\pi(i)}$ représente le temps d'achèvement du job i, C^* désigne le makespan optimal et $\text{random}\left[-\frac{C^*}{10}, \frac{C^*}{10}\right]$ est une valeur aléatoire dans l'intervalle $\left[-\frac{C^*}{10}, \frac{C^*}{10}\right]$.

Finalement, pour la résolution du problème avec contrainte de disponibilité des tâches les dates de disponibilité sont générées à partir d'une distribution uniforme (0, QP) comme nous l'avons effectué dans la partie de résolution exacte et indiqué dans les travaux de [PAR, 2015]. Où, Q est

un facteur qui détermine la plage de distribution et a été fixé à 0,4 et P représente la somme des temps opératoires de tous les jobs.

3.6.1. Mesures de performance

Étant donné que l'approche proposée repose sur le concept de dominance, la qualité des solutions non-dominées produites est très importante pour évaluer la performance d'un algorithme et aboutir à une comparaison correcte entre deux algorithmes multi-objectifs. Compte tenu de plusieurs facteurs, nous décrivons deux mesures de performances basées sur les solutions non-dominées obtenues en vue d'évaluer la qualité de recherche de notre algorithme vis-à-vis de l'algorithme HDE [QIA, 2009], à savoir la mesure de distance et du nombre global de solutions non-dominées, respectivement.

3.6.1.1. La mesure de distance (DM)

La mesure de distance (DM) est utilisée pour mesurer la diversité et la convergence de la solution S obtenue non dominée qui se rapporte à une solution de référence H*, elle est définie comme :

$$DM(\mathrm{S}, \mathrm{H}^*) = \frac{\sum_{z \epsilon H*} min(d(s,z) |\, s \epsilon S\,)}{|H*|} \tag{52}$$

Où d (s, z) désigne la distance entre une solution s et une référence z dans l'espace objectif normalisé dimensionnel Q.

$$d(s,z) = \sqrt{(f_1^*(z) - f_1^*(s))^2 + \cdots + (f_a^*(z) - f_a^*(s))^2} \tag{53}$$

Où $f_j^*(.)$ est le $j^{ème}$ critère normalisé par la solution de référence H*. Plus la valeur de DM (S, H*) est petite, plus l'approximation du front de Pareto est meilleure.

3.6.1.2. Le nombre global de solutions non-dominées (ONSN)

Le nombre total de solutions non dominées (ONSN) permet d'énumérer le nombre de solutions non dominées S_j par rapport à N qui ne sont dominées par aucune autre solution dans l'union des ensembles de solutions non dominées (S= S1US2U…SN), cette mesure de performance est définie comme :

$$ONSN\,(S_j) = \left|S_j - \left|\{x \epsilon S_j | \exists\, y \,\epsilon\, S : y > x\}\right|\right. \tag{54}$$

Où y>x signifie que la solution x est dominée par la solution y ; les solutions dominées x par d'autres solutions y dans S sont supprimées de S_j. Plus la valeur de ONSN (S_j) est grande, plus l'algorithme est dit performant.

3.6.2. Résultats de la RNSGA-II.ver.2

Les tableaux 3.7, 3.8 et la figure 3.12 rapportent les résultats des mesures ONSN et du DM fournis à travers la RNSGA-II.ver.2 proposée et la variation de l'ONSN en fonction de la capacité de stockage.

De plus, afin de découvrir la dispersion des solutions non dominées obtenues, les figures 3.13 et 3.14 présentent l'ensemble des solutions non-dominées obtenues pour le problème 40*20 pour les deux problèmes considérés.

D'après la littérature, il n'existe jusqu'à présent aucune référence qui traite le problème $F_m|block, r_k|C_{max}, T_{max}$. En effet, le tableau regroupant les mesures de performance ne contiendra

qu'une seule mesure de performance étant le facteur ONSN vu que le calcul de la deuxième mesure DM nécessite une solution de référence H*. De plus, nous signalons que la section faisant l'objet des comparaisons de l'approche proposée avec les différentes méthodes issues de la littérature, concernera uniquement le problème $F_m|block|C_{max}, T_{max}$.

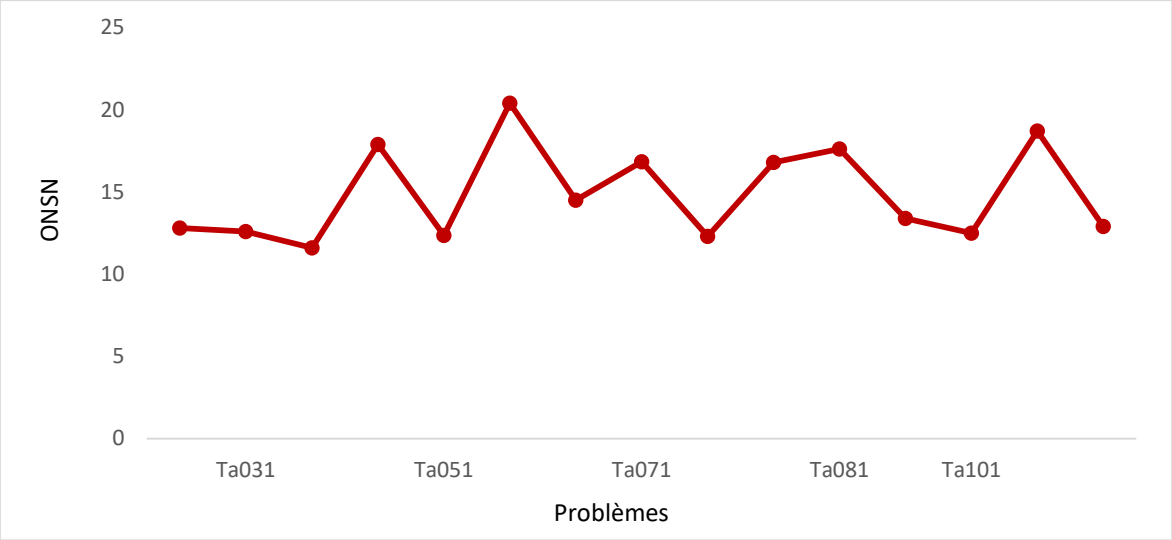


Figure 3.12 : Variation de la mesure ONSN de la R-NSGA-II.ver.2 selon les différentes capacités de stockage $F_m|block|C_{max}, T_{max}$

Problème	Capacités de stockage			
	Mesures de performance	**0**	**2**	**Variable**
Ta031	**ONSN**	11.600	12.600	12.800
	DM	0.000	0.000	0.000
Ta051	**ONSN**	20.400	12.360	17.900
	DM	0.500	0.000	0.000
Ta071	**ONSN**	12.300	16.400	14.500
	DM	0.000	0.000	0.510
Ta081	**ONSN**	13.400	17.620	16.800
	DM	2.134	0.000	0.000
Ta101	**ONSN**	12.900	18.700	12.500
	DM	1.806	0.000	0.000
Moyenne	**ONSN**	**14.88**		
	DM	**0.33**		

Tableau 3.7 : Résultats d'ONSN et de DM fournis par la RNSGA-II.ver.2 $F_m|block|C_{max}, T_{max}$

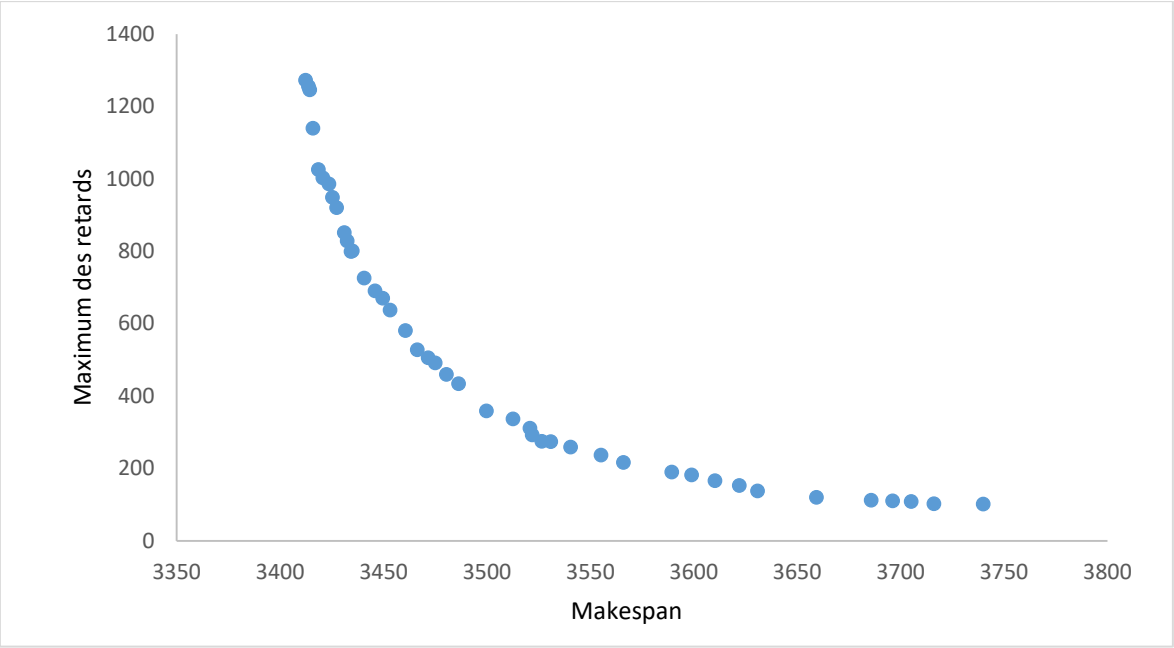


Figure 3.13 : L'ensemble des solutions non-dominées produites sur l'instance 40*20 (Front - Pareto) $F_m|block|C_{max}, T_{max}$

Problème	Capacités de stockage		
	0	**2**	**Variable**
Ta031	13.400	14.100	14.350
Ta051	23.100	13.660	16.500
Ta071	19.420	16.700	11.500
Ta081	12.150	19.260	17.200
Ta101	13.850	16.500	11.700
Moyenne	**15.55**		

Tableau 3.8 : Résultats d'ONSN fournis par la RNSGA-II.ver.2 $F_m|block, r_k|C_{max}, T_{max}$

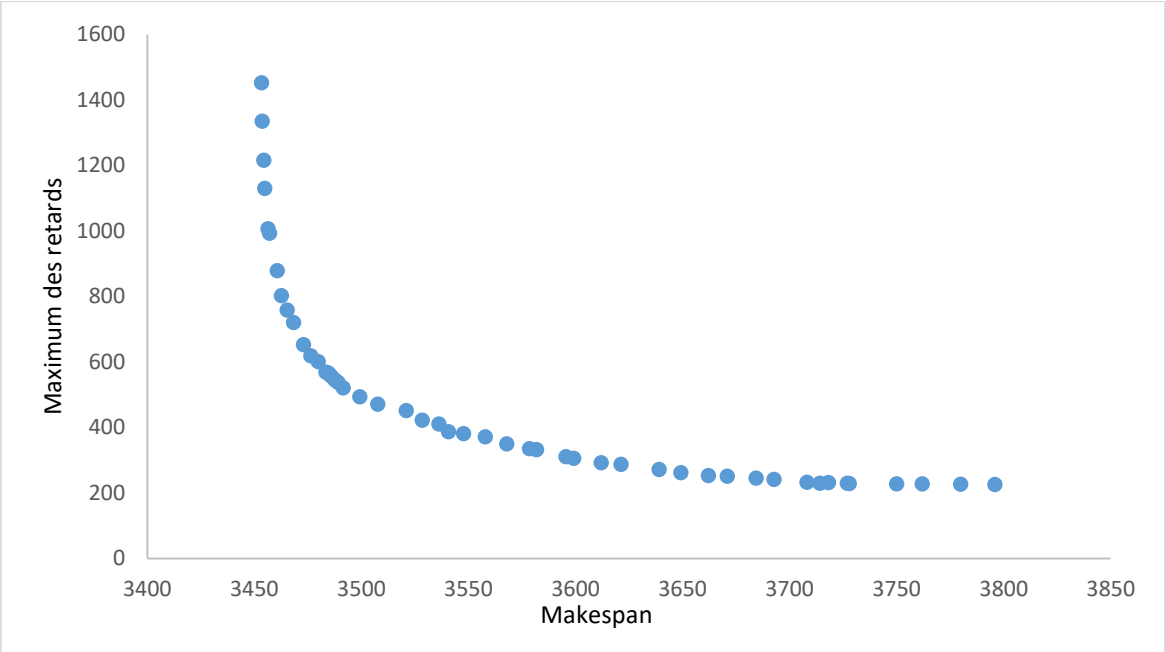


Figure 3.14 : L'ensemble des solutions non-dominées produites sur l'instance 40*20 (Front - Pareto) pour le problème $F_m|block, r_k|C_{max}, T_{max}$

3.6.3. Comparaison de la RNSGA-II.ver.2 avec d'autres approches issues de la littérature

D'après notre connaissance, l'approche HDE est reconnue par son pouvoir à trouver efficacement une sous-région plus prometteuse à travers les stratégies de croisement et de mutation adoptée, sa force lui a permis de dépasser largement l'approche IMOGLS2 [ISH, 2003] qui est capable de réaliser des solutions non-dominées uniformément réparties. Dans cet esprit, pour évaluer la performance de la R-NSGA-II.ver.2 proposée en vue de résoudre le problème de l'ordonnancement de l'atelier flow-shop avec blocage, des comparaisons en termes des deux métriques de performance ont été effectuées avec l'approche HDE dans les trois situations de capacité de stockage précitées.

Le tableau 3.9 et la figure 3.15 résument les résultats numériques des mesures de performance obtenus par les approches HDE et R-NSGA-II.ver.2. À partir de ce tableau, Il est bien clair qu'en ce qui concerne les résultats de la métrique ONSN, la R-NSGA-II.ver.2 proposée obtient les meilleurs avec une moyenne égale à 14,88 comparativement à la moyenne de la HDE qui est équivaut à 13,75. Ce constat signifie que l'approche proposée fournit des solutions plus compétitives et non-dominées. En outre, relativement à la métrique DM, nous pouvons clairement constater que les valeurs de cette mesure fournies par la R-NSGA-II.ver.2 sont inférieures à celles produites par la HDE, ce qui implique que les solutions obtenues par notre approche sont plus distribuées et couvrent plus de surface du front de Pareto. En dernière analyse, ces résultats permettent de conclure que nous pouvons conclure que les performances de la R-NSGA-II.ver.2 sont nettement meilleures que celles de la HDE dans différentes situations de capacité de stockage entre les machines consécutives.

Tableau 3.9 : Résultats de comparaison des mesures de performance de la HDE et la R-NSGA-II.ver.2

Problèmes	Capacitè de stockage	HDE		R-NSGA-II.ver.2	
		ONSN	DM	ONSN	DM

Ta031	Variable	11.600	0.000	**12,800**	0.000
	2	12.500	0.000	**12,600**	0.000
	0	10.400	0.000	**11.600**	0.000
Ta051	Variable	17.200	0.000	**17,900**	0.000
	2	12.500	0.798	12,360	**0.000**
	0	18.200	0.000	**20.400**	0.500
Ta071	Variable	13.400	0.000	**14,500**	0.510
	2	15.100	0.000	**16.840**	0.000
	0	10.800	0.000	**12.300**	0.000
Ta081	Variable	15.500	0.000	**16,800**	0.000
	2	15.800	0.000	**17.620**	0.110
	0	13.200	4.315	**13.400**	**2.134**
Ta101	Variable	10.100	2.124	**12,500**	**0.000**
	2	18.100	0.000	**18.700**	0.000
	0	11.900	2.637	**12.900**	**1.806**
Moyenne		13.75	0.65	**14.88**	**0.33**

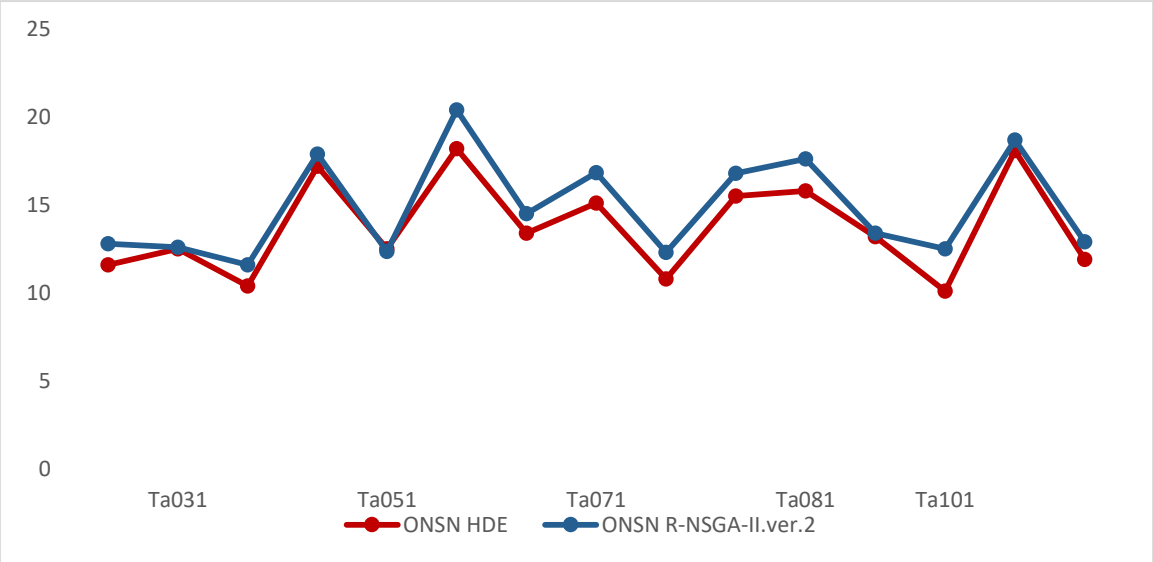


Figure 3.15 : Comparaison d'ONSN obtenus à travers la HDE et la R-NSGA-II.ver.2

3.6.4. Discussion et synthèse

En dépit du fait que les problèmes d'ordonnancement des ateliers de type flow-shop ont reçu une forte impulsion au cours de la dernière décennie, il y a encore beaucoup d'interrogations dans ce domaine. À notre connaissance, les études sur les problèmes flow-shop multi-objectifs ont certainement touché divers aspects ; or il reste encore quelques aspects plus intéressants et pertinents à aborder. Dans cette logique, la principale préoccupation de notre étude est de proposer une approche efficace pour résoudre le flow-shop multi-objectifs avec capacité de stockage limitée et différentes dates de disponibilité où le makespan et le maximum des retards absolus sont considérés comme des critères d'optimisation.

De ce fait, afin de valider l'approche proposée et comparer la performance des différents algorithmes de comparaison, deux paramètres sont utilisés dans notre étude, à savoir la mesure de distance et le nombre global des solutions non-dominées, comme les montrent les résultats présentés précédemment.

À partir de ces résultats, nous avons pu conclure que, dans différentes situations de capacité de stockage entre machines consécutives, les solutions produites par la R-NSGA-II.ver.2 sont plus proches de la solution de référence, plus dispersées et couvrent plus de surface du front Pareto. Dans cette optique, on peut évidemment confirmer que l'approche proposée permet une meilleure convergence et surpasse l'algorithme HDE de comparaison dans les cinq différents problèmes considérés.

En effet, les résultats de calcul confirment le besoin d'intégrer l'heuristique RNEH-WPT dans la phase d'initialisation de la population de la NSGA-II. Ce besoin s'explique par la force résultante de la RNSGA-II.ver.2 permettant de ne choisir que les bons individus au niveau de la première phase de recherche, ce qui permet bien évidemment d'améliorer efficacement la qualité des solutions fournies.

L'originalité de notre étude réside non seulement dans le fait qu'une version améliorée de la NSGA-II a été proposée, mais aussi dans le fait que la résolution du système flow-shop multi-objectifs qui minimise simultanément le makespan et le maximum des retards absolus en tenant en considération une capacité de stockage limitée et une disponibilité différentes des tâches sur la première machine, présente jusqu'à présent, un grand intérêt à la fois pour les chercheurs en théorie d'ordonnancement et les praticiens industriels, qui se trouvent souvent dans l'ambiguïté face à de tels problèmes d'ordonnancement qui sont justifiés par une grande complexité théorique et pratique.

3.7. Conclusions

Dans ce chapitre, nous avons abordé deux types de problèmes d'ordonnancement d'ateliers flow-shop multi-objectifs, le premier tient compte uniquement la contrainte de blocage pour laquelle la capacité de stockage est présumée être limitée entre les machines consécutives, tandis que le deuxième considère la contrainte de blocage et la contrainte pour laquelle les jobs ont des dates de disponibilité différentes sur la première machine. La résolution de ces problèmes vise à minimiser simultanément deux critères complexes, à savoir le maximum des retards absolus et le délai maximal d'exécution des travaux.

De ce fait, la première partie a porté sur une résolution exacte à travers la méthode de séparation et d'évaluation (B&B) intégrée dans le solveur CPLEX. Les résultats numériques obtenus ont démontré que pour les problèmes de petites tailles, la méthode exacte (B&B) a prouvé son efficacité vu qu'elles mènent vers des solutions optimales en un temps de calcul raisonnable. Par contre, elle reste inefficace face à des instances de très grandes tailles vue la quantité immodérée du temps de calcul consommé.

Tandis que dans la deuxième partie, une version améliorée de l'algorithme génétique élitiste de tri non dominé II (NSGA-II) a été proposée pour optimiser les mêmes critères de performance. Notre tâche consistait principalement à déterminer la solution correspondante au mieux aux préférences du décideur parmi les solutions de bon compromis entre le makespan et le maximum des retards absolus. Grâce à sa procédure d'assignement de fitness basée à la fois sur la notion de dominance et d'encombrement de l'espace de recherche, l'approche élitiste nommée NSGA-II permet de préserver la diversité des populations, en sauvegardant les meilleures solutions trouvées lors des générations précédentes. En dépit de ses capacités de sauvegarde et d'élitisme, notre attention a été portée particulièrement sur la phase d'initialisation de la population menant à développer un outil d'optimisation d'ordonnancement puissant nommé R-NSGA-II.ver.2. La force de la R-NSGA-II.ver.2 développée réside dans sa capacité à rejeter les solutions dominées dès le début du processus itératif en générant quatre solutions basées sur l'heuristique R-NEH-WPT. Les

résultats fournis ont prouvé qu'accorder une attention particulière à l'initialisation de la population, conduit bien évidemment à orienter la direction de la recherche vers un ensemble de solutions distribuées et non-dominées de haute qualité.

À cet échelon, nous mettons en place une technique de référence qui garantit efficacement la diversité des solutions Pareto-optimal, qui assure moins de complexité en termes de calcul et qui fournit un ensemble de solution non-dominées pertinent en termes de distribution et de convergence.

Sans omettre de signaler que les résultats obtenus dans la résolution du système multi-objectifs précité, tenant compte de la contrainte de disponibilité des tâches, représentent une première référence incontournable pour les praticiens et les chercheurs en théorie d'ordonnancement.

Dans le chapitre qui suit, une validation expérimentale des trois environnements d'ordonnancement étudiés sera effectuée. Cette concrétisation pratique sera un outil consistent pour la validation de l'efficacité des approches et des résultats obtenus d'une part, et pour répondre à une problématique industrielle rencontrée au niveau du système de production et de conditionnement des produits pharmaceutiques qui se trouve au sein du laboratoire de productique d'ECAM-EPMI.

Chapitre IV

Validation expérimentale: Cas de l'industrie pharmaceutique

Chapitre 4
Validation expérimentale : Cas de l'industrie pharmaceutique

4.1. Introduction

Dans les chapitres précédents, des modèles mathématiques, des approches exactes, approchées, combinées et basées sur le tri de non dominance, ont été proposés afin de supporter théoriquement les bonnes prises de décisions relatives à l'ordonnancement des systèmes industriels de type flow-shop mono-objectif et multi-objectifs. Toutefois, la qualité des solutions produites par les différentes approches proposées et qui ont permis effectivement de bien orienter cette prise de décision opérationnelle nous a laissé présager de réelles possibilités d'application industrielle.

Partant de ce constat, une collaboration avec le laboratoire de recherche en éco-innovation industrielle et énergétique (LR2E) situé à ECAM EPMI de Cergy pontoise a été effectuée. L'objectif de cette collaboration gravite autour de la validation expérimentale de toutes les techniques suggérées d'une part, et la résolution d'une problématique posée par un industriel chargé de la gestion de production de la ligne d'une autre part. En effet, le laboratoire LR2E est équipé d'une ligne de conditionnement des produits pharmaceutiques dans laquelle la contrainte de blocage domine fortement.

Pratiquement, la problématique posée au niveau cette ligne de production réelle touche de très près les systèmes d'ordonnancement étudiés, il s'agit de minimiser le temps de blocage dû au décalage de lancement des tâches résultant de la contrainte de disponibilité des opérations sur la première machine et l'absence des espaces de stockage entre les machines consécutives constituant un atelier d'ordonnancement de type flow-shop. Autrement dit, la principale contribution de ce chapitre réside dans la recherche de la séquence optimale des produits pharmaceutiques optimale qui correspond à un maximum des retards absolus et un makespan minimaux en respectant les contraintes de blocage et celle pour laquelle les tâches ont des dates de de disponibilité différentes. Pour ce faire, un couplage simulation/optimisation basé sur le module de gestion LCPP (Witness Horizon) conféré au fonctionnement réel de notre installation et les deux approches (SAGA et RNSGA-II.ver.2) proposées précédemment fera l'objet de cette dernière partie de recherche.

4.2. Problématique industrielle soulevée

Avant sa mise sur le marché, un produit pharmaceutique franchit plusieurs étapes dans différents ateliers de production :

- L'atelier de matière première composés des substances actifs et de l'excipient ;
- L'atelier des formes solides composés de produits semi-finis qui résultent de la transformation de la matière première en comprimés ou en gélules ;
- L'atelier de conditionnement dans lequel se met en pratique la mise en boîtes des comprimés et des gélules.

Le laboratoire de recherche productique de l'école d'ingénieurs ECAM-EPMI, situé à Cergy-pontoise, dispose d'une ligne de conditionnement de produits pharmaceutiques permettant de transformer les produits semi-finis à travers plusieurs postes, en produits finis. Cette ligne illustrée sur la figure 4.1 est équipée d'un atelier d'ordonnancement de type flow-shop caractérisé par un ensemble de palettes qui passent devant un ensemble de machines dans le même ordre. Réalisée par le groupe Schneider et sa filiale M2A, cet atelier réel est conçu pour le conditionnement des

produits pharmaceutiques. En termes de recherche, cette chaine de production représente un pilier complexe menant au développement de nouvelles techniques et approches en théorie de l'optimisation, de la modélisation, de la simulation et du pilotage des systèmes de conditionnement des produits pharmaceutiques principalement.

Partant de ce fait, Cette partie de recherche vise principalement à concrétiser et valider l'efficacité des approches de résolution et des modèles proposés dans le cadre de cette thèse, et répond particulièrement à une problématique industrielle posée par un industriel chargé de la gestion de production de cet atelier.

Figure 4.1 : Vue globale du système de production étudié

La problématique soulevée par l'industriel est la suivante :

Quelle est la séquence d'ordonnancement optimale à adopter pour minimiser le makespan et le maximum des retards absolus en tenant en considération la contrainte de blocage et de disponibilité des tâches sur le premier poste ?

Cependant, dans une ligne de production composée de plusieurs postes dont chacun ne possède aucune capacité de stockage intermédiaire, il est incontestable que le temps d'achèvement final consommé pour le traitement de toutes les tâches soit prohibitif. Pratiquement, si nous considérons les dates de réveil et nous lançons les palettes sur les convoyeurs selon une séquence aléatoire (voir figure 4.2), nous remarquons que la majorité des palettes restent bloquées une fois que la machine en amont soit occupée, et les postes ultérieurs auront tendance par la suite à attendre qu'une palette arrive d'un poste en amont pour pouvoir traiter son opération. Ce constat est bien évidement intuitif vu que l'atelier ne dispose pas d'un système de stockage intermédiaire et compte tenu du fait que la date de réveil ou de lancement de la fabrication d'un produit dépend de celui qui le précède. Ce qui engendre par la suite un blocage considérable menant non seulement à une augmentation importante et implicite du coût de production et du temps improductifs, mais également à une augmentation considérable des retards et du temps de conditionnement global de toutes les palettes. Dans ces circonstances, il existe bien évidement une séquence optimale à adopter afin de minimiser le makespan et le maximum des retards absolus dans ce système de production.

Figure 4.2 : Illustration d'une séquence des palettes

Pour y pallier efficacement et résoudre le problème d'ordonnancement au niveau des différents postes, nombreuses sont les approches de résolution qui peuvent être envisagées, parmi lesquelles, nous pouvons citer les méthodes basées sur le couplage simulation-optimisation. En effet, la mise en œuvre de ce couplage pour la résolution des systèmes de production complexes a suscité un large pan de littérature, notamment, [AZA, 1992] ; [FU, 1994] ; [DOL, 1997] ; [AND, 1998] ; [PIE, 2001] [OUA, 2001] ; [CHE, 2008] ; [FU, 2008] ; [KHA, 2015]. La procédure standard de l'optimisation à travers la simulation consiste à alterner régulièrement une phase de simulation et une autre d'optimisation. L'application de cette procédure sur un système industriel complexe représente un dilemme qui a attiré une vaste littérature, dont les difficultés soulevées gravitent principalement autour du choix de l'approche d'optimisation à proposer et de l'implémentation du couplage simulation- optimisation à mettre en œuvre.

Dans cet esprit, pour valider les modèles proposés et pour résoudre la problématique en question, nous nous sommes basés sur des approches de simulation/optimisation fondées sur le couplage des différentes approches de résolution (SAGA et RNSGA-II.ver.2) proposées dans les chapitres précédents avec un simulateur industriel très puissant en termes de flexibilité nommé WITNESS (LANNER). Ce couplage permettra de valider et concrétiser nos travaux de recherche établis pour résoudre le $F_m|block|C_{max}$, le $F_m|block|C_{max}, T_{max}$ et le $F_m|block, r_i\ |C_{max}, T_{max}$. Les principales étapes de ce couplage seront exposées dans des sections suivantes.

La projection des résultats théoriques obtenus sur cette ligne de conditionnement est justifiée par la concordance importante qui existe entre les hypothèses et les restrictions théoriques prises en compte dans les résolutions exactes et approchées établies auparavant et celles réelles existantes dans le processus de conditionnement des produits pharmaceutiques. Il est à noter que, l'atelier de conditionnement étudié fonctionne en flow-shop, dans lequel, chaque produit conditionné passe par l'ensemble des machines constituant la ligne de production dans un ordre unique. Chaque poste ne peut traiter qu'une seule opération à la fois, chaque job ne peut avoir qu'une seule opération en cours de réalisation simultanément, la préemption des opérations n'est pas autorisée, les tâches ont des dates de disponibilité différentes sur le premier poste, les temps de réglage sont inclus dans les temps opératoires des jobs et la capacité de stockage entre les postes consécutifs est nulle. Cette situation conduit à des blocages dans lesquels les jobs peuvent rester bloqués sur le poste tant que le poste suivant n'est pas disponible pour traiter l'opération en cours et la ligne ne dispose pas d'espaces de stockage pour conserver les opérations traitées.

4.3. Ligne de conditionnement des produits pharmaceutiques

La ligne de production étudiée est conçue pour conditionner des flacons composés chacun de deux types de produits pharmaceutiques (la vitamine B12 et le Fer) avec des pourcentages de dosage différents et défini préalablement pour chaque flacon. La principale cible de cet atelier de production est de transformer les produits semi-finis (les palettes) via plusieurs ressources matérielles en produits finis (les flacons remplis de produits pharmaceutiques).

Il convient de signaler que dans ce qui suit, la notion poste de travail est utilisée désormais pour désigner une machine et palette/ flacon pour dénoter un job. Par ailleurs, comme précité la ligne est composée de neufs postes (Annexe 4) dont : un représente un système de supervision et de gestion par ordinateur permettant de programmer les différentes gammes de production et lancer des ordres de fabrication. Un dédié au chargement et au départ des palettes et un autre isolé consacré à la fabrication des comprimés. Il est à noter que le premier poste de chargement des palettes et le poste de fabrication des comprimés ne sont pas considérés dans notre validation. Le premier poste (P1 : effectué manuellement par un opérateur) a été considéré comme un poste d'assemblage des flacons sur des palettes et de lancement des différentes séquences des palettes, et l'autre est disposé de façon isolée du circuit fermé de la ligne de conditionnement.

Par conséquent, les six postes de travail restants qui sont potentiellement reliés pas des convoyeurs à accumulation sont disposées comme montré sur la figure 4.3. Ils permettent d'effectuer plusieurs opérations, notamment :

- **Le remplissage du fer (P2) :** Ce poste est situé sur le convoyeur principal C1, il consiste à introduire un pourcentage de comprimés du fer bien défini par le superviseur dans le flacon. Il est composé d'un bol vibrant contenant les comprimés à doser. Une fois que la palette arrive à ce poste, elle est tirée en dessous du bol pour que le flacon soit rempli à travers une trémie ;
- **Le dosage du B12 (P3) :** Ce poste est situé sur un circuit indépendant du convoyeur principal, il est doté d'un autre convoyeur C2 dans lequel la palette est poussé via un vérin pneumatique afin d'arriver au poste de remplissage du B12. Une fois que l'opération est achevée, la palette est repoussée vers le convoyeur principal C1 à l'aide du même bras poussoir ;
- **Le bouchonnage des flacons (P4) :** Ce poste est également situé dans un circuit isolé contenant un autre convoyeur C3. Une fois que la palette arrive à un point du convoyeur principal, elle est poussée avec la même manière du dosage du B12 vers le convoyeur C3 afin que le flacon soit accessible pour le bouchonnage qui se fait avec le bouchon chargé sur la palette initialement ;
- **L'étiquetage des flacons (P5) :** Dans le même circuit (convoyeur C3), le flacon bouchonné est étiqueté avec un système à air comprimé, par des pastilles blanches imprimées, permettant de l'immatriculer en fin de conditionnement ;
- **Le déchargement des palettes (P6) :** Dans cette station, les palettes sont censées se décharger du flacon à travers un bras manipulateur qui pose le flacon dans un convoyeur tampon (C4). Cette station comporte deux tapis et deux vérins avec un système de ventouse nommés, transporteur tampon et transporteur de déchargement ;
- **Le conditionnement des flacons (P7) :** Le produit final que l'on obtient est un flacon séparé mécaniquement de la palette et conditionné pour le charger dans une boite en sortie du système, en attendant d'être transmis aux hôpitaux ou aux particuliers (pharmaciens et délégués médicaux).

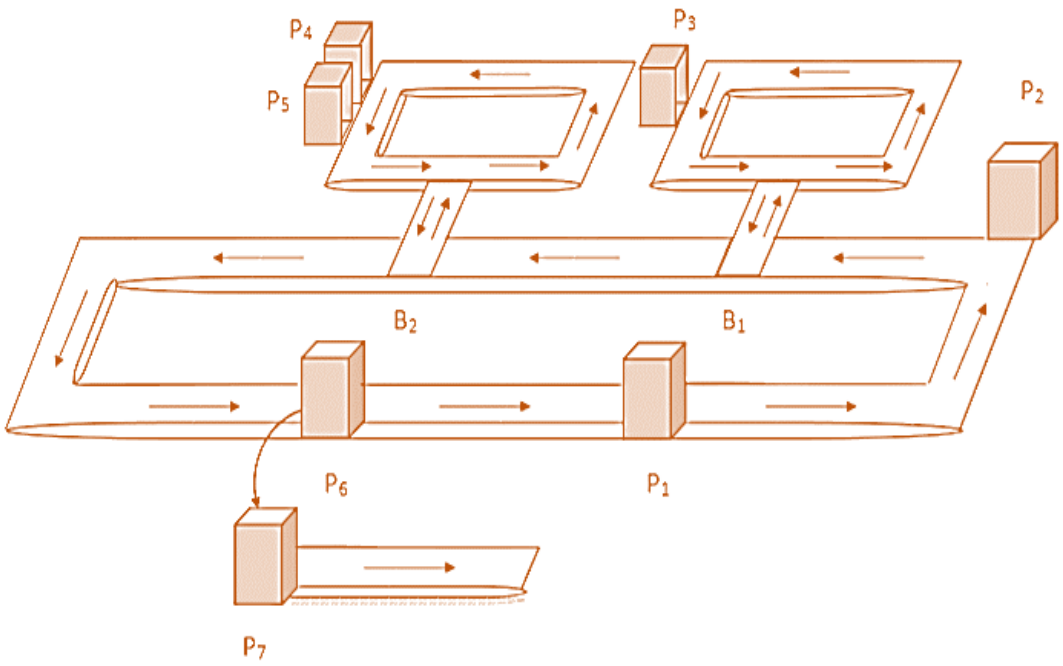


Figure 4.3 : Représentation du système de production étudié

4.4. Phase préparatoire de l'étude du système

4.4.1. Données préliminaires

Dans la ligne de conditionnement étudiée, nous manipulons deux gammes de production pour lesquelles les étapes de transformation sont les mêmes sur tous les postes de travail (Annexe 5). Pour les deux cas de figures, des durées opératoires correspondantes au traitement de chaque opération dans les différents postes de travail sont fixées. La première gamme (5*6) contient cinq palettes qui doivent être exécutées sur les six postes de travail, tandis que la deuxième (7*6) comporte sept palettes qui doivent être traitées semblablement sur les six postes de travail. Évidemment, les temps opératoires des postes de remplissage du fer et du dosage du B12 sont relatifs aux pourcentages des doses qui doivent être exécutées par ces postes. Les tableaux suivants regroupent les temps opératoires et les pourcentages des doses pour chacune des deux gammes précitées.

Tableau 4.1 : Pourcentage du Fer et du B12 de la gamme de production 1 (5*6)

Gamme de production n° 1		
Palette Np	**Pourcentage du fer (%) sur le poste de remplissage de fer (P2)**	**Pourcentage du B12 (%) sur le poste de dosage du B12 (P3)**
Palette N1	15	85
Palette N2	51	49
Palette N3	89	11
Palette N4	9	91
Palette N5	18	82

Tableau 4.2 : Pourcentage du Fer et du B12 de la gamme de production 2 (7*6)

Gamme de production n° 2		
Palette Np	**Pourcentage du fer (%) sur le poste de remplissage de fer (P2)**	**Pourcentage du B12 (%) sur le poste de dosage du B12 (P3)**

Palette N1	13	87
Palette N2	54	46
Palette N3	83	17
Palette N4	11	89
Palette N5	21	79
Palette N6	14	86
Palette N7	88	12

Tableau 4.3 : Durées opératoires de la gamme de production 1 (5*6)

Gamme de production n° 1						
Postes	P2	P3	P4	P5	P6	P7
Durées opératoires (secondes)	19	36	23	6	8	17
	47	35	23	6	8	17
	73	28	23	6	8	17
	18	34	23	6	8	17
	19	45	23	6	8	17

Tableau 4.4 : Durées opératoires de la gamme de production 2 (7*6)

Gamme de production n° 2						
Postes	P2	P3	P4	P5	P6	P7
Durées opératoires (secondes)	17	32	23	6	8	17
	51	35	23	6	8	17
	67	26	23	6	8	17
	19	32	23	6	8	17
	21	49	23	6	8	17
	18	36	23	6	8	17
	39	17	23	6	8	17

En outre, relativement à la validation et la résolution des systèmes $F_m|block|C_{max}, T_{max}$ et $F_m|block, r_i|C_{max}, T_{max}$, deux autres données temporelles s'ajoutent, notamment, les dates de disponibilité de chaque palette sur le premier poste de travail (Remplissage du fer) et la date d'échéance imposée par le client. Les deux configurations des deux gammes sont considérées comme suit :

- Pour la première gamme de production (5*6) :
  - Les dates de réveil (disponibilité) en secondes : ri = [5,26,75,45,33] ;
  - Les dates d'échéances en secondes : di = [92, 95,167, 206, 152].
- Pour la deuxième gamme de production (7*6) :
  - Les dates de réveil (disponibilité) en secondes : ri = [7, 37, 69,32, 85, 93, 57] ;
  - Les dates d'échéances en secondes : di = [101,92,182,219, 164, 228,193].

Un autre paramètre s'ajoute aux données temporelles précitées, il s'agit de la vitesse des convoyeurs, Sachant que cette vitesse peut être réglée sur l'intervalle [75%, 120%], la vitesse que nous considérons pour toutes nos simulations est :

V : 100% = 24.0 cm/seconde

4.4.2. Présentation du simulateur Witness Horizon

La nécessité de faire appel à un logiciel de simulation industrielle est née du fait que pour trouver la séquence optimale des systèmes étudiés et répondre efficacement à notre problématique, toutes les séquences réelles devaient être examinées pour les deux gammes considérées de chaque système afin d'en extraire la meilleure. Par conséquent, cinq palettes génèrent pratiquement 120

permutations possibles et sept génèrent potentiellement 5040 séquences possibles. Partant de ce fait, afin de pouvoir traiter concrètement l'ensemble des permutations précitées et tester réellement tous les problèmes d'ordonnancement considérés, nous étions dans l'obligation de modéliser et simuler l'atelier flow-shop de conditionnement dans un des logiciels de simulation industriels performants.

Depuis 1980, un essor important des outils de simulation industriels s'est produit grâce à l'évolution profonde que les systèmes informatiques ont connu. Plus de 50 logiciels de simulation et d'analyse des systèmes complexes à base de flux ont vu le jour et la plupart existent jusqu'à présent avec des versions évoluées, parmi lesquels, nous citons : SLAM, SIMAN, ARENA, GENETIK, HOCUS et WITNESS.

Dans notre travail de recherche, notre modèle de simulation a été implémenté sur le progiciel Witness. Le choix de ce simulateur est dicté d'une part par sa disponibilité dans notre local de recherche (le laboratoire productique d'ECAM-EPMI), et d'autre part par sa reconnaissance comme l'un des leaders du marché mondial des outils de simulation à événements discrets. En effet, développé par la société britannique LANNER, ce logiciel de simulation industrielle prédictive a été utilisé avec succès par des milliers de modélisateurs au cours des 20 dernières années. Son succès s'explique par sa capacité à offrir à l'utilisateur un support de modélisation, et du développement des modèles riches en fonctionnalités et des applications de simulation, grâce à une visualisation dynamique et très conviviale des données relatives aux systèmes complexe à base de flux.

4.4.3. Modélisation de la ligne de conditionnement des produits pharmaceutiques

Pour modéliser et simuler graphiquement la ligne de conditionnement des produits pharmaceutiques sur Witness, nous avons utilisé un attribut correspondant à une étiquette qui stocke des informations représentatives de chaque palette. Ensuite, nous avons créé des variables permettant de stocker les temps de cycle nécessaires pour le traitement des opérations au niveau de chaque poste de travail sous forme d'un tableau sous Excel qui donne en ligne le nom du poste de travail, en colonne le type de palette et à l'intérieur du tableau le temps consommé sur un poste donné pour le traitement d'un type de palette donné. Signalons qu'au moment du lancement d'une palette il faudra décider son type et donc la valeur de son attribut. À ce stade, nous nous sommes trouvés dans l'ambiguïté de paramétrer les postes de travail de façons à ce qu'ils traitent chaque type de palette avec son temps de cycle correspondant. Toutefois, grâce à l'aide précieuse du personnel de la société LANNER, nous avons finalement pu modéliser l'atelier flow-shop de conditionnement des produits pharmaceutiques (LCPP) comme illustré sur la figure 4.4. Le problème soulevé résidait effectivement au niveau du paramétrage des temps de cycle où il fallait renseigner une expression qui renvoie chaque poste de travail courant à sa position dans le tableau des temps opératoires.

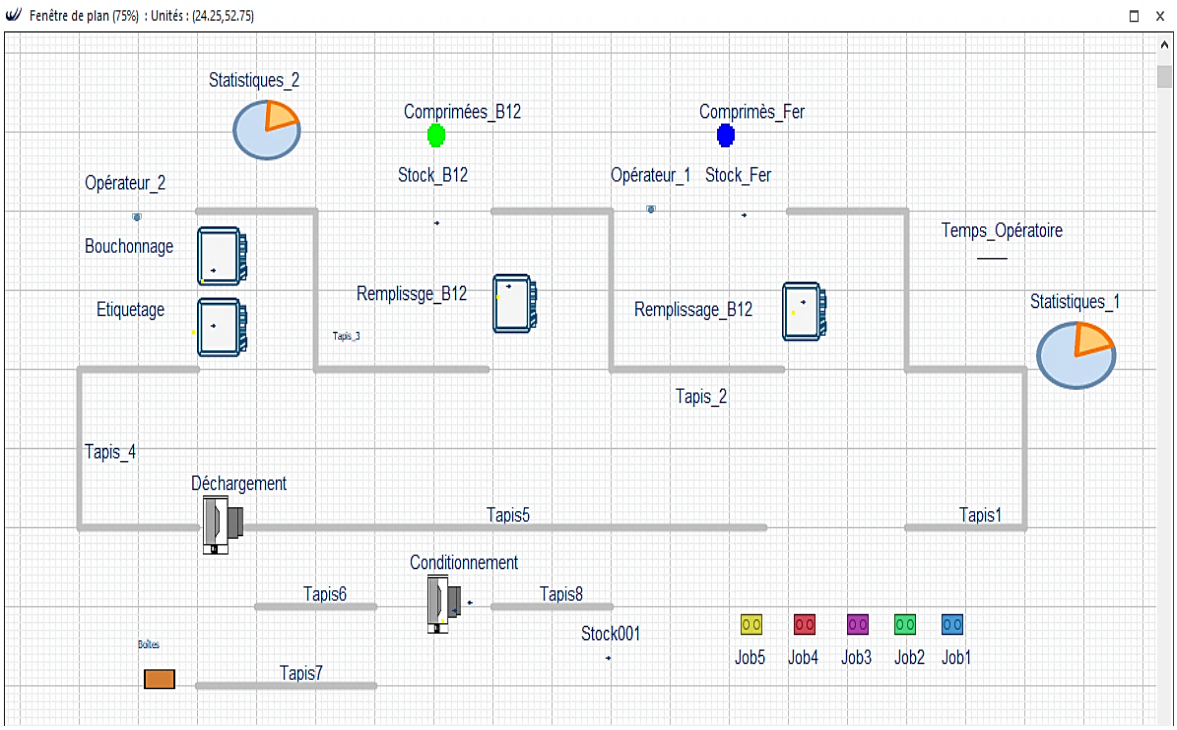


Figure 4.4 : Modélisation de la ligne de conditionnement étudiée sur le simulateur Witness

4.4.4. Résultats des simulations préliminaires

Comme étude préliminaire, nous avons réalisé un jeu de simulations à travers Witness de l'atelier flow-shop de conditionnement sur 20 heures pour les deux différentes gammes de production (5*6) et (7*6), appliquées sur les différents ateliers considérés : le $F_m|block|C_{max}$, le $F_m|block|C_{max}, T_{max}$ et le $F_m|block, r_i\ |C_{max}, T_{max}$.

Les résultats de ces simulations sont regroupés dans les figures 4.5, 4.6, 4.7, 4.8, 4.9 et 4.10. Comme on le constate sur les graphiques présentés, en générant des séquences aléatoires pour les deux gammes de production considérées, les temps de blocage augmentent significativement avec des pourcentages de 66%, 74% 77%, 84%, 89% et 93% pour le $F_m|block|C_{max}$ (5*6), le $F_m|block|C_{max}$ (7*6), le $F_m|block|C_{max}, T_{max}$(5*6), le $F_m|block|C_{max}, T_{max}$ (7*6), le $F_m|block, ri|C_{max}, T_{max}$ (5*6) et le $F_m|block, ri|C_{max}, T_{max}$ (7*6), respectivement. Ce blocage qui domine fortement dans les diffèrents systèmes peut s'expliquer par l'absence des zones de stockage entre les postes de travail consécutifs du système de production étudié, il survient quand la palette reste bloquée sur un poste de travail par manque de place et le poste en amont n'est pas non plus disponible pour commencer son traitement. Ainsi, pour le cas où les palettes ont des dates de disponibilité différentes sur le premier poste de travail, le temps de blocage connait une augmentation supplémentaire dû au fait qu'en lançant les premières tâches avec un temps initial décalé, les tâches en amont subissent des décalages au niveau du lancement, ce qui génère par la suite un temps global de blocage considérable.

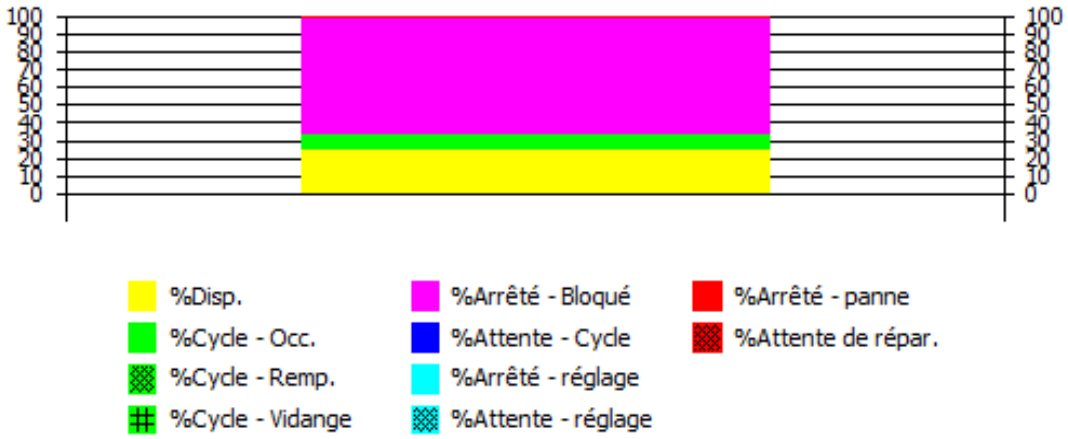


Figure 4.5 : Temps de blocage du $F_m|block|C_{max}$ (5*6)

Figure 4.6 : Temps de blocage du $F_m|block|C_{max}$ (7*6)

Figure 4.7 : Temps de blocage du $F_m|block|C_{max}, T_{max}$ (5*6)

Figure 4.8 : Temps de blocage du $F_m|block|C_{max}, T_{max}$ (7*6)

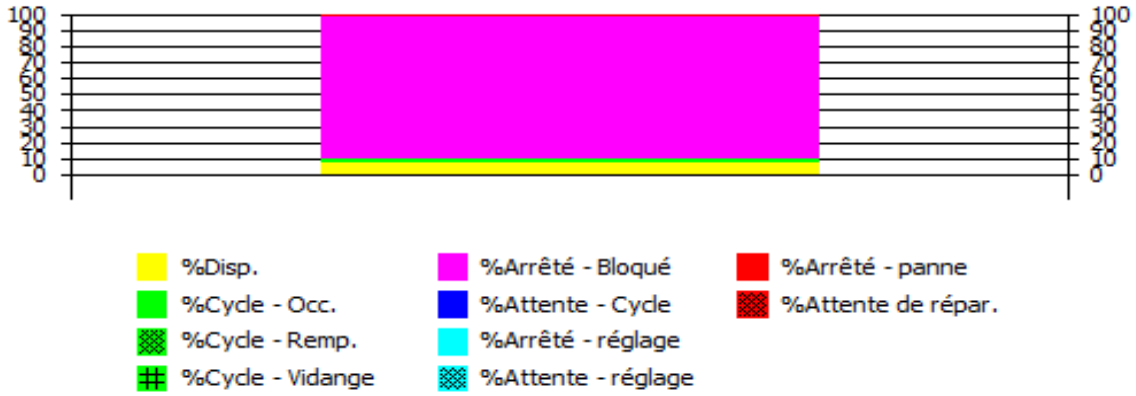


Figure 4.9 : Temps de blocage du $F_m|block, ri|C_{max}, T_{max}$ (5*6)

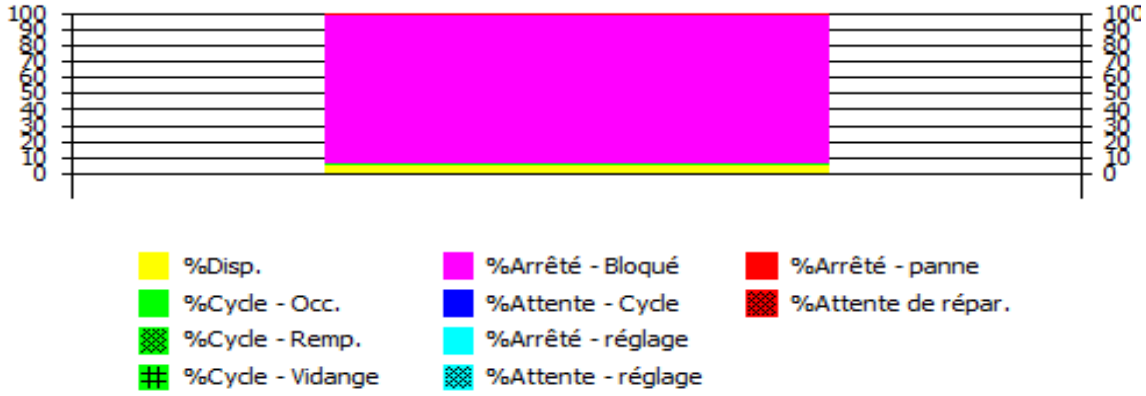


Figure 4.10 : Temps de blocage du $F_m|block, ri|C_{max}, T_{max}$ (7*6)

Nous rappelons que cette phase préparatoire a été effectuée afin de valider le modèle de simulation en confrontant les résultats obtenus au fonctionnement réel de notre système et d'évaluer également l'efficacité des différentes approches proposées précédemment ainsi que notre approche de simulation / optimisation qui fait l'objet de la suite de ce travail.

4.5. Couplage simulation/optimisation

4.5.1. Procédure générale

Le couplage simulation/optimisation tient une place prépondérante dans la littérature du fait qu'il permet d'étudier profondément le comportement dynamique des systèmes complexes. Ce couplage consiste à alterner régulièrement deux phases principales : une phase de simulation et une autre d'optimisation comme illustre la figure 4.11.

Relativement à nos travaux de recherche, le bloc d'optimisation correspond aux différentes approches de résolution proposées précédemment en vue de trouver une solution optimale pour les problèmes d'ordonnancement considérés. Tandis que le bloc de simulation s'accorde principalement au modèle simulé LCPP permettant d'évaluer réellement la fonction objectif en intégrant toutes les contraintes du problème.

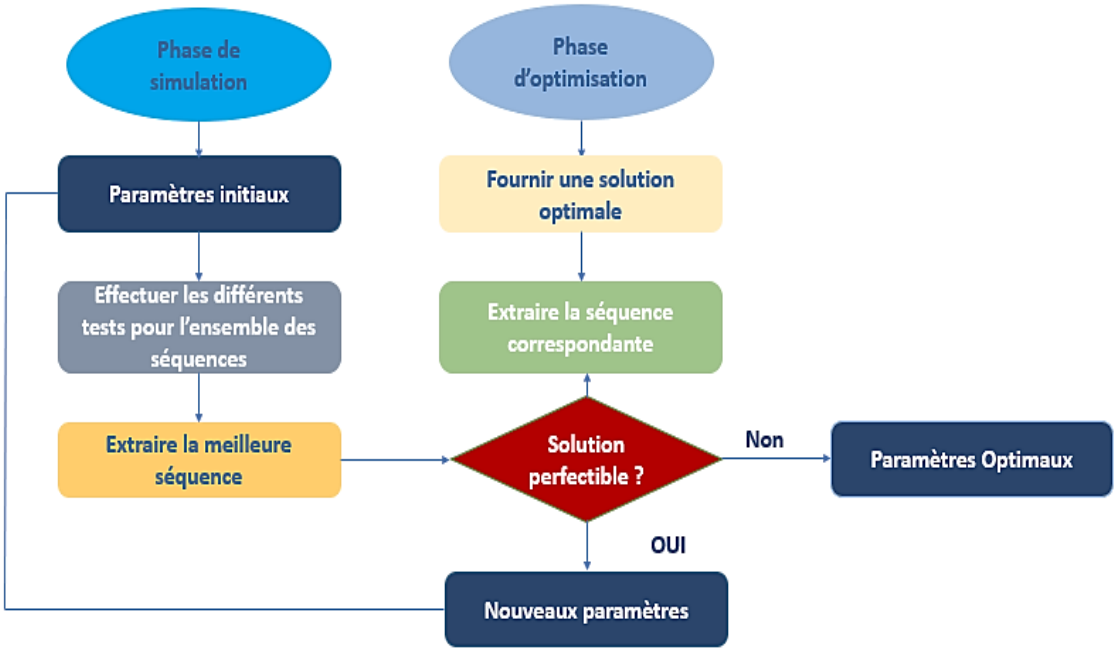


Figure 4.11 : Démarche générale du couplage simulation/optimisation

Les sections suivantes dérivent les couplages simulation-optimisation établis afin de résoudre la problématique posée par l'industriel et valider les approches et les modèles proposés dans nos travaux de recherche. Par ailleurs, les figures 4.12 et 4.13 décrivent clairement les principales étapes du couplage du modèle LCPP avec les deux approches originales que nous avons proposées. Le premier couplage, illustré sur la figure 4.12 correspond au LCPP avec l'approche « Simulated Annealing Genetic Algorithme » (SAGA) proposée pour résoudre le flow-shop mono-objectif avec contrainte de blocage visant à minimiser le makespan. Le deuxième couplage, rapporté sur la figure 4.13 représente la synergie du LCPP avec l'approche « Non-Dominated Genetic Algorithm - II.ver.2 » proposée pour résoudre :

- Le flow-shop multi-objectifs qui minimise simultanément le makespan et le maximum des retards absolus en tenant en considération la contrainte de blocage ;
- Le flow-shop multi-objectifs qui minimise simultanément le makespan et le maximum des retards absolus en tenant en considération la contrainte de blocage et la contrainte pour laquelle les tâches ont des dates de disponibilité différentes sur le premier poste de travail.

4.5.2. Coulage LCPP/ SAGA

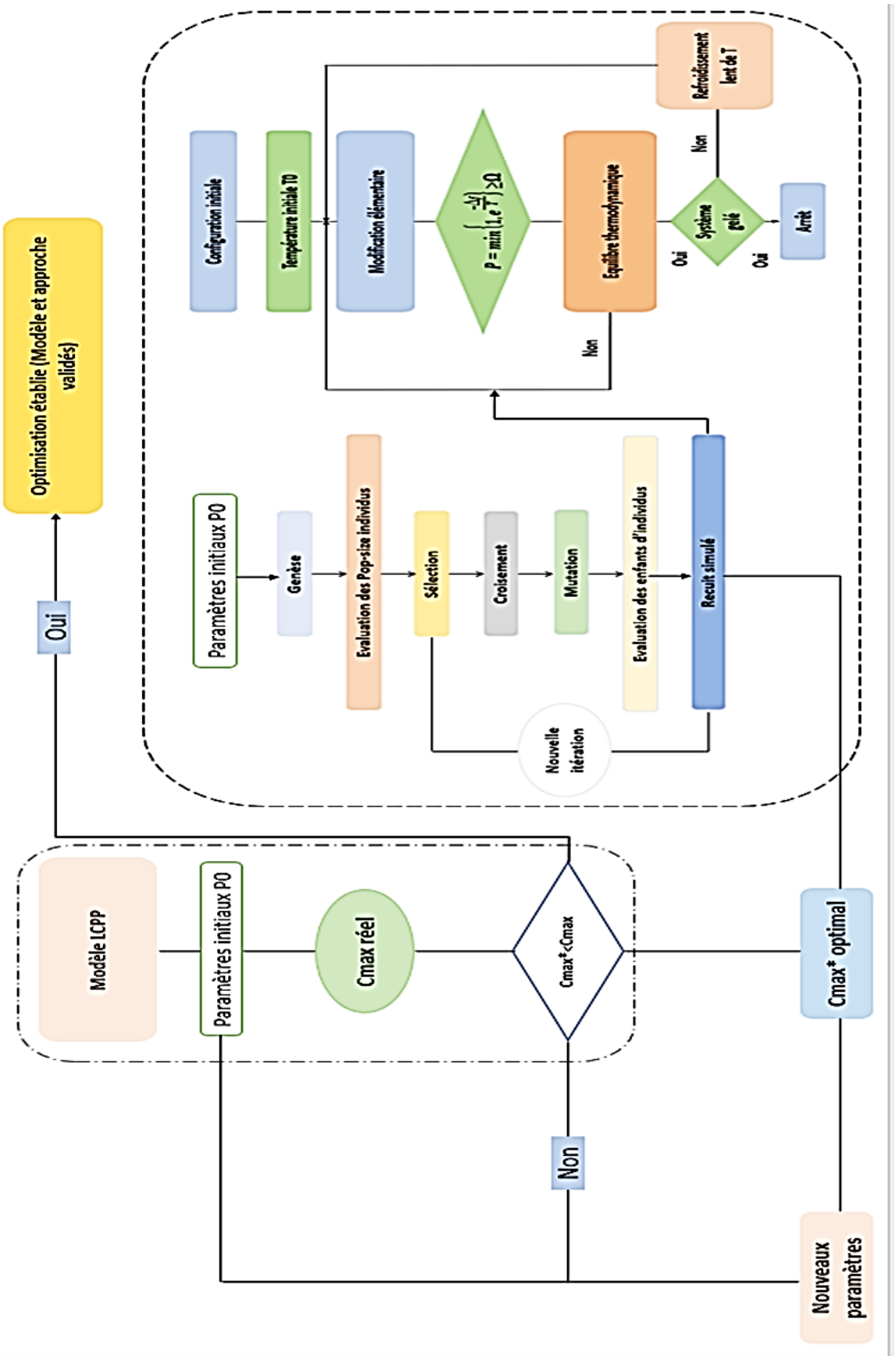


Figure 4.12 : Schéma général du couplage LCPP/ SAGA

4.5.3. Coulage LCPP/ RNSGA-II.ver.2.

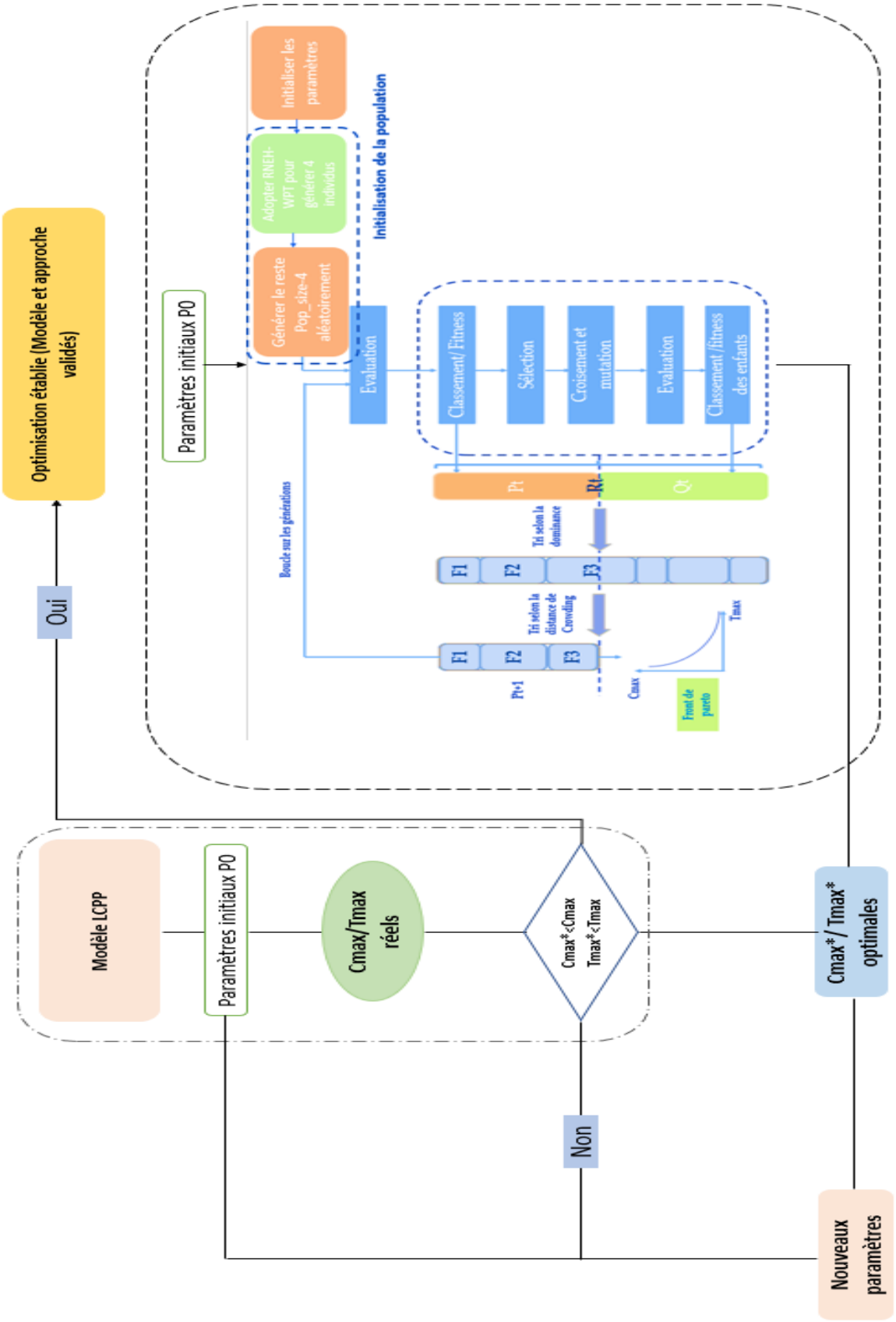


Figure 4.13 : Schéma général du couplage LCPP/ RNSGA-II. Ver.2

4.6. Simulations et résultats expérimentaux

Dans cette partie, nous présentons une série de tests expérimentaux réalisée pour évaluer les différentes approches présentées précédemment. Nous rappelons que ces approches sont appliquées sur la ligne de production réelle étudiée et les paramètres relatifs aux approches sont calibrés avec les mêmes niveaux sélectionnés dans les deux chapitres précédents à travers les plans dynamiques de Taguchi. Le tableau 4.5 rappelle les paramètres optimaux sélectionnés en vue d'optimiser les trois différents problèmes.

Tableau 4.5 : Paramètres optimaux des trois différents problèmes

Problèmes	Approches	Pop_Size	Max-iter	Pc	Pm	Cool_rate	Temp
$F_m\|block\|C_{max}$	SAGA	400	600	90%	10%	0.5	500
$F_m\|block\|C_{max}, T_{max}$	RNSGA-II.ver.2	50	100	60%	10%	-	-
$F_m\|block, r_k\|C_{max}, T_{max}$	RNSGA-II.ver.2	200	300	40%	40%	-	-

Les résultats d'optimisation obtenus sont classés en six catégories suivant les deux approches proposées, les deux gammes de production considérées et les différents environnements d'ordonnancement traités.

4.6.1. Résultats de simulation/optimisation via LCPP/SAGA (5*6)

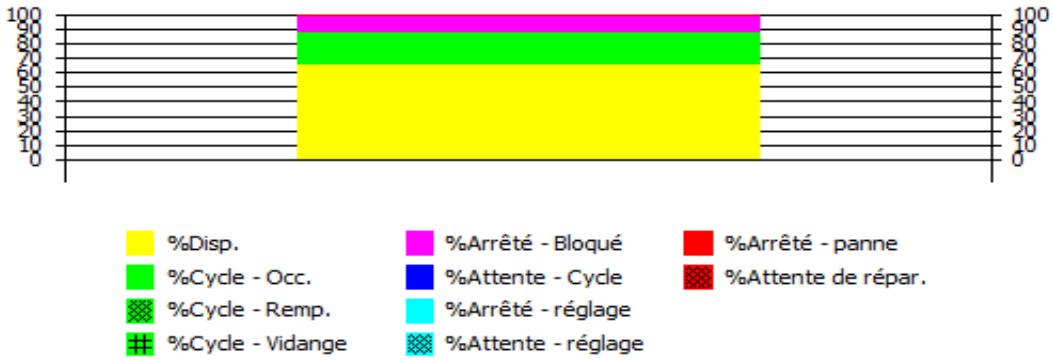


Figure 4.14 : Temps de blocage après optimisation du $F_m|block|C_{max}$ (5*6)

4.6.2. Résultats de simulation/optimisation via LCPP/SAGA (7*6)

Figure 4.15 : Temps de blocage après optimisation du $F_m|block|C_{max}$ (7*6)

4.6.3. Résultats de simulation/optimisation via LCPP/RNSGA-II. Ver. 2 (5*6)

4.6.3.1. Résultats du Fm/block/Cmax, Tmax

Figure 4.16 : Temps de blocage après optimisation du $F_m|block|C_{max}, T_{max}$ (5*6)

4.6.3.2. Résultats du Fm/Block, ri/Cmax, Tmax

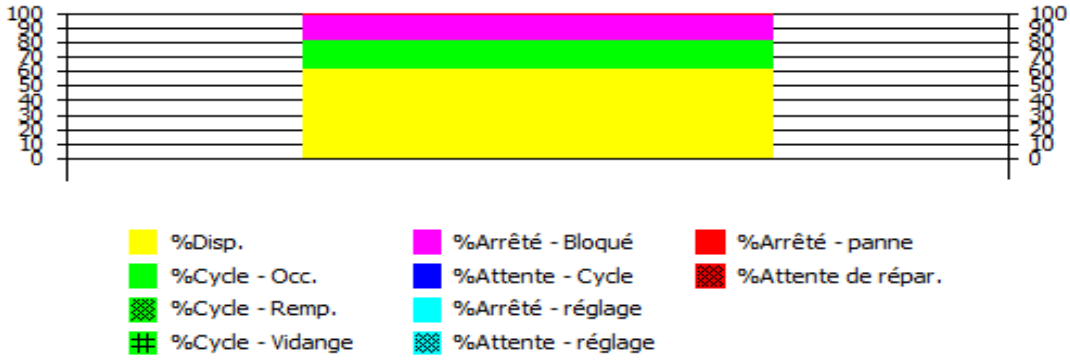


Figure 4.17 : Temps de blocage après optimisation du $F_m|block, ri|C_{max}, T_{max}$ (5*6)

4.6.4. Résultats de simulation/optimisation via LCPP/RNSGA-II. Ver. 2 (7*6)

4.6.4.1. Résultats du Fm/block/Cmax, Tmax

Figure 4.18 : Temps de blocage après optimisation du $F_m|block|C_{max}, T_{max}$ (7*6)

4.6.4.2. Résultats du Fm/Block, ri/Cmax, Tmax

Figure 4.19 : Temps de blocage après optimisation du $F_m|block, ri|C_{max}, T_{max}$ (7*6)

4.6.5. Analyse et discussions

Les résultats expérimentaux présentés sur les figures (4.14, 4.15, 4.16, 4.17, 4.16 et 4.17) confirment le besoin de considérer implicitement des méthodes de résolution robustes, pour répondre au mieux aux problématiques d'ordonnancement industrielles rencontrées au niveau de la ligne de production des produits pharmaceutiques en question. Ces figures correspondent aux statistiques recueillies au niveau des postes de travail et rapportent les pourcentages de disponibilité et de blocage relatifs aux séquences optimales produites par les couplages LCPP/SAGA et LCPP/RNSGA-II.ver.2, respectivement.

En effet, le couplage de la SAGA et la RNSGA-II.ver.2 avec le modèle LCPP simulé à travers Witness a permis de minimiser les temps de blocage, relatifs à l'ensemble des environnements d'ordonnancement considérés et aux différentes gammes de production étudiées, avec des pourcentages de (12%, 17%, 16%, 18%, 26% et 29%) comparativement aux pourcentages fournis dans la phase de simulation préliminaire présentée dans la section précédente (66%,74%77%, 89%, 84% et 93%). Par conséquent, une augmentation importante au niveau des temps de disponibilité est engendrée comme illustrée sur les différentes figures, ce qui permet de conclure que l'ensemble des approches proposées permettent d'améliorer théoriquement la qualité des solutions produites et d'assurer pratiquement une utilisation maximale des postes de travail constituant l'atelier étudié.

Il convient d'énoncer que les séquences optimales trouvées ont été testées réellement sur la ligne de production des produits pharmaceutiques en les confrontant au fonctionnement réel de l'atelier d'ordonnancement traité. Dans cet esprit, les résultats obtenus ont pratiquement validé l'efficacité des deux approches d'optimisation proposées, ont confirmé la pertinence des solutions produites à travers l'approche du couplage simulation/ optimisation et ont répondu efficacement à la problématique aperçue au niveau du système de production étudié.

4.7. Conclusions

L'objectif de ce chapitre était de concrétiser et montrer pratiquement l'efficacité de toutes les approches qui ont été introduites dans les deux derniers chapitres, à savoir, les modèles mathématiques confrontés aux problèmes d'ordonnancement flow-shop et les outils d'optimisation qui ont fait l'objet de leurs résolutions.

Au début de ce chapitre, nous avons éclaircit la problématique industrielle affrontée par l'industriel chargé de la gestion de production de la ligne de conditionnement des produits

pharmaceutique considérée. La problématique consistait à réduire le temps de blocage qui résulte de l'absence des zones de stockage dans l'espace de production et du décalage du lancement des tâches dû à la contrainte de disponibilité des tâches sur la première machine.

Après une description globale de la ligne de conditionnement en question, une étude préliminaire permettant de sélectionner les meilleurs paramètres et de valider le modèle LCPP simulé et modélisé à travers le simulateur Witness Horizon a été conduite. Cette phase préparatoire a permis bien évidement de définir l'ensemble des données conférées au fonctionnement du système.

Dès que la phase de modélisation préliminaire a été bien accomplie, nous nous sommes intéressés à l'étude des performances du système considéré en exposant les temps d'attente qui existent réellement entre ses postes de travail. L'accroissance exponentielle des temps d'attente constatée entre les différents postes de travail consécutifs, nous a poussé à développer deux approches de résolution distinctes en vue de répondre aux mieux aux problématiques modélisés. Ces approches sont basées sur le couplage du modèle LCPP et de la SAGA dans le cas de la résolution mono-objectif du flow-shop avec blocage. Et sur le couplage du LCPP avec la RNSGA-II.ver.2 dans les deux autres cas de résolution multi-objectifs : notamment, le flow-shop qui minimise le makespan et le maximum des retards absolus en considérant le blocage, et le flow-shop qui minimise simultanément le makespan et le maximum des retards absolus en considérant le blocage et la disponibilité des tâches sur le premier poste. Au vu des résultats pratiques obtenus, nous avons conclu que les approches proposées ont abouti à une utilisation des ressources maximale qui minimise largement les temps d'attente, qui optimise efficacement les critères de performances demandés par l'industriel et qui valide pratiquement l'efficacité des approches proposées et des modèles mathématiques développés dans la globalité de ces travaux de recherche.

Nous allons finalement conclure ce mémoire de thèse en donnant un bilan général de ce qui a été effectué et en fixant des pistes de recherche futur

Conclusion générale et perspectives

L'amélioration de la performance et l'optimisation des systèmes de production représentent un enjeu crucial pour les entités industrielles. Parmi les problèmes d'optimisation qui touchent de plus près la performance de ces systèmes de production, les problèmes d'ordonnancement sont au cœur de nombreux sujets de recherche. Dans cette thèse, nous avons contribué à la résolution des problèmes d'ordonnancement des ateliers de type flow-shop monocritère et multicritères soumis simultanément à différents types de contraintes.

Dans un premier temps, nous nous sommes focalisés sur l'étude de l'ordonnancement dans sa globalité en introduisant ses différents composants qui vont de la définition des éléments de base des ateliers d'ordonnancement (tâches, ressources, contraintes, critères) à la description du large pan des approches d'optimisation qui sont capables de retourner des solutions de qualité pour ces problèmes qui sont considérés, de point de vue complexité très difficile. Une fois que les outils et les composants de base sur lesquels repose le système flow-shop que nous mettons en œuvre au sein de cette thèse sont identifiés, nous avons porté nos efforts pour présenter une classification des travaux menés dans la littérature pour résoudre le système de production ètudiè. Une recherche bibliographique approfondies de cette classification a décelé :

- L'intérêt pratique et théorique de la résolution des problèmes d'ordonnancement dans les ateliers de type flow-shop particulièrement ;
- Une résolution exubérante et de différentes natures est effectuée dans la littérature, or plusieurs aspects techniques issus des situations industrielles réelles sont négligés, ce qui justifie le besoin et la nécessité d'aborder exhaustivement les volets dédaignés.

D'après cette bibliographie qui a fait émerger les enjeux industriels et méthodologiques de notre thèse, nous avons repéré trois voies de recherche pour lesquelles trois modèles mathématiques minimisant différents critères et qui considèrent plusieurs contraintes ont été introduits, notamment :

- Le modèle mono-objectif qui minimise le makespan avec contrainte de blocage dans un atelier $F_m|block|C_{max}$;
- Le modèle multi-objectifs qui minimise le makespan et le maximum des retards absolus avec contrainte de blocage dans un atelier $F_m|block|C_{max}, T_{max}$;
- Le modèle multi-objectifs qui minimise le makespan et le maximum des retards absolus avec contrainte de blocage et de disponibilité des tâches sur la première machine dans un atelier $F_m|block, r_k|C_{max}, T_{max}$.

D'un point de vue méthodologique, nous nous sommes intéressés aux approches à bases d'algorithmes génétiques pour la résolution des problèmes d'ordonnancement mono-objectif, notre visée étant d'exploiter leur capacité de trouver de bonnes solutions sur des problèmes très complexes, et trop éloignés des problèmes combinatoires classiques. En ce qui a trait aux problèmes d'ordonnancement multi-objectifs, nous avons porté notre focus sur les approches basées sur la notion de dominance au sens de Pareto, et plus particulièrement la méthode des algorithmes génétiques de tri non dominé, motivé par le fait que cette dernière a permis de résoudre efficacement de nombreux problèmes d'optimisation combinatoire.

Notre premier apport a consisté en l'étude du problème d'ordonnancement mono-objectif qui minimise le makespan dans un atelier de type flow-shop avec une capacité de stockage limitée entre les machines consécutives. Ce problème a été résolu initialement à l'aide d'un outil de

résolution exacte (CPLEX). Les résultats numériques ont démontré que pour les problèmes de petites tailles, la résolution exacte a pu prouver son efficacité vue qu'elle a mené vers des solutions optimales en un temps de calcul raisonnable. Cependant, elle reste inefficace face à des instances de très grandes tailles vue la quantité immodérée du temps de calcul consommé. Etant convaincus que les méthodes exactes tels que celle de séparation et d'évaluation sont inexploitables et limitées pour les problèmes de grande taille, qui sont en fait les plus répandus dans l'industrie, il a été choisi de proposer une approche par métaheuristique (algorithmes génétiques) afin d'obtenir des solutions pertinentes dans un laps de temps relativement réduit. En dépit de sa bonne exploration de l'espace de recherche et sa flexibilité, les résultats numériques fournis à travers les algorithmes génétiques ont mis en évidence leur faiblesse qui réside dans leur convergence prématurée. En conséquence, afin d'éliminer ces faiblesses et d'améliorer la qualité des solutions fournies, une méthode hybride constituée de la méthode précitée et celle du recuit simulé a été proposée dans la deuxième partie de cette phase de recherche. Cette incorporation consistait à exploiter les avantages respectifs des deux méthodes en combinant leurs algorithmes suivant une approche synergétique. De plus, étant donné que les paramètres du problème et des méthodes utilisées nécessitent un calibrage empirique, la méthode de Taguchi a été utilisée permettant de sélectionner les paramètres menant à une résolution optimale. A la fin de cette tâche, une étude comparative a été menée sur les résultats numériques obtenus par l'hybridation séquentielle suggérée et ceux obtenus par les algorithmes robustes existants dans la littérature. L'analyse de l'étude comparative établie a décelé que, l'approche que nous avons proposée fournit des résultats intéressants sur les différents tests effectués comparativement aux meilleurs résultats trouvés auparavant dans la littérature pour la minimisation du makespan dans un atelier flow-shop soumis à des contraintes de blocage.

La deuxième contribution se manifeste par l'étude de deux problèmes d'ordonnancement multicritères incorporant des hypothèses plus réalistes. La première porte sur la résolution du problème d'ordonnancement qui minimise simultanément le makespan et le maximum des retards absolus dans un atelier de type flow-shop dans lequel la contrainte de blocage domine potentiellement. Tandis que le deuxième traite le même problème multi-objectifs précité en considérant une contrainte de disponibilité des tâches supplémentaire. Pour ce faire, nous avons là encore utilisé le même outil de résolution exacte (CPLEX) en se basant sur les modèles mathématiques proposés. Cependant, pour la même raison, nous avons penché principalement vers les méthodes approchées. Dans ce contexte, une méthode fondée sur Pareto traitant séparément les objectifs a été introduite en vue de déterminer la solution correspondante au mieux aux préférences du décideur parmi les solutions de bon compromis entre le makespan et le maximum des retards absolus Grâce à sa procédure d'assignement de fitness basée sur la notion de dominance, cette méthode élitiste nommée NSGA II (Non dominated Sorting Genetic Algorithm-II) permet de préserver la diversité des populations, en sauvegardant les meilleures solutions trouvées lors des générations précédentes. Pour aboutir à des résultats plus performants, nous avons effectué une amélioration au niveau de la phase d'initialisation de la population de la NSGA-II en se basant sur une variante adaptée de la fameuse heuristique NEH donnant lieu à une nouvelle technique d'optimisation multi-objectifs des problèmes d'ordonnancement nommée RNSGA-II.ver.2. Pour les deux types de problèmes, les résultats numériques ont prouvé que l'approche proposée a permis de fournir des solutions Pareto-optimal pertinentes en termes de qualité, de diversité et de dispersion.

L'attention a été, enfin, focalisée sur la validation expérimentale de l'ensemble des approches et des modèles proposés. Elle a été effectuée sur une ligne de conditionnement des produits pharmaceutiques à travers une approche de couplage simulation/optimisation. Cette approche est basée sur le couplage des approches proposées avec un modèles LCPP modélisé et simulé sur le logiciel Witness Horizon. Une étude comparative a été établi entre les résultats obtenus dans la

phase préparatoire sans couplage et ceux fournis à travers le couplage. Cette étude a démontré l'efficacité des approches de résolution suggérées et des modèles mathématiques développés dans l'intégrité des travaux de recherche établis dans le cadre de ce projet de doctorat.

Comme perspectives de recherche futures, plusieurs pistes nous semblent intéressantes à explorer. Comme première ouverture, nous envisageons l'adaptation de l'ensemble des techniques développées sur l'extension de l'atelier d'ordonnancement à l'étude étant l'atelier à cheminement unique avec machines en exemplaires multiples nommé flow-shop hybride. Nous signalons que d'ores et déjà une revue de littérature suivie d'une étude bibliographique sur les problèmes d'ordonnancement des ateliers de type flow-shop hybride a été effectuée et publiée. Nous envisageons ainsi de projeter les résultats obtenus dans la résolution du système hybride sur le cas de la ligne de production des produits pharmaceutiques afin de prouver davantage l'efficacité des outils utilisés et d'améliorer de plus en plus les couplages simulation/ optimisation suggères.

Une autre perspective pour la phase de la validation expérimentale serait de prendre en compte des incertitudes dans l'estimation des paramètres afin d'approximer au plus juste le comportement du système. Il serait aussi judicieux d'étendre les approches proposées à d'autres industries qui présentent les mêmes types de contraintes telles que les industries chimiques ou agroalimentaires.

De surcroit, l'intégration d'autres aspects possédant des sens forts dans les cas concrets comme la considération des contraintes de blocage mixtes et la disponibilité des ressources semble être également une direction intéressante. De plus, d'autres techniques et méthodes peuvent être étudiées et utilisées pour améliorer la convergence et le temps de calcul des variables requises dans les modèles actuels.

Parallèlement à ces perspectives, l'application des principes du raisonnement énergétique est également une piste de recherche très enrichissante à suivre. De nos jours, la maîtrise de la consommation d'énergie est devenue un enjeu important, notamment, pour les chercheurs et pour les industriels qui font face à des difficultés quotidiennes résultantes de l'augmentation des coûts énergétiques et des préoccupations environnementales. A cet égard, la considération de cet aspect énergétique a pour finalité de garantir la durabilité de l'ensemble des activités des entreprises en minimisant les effets secondaires sur l'environnement. Ceci peut se présenter par la minimisation de la consommation énergétique et aussi par la réduction du coût énergétique; c'est dans ce contexte que cette perspective se situe, nous envisageons principalement le développement d'un formalisme mathématique portant sur l'optimisation multi-objectifs des critères abordés dans ce mémoire pour la résolution des systèmes d'ordonnancement étudiés et leur extension en considérant l'aspect énergétique non seulement en tant que restriction à respecter mais potentiellement en tant que critère majeur d'optimisation.

Finalement, à long terme, nous prévoyons d'étendre nos recherches à d'autres thématiques liées aux systèmes de production

Bibliographie

Bibliographie

[ABA, 2000] Abdi INK, Hall NG, "Sriskandarajh C. Minimizing cycle time in a blocking flowshop", Operations Research, 48, pp. 177–80, 2000.

[AHM, 2012] Ahmadizar, Fardin, "A new ant colony algorithm for makespan minimization in permutation flow shops", Computers & industrial engineering, 63(2), pp. 355-361, 2012.

[ALI, 2018] Allahverdi, ali, Aydilek, Harun, Et Aydilek, Asiye, "No-wait flowshop scheduling problem with two criteria, total tardiness and makespan", European Journal of Operational Research, 269(2), pp. 590-601, 2018.

[ALL, 2004] Allahverdi, A., "A new heuristic for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness", Computers & Operations Research, 31(2), pp. 157-180, 2004.

[ALL, 2008] Allahverdi, A., C. T. Ng, T. E. Cheng, and M. Y. Kovalyov, "A Survey of Scheduling Problems with Setup Times or Costs", European Journal of Operational Research, 187(3), pp. 985–1032, 2008.

[AND, 1998] Andradottir S., "Simulation optimization", (edited by Jerry Banks, John Wiley & sons inc. New York), in Handbook of simulation, pp. 307-334, 1998.

[ARD, 2013] Abouei Ardakan, Mostafa, Ali Hakimian, and Mohammad Taghi Rezvan, "A branch-and-bound algorithm for minimising the number of tardy jobs in a two-machine flow-shop problem with release dates", International Journal of Computer Integrated Manufacturing, 27(6), pp. 519-528, 2014.

[ARM, 2004] Armentano, V. A., & Claudio, J. E., "An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem", Journal of Heuristics, 10(5), pp. 463-481, 2004.

[ARM, 2004] Arroyo, J. E. C., and V. A. Armentano, "A partial enumeration heuristic for multi-objective flowshop scheduling problems", Journal of the Operational Research Society, 55(9), pp. 1000-1007, 2004.

[ARR, 2005] J.C. Arroyo, V.A. Armentano, "Genetic local search for multi-objective flowshop scheduling problems", Eur J Oper Res, 167(3), pp. 717–738, 2005.

[ARR, 2011] Arroyo, José Elias Claudio, and Ana Amélia de Souza Pereira, "A GRASP heuristic for the multi-objective permutation flowshop scheduling problem", The International Journal of Advanced Manufacturing Technology, 55(5-8), pp. 741-753, 2011.

[AWA, 2012] A.EL Awady, K. H. Edi and P. Duquenne, "Flexible resources allocation techniques: characteristics and modelling", Int. J. Operational Research, 14(2), pp. 251-254, 2012.

[AXE, 1987] Axelrod, "The evolution of strategies in the iterated prisoner's dilemma", in L. D. Davis, ed., "Genetic algorithms and simulated annealing", Morgan Kaufmann, R., 1987.

[AZA, 1992] Azadivar, F., "A tutorial on simulation optimization", In Proceedings of the 24th conference on Winter simulation, ACM, pp. 198-204, 1992.

[BAK, 2010] Baker, K. R., & Keller, B., "Solving the single-machine sequencing problem using integer programming", Computers & Industrial Engineering, 59(4), pp. 730-735, 2010.

[BEL, 1954] R. Bellman, "The theory of dynamic programming", Bulletin of the American Mathematical Society, 60(6), pp. 503-515, 1954.

[CAR, 1988] CARLIER, P. CHRETIENNE, "Problèmes d'Ordonnancement : Modélisation, Complexité, Algorithmes" (330 pages), MASSON, Collection Etudes et Recherches en Informatique, Janvier 1988.

[CAR, 2001] Caraffa, V., S. Ianes, T. P. Bagchi, and C. Sriskandarajah, "Minimizing Makespan in a Blocking Flowshop using Genetic Algorithms", International Journal of Production Economics, 70(2), pp. 101–115, 2001.

[CHA, 1999] K. Chakravarty, C. Rajendran, "A heuristic for scheduling in flowshop with bicriteria of makespan and maximum tardiness minimization", Prod Plan Control, 10(7), pp. 707–714, 1999.

[CHA, 2003] P.-C. Chang, J.-C.Hsieh, S.-G. Lin, "The development of gradual-priority weighting approach for the muti-objective flowshop scheduling prblem", Int J Prod Econ, 79(3), pp. 171-183, 2003.

[CHE, 2008] Chen C.H., Fu M., and Shi L., "Simulation and Optimization", Tutorials in Operations Research, Informs, Hanover, MD, pp. 247-260, 2008.

[CHE, 2005] Chen*, J-S., and JC-H. Pan, "Minimising mean tardiness with alternative operations in two-machine flow-shop scheduling", International journal of systems science, 36(12), pp. 757-766, 2005.

[CHU, 1996] Chu, C. & Proth, J.-M., "L'ordonnancement et ses applications", Paris: Masson, 1996.

[COL, 2002] Collette Y., Siarry P., "Optimisation mutliobjectif", Eyrolles, Paris, 2002.

[DAM, 2012] Damodaran, Purushothaman, Anantha Gangadhara Rao, and Siddharth Mestry, "Particle swarm optimization for scheduling batch processing machines in a permutation flowshop", The International Journal of Advanced Manufacturing Technology, 64(5-8), pp. 989-1000, 2013.

[DAN, 1990] R.L. Daniels, R.J. Chambers, "Multiobjective flowshop scheduling", Nav Res Logist Q, 37(6), pp. 981–995, 1990.

[DEB, 2002] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T., "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE transactions on evolutionary computation, 6(2), pp. 182-197, 2002.

[DET, 2015] Detienne, Boris, Ruslan Sadykov, and Shunji Tanaka, "The two-machine flowshop total completion time problem: A branch-and-bound based on Network-flow formulation", 7th Multidisciplinary International Conference on Scheduling: Theory and Applications, 2015.

[DOL, 1997] Dolgui A. et Ofitserov D., "A stochastic method for discrete and continuous optimization in manufacturing system", Journal of Intelligent manufacturing, 8(5), pp. 405-515, 1997.

[DOR, 1998] Dorigo M., V. Maniezzo et A. Colorni, "Positive feedback as a search strategy", rapport technique numéro 91-016, Dip. Elettronica, Politecnico di Milano, Italy, 1991.

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

[DUB, 2011] Dubois-Lacoste, Jérémie, Manuel López-Ibáñez, and Thomas Stützle, "A hybrid TP+ PLS algorithm for bi-objective flow-shop scheduling problems", Computers & Operations Research, 38(8), pp. 1219-1236, 2011.

[DUV, 2000] D. Duvier, Etude de l'hybridation des métaheuristiques, "application à un problème d'ordonnancement de type jobshop", Thèse de Doctorat à Bibliographie 117 l'Université du Littoral Côte d'Opale, Calais, 2000.

[Edi, 2007] K. H. Edi, "Affectation flexible des ressources dans la planification des activités industrielles : prise en compte de la modulation d'horaires et de la polyvalence", Thèse de doctorat, Université Paul Sabatier Toulouse (France), 2007.

[FOU, 1985] Fourman, "Compaction of Symbolic Layout using Genetic Algorithms", In Genetic Algorithms and their Applications : Proceedings of the First International Conference on Genetic Algorithm, pp. 141-153, 1985.

[FRA, 2009] J.M. Framinan, "A fitness-based weighting mechanism for multicriteria flowshop scheduling using genetic algorithms", Int J Adv Manuf Technol, 43(9-10), pp. 939–948, 2009.

[FU, 1994] Fu, M.C., "Optimization via simulation", Annals of O.R., 53, pp. 199-248, 1994.

[FU, 2008] Fu, M., Chen, C.H., Shi, L., "Some topics for simulation optimization" Proceedings of the 2008 Winter Simulation Conference, 2008.

[GEI, 2007] M.J. Geiger, "On operators and search space topology in multi-objective flow shop scheduling", Eur J Oper Res, 181(1), pp. 195-206, 2007.

[GEI, 2011] Geiger, Martin Josef, "Decision support for multi-objective flow shop scheduling by the Pareto iterated local search methodology", Computers & Industrial Engineering, 61(3), pp. 805-812, 2011.

[GLO, 1998] Glover, F., & Laguna, M., "Tabu search. In Handbook of combinatorial optimization", Boston, MA, pp. 2093-2229, 1998.

[GOT, 1993] Lopez, P. & Roubellat, F., "Ordonnancement de la production", Paris: Hermès science publications, 2001.

[GRA, 1979] Graham, R.L., Lawler, E.L. & Rinnooy Kan, A.H.G., "Optimization and approximation in deterministic sequencing and scheduling : a survey", Annals of Discrete Mathematics, 5, pp. 287-326, 1979.

[GRA, 2000] Grabowski, J. and J. Pempera, "Sequencing of Jobs in Some Production System", European Journal of Operational Research, 125, pp. 535–550, 2000.

[GUA, 2018] Zhang, Guanghui, Xing, Keyi, et Cao, Feng, "Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion", Engineering, Applications of Artificial Intelligence, 76, pp. 96-107, 2018.

[GUP, 1999] Gupta, Jatinder ND., "Designing a tabu search algorithm for the two-stage flow shop problem with secondary criterion", Production Planning & Control, 10(3), pp. 251-265, 1999.

[GUP, 2002] Gupta, J. N., Hennig, K., & Werner, F., "Local search heuristics for two-stage flow shop problems with secondary criterion", Computers & Operations Research, 29(2), pp. 123-149, 2002.

[HAL, 1996] Hall NG, "Sriskandarajah CA survey of machine scheduling problems with blocking and no-wait in process", Operations Research, 44, pp. 510–25, 1996.

[HAN, 2012] Han, Y. Y., Q. K. Pan, J. Q. Li, and H. Y. Sang, "An Improved Artificial Bee Colony Algorithm for the Blocking Flowshop Scheduling Problem", International Journal of Advanced Manufacturing Technology, 60, pp. 1149–1159, 2012.

[HAN, 2014] Han, Y. Y., Gong, D. W., Sun, X. Y., & Pan, Q. K., "An improved NSGA-II algorithm for multi-objective lot-streaming flow shop scheduling problem", International Journal of Production Research, 52(8), pp. 2211-2231, 2014.

[HAN, 2015] Han, Y. Y., Gong, D., & Sun, X., "A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking", Engineering Optimization, 47(7), pp. 927-946, 2015.

[HAN, 2016] Han, Y., Gong, D., Li, J., & Zhang, Y., "Solving the blocking flowshop scheduling problem with makespan using a modified fruit fly optimisation algorithm", International Journal of Production Research, 54(22), pp. 6782-6797, 2016.

[HOL, 1970] J. H. Holland, "Adaptation In Natural And Artificial Systems", University of Michigan Press, 1975.

[HOO, 2005] Hoogeveen, H. Multicriteria scheduling, "European Journal of Operational Research", 167, pp.592-623, 2005.

[HOU, 2003] Haouari, M., and T. Ladhari, "A branch-and-bound-based local search method for the flow shop problem", Journal of the Operational Research Society, 54(10), pp. 1076-1084, 2003.

[HUA, 2009] Huang, Rong-Hwa, and Chang-Lin Yang, "Solving a multi-objective overlapping flow-shop scheduling", The International Journal of Advanced Manufacturing Technology, 42(9-10), pp. 955-962, 2009.

[ISH, 1998] Ishibuchi, Hisao, and Tadahiko Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling", Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 28(3), pp. 392-403, 1998.

[ISH, 2003] Ishibuchi H, Yoshida T, Murata T., "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling", IEEE Transactions on Evolutionary Computation, 7, pp. 204–23, 2003.

[ISH, 2003] Ishibuchi, H., Yoshida, T., & Murata, T., "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling", IEEE transactions on evolutionary computation, 7(2), pp. 204-223, 2003.

[JOL, 2013] Jolai, F., Asefi, H., Rabiee, M., & Ramezani, P., "Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem", Scientia Iranica, 20(3), pp. 861-872, 2013.

[KAC, 2003] I. Kacem, "Ordonnancement multicritère des job-Shop flexibles : formulation, bornes inférieures et approche évolutionniste coopérative", Thèse de Doctorat, Université des Sciences et Techniques de Lille1, 2003.

[Kar, 2012] Asma Karray ; "Contribution a` l'ordonnancement d'ateliers agroalimentaires utilisant des m´ethodes d'optimisation hybrides", Autre, Ecole Centrale de Lille, Français, 2011.

[KEN, 1995] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", In : Proceedings of the IEEE International Conference on Neural Networks IV, Perth, Australia, pp. 1942–1948, 1995.

[KHA, 2011] Khan, B. Shahul Hamid, and Kannan Govindan, "A multi-objective simulated annealing algorithm for permutation flow shop scheduling problem", International Journal of Advanced Operations Management, 3(1), pp. 88-100, 2011.

[KHA, 2015] Housseyn Amin KAHOUADJI, "Contribution à l'évaluation et à l'optimisation de performances des systèmes de production à l'aide des Réseaux de Petri", Université de Tlemcen, 2015.

[KHA, 2007] Shahul Hamid Khan, B., G. Prabhaharan, and P. Asokan, "A grasp algorithm for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness", International Journal of Computer Mathematics, 84(12), pp. 1731-1741, 2007.

[KIR, 1983] KIRKPATRICK, Scott, GELATT, C. Daniel, et VECCHI, Mario P., "Optimization by simulated annealing", science, 220(4598), pp. 671-680, 1983.

[KRI, 1992] Krishnakumar, K. & Goldberg, D., "Control system optimization using genetic algorithm", Journal of Guidance, Control, and Dynamics, 15(3), pp. 735–740, 1992.

[KUM, 2016] Sanjeev Kumar, R., Padmanaban, K. P., & Rajkumar, M., "Minimizing makespan and total flow time in permutation flow shop scheduling problems using modified gravitational emulation local search algorithm", Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 0954405416645775, 2016.

[LAN, 1960] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems", doi:10.2307/1910129, Econometrica, 28(3), pp. 497–520, 1960.

[LEE, 2001] Lee, Wen-Chiung, and Chin-Chia Wu., "Minimizing the total flow time and the tardiness in a two-machine flow shop", International Journal of Systems Science, 32(3), pp. 365-373, 2001.

[LEI, 1990] Leisten, R., "Flowshop sequencing problems with limited buffer storage", International Journal of Production Research, 28, pp. 2085–100, 1990.

[LIA, 1997] Liao, Ch J., W. C. Yu, and C. B. Joe., "Bicriterion scheduling in the two-machine flowshop", Journal of the Operational Research Society, 48(9), pp. 929-935, 1997.

[LIA, 2007] C.-J. Liao, C.-T. Tseng, P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems", Comput Oper Res, 34(10), pp. 3099–3111, 2007.

[LIN, 2006] Lin, B. M. T., and J. M. Wu., "Bicriteria scheduling in a two-machine permutation flowshop", International journal of production research, 44(12), pp. 2299-2312, 2006.

[LIN, 2008] B.M.T. Lin, C.Y. Lu, S.J. Shyu, C.Y. Tsai, "Development of new features of ant colony optimization for flowshop scheduling", Int. J. Production Economics, 112(2), pp. 742–755, 2008.

[LIN, 2009] Lin, S-W., and K-C. Ying, "Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems", International Journal of Production Research, 47(5), pp. 1411-1424, 2009.

[LIN, 2011] Lin, Shih-Wei, et al., "Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates", Computers & Operations Research, 38(5), pp. 809-815, 2011.

[LIN, 2016] Lin, Shih-Wei, and Kuo-Ching Ying, "Minimizing makespan for solving the distributed no-wait flowshop scheduling problem", Computers & Industrial Engineering, 99, pp. 202-209, 2016.

[LIU, 2013] Liu, Yen-Cheng, Kuei-Tang Fang, and Bertrand Lin, "A branch-and-bound algorithm for makespan minimization in differentiation flow shops",Engineering Optimization, 45(12), pp. 1397-1408, 2013.

[LIU, 2001] Liu JY, Reeves CR. "Constructive and composite heuristic solutions to the P//Sci scheduling problem", European Journal of Operational Research, 132(2), pp. 439–52, 2001.

[LOP, 2001] Lopez, P. & Roubellat, F., "Ordonnancement de la production", Paris: Hermès science publications, 2001.

[LOU, 2003] Lourenço, H., Martin, O. & Stützle, T., "Iterated local search. In F. Glover & G. Kochenberger", eds. Handbook of Metaheuristics, International Series in Operations Research & Management Science. Kluwer Academic Publishers, 57, pp. 320-53, 2003.

[LOU, 2005] Loukil, T., Teghem, J., & Tuyttens, D., "Solving multi-objective production scheduling problems using metaheuristics", European journal of operational research, 161(1), pp. 42-61, 2005.

[MAC, 1993] B.L. McCarthy, J. Liu, "Addressing the gap in schedulingresearch: a review of optimization and heuristic methods in production scheduling", International Journal of Production Research, 31(1), pp. 59-79, 1993.

[MAD, 2009] Madhushini, N., Chandrasekharan Rajendran, and Y. Deepa, "Branch-and-bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs". Journal of the Operational Research Society, 60(7), pp. 991-1004, 2009.

[MAN, 2016] Eddaly, M., Jarboui, B., & Siarry, P., "Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem", Journal of Computational Design and Engineering, 3(4), pp. 295-311, 2016.

[MAR, 1996] Marco, N., Godart, C., Désidéri, J.-A., Mantel, B. & Périaux, J., "A genetic algorithm compared with a gradient-based method for the solution of an active-control model problem, Technical report, INRIA", Rapport de Recherche de l'INRIA - Projet SINUS, (2948), 1996.

[MAR, 2013] Marinakis, Yannis, and Magdalene Marinaki, "Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem", Soft Computing, 17(7), pp. 1159-1173, 2013.

[MAT, 2017] Matin, Hossein NZ, Nasser Salmasi, and Omid Shahvari, "Makespan minimization in flowshop batch processing problem with different batch compositions on machines", International Journal of Production Economics, 193, pp. 832-844, 2017.

[MCC, 1989] McCormick, S. T., M. L. Pinedo, S. Shenker, and B. Wolf, "Sequencing in an Assembly Line with Blocking to Minimize Cycle Time", Operations Research, 37(6), pp. 925–935, 1989.

[MES, 2010] Mesgarpour, Mohammad, Nureddin Kirkavak, and Hakan Ozaktas, "Bicriteria scheduling problem on the two-machine flowshop using simulated annealing", Evolutionary Computation in Combinatorial Optimization, Springer Berlin Heidelberg, pp. 166-177, 2010.

[MIC, 1992] Michalewicz, Z., "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, 1992.

[MOL, 2015] Molina-Sánchez, L., and E. González-Neira, "GRASP to minimize total weighted tardiness in a permutation flow shop environment", International Journal of Industrial Engineering Computations, 7(1), pp. 161-176, 2015.

[MOR, 2014] Morizawa, Kazuko, "A Branch-and-Bound Based Heuristic Algorithm for Minimizing Makespan in Machining-Assembly Flowshop Scheduling", Engineering, 6(13), pp. 877, 2014.

[MOS, 2013] Moslehi, Ghasem, and Danial Khorasanian, "Optimizing blocking flow shop scheduling problem with total completion time criterion", Computers & Operations Research, 40(7), pp. 1874-1883, 2013.

[MUR, 1996] Murata, Tadahiko, Hisao Ishibuchi, and Hideo Tanaka, "Multi-objective genetic algorithm and its applications to flowshop scheduling", Computers & Industrial Engineering, 30(4), pp. 957-968, 1996.

[MUR, 2001] Murata, Tadahiko, Hisao Ishibuchi, and Mitsuo Gen, "Specification of genetic search directions in cellular multi-objective genetic algorithms", Evolutionary multi-criterion optimization. Springer Berlin Heidelberg, 2001.

[NAG, 1995] A. Nagar, S.S. Heragu, J. Haddock, "A branch and bound approach for two-machine flowshop scheduling problem", J Oper Res Soc, 46(6), pp. 721–734, 1995.

[NAG, 1996] A.Nagar, S.S. Heragu, J. Haddock, "A combined branch anad bound anad genetic algorithm based approach for a flowshoop scheduling problem", Ann Oper Res, 63(3), pp. 397-414, 1996.

[NAW, 1983] Nawaz, M., Enscore, E.E.J. and Ham, I., "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem", OMEGA – International Journal of Management Science, 11, pp. 91–95, 1983.

[NEP, 1996] Neppalli, Venkata Ranga, Chuen-Lung Chen, and Jatinder ND Gupta, "Genetic algorithms for the two-stage bicriteria flowshop problem", European journal of operational research, 95(2), pp. 356-373, 1996.

[NES, 2012] Nessah, R. and Kacem, I., "Branch-and-bound method for minimizing the weighted completion time scheduling problem on a single machine with release dates", Computers & OR, 39(3), pp. 471-478, 2012.

[NOU, 2017] Nouri, N., & Ladhari, T., "Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking", Annals of Operations Research, pp. 1-18, 2017.

[OSM, 1989] M.L. Smith, R.A. Dudek, "A general algorithm for solution of the n-job, m-machine sequencing problem of the flowshop", Oper Res, 15 (1), pp. 71–82, 1967.

[OUA, 2001] Ouabiba M., Castagna P., et Mebarki N., "Couplage entre des méthodes d'optimisation itératives et Des modèles de simulation a événements discrets", Actes de la 3ème conférence francophone de Modélisation et Simulation MOSIM'01, Troyes, France, 2001.

[PAN, 2002] Pan, Jason Chao-Hsien, Jen-Shiang Chen, and Chii-Ming Chao, "Minimizing tardiness in a two-machine flow-shop", Computers & Operations Research, 29(7), pp. 869-885, 2002.

[Pan, 2010] Pan, Q. K., and L. Wang, "Effective Heuristics for the Blocking Flowshop Scheduling Problem with Makespan Minimization", Omega, 40(2), pp. 218–229, 2012.

[PAN, 2012] Pan, Quan-Ke, and Rubén Ruiz, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times", Omega, 40(2), pp. 166-180, 2012.

[PAP, 1980] Papadimitriou CH, Kanellakis PC, "Flowshop scheduling with limited temporary storage", Journal of the Association for Computing Machinery, 27, pp. 533–49, 1980.

[PAR, 1985] Parunak, H.V.D., "Manufacturing experience with the contract net. In Proceedings of the Fith Workshop on Distributed Artificial Intelligence", 1985.

[PAR, 2015] Parsamanesh, A. H., & Sahraeian, R., "Solving Single Machine Sequencing to Minimize Maximum Lateness Problem Using Mixed Integer Programming", Journal of Quality Engineering and Production Optimization, doi: 10.22070/jqepo.2015.187, 1(1), pp. 33-42, 2015.

[PAS, 2006] T. Pasupathy, C. Rajendran, R.K. Suresh, "A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs", Int J Adv Manuf Technol, 27, pp. 804–815, 2006.

[PER, 2000] Pereira, R., "Genetic algorithm optimisation for finance and investment", Technical report, La Trobe University, 2000.

[PIE, 2001] Pierreval, H., et Paris J.L., "De l'optimisation de systèmes via la simulation à la configuration des systèmes via la simulation", Actes de la 3ème conférence francophone de Modélisation et Simulation MOSIM'01, Troyes, France, 2001.

[PON, 2004] Ponnambalam, S. G., Jagannathan, H., Kataria, M., & Gadicherla, A., "A TSP-GA multi-objective algorithm for flow-shop scheduling", The International Journal of Advanced Manufacturing Technology, 23, pp. 909-915, 2004.

[POT, 1982] Potts, C. N., & Van Wassenhove, L. N., "A decomposition algorithm for the single machine total tardiness problem", Operations Research Letters, 1(5), pp. 177-181, 1982.

[PUC, 2005] Puchinger, J., Raidl, G.R., "Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification", In: Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, LNCS, Springer, Berlin, 3562, pp. 41–53, 2005 .

[PUG, 2014] Pugazhenthi, R., "A Genetic Algorithm Applied Heuristic to Minimize the Makespan in a Flow Shop", Procedia Engineering, 97, pp. 1735-1744, 2014.

[QIA, 2009] Qian, B., Wang, L., Huang, D. X., Wang, W. L., & Wang, X., "An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers", Computers & Operations Research, 36(1), pp. 209-233, 2009.

[RAB, 2011] A. Rabanimotlagh, "An efficient ant colony optimization algorithm for multiobjective flow shop scheduling problem", World Acad of Sci Eng Technol, 5(3), pp. 598–604, 2011.

[RAH, 2007] A.R. Rahimi-Vahed, S.M. Mirghorbani, "A multi-objective particle swarm for a flowshop scheduling problem", J Comb Optim, 13(1), pp. 79–102, 2007.

[RAH, 2007] Rahimi-Vahed, A., & Mirzaei, A. H. (2008), "Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm", Soft Computing, 12(5), pp. 435-452, 2007.

[RAJ, 1992] C. Rajendran, "Two-stage flowshop scheduling problem with bicriteria", J Oper Res Soc, 43(9), pp. 871–884, 1992.

[RAJ, 2004] Chandrasekharan Rajendran ,Hans Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs", European Journal of Operational Research, 155(2), pp. 426–438, 2004.

[RAJ, 2009] R. Rajkumar, P. Shahabudeen, "Bi-criteria improved genetic algorithm for scheduling in flowshops to minimise makespan and total flowtime of jobs", Int J Comp Integ M, 22(10), pp. 987–998, 2009.

[RAV, 2005] Ravindran, D., Selvakumar, S. J., Sivaraman, R., & Haq, A. N., "Flow shop scheduling with multiple objective of minimizing makespan and total flow time", The international journal of advanced manufacturing technology, 25(9-10), pp. 1007-1012, 2005.

[REN, 2015] Ren, Tao, et al., "A Local Search Algorithm for the Flow Shop Scheduling Problem with Release Dates", Discrete Dynamics in Nature and Society, 2015.

[RIB, 2011] Ribas I, Companys R, Tort-Martorell X., "An iterated greedy algorithm for the flowshop scheduling with blocking", OMEGA-The international Journal of Management Science, 39, pp. 293–301, 2011.

[RIN, 1976] A. H. G. Rinnooy Kan, "Machine Sequencing Problems: Classification, Complexity and Computation", Nijhoff, The Hague, 1976

[ROD, 1988] F.A. Rodammer et K. Preston White, "A recent survey of production scheduling", IEEE Transaction on Systems, Man and Cybernetics, pp. 6-18, 1999.

[RON, 2001] Ronconi, D. P., and V. A. Armentano, "Lower Bounding Schemes for Flowshops with Blocking in-Process", Journal of the Operational Research Society, 52, pp. 1289–1297, 2001.

[RON, 2004] Ronconi, D. P., "A note on constructive heuristics for the flowshop problem with blocking", International Journal of Production Economics, 87(1), pp. 39-48, 2004.

[RON, 2005] Ronconi, D.P., "A branch-and-bound algorithm to minimize the makespan in a flowshop problem with blocking", Annals of Operations Research, 138(1), pp. 53–56, 2005.

[RON, 2012] Ronconi, D. P. & Birgin, E. G., "Mixed-integer programming models for flow shop scheduling problems minimizing the total earliness and tardiness", In Just-in-Time systems, Springer, New York, NY, pp. 91-105, 2012.

[ROY, 1993] Roy, B., & Bouyssou, D., "Aide multicritère à la décision: méthodes et cas", Paris: Economica, p. 695, 1993.

[RUB, 2019] RUIZ, Rubén, PAN, Quan-Ke, et NADERI, "Bahman. Iterated Greedy methods for the distributed permutation flowshop scheduling problem", Omega, 83, pp. 213-222, 2019.

[SAD, 2015] Sadaqa, M., & Moraga, R. J., "Scheduling Blocking Flowshops Using Meta-RaPS", In Procedia Computer Science. 61, pp. 533-538, 2015.

[SAY, 1999] Sayin, Serpil, and Selcuk Karabati, "Theory and Methodology A bicriteria approach to the two-machine flow shop scheduling problem", European journal of operational research, 113(2), pp. 435-449, 1999.

[SCH, 1985] Schaffer David, pj p, "Multiple Objective Optimisation with Vector Evaluated Genetic Algorithm", In genetic Algorithm and their Applications : Proceedings of the First International Conference on Genetic Algorithm, pp. 93-100, 1985.

[SHI, 2015] Wang, Shijin, Ming Liu, and Chengbin Chu, "A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling", International Journal of Production Research, 53(4), pp. 1143-1167, 2015.

[SMI ,1967] M.L. Smith, R.A. Dudek, "A general algorithm for solution of the n-job, m-machine sequencing problem of the flowshop", Oper Res, 15(1), pp. 71–82, 1967.

[SMU, 2015] Smutnicki, C., Pempera, J., Rudy, J., & Żelazny, D., "A new approach for multi-criteria scheduling", Computers & Industrial Engineering, 90, pp. 212-220, 2015.

[SRI, 1996] C. Sridhar, C. Rajendran, "Scheduling in flowshop and cellular manufacturing systems with multiple objectives:A genetic algorithmic approach", Prod Plan Control, 7(4), pp. 374–382, 1996.

[SUR, 2004] Suresh, R. K., and K. M. Mohanasundaram, "Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives", Cybernetics and Intelligent Systems, IEEE Conference on, 2, 2004.

[T'KI, 2002] T'kindt, V. & Billaut, J.-C., "Multicriteria Scheduling : Theory, Models and Algorithms", Berlin: Srpinger, 2002.

[T'KI, 2003] T'kindt, Vincent, Jatinder ND Gupta, and Jean-Charles Billaut, "Two-machine flowshop scheduling with a secondary criterion", Computers & Operations Research, 30(4), pp. 505-526, 2003.

[TAG, 1962] Genichi TAGUCHI, "Studies on mathematical statistics for quality control", Doctoral thesis, Kushu University, March 1962.

[TAI, 1983] Taillard, E., "Benchmarks for Basic Scheduling Problems", European Journal of Operational Research, 64, pp. 278–285, 1993.

[TAS, 2015] Tasgetiren, M. Fatih, et al., "A populated local search with differential evolution for blocking flowshop scheduling problem", Evolutionary Computation (CEC), 2015 IEEE Congress on. IEEE, 2015.

[TAV, 2007] Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. R., & Mirzaei, A. H., "Solving a bi-criteria permutation flow shop problem using immune algorithm", In Computational Intelligence in Scheduling, SCIS'07, IEEE Symposium on, pp. 49-56, 2007.

[TOK, 2004] Toktaş, Berkin, Meral Azizoğlu, and Suna Kondakcı Köksalan, "Two-machine flow shop scheduling with two criteria: Maximum earliness and makespan", European Journal of Operational Research, 157(2), pp. 286-295, 2004.

[TRA, 2013] TRABELSI, Wajdi, "Ordonnancement des systèmes de production flexibles soumis à différents types de contraintes de blocage", Thèse de doctorat. Universite de Lorraine, 2012.

[TSA, 2014] Tsai, Jinn-Tsong, Ching-I. Yang, and Jyh-Horng Chou, "Hybrid sliding level Taguchi-based particle swarm optimization for flowshop scheduling problems", Applied Soft Computing, 15, pp. 177-192, 2014.

[VAI, 2012] Schaffer David, "pj p Multiple Objective Optimisation with Vector Evaluated Genetic Algorithm", In genetic Algorithm and their Applications : Proceedings of the First International Conference on Genetic Algorithm, pp. 93-100, 1985.

[VAR, 2005] Varadharajan, T. K., and Chandrasekharan Rajendran, "A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs", European Journal of Operational Research, 167(3), pp. 772-795, 2005.

[WAN, 2010] Wang L, Pan QK, Suganthan PN, Wang WH, Wang YM, "A novel hybrid discrete differential evolution algorithm for blocking flowshop scheduling problems", Computers and Operations Research, 37(3), pp. 509–20, 2010.

[WAN, 2010] Wang, L., Pan, Q. K., & Tasgetiren, M. F., "Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithm", Expert Systems with Applications, 37(12), pp. 7929-7936, 2010.

[WEI, 2011] Wei, X., Zhang, W., Weng, W., & Fujimura, S., "Multi-objective local search combined with NSGA-II for Bi-criteria permutation flow shop scheduling problem", IEEJ Transactions on Electronics, Information and Systems, 132(1), pp. 32-41, 2012.

[WEI, 2012] Wei, X., Zhang, W., Weng, W., & Fujimura, S., "Multi-objective local search combined with NSGA-II for Bi-criteria permutation flow shop scheduling problem", IEEJ Transactions on Electronics, Information and Systems, 132(1), pp. 32-41, 2012.

[YAG, 2008] B. Yagmahan, M.M. Yenisey, "Ant colony optimization for multi-objective flow shop scheduling problem", Comput Ind Eng, 54(3), pp. 411–420, 2008.

[YAG, 2010] B. Yagmahan, M.M. Yenisey, "A multi-objective ant colony system algorithm for flow shop scheduling problem", Expert Syst Appl, 37(2), pp. 1361–1368, 2010.

[YAN, 2007] Yandra, H. Tamura, "A new multiobjective genetic algorithm with heterogeneous population for solving flowshop scheduling problems", Int J Comp Integ M, 20(5), pp. 465–477, 2007.

[YEH, 1999] Yeh, Wei-Chang, "A new branch-and-bound approach for the n/2/flowshop/αF+βC max flowshop scheduling problem", Computers & operations research, 26(13), pp. 1293-1310, 1999.

[YEH, 2001] Yeh, Wei-Chang, "An efficient branch-and-bound algorithm for the two-machine bicriteria flowshop scheduling problem", Journal of Manufacturing Systems, 20(2), pp. 113-123, 2001.

[ZIT, 1999] Zitzler, E., & Thiele, L., "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach", IEEE transactions on Evolutionary Computation, 3(4), pp. 257-271, 1999.

Annexes

Annexe 1 : Liste des abbrèviations utilisèes dans la figure 2.1

ILOG CPLEX - Le cœur du système qui résout les problèmes de programmation mathématique.

ILOG CP- Fournit un moteur de programmation par contraintes.

ILOG IPL- Fournit des outils de développement des modèles.

ILOG Optimization- Fournit des outils de développement des applications.

ILOG Solver - La partie principale du système résout des applications en utilisant la programmation par contraintes.

ILOG Scheduler — Fournit des extensions pour résoudre des problèmes de planification.

ILOG Dispatcher — Fournit des extensions pour la résolution de problèmes de tournées de véhicules.

Annexe 2 : Rappel sur la complexité des problèmes d'ordonnancement

La complexité des problèmes d'ordonnancement se rapporte à plusieurs données dont nous pouvons citer : le nombre de machines et le nombre de tâches, la complexité des méthodes de résolution et celle des algorithmes employés. Nous dénombrons un certain nombre de types de notations grand-O :

- O (1) : Le traitement prend toujours le même temps, quel que soit le nombre de données.
- O(n) : Le temps de traitement est constamment proportionnel au nombre de données, quasiment linéaire.
- O (n !) : Le temps de traitement correspond à une factorielle du nombre de données.
- O(log(n)) : Evolution logarithmique.
- O(n*log(n)) : Evolution linéarithmique.
- O(c^n) : Evolution exponentielle.

Ainsi, nous distinguons deux types de complexité :

La complexité algorithmique : Dans l'étude de la complexité algorithmique, le temps d'exécution et l'espace mémoire sont les facteurs principaux permettant de comparer l'efficacité d'algorithmes résolvant le même problème. Dans une situation donnée, cela permet donc d'établir lequel des algorithmes disponibles est le plus optimal.

- **Complexité en temps :** Réaliser un calcul de complexité en temps revient à décompter le nombre d'opérations élémentaires (affectation, calcul arithmétique ou logique, comparaison…) effectuées par l'algorithme. Selon [KAR, 2012], la complexité en temps peut être définie comme suit :

  Définition 1 :

  On appelle complexité en temps d'un algorithme la fonction (f n) qui représente le nombre maximum d'opérations élémentaires effectuées pour résoudre un problème de taille n (n étant le nombre de variables décrivant le problème). On associe ainsi une unité de temps à chaque opération élémentaire.

Définition 2 :

On dit que f(n) ∈ O(g(n)), s'il existe une constante c>0 et un entier n_0 tels que ∀ n≥ n_0, |f(n)| ≤ c|g(n)|.

Définition 3 *:*

Un algorithme est dit polynômial si sa fonction de complexité f(n) ∈ O(p(n)), p étant un polynôme en n, c'est-à-dire, s'il existe une constante k>0 telle que (n) ∈ O(n^k).

- **Complexité en espace :** La complexité en espace est quant à elle la taille de la mémoire nécessaire pour stocker les différentes structures de données utilisées lors de l'exécution de l'algorithme.

La complexité problématique : La complexité problématique est relative principalement à la difficulté du problème à résoudre et du nombre des opérations élémentaires qu'un algorithme peut exécuter pour trouver l'optimum en fonction de la taille du problème. En fonction de son niveau de complexité, un problème peut s'approprier à l'une des quatre classes suivantes :

- **Les problèmes les plus difficiles :** Ce sont des problèmes pour lesquels il n'existe aucune approche de résolution ; ils sont dits indécidables ;
- **Les problèmes de la classe P (Polynomial) :** ils sont considérés comme « faisables ». Ce sont des problèmes décidés en temps polynomial pour lesquels il existe un algorithme efficace de complexité polynomiale pour leur résolution ;
- **Les problèmes de la classe NP (Non Déterministe Polynomial) :** Ce sont des problèmes NP-difficiles, qui ne peuvent à priori être résolus en un temps polynomial que par un algorithme polynomial non déterministe (Méthodes approchées) ;
- **Les problèmes NP-Complets :** Un problème est dit NP-complet est un problème de décision vérifiant les propriétés suivantes :
  - Il est possible de vérifier une solution efficacement (en temps polynomial) ; la classe des problèmes vérifiant cette propriété est notée NP ;
  - Tous les problèmes de la classe NP se ramènent à celui-ci via une réduction polynomiale ; cela signifie que le problème est au moins aussi difficile que tous les autres problèmes de la classe NP.

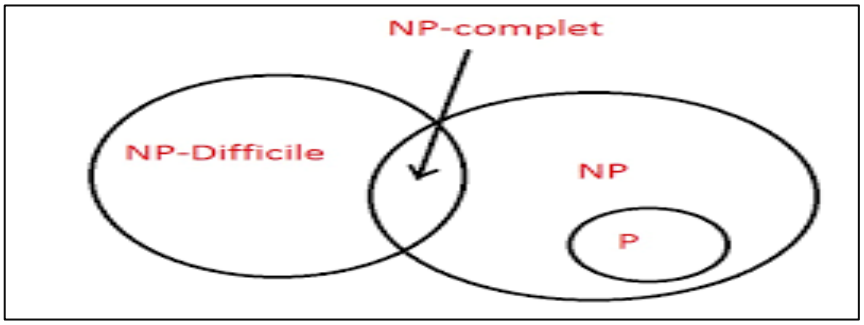


Figure A2.1 : Type de complexité des problèmes

Annexe 3 : Modélisation et représentation de l'ordonnancement

La modélisation des problèmes d'ordonnancement peut être effectuée avec des formalismes mathématiques ou à travers des représentations graphiques permettant de construire facilement les différentes contraintes entre les temps de début et de fin des opérations.

Les modèles mathématiques : Les variables, les restrictions à respecter, les critères d'optimisation et la fonction objectif sont formulées sous forme d'équations et d'inéquations mathématiques. Ces méthodes, généralement utilisées, sont directement exploitables par les approches de résolution.

Les méthodes graphiques : Un problème de planification d'atelier peut être modélisé par un graphe disjonctif (ou graphe potentiel – tâches). Sachant que chaque arc est pondéré par la durée de l'opération, dans ce type de graphe on trouve :

- Les nœuds représentant : les débuts des opérations du problème d'ordonnancement, les opérations fictives qui correspondent à la fin de chaque job et les dates du début et de la fin de l'ordonnancement.
- Les arcs conjonctifs liant deux opérations consécutives du même job, ils décrivent ainsi les contraintes de précédence qui existent entre ces opérations,
- Les arcs disjonctifs liant deux opérations appartenant à des jobs différents qui utilisent la même machine, ils précisent ainsi les conflits potentiels d'utilisation des ressources.

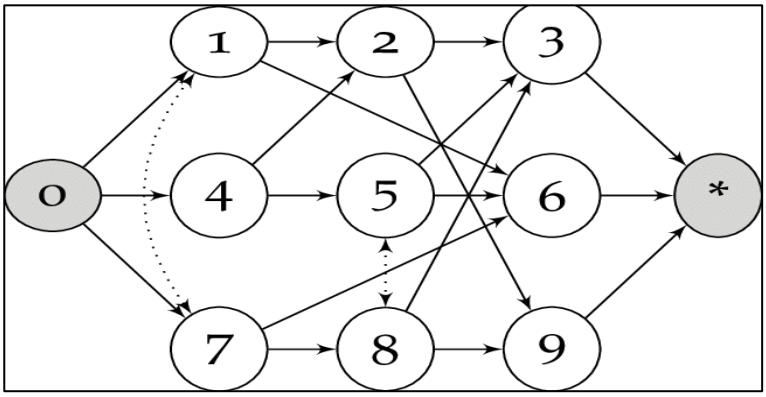


Figure A3.1 : Graphe disjonctif classique

Annexe 4 : La ligne de conditionnement des produits pharmaceutiques étudiée

- **Vue globale de la ligne étudiée :**

Figure A4.1 : Vue globale de la ligne de conditionnement des produits pharmaceutique

- **Le superviseur :**

Le système global est supervisé par un ordinateur permettant de programmer différentes gammes de production et le lancement des ordres de fabrication. Ce poste est muni d'un système de supervision et de gestion par ordinateur qui permet :

- La définition et le lancement des gammes : Définition des opérations par lesquelles doivent passer les palettes, le pourcentage du fer et la dose du B12 à mettre dans un flacon ;
- Les commandes ;
- Attribution d'une référence commerciale et d'un numéro de lot ;
- Gestion de la quantité à livrer, suivi et contrôle de la production.

Figure A4.2 : Illustration du poste superviseur

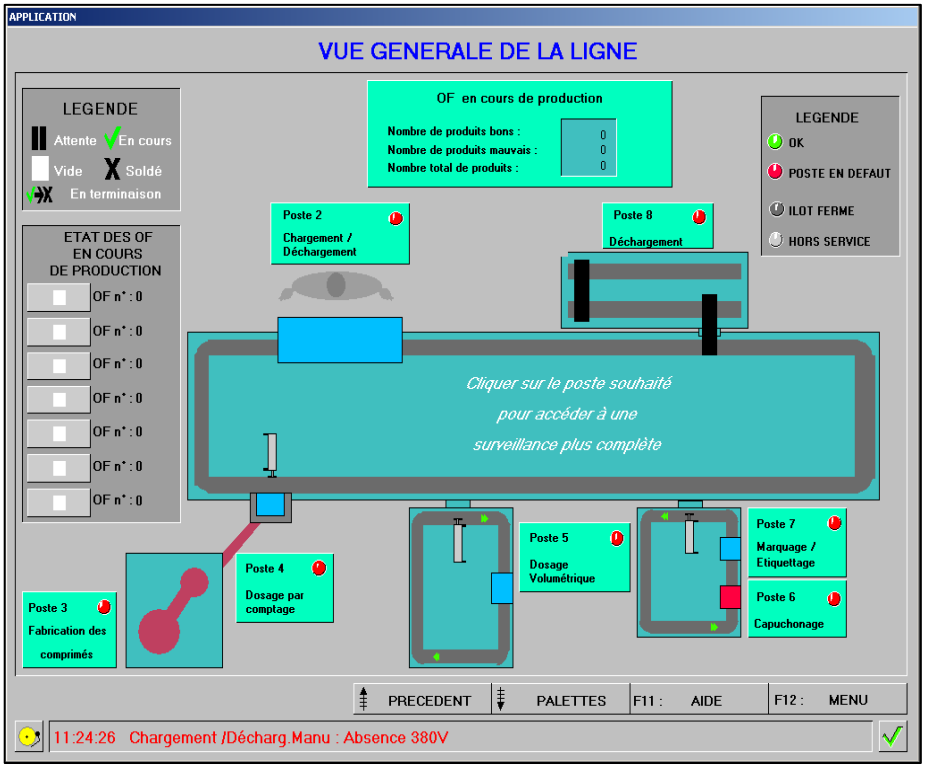


Figure A4.3 : Illustration du logiciel du poste superviseur

- ## Chargement et départ de palettes :

L'assemblage des flacons avec les bouchons sur des palettes arrivant par convoyeur est effectué à ce niveau. Cette tâche est effectuée manuellement par un opérateur.

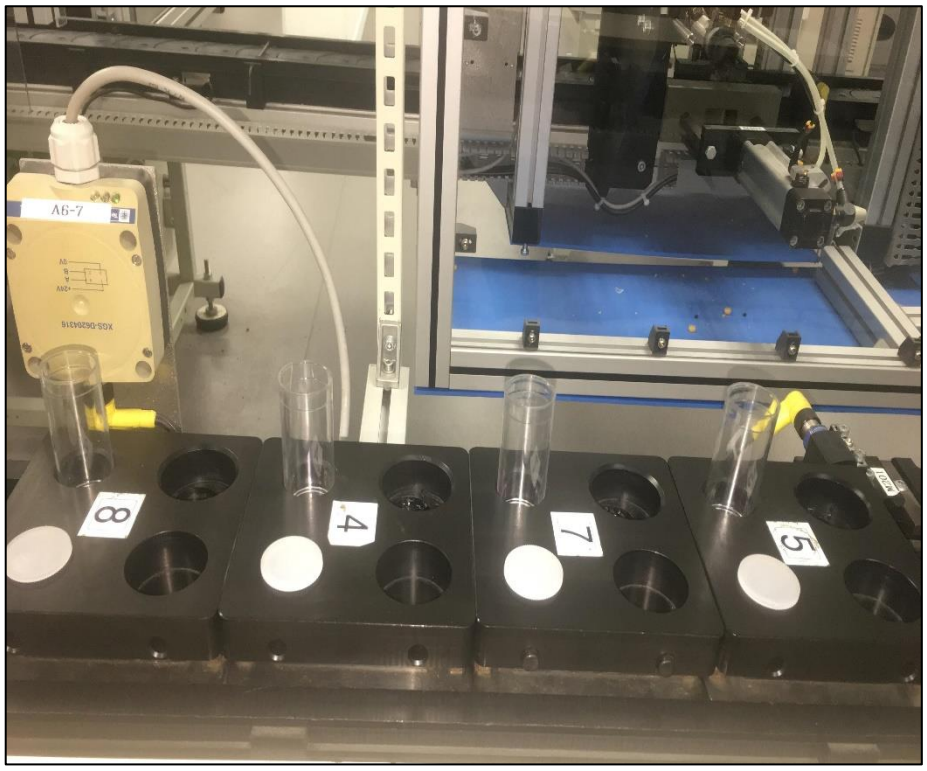


Figure A4.4 : Séquence des palettes prêtes pour le lancement

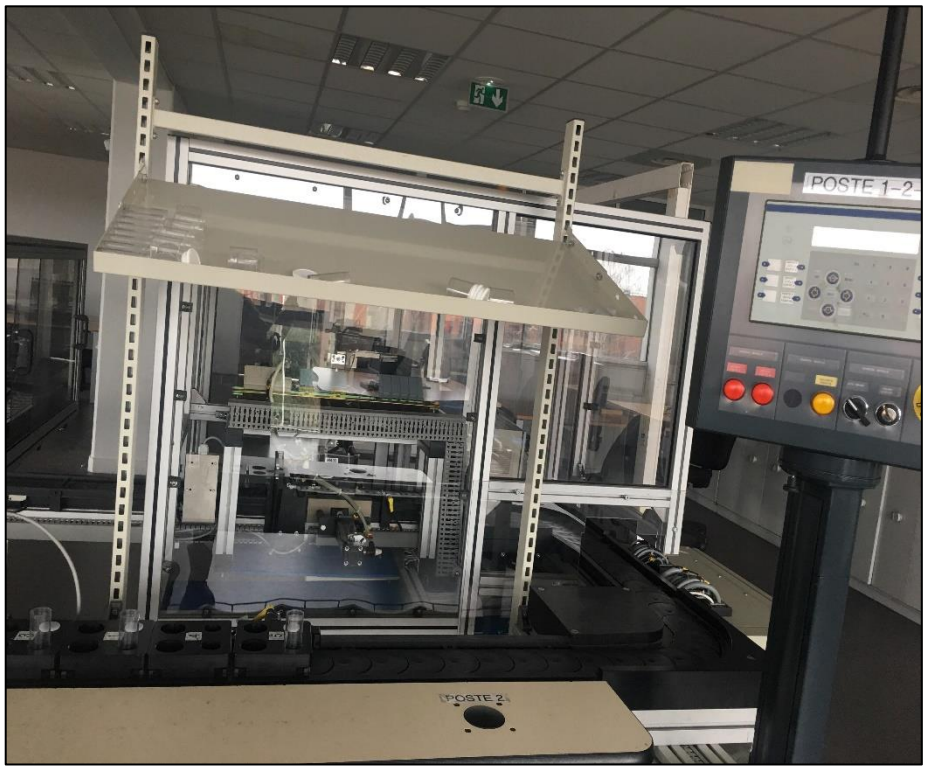


Figure A4.5 : Illustration du poste de lancement

- **Poste de remplissage des comprimés de fer :**

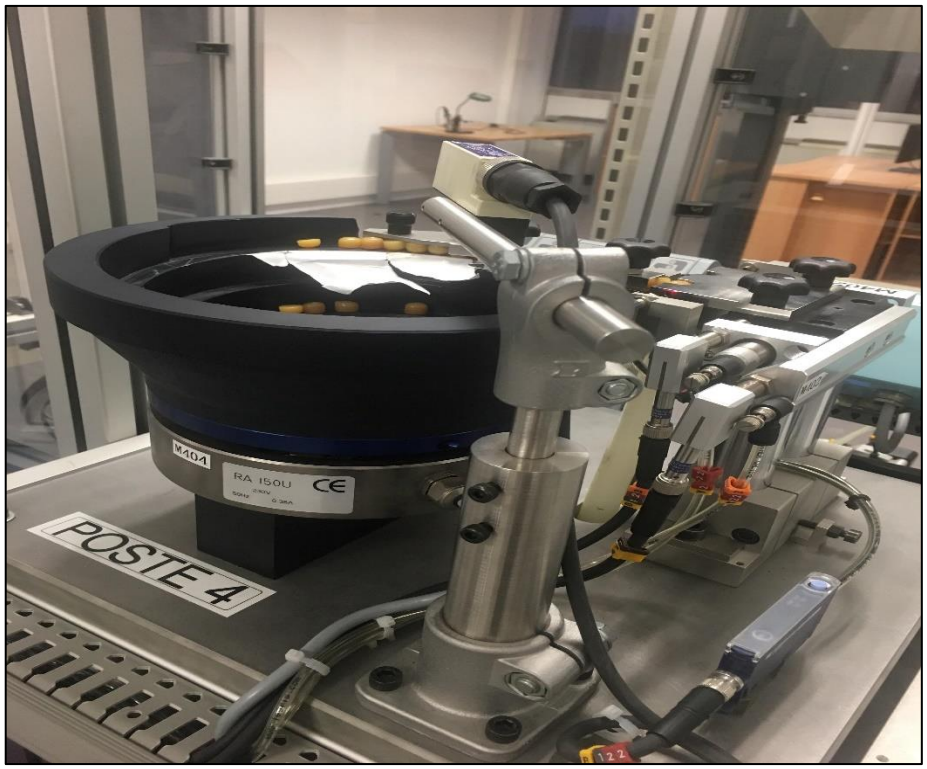
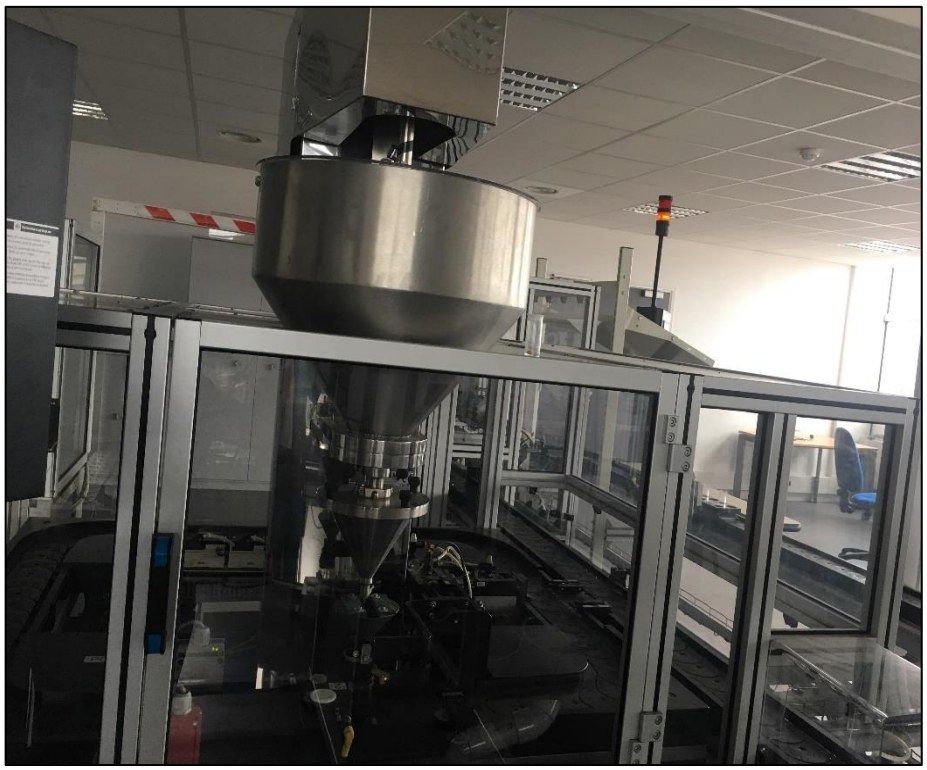


Figure A4.6 : Illustration du poste de remplissage des comprimés de fer



- **Poste de dosage du B12 :**

Figure A4.7 : Illustration du poste de dosage du B12

- **Postes de bouchonnage et étiquetage des flacons :**

Figure A4.8 : Illustration du poste du bouchonnage et étiquetage des flacons

- **Poste de déchargement des palettes :**

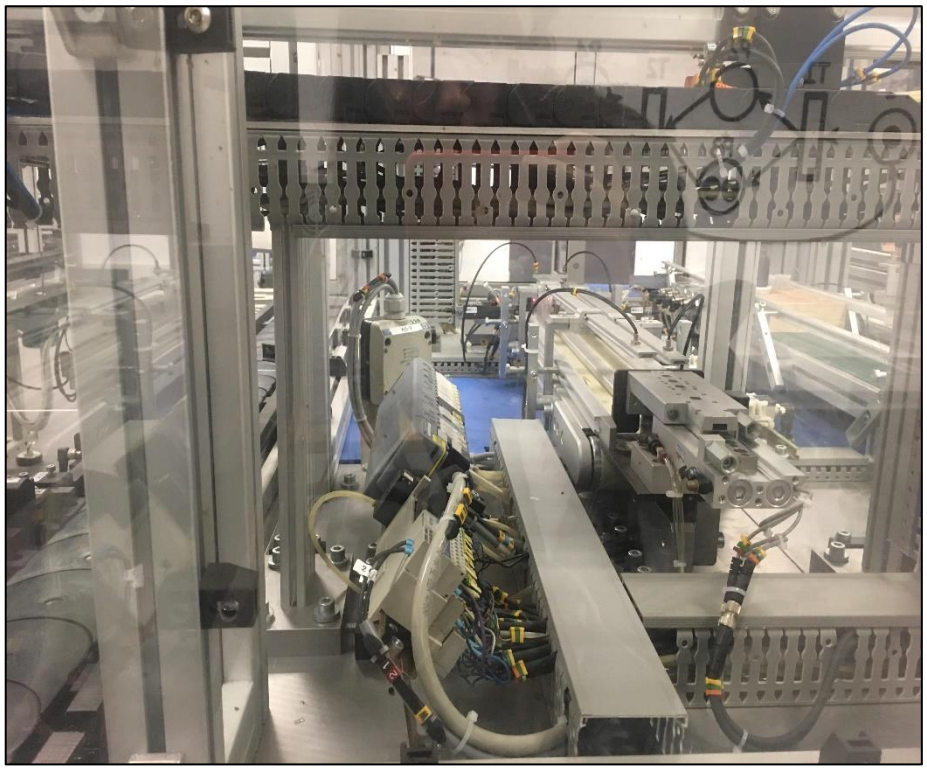


Figure A4.9 : Illustration du poste de déchargement des palettes

- **Poste de conditionnement des flacons :**

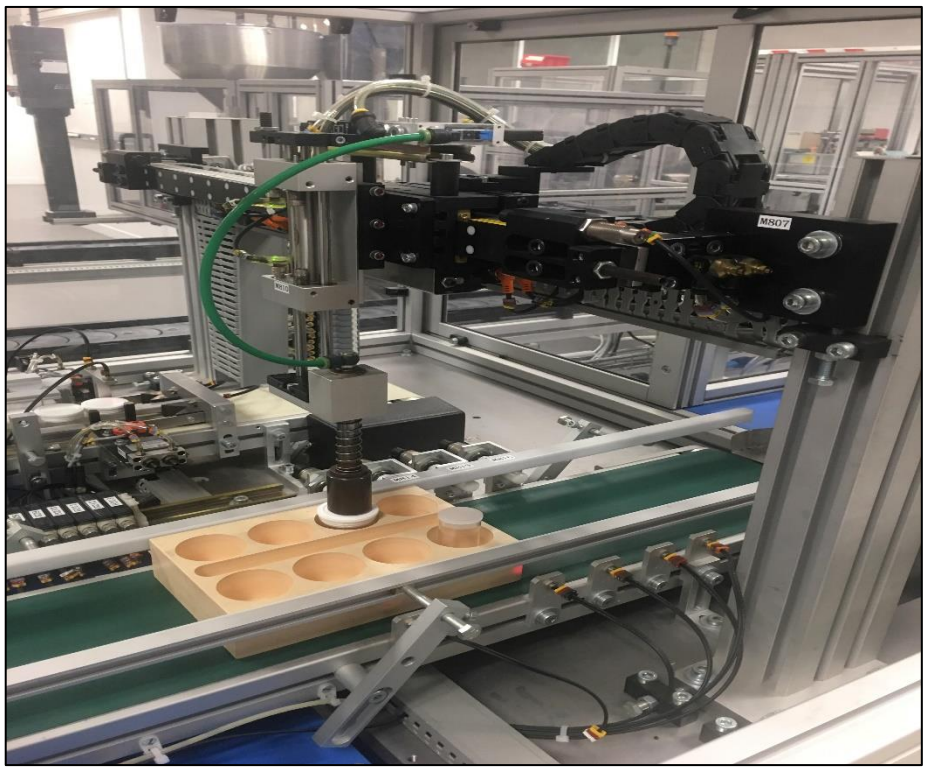


Figure A4.10 : Illustration du poste de conditionnement des flacons

Nous signalons que l'installation dispose aussi d'un poste fabrication de comprimés (voir figure A4.10), or, du fait qu'il est indépendant du circuit fermé de la ligne de conditionnement, nous ne l'avons pas pris en considération dans notre validation expérimentale.

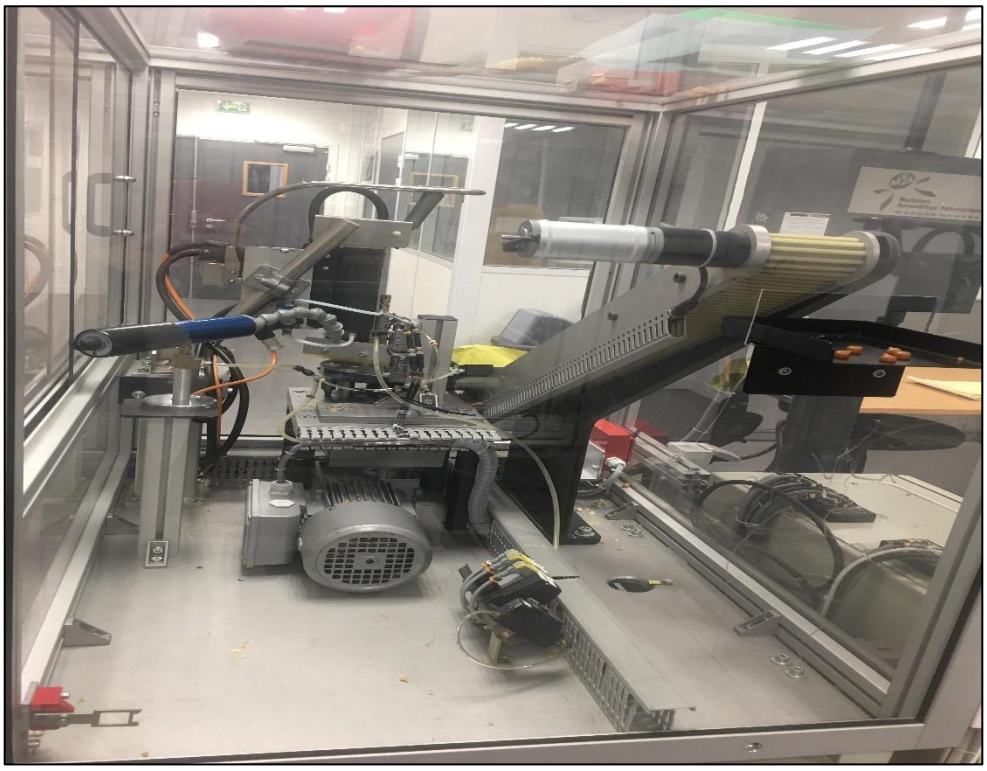


Figure A4.11 : Illustration du poste de fabrication des comprimés

Annexe 5: Les gammes de production

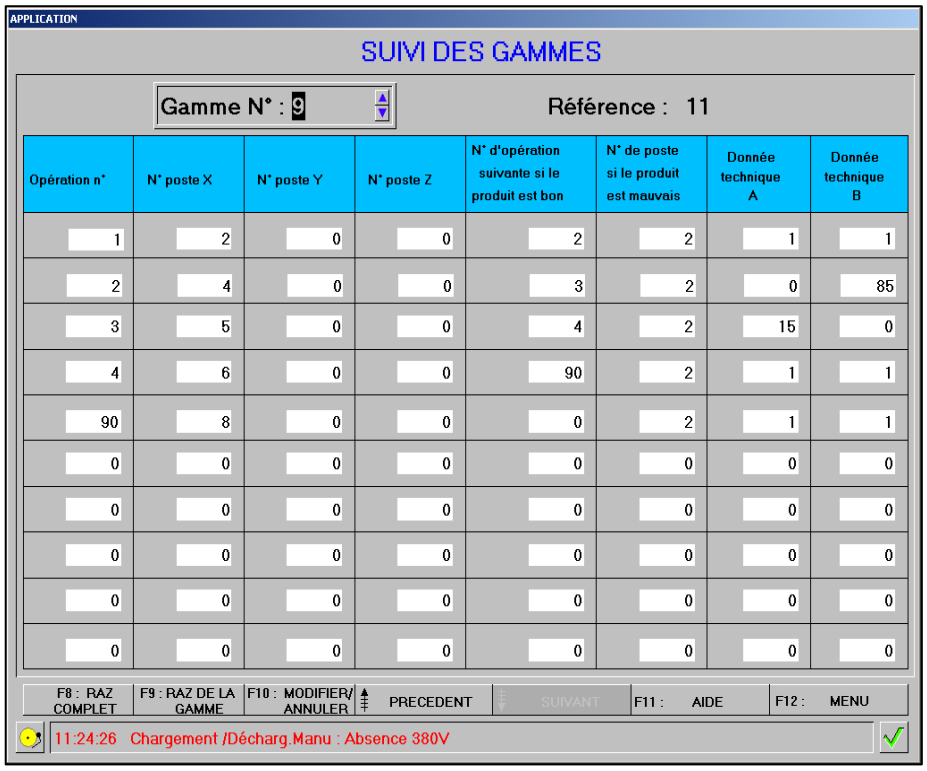


Figure A5.1 : Gamme de production du job 1 de la configuration (5*6)

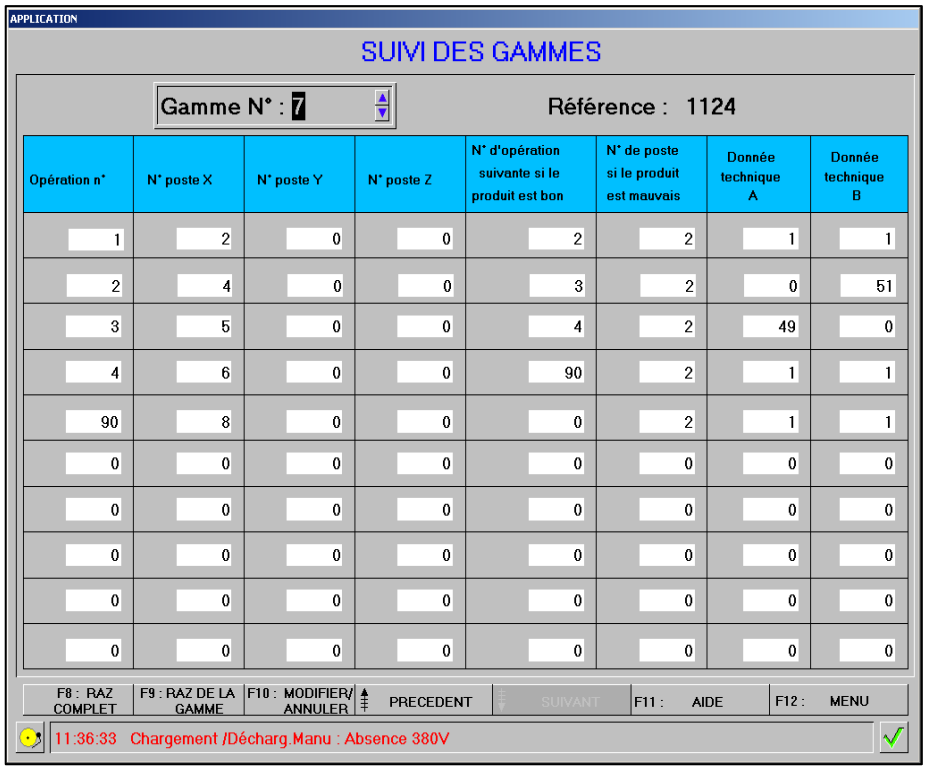


Figure A5.2 : Gamme de production du job 2 de la configuration (5*6)

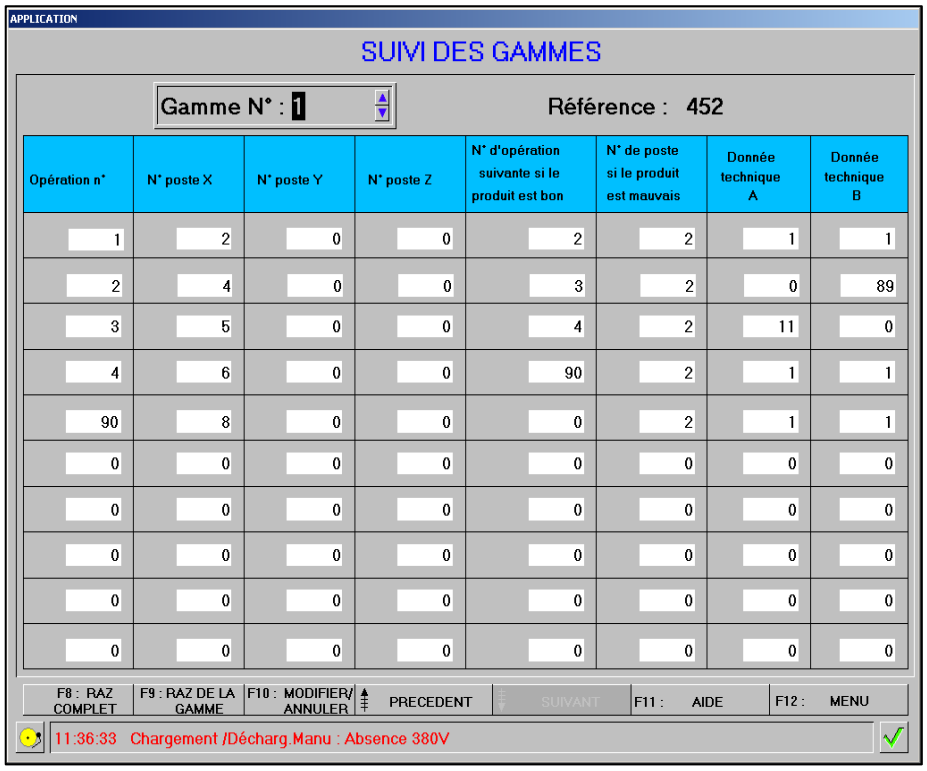


Figure A5.3 : Gamme de production du job 3 de la configuration (5*6)

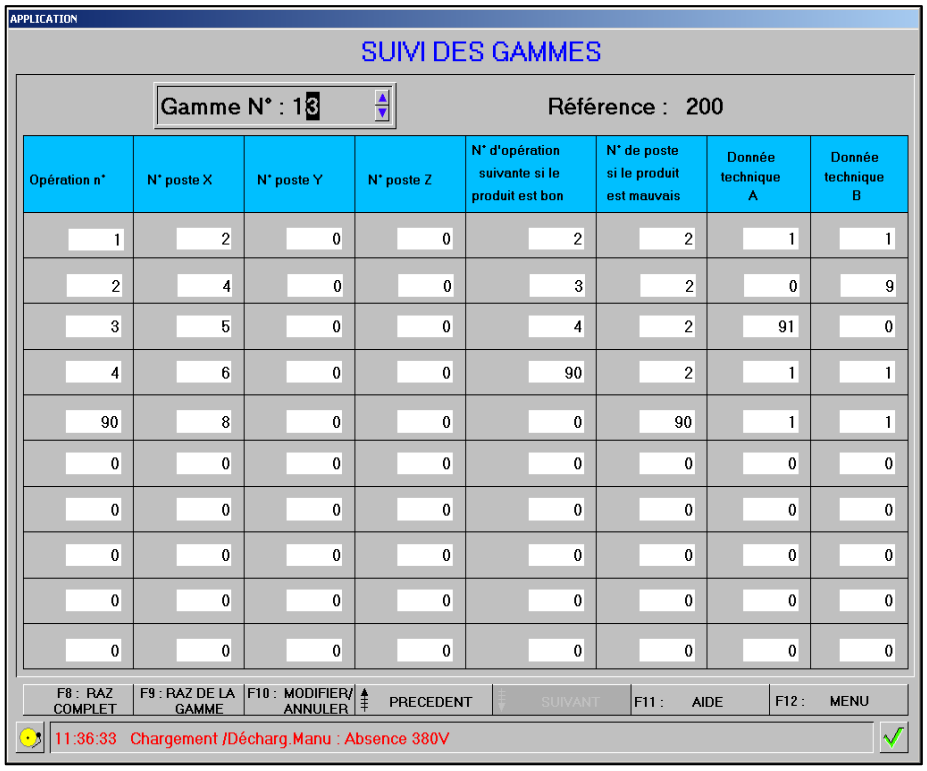


Figure A5.4 : Gamme de production du job 4 de la configuration (5*6)

APPLICATION

SUIVI DES GAMMES

Gamme N° : 5 Référence : 231

Opération n°	N° poste X	N° poste Y	N° poste Z	N° d'opération suivante si le produit est bon	N° de poste si le produit est mauvais	Donnée technique A	Donnée technique B
1	2	0	0	2	2	1	1
2	4	0	0	3	2	0	18
3	5	0	0	4	2	82	1
4	6	0	0	90	2	1	1
90	8	0	0	0	2	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

F8 : RAZ COMPLET | F9 : RAZ DE LA GAMME | F10 : MODIFIER/ ANNULER | PRECEDENT | SUIVANT | F11 : AIDE | F12 : MENU

11:36:33 Chargement /Décharg.Manu : Absence 380V

Figure A5.5 : Gamme de production du job 5 de la configuration (5*6)

APPLICATION

SUIVI DES GAMMES

Gamme N° : 7 Référence : 1124

Opération n°	N° poste X	N° poste Y	N° poste Z	N° d'opération suivante si le produit est bon	N° de poste si le produit est mauvais	Donnée technique A	Donnée technique B
1	2	0	0	2	2	1	1
2	4	0	0	3	2	0	13
3	5	0	0	4	2	87	0
4	6	0	0	90	2	1	1
90	8	0	0	0	2	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

F8 : RAZ COMPLET | F9 : RAZ DE LA GAMME | F10 : MODIFIER/ ANNULER | PRECEDENT | SUIVANT | F11 : AIDE | F12 : MENU

11:36:33 Chargement /Décharg.Manu : Absence 380V

Figure A5.6 : Gamme de production du job 1 de la configuration (7*6)

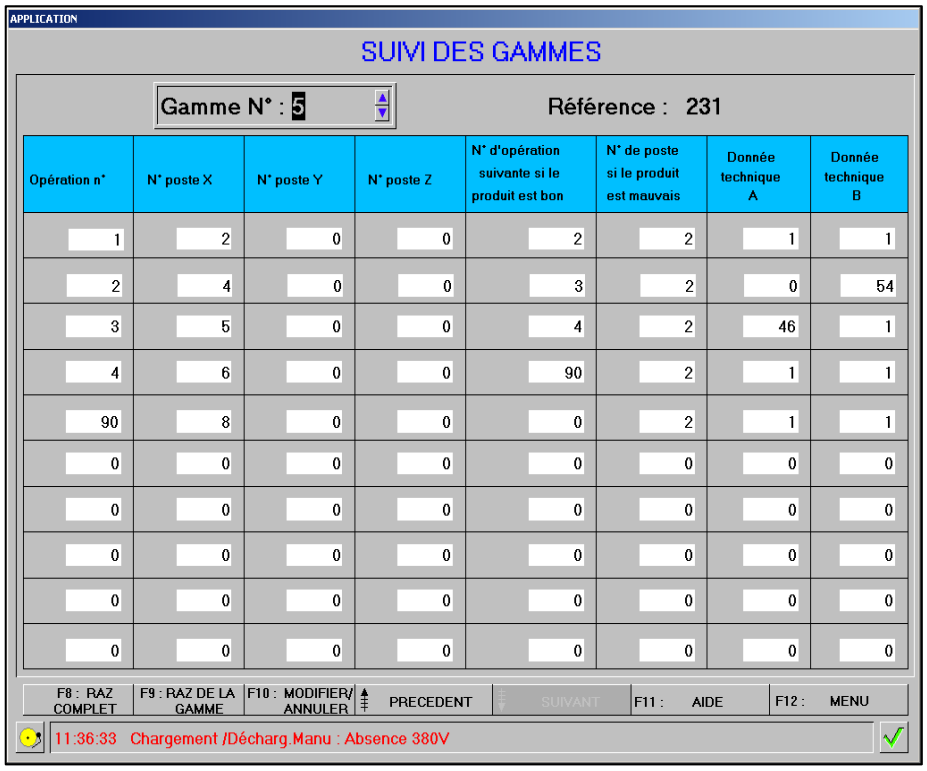


Figure A5.7 : Gamme de production du job 2 de la configuration (7*6)

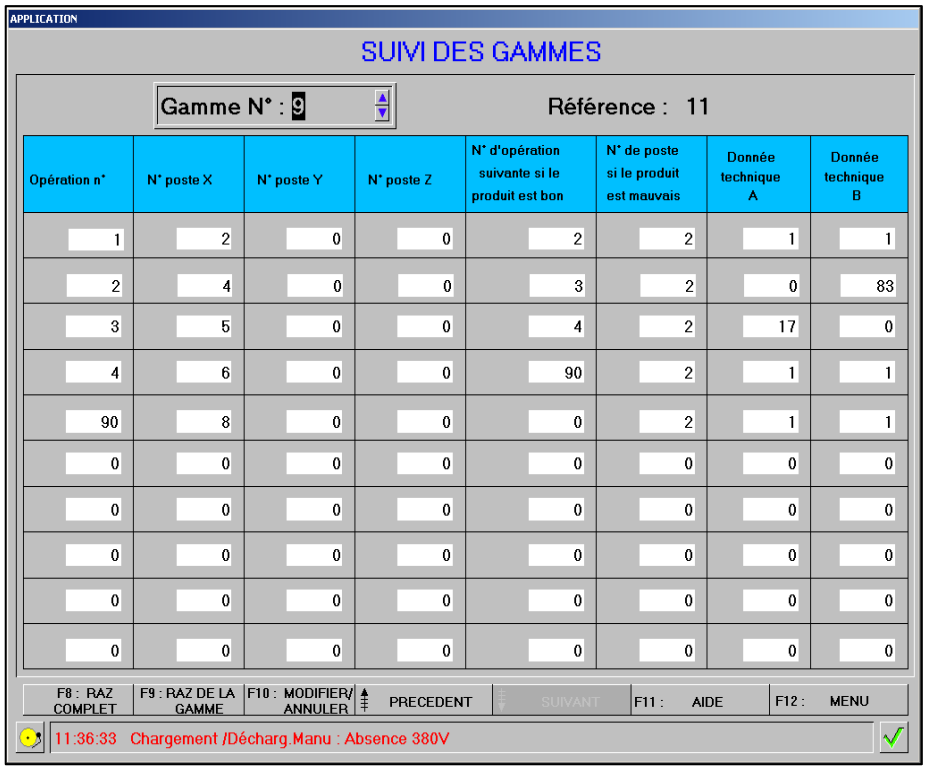


Figure A5.8 : Gamme de production du job 3 de la configuration (7*6)

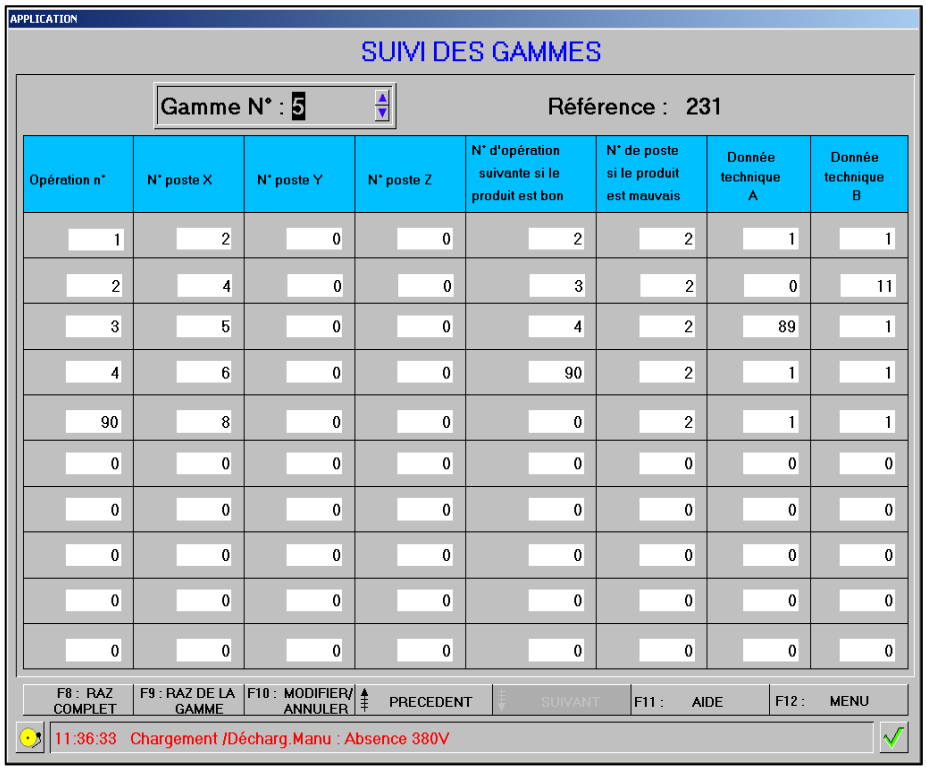


Figure A5.9 : Gamme de production du job 4 de la configuration (7*6)

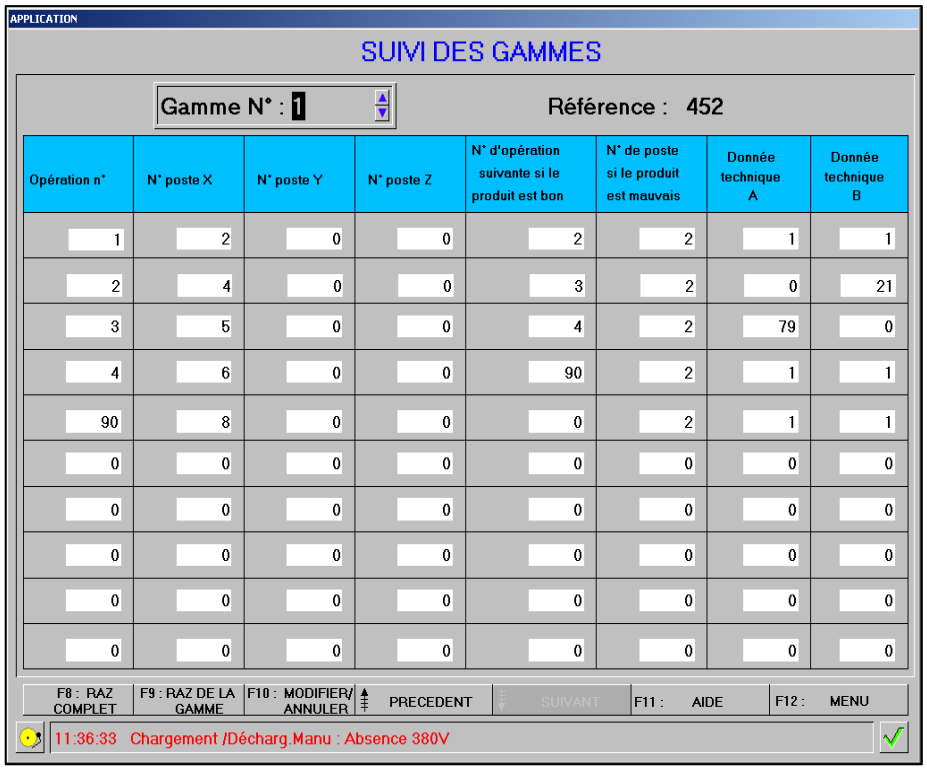


Figure A5.10 : Gamme de production du job 5 de la configuration (7*6)

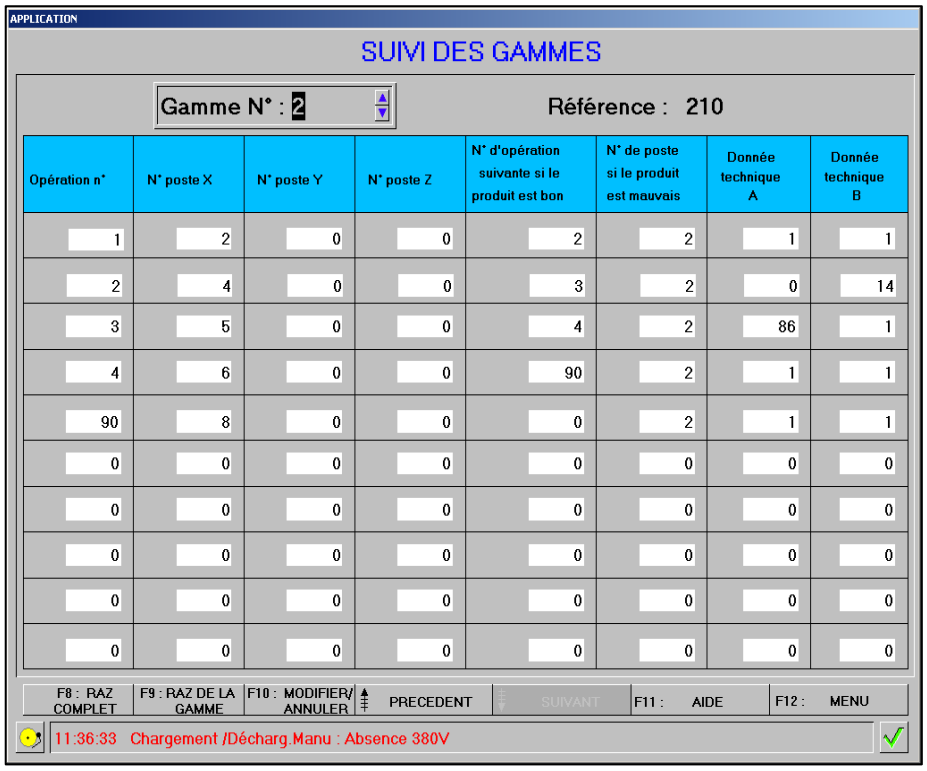


Figure A5.11 : Gamme de production du job 6 de la configuration (7*6)

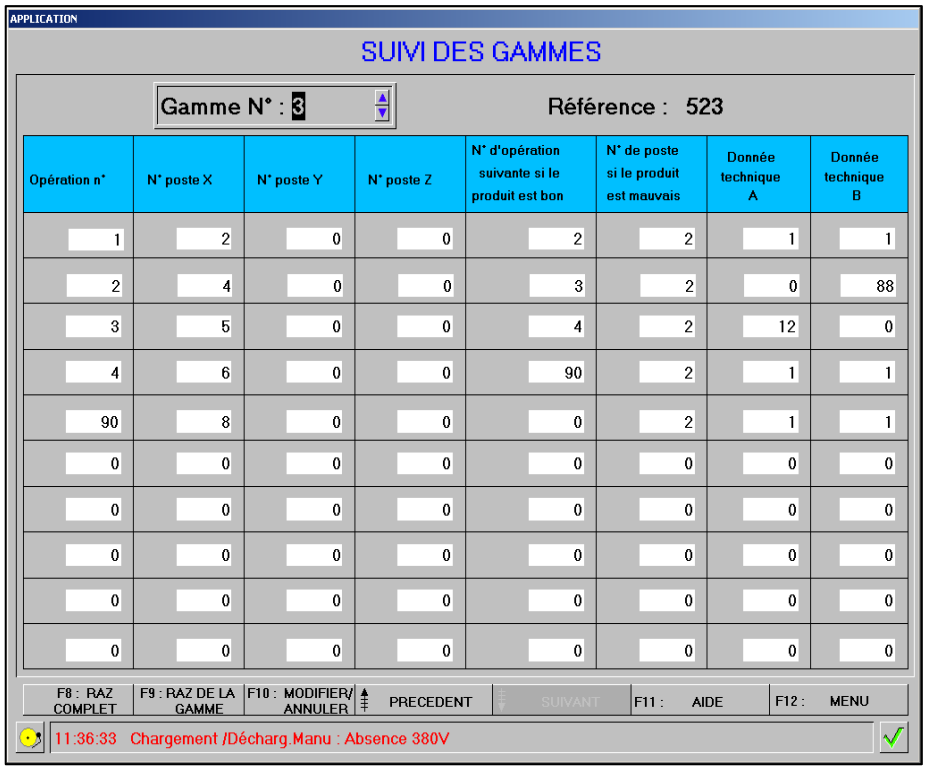


Figure A5.12 : Gamme de production du job 7 de la configuration (7*6)