Ordonnancement d'un système de production industriel complexe : flow shop hybride avec des machines dédiées soumis à différentes contraintes temporelles

Houda Harbaoui

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique et applications section CNU 27*

&

INSTITUT SUPERIEUR D'INFORMATIQUE ET TECHNIQUES DE COMMUNICATION, UNIVERSITE DE SOUSSE

Par

Houda HARBAOUI

"Ordonnancement d'un système de production industriel complexe: flow shop hybride avec des machines dédiées soumis à différentes contraintes temporelles"

Thèse présentée et soutenue à Nantes le 14 décembre 2018
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)
Thèse N° : 2018IMTA0114

Composition du Jury :

Président : Vincent T'KINDT		Professeur, Université de Tours, France
Rapporteurs :	Safia KEDAD-SIDHOUM	Professeur, CNAM Paris, France
	Anis GHARBI	Professeur, TBS Tunis, Tunisie
Examinateurs : Najoua DRIDI		Professeur, ENIT, Tunisie
Directeur de thèse : Odile BELLENGUEZ-MORINEAU		H.D.R, IMT Atlantique, Nantes, France
Co-directeur. de thèse : Mohamed HAOUARI		Professeur, Université Old Dominion, USA

Invité(s)

Soulef KHALFALLAH, Maitre assistant, ISG Sousse, Tunisie

Dédicace

A la mémoire de ma grand-mère

À la mémoire de mon père

À ma chère mère

A mon frère Wahid

A mes sœurs Hela, Hana et Amal

À mes amis

À toute ma famille

Houda

Remerciements

Au terme de cette thèse, je tiens à adresser mes remerciements les plus sincères à mes directeurs de thèse ***Mme. Odile BELLENGUEZ-MORINEAU et Mr Mohamed HAOUARI****, de m'avoir accordé de précieux conseils, et un soutient permanent, sans leur patience et leur aide, ce travail n'aurait pas pu être finalisé. Qu'ils trouvent ici l'expression de ma gratitude et mon respect infini.*

Bien entendu, j'adresse également mes vifs remerciements à Madame ***Soulef KHALFALLAH****, qu'elle n'a cessé de me prodiguer quotidiennement et de son admirable patience envers moi.*

Je remercie également, chaleureusement, les membres du jury : professeur ***Vincent T'KINDT****, professeur* ***Safia KEDAD-SIDHOUM, professeur Anis GHARBI*** *et professeur* ***Najoua DRIDI*** *qui ont eu l'amabilité de lire cette thèse et d'évaluer son contenu et de me faire part de leurs remarques précieuses.*

Je ne peux oublier mes remerciements à l'ensemble du corps professoral et administratif de l'IMT Atlantique et l'l'ISTCOM de Hammam Sousse pour avoir contribué à ma formation.

Mes reconnaissances les plus profondes s'adressent à ma famille et à tous mes amis qui n'ont cessé de m'encourager.

Enfin, je remercie tous ceux qui de prés et de loin ont contribué à l'élaboration de la thèse.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

L'ordonnancement joue un rôle important dans la plupart des industries et des entreprises de services. C'est un élément clé de la gestion de production qui doit contribuer aux résultats économiques et financiers de l'entreprise. Dans un univers concurrentiel, les entreprises industrielles doivent en permanence maximiser la productivité pour augmenter le gain tout en réduisant les pertes de matières et les coûts de maintenance. Pour atteindre ce but, elles doivent entre autres optimiser l'ordonnancement des opérations dans les ateliers de production en tenant compte des contraintes de temps et de ressources. Selon Pinedo [Pinedo, 2008], "*Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives*".
On rencontre différents types de problèmes d'ordonnancement d'atelier selon l'ordre d'enchaînement des opérations et selon le nombre et la disposition des machines. On trouve en particulier les problèmes d'ordonnancement de type flow shop classique et hybride définis par le fait que les tâches suivent une gamme de production unique.
Cette thèse s'intéresse à l'ordonnancement dans un système industriel réel à deux étages, inspiré d'une activité industrielle rencontrée dans une entreprise agroalimentaire partenaire.
L'objectif principal de l'ordonnancement dans cette thèse concerne la minimisation des délais de production appelée makespan(C_{max}). Un second objectif consiste à minimiser les déchets non recyclables dues aux longs arrêts des machines. Nous verrons que ce problème est classé difficile.

Organisation du manuscrit

Cette thèse est organisée en sept chapitres comme suit :

Le chapitre 1 présente l'introduction générale.

Le chapitre 2 présente le problème industriel réel et donne sa classification scientifique. De plus, nous décrivons la génération des données utilisées pour tester nos approches, à savoir des instances issues de la littérature et d'autres de l'entreprise. Deux critères d'optimisation y seront présentés et leur importance pour l'entreprise discutée. Par la suite, nous nous intéresserons tout d'abord au premier critère, qui est la date d'achèvement, noté C_{max}. Celui-ci sera considéré

jusqu'au chapitre 5. Le second critère est la minimisation des pertes matières dûes aux arrêts des machines. Il sera étudié dans le chapitre 7.

Le chapitre 3 est dédié à l'état de l'art relatif aux problèmes de la littérature les plus proches de notre cas, et dont nous détaillerons les contraintes de setup, setup par famille, no-wait et time lag. De plus, nous étudierons le flow shop hybride avec machines dédiées.

Le chapitre 4 décrit deux modèles mathématiques linéaires en nombres entiers" (PLNE) pour modéliser le problème étudié. Les modèles nous permettent d'atteindre une solution optimale pour des instances de petite taille dans un temps admissible du point de vue industriel. De plus, nous proposons trois bornes inférieures qui vont nous servir à évaluer la qualité de nos solutions. Une comparaison des deux modèles est également établie.

Afin de pouvoir traiter des instances de taille plus importante, le chapitre 5 propose des méthodes de résolution approchées basées sur deux algorithmes génétiques (AG) et un algorithme mémétique (AM). Le premier AG est basé sur un codage de séquence de jobs sur l'étage 2 (E2). Le second AG est basé sur un codage de séquence de jobs sur l'étage 1 (E1). Ensuite, nous présentons des versions différentes des AGs : des AGs purs et des AGs hybrides. Nous clôturons le chapitre par une comparaison des algorithmes et une évaluation des résultats.

Le chapitre 6 est consacré à une troisième méthode de résolution, soit une heuristique arborescente. D'abord, nous présentons une description générale de la méthode. Ensuite nous proposons une version dédiée au problème étudié. Nous utilisons des bornes inférieures et supérieures présentés dans les chapitre 4 et 5. Pour valider et évaluer les solutions générées, nous introduisons un outil de résolution exacte soit le MIPstart de CPLEX dont plusieurs réglages sont effectués. Nous clôturons le chapitre par une évaluation de l'algorithme proposé en comparant sa capacité à résoudre à l'optimalité le jeu de données utilisé.

Le chapitre 7 est consacré au second objectif qui vise à minimiser les quantités de matières perdues suite aux arrêts d'une durée de temps fixe, des machines tel que spécifié au chapitre 1. Dans ce chapitre, nous présentons d'abord une étude bibliographique du flow shop classique qui traite la contrainte de "idle-time". Ensuite, nous proposons une extension des modèles présentés dans le chapitre 4 en introduisant la nouvelle contrainte ce qui permet de compter le nombre de idle-time maximal.

Enfin, le dernier chapitre 8 sera un bilan des travaux présentés dans cette thèse afin d'établir une conclusion et présenter des perspectives de recherche.

Chapitre 2

Contexte du problème

Sommaire

Dans ce premier chapitre, nous allons décrire le problème industriel à résoudre dans cette thèse. Dans la section 2.1, nous présentons les caractéristiques, les contraintes et les objectifs du problème réel concerné. Ensuite, dans les sections 2.2 et 2.3, nous étudions respectivement la complexité du problème ainsi que les notations générales à utiliser dans ce manuscrit. Finalement, une description de jeux de données sera présentée dans la section 2.4.

2.1 Description du problème réel

Nous allons tout d'abord décrire le problème rencontré par l'entreprise agroalimentaire partenaire. Notre étude est en effet basée sur un cas réel : dans la région de Sousse, en Tunisie, au sein de l'entreprise : "Pâte Warda".

2.1.1 Définition et caractéristiques du cas d'étude

1. **Définition**

   L'industrie agroalimentaire est définie par l'Union Européenne (UE) comme étant l'ensemble des industries de transformation des matières premières, d'origine végétale ou animale, en produits destinés à l'alimentation humaine ou animale. Il y a plusieurs familles d'industries appartenant au secteur agro-alimentaire :

   — les industries alimentaires (pâtes alimentaires, sucre, confiserie, chocolaterie, pain, pâtisserie, café, thé, aliments adaptés à l'enfant et diététiques, condiments)

   — l'industrie laitière

   — l'industrie du poisson

   — l'industrie des viandes

   — l'industrie des fruits et légumes

   — l'industrie des boissons

   — la fabrication d'aliments pour animaux ;

2. **Caractéristiques**

   Étant donné que l'industrie agro-alimentaire est destinée à la production de notre alimentation, elle doit être très attentive à l'hygiène. Par ailleurs, un contrôle doit toujours accompagner la production pour assurer la sécurité sanitaire des produits et les équipements de travail en nettoyant les locaux et éliminant les salissures apportées par les machines employées et les déchets des produits traités dans le procédé industriel. D'autre part, les risques alimentaires dans l'industrie agroalimentaire induisent également un cycle de vie court des matières premières car ce sont des denrées périssables. Ces différentes considérations vont engendrer des contraintes temporelles sur la production, que l'on retrouvera dans la définition du problème à venir.

2.1.2 Présentation de l'entreprise

Cette société a été crée en 1995. Elle est spécialisée dans la fabrication des pâtes et du couscous. Sa capacité de production est de 205 T/J pour la pâte et de 35 T/J pour le couscous. Ses produits sont exportés dans plus de 20 pays et le chiffre d'affaires à l'export représente 40% de son chiffre d'affaires total. L'entreprise concernée utilise un ERP qui n'intègre pas un module d'ordonnancement. Actuellement, l'ordonnancement se fait manuellement.

2.1.3 Description de la gamme de production

Avant de décrire les lignes de production, il convient de signaler que la gamme de production des pâtes est constituée de deux étapes principales : la fabrication des produits et leur séchage.

— La première phase de production consiste à fabriquer différents produits suivant un document technique élaboré (recette) par l'entreprise. Cette phase est réalisée par deux machines

(présentées dans la figure 2.1), selon le type de produit. A la sortie de cette étape de production, les produits sont mis sur des plateaux qui sont disposés sur des chariots de transport de taille fixe (32 plateaux). Nous assumons ici que les quantités fabriquées correspondent toujours à un multiple des chariots pleins. Il convient de signaler qu'une quantité fabriquée correspond exactement à 8 chariots.

— Après la phase de fabrication, les produits doivent passer par la phase de séchage : 10 cabines de séchage identiques sont disponibles pour cette opération et constituent le deuxième étage de production. En effet, à chaque fois qu'un chariot est rempli et organisé, un ouvrier le place dans la cabine de séchage déjà affectée et attendant le reste des chariots. Une opération de séchage ne peut être lancée que lorsque la cabine est pleine (8 chariots : $8 \times 32 = 256$ plateaux). Dès lors la cuisson peut commencer. Le réglage de chaque cabine se fait en amont avec le chargement et le transport des chariots. Une fois la cabine chargée, l'opération de séchage est tout de suite lancée. L'entreprise fait donc le choix de ne pas prendre en considération le temps de réglage des cabines lors de la construction de l'ordonnancement vu qu'il se fait en même temps que la préparation des cabines.

  Il convient de signaler que le transfert se fait en même temps que le remplissage des chariots. L'entreprise fait donc le choix de ne pas le prendre en considération lors de la construction de l'ordonnancement.

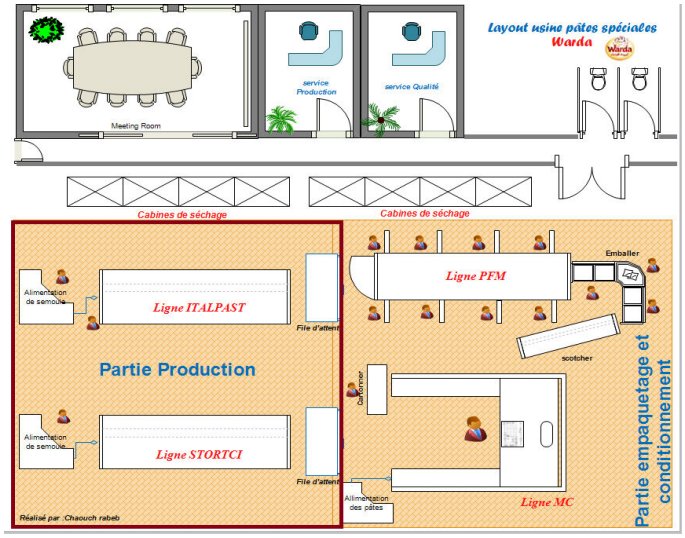


FIGURE 2.1 – Layout de l'atelier de production [CHAOUCH, 2015]

Dans la figure 2.1, la partie de production encadrée en rouge est la partie étudiée dans cette thèse. Les deux lignes PFM et MC présentent deux lignes d'emballage.

2.1.4 Description des lignes de production de l'entreprise

La phase de fabrication que nous venons de décrire est réalisée par deux machines de production indépendantes. Les machines de production de ce *premier étage* (E1) sont :

— La machine « $ITALPAST$ » (figure 2.2) : est une machine qui fabrique une 1ère famille de produits (F1) soit les pâtes courtes. Sa capacité est de 170Kg/heure pour l'article Nwaser et 204Kg/heure pour l'article Lasagne.

FIGURE 2.2 – Machine $ITALPAST$

— La machine « $STORCI$ » (figure 2.3) est une machine qui fabrique une 2ème famille de produits (F2) soit les pâtes spéciales (Cannelloni, Jumbo avec ses 3 types, Tagliatelle 1, Tagliatelle 2). Sa capacité est de 190 Kg/heure pour l'article cannelloni, 170 Kg/heure pour l'article Jumbo et 200 Kg/heure pour l'article Tagliatelle

FIGURE 2.3 – Machine $STORCI$

2.1.5 Description du séchage

Le deuxième étage de production (E2) correspond aux cabines de séchage qui sont toutes identiques.

— Les cabines de séchage ont chacune une capacité unique de 8 chariots contenant chacun 32 plateaux.

— Chaque machine du premier étage produit exactement le contenu d'une cabine, soit 8 chariots de 32 plateaux.

Pour des raisons techniques, une cabine de séchage a les propriétés suivantes :

— Elle ne peut pas contenir plus qu'un produit ;

— Elle doit nécessairement être pleine

Pour cela, une machine de production au niveau de l'étage 1, munie du moule approprié, est alimentée par exactement une quantité de semoule (matière première) qui permet de remplir une cabine de séchage. Pour ces raisons, un « job » qui correspond à un produit donné, représente le contenu d'une cabine de séchage. Dans la littérature [Mathirajan and Sivakumar, 2006], plusieurs problèmes de « fournées » sont des ordonnancements sur des machines à traitement par lots « batch ». L'objectif principal de l'ordonnancement par batch est de regrouper un certain nombre de jobs, ayant des caractéristiques non similaires (durée d'exécution, température de cuisson...), dans une même fournée. Cependant, en pratique il est, par exemple, impossible de cuire deux produits à des températures différentes dans un même four. En effet, le but de traitement par lot est d'optimiser les setups ou minimiser la consommation d'énergie. Ceci ne s'applique pas dans le cas de notre problème car les jobs ayant différentes caractéristiques ne peuvent pas être regroupés au niveau d'une même cabine de séchage.
D'autre part, les hypothèses du problème industriel impliquent que les travaux d'un même groupe ont la même durée de traitement.

2.1.6 Description des contraintes

Pour ordonnancer la production décrite auparavant, il y a plusieurs contraintes à respecter, à savoir :

1. **Contraintes de production**
   — La production sur les machines «$STORCI$» et «$ITALPAST$» nécessite différents moules selon le type de produit à fabriquer. Les produits sont regroupés en 2 familles (comme indiqué dans la section 2.1.4), chacune contenant plusieurs groupes d'articles. Ainsi, lorsqu'on passe d'un groupe de produits à un autre, il est nécessaire de changer de moules. Ce temps de changement est un temps pendant lequel la machine ne produit rien. Le montage des nouveaux moules est nécessaire lorsqu'on change de groupe de produits. Il sera désigné par la suite en tant que setup, et constitue une contrainte de notre problème. Ce temps peut être compris entre 05 et 20 minutes.
   — Les produits de la ligne 1 (famille 1 (F1)) qui passent sur M1 peuvent attendre un délai maximum de 30 minutes avant de passer en cabine de séchage. Ainsi, la production peut être lancée même s'il n'y a pas de cabine disponible (c'est-à-dire elle peut être déclenchée 30 minutes avant au maximum). Dans ce cas, l'opération est soumise à la contrainte de time lag maximal.

— Les produits de la ligne 2 (famille 2 (F2)) qui passent sur M2 ne peuvent pas attendre à la fin de leur production pour partir au séchage. Donc on peut dire que la production est soumise à la contrainte de no-wait dans ce cas.
En résumé, la production des pâtes dans cette étude est soumise à la contrainte de *time lag maximal*. Celui-ci est de 30 minutes sur M1 et de 0 minutes sur M2.

2. **Contraintes de séchage**
   — Le temps de nettoyage final de la cabine, qui dure entre 02 et 05 mn, est intégré dans le temps de séchage car il est réalisé de façon systématique
   — Lorsque le séchage des produits est terminé, un employé du service qualité prend des échantillons de l'article séché pour une analyse de qualité. Un test de qualité sera effectué. Le test de qualité pour tous les types de produit dure environ 15 mn, en conséquence les cabines de séchages, ayant fini leur opération, restent encore bloquées par les jobs pendant cette durée. Dans ce travail, nous considérons cette durée incluse dans la durée de séchage. Pour bien illustrer, nous présentons l'exemple suivant :
   Soit un job i a une durée de séchage réel de 135 minutes alors la durée de séchage planifiée sera de 150 minutes. Ceci est dû au fait que la cabine reste réservée au traitement de ce job pour une durée supplémentaire de 15 minutes. En discutant avec le chef de production, les durées de séchage fournies à notre étude sont des durées fixes. En ce qui concerne le cas où le résultat de test ne respecte pas les normes de qualité engendrant un temps de séchage additionnel, cet aléa rare, n'est pas considéré dans la construction de l'ordonnancement. Du point de vu pratique et sur recommandation du chef de production, nous avons donc considéré cette donnée comme déterministe.

2.1.7 Description des objectifs

Dans le cadre de cette étude, l'entreprise souhaite explorer différentes finalités afin d'être en capacité de répondre à différentes exigences d'ordonnancement et établir un comparatif. La plupart du temps, l'entreprise fabrique pour le stock. C'est-à-dire elle œuvre pour garder un certain niveau de produits pour lesquels il existe une demande plus ou moins stable. Nous considérons deux cas possibles :

1. **Premier objectif principal : minimiser le temps total de travail**

   L'entreprise cherche en priorité à diminuer le délai de production, c'est pourquoi elle doit bien gérer les facteurs entrant dans la production, essentiellement les matières premières et les machines. L'entreprise cherche à répondre à cet objectif car elle fabrique sur stock d'où elle doit :
   — gérer son stock et combler toute rupture.
   — utiliser rationnellement ses ressources. L'entreprise doit gérer les coûts variables d'utilisation de ses matériaux (par exemple les coûts d'exploitation des machines).

2. **Deuxième objectif : minimiser les pertes en minimisant le nombre d'arrêts des machines**

   L'entreprise souhaite également tester un autre critère car les coûts de production

sont liés également aux pertes de matières travaillées. Pour minimiser les coûts de production, l'entreprise vise à minimiser les déchets (des produits résiduels inutilisables). Étant donné que les pâtes sont des produits périssables, il existe un phénomène, qu'on appelle perte de matière, et qui se présente dans les cas suivants :

— lors de changement de produit :
A chaque changement du groupe de produit, il y aura un changement de moule sans arrêter les machines. En effet, les machines continuent à tourner afin d'évacuer les déchets. Dans ce cas, il y a perte des matières encore valables appelées des déchets réutilisables puisque ils constituent une matière première pour une nouvelle production (le recyclage). Ces pertes de matières ne sont pas considérées dans notre critère puisqu'elles sont réutilisables.

— lors d'un arrêt des machines :
Lorsque la machine s'arrête pour une durée de temps supérieure à une borne connue, le résidu des pâtes devient nocif. Il n'est plus possible de le réutiliser : c'est ce qu'on appelle des déchets perdus. Ce résidu représente une quantité non négligeable qui dépend de la machine. La durée d'arrêt pour laquelle il y aura des pertes non recyclables est de 30 minutes pour les 2 machines (figure 2.4).

Il convient de signaler que la quantité de déchet non recyclable est de 50 kg pour les 2 machines.

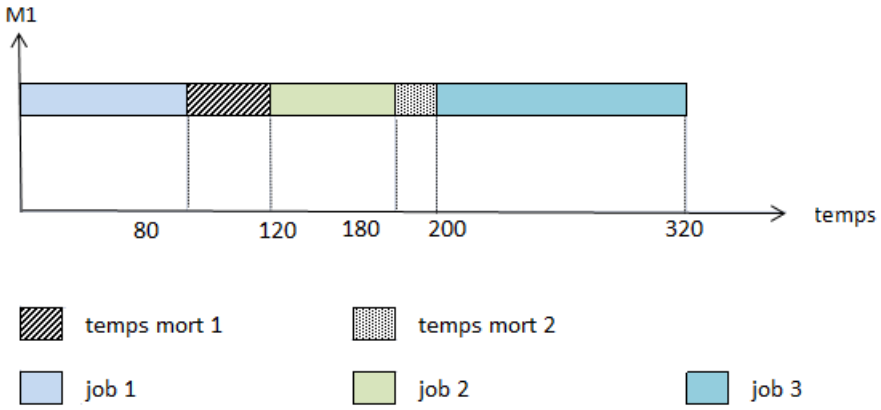


FIGURE 2.4 – Arrêt de longue durée sur les machines dédiées

Dans la figure 2.4, le temps mort 1 présente un "arrêt de machine" puisque ce temps mort est supérieur à 30 minutes. Le temps mort 2 est un changement de job sans arrêt de machine. L'ordonnancement de la figure 2.4 engendre une perte de matière de 50 kg. Ces quantités sont fixes et non récupérables.
Il sera donc intéressant de minimiser le nombre d'arrêts de machines dépassant les bornes fixées pour minimiser les pertes de matières. Comme il n'existe pas de date de début au plus tôt sur les jobs, la seule raison provoquant ces arrêts longs est la non-disponibilité des cabines. En effet, le time-lag maximum oblige à décaler les opérations du premier étage, ce qui peut entrainer par la suite des arrêts longs.

Dans ce qui suit, nous allons essayer de traduire l'information collectée auprès de l'entreprise afin de construire une description scientifique du problème.

2.1.8 Définition des problèmes d'ordonnancement

D'après nos observations, l'atelier de production étudié est composé de 2 étages, tout d'abord les machines de production puis les séchoirs. Ainsi les produits suivent un cheminement unidirectionnel d'un étage à un autre. Dans la littérature scientifique publiée sur ce sujet, le problème industriel étudié peut être vu sous 2 angles différents. Dans tous les cas, la gamme unidirectionnelle, nous permet d'identifier un flow shop (FS). En se basant sur l'article de [Ruiz and Rodriguez, 2010] nous pouvons identifier le problème comme :

— Un flow shop hybride (FH) à deux étages avec deux machines parallèles dédiées à l'étage 1 et plusieurs machines parallèles identiques à l'étage 2.

En se basant sur l'article de [Ribas et al., 2010], nous pouvons identifier le problème comme :

— Un FH à trois étages avec une machine à l'étage 1, une machine à l'étage 2, et plusieurs machines identiques à l'étage 3. Les jobs de la famille 1 vont sauter l'étage 2 (un job pourrait ignorer un certain nombre d'étages) et les jobs de la famille 2 vont sauter l'étage 1.

Dans la suite de la thèse, nous allons garder la première classification qui est plus utilisée dans la littérature (figure 2.5), et qui permet de compacter davantage la formulation de ce problème.

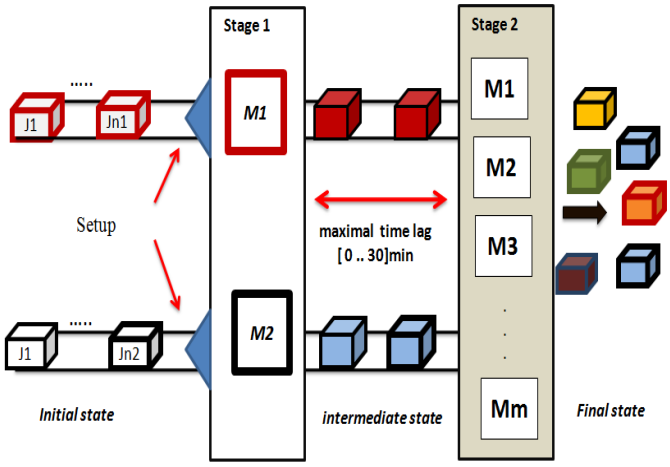


FIGURE 2.5 – Layout du problème industriel

2.1.9 Classification scientifique du problème

Ainsi, notre problème est un flow shop hybride (FH) à 2 étages. En effet, le problème posé dans le cas étudié est une généralisation de 2 problèmes :

— le problème de flow shop puisque les jobs suivent un ordre d'exécution unidirectionnel sur 2 étages

— le problème d'ordonnancement à machines parallèles au second étage puisque les machines sont des cabines de séchages identiques parallèles. Le problème étudié est de type flow shop hybride à 2 étages (FH) avec machines parallèles dédiées à l'étage 1 et machines parallèles au 2ème étage.

Selon le critère pris en compte, nous considérons deux problèmes distincts. Le premier s'intéresse à l'objectif C_{max}. Selon la notation $\alpha|\beta|\gamma$ proposée par [Graham et al., 1979], notre problème s'écrit comme suit :

— FH2 (PD2,Pm) $|ST_{g,sd}, l_i|C_{max}$

Le deuxième s'intéresse à l'objectif de minimisation du nombre d'arrêts des machines. Puisque le critère minimisation du nombre d'arrêts n'a pas été considéré avant dans la littérature à notre connaissance, nous proposons la notation suivante pour le critère d'évaluation :

— FH2 (PD2,Pm) $|ST_{g,sd}, l_i|N_{idle-time-max}$

$ST_{g,sd}$: désigne la contrainte de "sequence dependent group setup".
l_i : le time-lag maximal définit la durée d'attente maximale entre les 2 étages.
C_{max} : définit le temps total de traitement de tous les jobs.
$N_{idle-time-max}$: désigne le nombre des arrêts maximal.

2.2 Complexité

En se basant sur la revue de la littérature, nous présentons la complexité des problèmes proches de notre étude :

1. [Hoogeveen et al., 1996] ont montré que le problème du flow shop hybride à deux étages avec une seule machine sur l'étage 1 et deux machines parallèles identiques sur l'étage 2 (m = 2) et lorsque les machines sont toujours disponibles est connu comme NP-difficile au sens fort ;
2. [Kurz and Askin, 2004] ont montré que le problème de flow shop hybride avec la contrainte de setup est NP-difficile au sens fort ;
3. [Su, 2003] ont montré que le problème de flow shop hybride à 2 étages soumis à la contrainte de délai d'attente maximal est NP-difficile au sens fort ;
4. [Oguz et al., 1997] ont montré que le problème de flow shop hybride avec des machines dédiées sur le premier étage est NP-difficile au sens fort.

En se référant au travail de [Su, 2003], nous constatons qu'une instance de ce problème est incluse dans les instances du problème traité dans cette thèse. En effet, si tous les jobs traités appartiennent au même groupe (un cas particulier de notre problème), nous nous trouvons avec un problème de flow shop hybride à deux étage soumis à la contrainte de setup. Ce problème est NP-difficile au sens fort.

2.3 Notations générales

$E = \{ E1,E2 \}$: ensemble des étages
M_k : ensemble des machines par étage où $M_k = \{1..m_k \}$
F_1 : famille 1 de jobs
F_2 : famille 2 de jobs
$J = \{ J_1,J_2...J_n \}$: ensemble des travaux ou jobs à réaliser
$J1 = \{ J_1,J_2...J_{n1} \}$: ensemble des travaux ou jobs à réaliser $\in$ F1 passant sur la première machine dédiée
$J2 = \{ J_1,J_2...J_{n2} \}$: ensemble des travaux ou jobs à réaliser $\in$ F2 passant sur la deuxième machine dédiée
$S_{u,m}$: setup de groupe u sur la machine m
m_k : nombre de machine à l'étage k
n : nombre de jobs
$p_{i,k}$: durée d'exécution du job i sur l'étage k
$S_{i,k}$: durée de setup du job J_i à l'étage k
l_i : la durée de time lag maximal du job J_i entre l'étage 1 à l'étage 2
A_i : numéro de machine affectée au job J_i à l'étage 1
u_1 : nombre de groupes de jobs $\in J1$
u_2 : nombre de groupes de jobs $\in J2$
b_i : groupe auquel appartient le job J_i
B_{ij} : conformité entre le groupe (sous famille) du job i et le groupe du job j : $B_{ij} = 1$ si i et j $\in$ même groupe, 0 sinon
$Y_i^{m,k}$: matrice binaire de pré-affectation de job i à la machine m de l'étage k :

- Si k=1 (cas des machines dédiées) $\mathbf{Y}_i^{m,1} = 1$ si le job i est affecté à la machine m de l'étage 1, 0 sinon.
- Si k=2, $Y_i^{m,2} = 1$, $\forall$ m= $\{1..m_2\}$ ceci signifie qu'il n'y a pas une machine parallèle m à laquelle il ne pourrait pas être affecté un job i.

CT^{M1} : la date de fin de traitement des jobs ordonnancés sur la machine M1 de l'étage 1
CT^{M2} : la date de fin de traitement des jobs ordonnancés sur la machine M2 de l'étage 1
$CT_{i,k}$: la date de fin de traitement du job i sur l'étage k
AVi : la date de disponibilité de la machine parallèle i de l'étage 2

2.4 Génération de jeux de données

2.4.1 Données de test

Pour valider les approches proposées dans ce travail, nous présentons dans cette partie une description des instances de tests qui ont été utilisées pour notre travail. Nous avons travaillé avec deux jeux d'instances différents. Le premier jeu est directement construit en partant des données réelles de l'entreprise. Le second jeu correspond à des instances inspirées de la littérature.
En ce qui concerne le premier jeu de données, on a visité l'entreprise et on a récolté leurs données. Malheureusement, il n'y a pas de jeu de données directement exploitable. D'autre part, la

préparation des ordres de fabrication et des ordonnancements est basée sur l'expérience du responsable de production. Nous avons alors examiné toutes les données liées aux produits, aux groupes de produits, les durées de fabrication sur les machines de deux étages, les durées de changement de moules et les durées d'attentes autorisés entre les deux étages. A partir de ces observations, nous avons généré plusieurs instances de données réelles en fonction des paramètres décrits auparavant.

1. **Catégorie 1 : instances réelles**
   Dans ce jeu d'instances (voir le tableau 2.1), les données sont générées en respectant les données industrielles réelles. En effet, dans le cas réel, les durées de traitement sur l'étage 2 sont 5 fois supérieures aux durées de traitement sur l'étage 1. Il y a 2 machines dédiées sur l'étage 1 et 10 machines identiques parallèles sur l'étage 2. Les données de cette catégorie sont entre autres basées sur les caractéristiques des jobs, comme présenté auparavant, à savoir :
   - une famille : c'est le type de produit. Chaque type est fabriqué uniquement sur une seule machine. En d'autres termes, le nombre de familles correspond au nombre de machines dédiées à l'étage 1 ;
   - sous-famille ou groupe dont chaque groupe a son propre moule. Le terme "groupe" sera adopté dans le reste de la thèse.
   - Une durée de setup entre 05 minutes et 20 minutes et les jobs du même groupe ont la même durée de setup ;
   - une première opération sur une machine dédiée de l'étage 1 d'une durée qui varie entre 55 minutes et 150 minutes ;
   - une deuxième opération sur n'importe quelle machine de l'étage 2 dont la durée varie entre 180 minutes et une valeur maximale 780 minutes ;
   - Un time lag maximal : qui respecte le cas réel c'est à dire une valeur nulle sur une première machine dédiée et une valeur strictement positive sur l'autre. En effet, nous avons choisi de faire varier la valeur de time lag afin de varier les instances de test. Par ailleurs, il faut signaler que les jobs appartenant à la même famille ont la même durée d'attente, c'est à dire les jobs passant sur la même machine dédiée ont le même time lag.

TABLE 2.1 – Caractéristiques de la 1ère catégorie de données

Paramètre	**Valeur**			
Nombre de job	10	20	50	100
Nombre de machines/E1	2			
Nombre de machines/E2	10			
Nombre de groupe/machine	7 groupes			
Durée de traitement/E1 (minutes)	[55,150]			
Durée de traitement/E2 (minutes)	[180,780]			
Durée de setup (minutes)	[5,20]			
Durée de time lag maximal (minutes) des jobs passant sur M1	[1, 30]			
Durée de time lag maximal (minutes) des jobs passant sur M2	0			

Il faut noter que les durées opératoires des jobs appartenant au même groupe sont les mêmes que ce soit sur l'étage 1 ou bien sur l'étage 2.
Il convient toutefois de signaler que 24 heures avant le lancement de la production, un test est effectué sur la matière première pour ajuster les durées de traitement au maximum de ±5%. Dans le cadre de cette thèse, nous avons décidé de générer des durées opératoires différentes pour les travaux d'un même groupe. Nous retenons le fait que les travaux d'un même groupe nécessite le même moule au niveau de la phase de fabrication et qu'un job correspond au contenu d'une cabine de séchage.

2. **Catégorie 2 : instances de la littérature**

Les instances réelles sont des instances limitées et elles reflètent un cas particulier de problème. Dans le but d'illustrer les performances de nos approches, nous aurons besoin de les tester via différentes instances générales de la littérature. Cependant, il était difficile de trouver des instances qui correspondent à notre problème et la plupart des auteurs de la littérature ont utilisé leurs propres instances. Donc, nous avons généré une deuxième catégorie de données selon des caractéristiques groupées de la littérature.
En se basant sur le travail de [Yang, 2011, Yang, 2015a], les données sont générées selon différent facteurs : n, n_1, n_2 =n -n_1, $p_{j,k}$ où k $\in \{1,2\}$, j $\in \{1..n\}$, avec n, n_1, n_2, $p_{i,k}$ présentent respectivement le nombre total de jobs, le nombre de jobs passant sur la première machine dédiée, le nombre de job passant sur la deuxième machine dédiée, la durée de traitement sur l'étage 1 ainsi que sur l'étage 2.
Le problème étudié dans les travaux de [Wang and Liu, 2013b, Yang, 2015a] est proche de notre problème : le flow shop hybride avec machines dédiées traitant la contrainte no-wait. Les auteurs ont considéré 3 combinaisons de $n_i, i \in 1, 2$ sachant que n= n_1 + n_2 : Selon [Yang, 2011, Wang and Liu, 2013b, Yang, 2015a],

— n_1= 0.5 n

— n_1= 0.6 n

— n_1= 0.7 n

Généralement, les machines parallèles identiques sont utilisées dans les industries pour traiter les opérations de longue durée. Ceci est bien conforme avec notre étude, car on constate que : $\frac{|p_{i,2}|}{|p_{i,1}|} \equiv \frac{|M2|}{|M1|}$

Il faut signaler que nous avons respecté cette formule dans le choix de chaque intervalle de temps de traitement des jobs.

Cependant, le travail de [Yang, 2015b] ne respecte pas cette règle, donc nous avons décidé de créer nos propres critères de génération de données. D'autre part, nous suivons [Rios-Mercado and Bard, 1998] pour générer le setup et le time lag maximal. Les auteurs [Rios-Mercado and Bard, 1998] ont travaillé sur la contrainte de setup.

2.4.2 Génération de jeux d'instances

Nous proposons deux catégories de données. La première catégorie est basée sur une classe unique des données alors que les instances de la deuxième catégorie sont en plus regroupées en trois classes selon les durées de traitement sur E1 et sur E2.

Pour la première catégorie, on a généré 30 instances pour chaque valeur de n (10, 20, 50, 100) tout en considérant les autres facteurs (setup et time lag). En croisant les différents facteurs 30×4, on aura 120 problèmes. Pour cette première catégorie, le setup, le time lag maximal, ainsi que (n,n1) sont générées aléatoirement.

En ce qui concerne la deuxième catégorie, pour chaque combinaison de n et n_1 (n= 10, 20, 50, 100, 200 ; n_1= 0.5n, 0.6n, 0.7n) 30 instances sont générées. D'autre part, on a 3 distributions ou classes de processing time : ($p_{i,1}$: [1,40] $p_{i,2}$: [5,200]), ($p_{i,1}$: [40,80] $p_{i,2}$: [200,400]) ($p_{i,1}$ (80,100) $p_{i,2}$(400,600), ce qui augmente le nombre des instances générées de cette catégorie à 1350 ($5 \times 3 \times 3 \times 30$). Finalement, le nombre total des instances est égal à 1470 instances. Le tableau ci-dessous résume la génération de la catégorie 2 :

TABLE 2.2 – Caractéristiques de la 2ème catégorie de données

Paramètre	**Valeur**				
Nombre de job	10	20	50	100	200
Nombre de groupe	[2, 4]	[3, 7]		[5, 10]	
Nombre de machines/E1	2				
Nombre de machines/E2	10				
Durée de traitement/E1 (minutes)	[1, 40] - [40, 80] - [80, 100]				
Durée de traitement/E2 (minutes)	[5, 200] - [200, 400] - [400, 600]				
Durée de setup (minutes)	[5, 20]				
Durée de time lag maximal (minutes)	[0, 30]				

2.4.3 Étude de la difficulté des instances de données

Les deux catégories de données se caractérisent par un degré de difficulté différent. La spécificité des instances dans la première catégorie de données tient à sa résolution la plus simple car la contrainte no-wait est plus présente. Le problème étudié constitue une configuration plus facile FH2(PD2,Pm)|no-wait|C_{max} et qu'une solution optimale est vite trouvée sur la base de ECT-FAM.
Dans la deuxième catégorie de donnée, il existe 3 classes différentes avec 3 distributions des charges de travail :
- Dans la classe 1, il y a un chevauchement entre les intervalles des durées de traitement, fait qu'en général, que les machines de l'E1 sont plus chargées.
- Dans la classe 2, les intervalles des durées de traitement ne sont pas chevauchés et l'écart entre les durées de traitement est faible. Ceci permet d'avoir plus de machines disponibles sur les deux étages ce qui rend cette classe plus facile à résoudre.
- Dans la classe 3, les intervalles des durées de traitement ne sont pas chevauchés. De plus, il y a un écart très important entre les durées de traitement sur les deux étages. Cette spécificité va engendrer des retards de lancement des jobs sur l'E1 et renforcer l'E2 à être un étage goulot.
D'autre part, dans les 3 classes, les instances ayant une balance des charges entre le machines dédiées sont les plus difficiles à résoudre. Il est à signaler aussi que, plus que le nombre de jobs affectés à une machine dédiée est élevé (n1 = 0.7n), plus les solutions optimales sont plus faciles à trouver.

Chapitre 3

État de l'art

Sommaire

Dans ce chapitre, nous présentons les principales références relatives aux différents problèmes d'ordonnancement, de type flow shop hybride, qui s'approchent de notre problème.
Elles porteront successivement sur les différentes contraintes rencontrées dans notre étude. Cette revue de la littérature sera organisée comme suit : nous évoquons dans un premier temps, dans la section 3.2, les principales références qui traitent du flow shop standard. Puis dans un second temps, dans la section 3.3, nous présentons une étude de toutes les contraintes additionnelles liées à notre problème. Dans les sous-sections 3.3.1 et 3.3.2, nous verrons principalement les deux contraintes "no-wait" et "time lag maximal". Dans la sous-section 3.3.4, nous ferons une synthèse de la littérature portant sur la contrainte de "temps de setup dépendant de la famille de job". La section 3.4 sera dédiée aux travaux portant sur le flow shop hybride avec des machines dédiées. Dans la section 3.5, nous résumons les travaux existants. Nous clôturons ce chapitre par présenter les principales contributions.

3.1 Introduction

Comme mentionné dans le chapitre précédent, nous considérons un atelier de type flow shop hybride (FH) à deux étages. L'étage 1 (E1) est composé de deux machines dédiées alors que l'étage 2 (E2) est composé de dix machines parallèles identiques. Nous avons constaté que les revues les plus exhaustives et les plus complètes relatives au FH ont été publiées en 2010 dans

deux journaux différents ([Ruiz and Rodriguez, 2010] et [Ribas et al., 2010]). Dans ces revues, les auteurs ont rappelé la complexité du problème et sa pertinence pratique. Ensuite, ils ont récapitulé les différentes méthodes exactes et approchées pour le résoudre ainsi que les problèmes connexes. Malgré le nombre important de travaux qui portent sur le flow shop hybride, il est encore difficile dans le cas du FH standard de bien comparer deux algorithmes entre eux. Ceci est dû au fait que la plupart des chercheurs génèrent leurs propres instances ou bien utilisent des instances industrielles. Selon une analyse statistique faite par [Ruiz and Rodriguez, 2010], 60% des articles se sont intéressés au critère C_{max}.
Selon la notation unifiée $\alpha|\beta|\gamma$ proposée par [Graham et al., 1979], puis étendue par [Vignier, 1997], nous avons les caractéristiques suivantes :

TABLE 3.1 – Classification $\alpha|\beta|\gamma$ étendue par [Vignier, 1997]

α	
$\alpha 1$	Type d'atelier
α 2	Nombre d'étage
$(\alpha$ 3 α 4$)^k$	α 4 machines de type α 3 dans l'étage k
β	
les conditions d'exécution des jobs, les états des ressources présentées dans l'atelier	
γ	
Critères d'évaluation	

Suivant cette classification, notre problème est représenté par le triptyque suivant :
$FH2, (PD2, Pm)|ST_{g,sd}, l_i|C_{max}$. Nous rappelons que dans le chapitre 2, nous avons bien classifié le problème étudié en flow shop hybride à 2 étages (FH2) avec 2 machines parallèles dédiées (PD2) et plusieurs machines parallèles identiques (Pm). Nous rappelons aussi que les contraintes étudiées sont : "temps de setup dépendant de la famille de job" appelée dans notre cas "sequence dependent group setup" ($ST_{g,sd}$) car le terme "family", dans notre étude, désigne la famille des jobs correspondant à chaque machine dédiée, time lag maximal (l_i) avec un objectif de minimisation des dates de fin de production (C_{max}). Vu la richesse de la littérature relative au FH, nous nous concentrons, dans ce qui suit, principalement sur le FH à deux étages avec machines identiques et nous allons présenter les travaux les plus pertinents pour notre cas d'étude.

3.2 FH Standard

Le FH est considéré comme un flow shop général, auquel s'ajoutent des machines dupliquées à un ou plusieurs étages. Ce problème a été prouvé NP-Complet par [Garey and Johnson, 1979]. Plus particulièrement, le problème à deux étages est NP-difficile même si l'un des étages contient une seule machine ([Gupta, 1988]).
Le travail de ces derniers auteurs est considéré parmi les premiers travaux publiés dans lequel les

auteurs ont proposé une heuristique basée sur la règle de Johnson [Johnson, 1954] pour résoudre un cas particulier de FH à deux étages avec m machines identiques sur le premier étage et une seule machine sur le deuxième étage. L'approche a été testée sur des instances aléatoires pour la plupart résolues à l'optimum.
De plus, [Lee and Vairaktarakis, 1994] ont proposé une heuristique basée aussi sur la règle de Johnson [Johnson, 1954] pour résoudre $FH2(Pm, Pm)||C_{max}$. Les auteurs ont utilisé deux techniques d'affectation des jobs aux machines parallèles, à savoir, FAM (First Available Machine) et LBM (Last Busy Machine). Pour évaluer l'efficacité de l'heuristique, ils ont introduit également des bornes inférieures intuitives. L'algorithme produit des solutions de bonne qualité en réalisant un écart faible qui ne dépasse pas 3%.
Une modélisation générale du problème FH à plusieurs machines sur les deux étages FH2 (Pm, Pm)| | C_{max} a été proposée par [Haouari and M'Hallah, 1997]. Les auteurs ont conçu deux heuristiques l'une étant un recuit simulé et l'autre une recherche tabou. L'approche proposée améliore les résultats obtenus par [Lee and Vairaktarakis, 1994] et ont permis d'obtenir des solutions optimales pour 35% des instances aléatoires.
Un algorithme exact utilisant une méthode de Branch and Bound a été proposé par [Haouari et al., 2006] pour résoudre FH2(Pm,Pm)| | C_{max}. Les auteurs ont présenté des bornes inférieures basées sur la règle de priorité SPT. Des nouvelles instances aléatoires ont été résolues. Les auteurs ont signalé que les cas les plus difficiles à résoudre sont ceux où les charges de travail dans les deux étages sont équilibrées. (82% des instances équilibrées (ensemble B) et 95% des instances déséquilibrées (ensemble C) ont été résolues à l'optimalité).
Deux années plus tard, [Kahraman et al., 2008] ont rappelé la modélisation mathématique d'un FHm(Pm)||C_{max}. Dans un second temps, un algorithme génétique est proposé pour résoudre le problème et est comparé à deux travaux de la littérature, à savoir, la PSE de [Néron et al., 2001] et la méta-heuristique AIS (Artificial Immune Systems) de [Engin and Döyen, 2004]. Le taux de résolution obtenu dépasse légèrement celui réalisé par [Néron et al., 2001, Engin and Döyen, 2004] (4%).
[Engin et al., 2011] ont opté aussi pour un AG afin de résoudre le même problème. Une comparaison aux solutions de l'AG proposé par [Oğuz and Ercan, 2005] et l'algorithme glouton parallèle de [Kahraman et al., 2010] a été élaborée. Dans ce travail, des améliorations de 14,80% par rapport à la première étude, et de 1,47% par rapport à la seconde étude sont obtenues.
Un modèle mathématique concernant une version particulière de FH inspirée d'une application réelle a été proposé par [Yalaoui et al., 2013]. Le cas industriel étudié est spécifique étant donné que les machines parallèles ne sont pas toutes identiques. Les auteurs ont étudié la minimisation de makespan et la somme de retards ainsi que des contraintes de pré-affectation des jobs. Pour cela, ils ont proposé un algorithme génétique (GA) et une méthode d'optimisation par essaims particulaires (PSO). Pour tester leur méthode, les auteurs ont créé un nouveau jeu de données en se basant sur les instances de [Azizoglu and Kirca, 1998]et [Yalaoui and Chu, 2002]. Les résultats ne dépassent pas 1% à la solution optimale obtenue par le MIP en ce qui concerne les instances de petite taille.
On peut présenter aussi le travail de [Akkoyunlu et al., 2015] qui porte sur le problème FHm(Pm)|| C_{max} en utilisant la recherche par harmonie (Harmony Search : HS). Cette méthode est inspirée du processus musical proposé par [Geem et al., 2001]. Les auteurs ont comparé leur approche à celle de [Kahraman et al., 2008] et [Engin et al., 2011] en utilisant les mêmes instances de test. Seulement 20 instances ont été améliorées.

Dans cette section, nous pouvons remarquer qu'il existe un grand nombre de méthodes pour résoudre le problème de FH. Néanmoins, notre problème considère des contraintes additionnelles. Nous allons les traiter dans les sections qui suivent. Cependant, nous pouvons retenir de la littérature la modélisation par un MIP car il s'agit effectivement d'un FH. En effet, les formulations relatives au FH se basent sur trois familles de variables, à savoir :
-la première famille concerne les temps de fin de traitement ou du début de traitement de chaque job sur chaque étage ;
-la deuxième famille concerne l'affectation des jobs aux machines parallèles ;
-la troisième famille l'ordre de traitement des jobs sur chaque machine.
Dans notre cas d'étude, les travaux de [Kahraman et al., 2008] et [Liao et al., 2012] semblent intéressants car ils séparent la variable d'affectation de la variable relative à l'ordre de traitement.
Dans ce cadre, nous allons adapter ce modèle à notre problème dans le chapitre 4.
D'autre part, au niveau de la formulation mathématique (Chapitre 4), nous allons aussi nous inspirer de l'idée de [Yalaoui et al., 2013] pour représenter la pré-affectation des jobs aux machine dédiées sur E1. L'idée des auteurs était de représenter la pré-affectation des jobs sur les machines dédiées par une matrice Y où les lignes représentent les jobs et les colonnes représentent les machines dédiées. A titre d'exemple, $Y_3^{2,1} = 1$, signifie que le job 3 est pré-affecté à la machine 2 au niveau de l'étage 1. En ce qui concerne le codage d'une solution, qui est une étape nécessaire pour l'application d'une méta-heuristique ou l'application du PSE, nous allons nous inspirer de la proposition de ([Haouari et al., 2006]). Dans ce cadre, il convient de signaler que les premiers articles qui ont abordé la PSE pour le FH, tels que ([Brah and Hunsucker, 1991]), avaient proposé un codage basé sur une permutation par machine. Plus tard, on a constaté qu'il suffit de coder une solution par une permutation unique pour un étage donné selon l'ordre non décroissant du temps de début de traitement sur un étage donnée. Dans le cas du problème traité dans cette thèse, et puisque l'E2 contient des machines parallèles identiques, nous proposons au niveau de AG1 (Chapitre 5) de coder la solution par une permutation sur E2 selon l'ordre non décroissant du temps de début de traitement, ensuite construire une permutation pour chacune des machines dédiées de E1 par « backtracking ».
Cette littérature de problèmes portant sur le FH montre bien la difficulté du problème à résoudre. Dans ce qui suit nous allons examiner la littérature relative à ce problème avec des contraintes additionnelles.

3.3 Revue de la littérature du FH suivant les contraintes additionnelles

Dans cette section, nous allons présenter des travaux qui traitent les contraintes additionnelles considérées dans cette thèse, à savoir : sans attente « no-wait », durée d'attente limitée « time lag maximal (l_i) » et setup time « ST_{sd}».

3.3.1 FH avec no-wait

Dans l'industrie, il existe de différents systèmes de production. Parmi lesquels, on trouve les systèmes soumis à la contrainte de "no-wait". Un exemple de ce type de production est rencontré

dans l'industrie de l'acier où le métal chauffé passe par différentes opérations telles que la fusion, la coulée et le laminage. Un autre exemple se produit dans l'industrie agro-alimentaire où il est nécessaire que la nourriture puisse être conservée en temps réel dans un état frais et utilisable. D'autres exemples peuvent être cités, à savoir, la fabrication des produits pharmaceutiques ([Raaymakers and Hoogeveen, 2000]), le traitement alimentaire ([Hall and Sriskandarajah, 1996]), le traitement chimique ([Rajendran, 1994]), l'industrie électronique ([Marcelo et al., 2012]).
En général, dans un système de production soumis à la contrainte de "no-wait", les travaux sont traités sur une machine et passent sur la suivante sans autoriser un temps d'attente [Huang et al., 2009]. Une fois qu'un job est commencé sur la première machine, il doit donc être traité en continu par les machines suivantes sans interruption (figure 3.1 où les J1, J2 et J3 sont no-wait). Par conséquent, le début d'une tâche sur la première machine doit être retardée afin de répondre à l'exigence de no-wait [Tasgetiren et al., 2007].

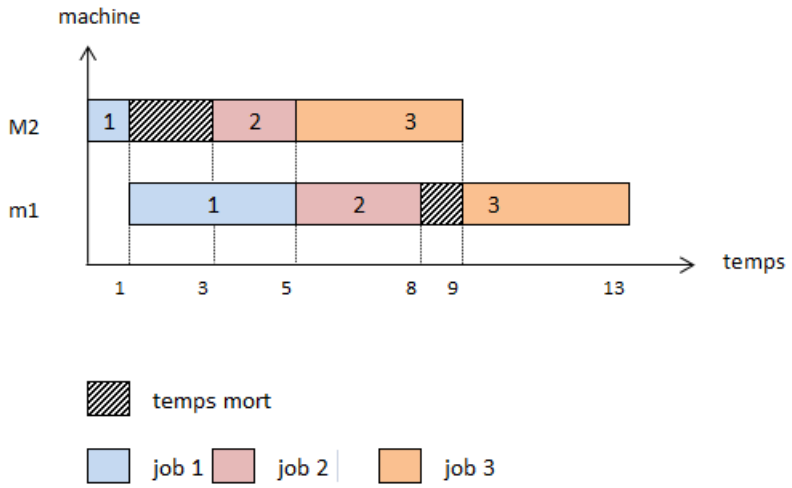


FIGURE 3.1 – Diagramme de Gantt d'une situation "no-wait"

En raison de l'importance de l'ordonnancement sans attente en théorie et en pratique dans les entreprises industrielles, ce genre de problèmes d'ordonnancement a attiré l'attention de nombreux chercheurs [[Song and Tang, 2008], [Ruiz and Allahverdi, 2009], [Jolai et al., 2009], [Moradinasab et al., 2013], [Jolai et al., 2012],[Rabiee et al., 2012], [Naderi et al., 2014], [Pang, 2013]].
Bien que de nombreuses recherches ont été dédiées au flow shop sans attente, il y a eu peu de tentatives dédiées au FH sans attente. Une première revue de littérature approfondie sur les problèmes d'ordonnancement avec blocage et no-wait a été élaborée par [Hall and Sriskandarajah, 1996] (il convient de rappeler que le blocage survient quand il n'y a pas d'espace de stockage entre les machines). Une revue plus récente a été publiée par [Allahverdi, 2016] en 2016.
Dans le cadre du FH avec no-wait, nous trouvons par exemple, [Liu et al., 2003] qui ont présenté une heuristique pour un flow shop hybride FH2(1,Pm)|no-wait| C_{max}. L'idée de base de cet algorithme est de réduire le temps d'inactivité sur les machines lorsque les jobs sont affectés. L'heuristique a été testée sur des instances aléatoire et comparée aux algorithmes de [Sriskandarajah and Sethi, 1989, Sriskandarajah, 1993]. En outre, les résultats ont montré que

l'approche proposée est plus performante qu'une version adaptée de l'algorithme de Johnson [Johnson, 1954].
Cette étude a été étendu par [Xie et al., 2004] qui a utilisé la même méthode de résolution pour traiter un cas plus général de flow shop hybride : FH2(Pm,Pm)|no-wait|C_{max}. Les solutions d'un jeu de données aléatoires sont obtenues avec un gap de 6% par rapport à une borne inférieure proposée.
Nous trouvons aussi [Song and Tang, 2008] qui ont proposé un modèle mathématique et une méthode d'optimisation par essaims particulaires (PSO) pour résoudre un problème réel de type FH2(Pm,Pm)|no-wait | C_{max}. Ce problème est rencontré dans une usine industrielle d'acier. Les auteurs ont montré l'efficacité de la solution proposée en la comparant à la règle de priorité LPT.
En 2011, [Shafaei et al., 2011] ont proposé six heuristiques pour résoudre un FH2(Pm,Pm)|no-wait|C_{max}. En outre, une heuristique basée sur un réseau de neurones artificiels (Adaptive Neuro Fuzzy Inference System : ANFIS) est appliquée pour calculer la durée du makespan. Les auteurs ont comparé les résultats trouvés à des règles de priorité et des travaux de la littérature. La méthode proposée est avérée efficace à résoudre ce type de problème.
D'autre part, trois autres algorithmes sont proposés par [Jolai et al., 2012] pour résoudre FH2(Pm,Pm)|no-wait, $ST_{sd}|C_{max}$, à savoir le recuit simulé, l'algorithme compétitif impérialiste adaptée (adapted imperialist competitive algorithm : AICA) et leur hybridation. Les algorithmes ont été testés sur des instances aléatoires. Les auteurs ont comparé les résultats entre eux pour montrer que la dernière méthode hybride est la plus prometteuse.
La méthode approchée AICA et un algorithme génétique ont été proposés par [Moradinasab et al., 2013] pour résoudre un FH2(Pm,Pm)|no-wait, $ST_{sd}|C_{max}$. Les algorithmes sont comparés à un algorithme de colonies de fourmis(ACO). Les résultats trouvés indiquent que l'AICA surpasse l'AG et ACO d'une manière significative.
[Wang and Liu, 2013a] ont utilisé un algorithme génétique pour résoudre le problème FH2(1, Pm)|no-wait|C_{max}. Dans cet algorithme, une permutation sur la machine critique de l'étage 1 est définie ensuite les jobs sont affectés aux machines parallèles de l'étage 2 suivant la règle FAM. Sur des instances générées suivant [Garey and Johnson, 1979], des solutions avec un gap de 5% par rapport à des bornes inférieures globales sont trouvées. Plus récemment, [Naderi et al., 2014] ont étudié un FHm(Pm)|no-wait|C_{max}. Pour le résoudre, les auteurs ont proposé dans un premier temps, un modèle mathématique. Dans un second temps, les auteurs ont présenté une méthode heuristique : un algorithme de recherche de chasse (HSA : Hunting Search Algorithm). Les résultats obtenus montrent que le HSA proposé est plus performant que d'autres algorithmes de la littérature.

3.3.2 FH avec time lag maximal

Bien qu'il existe des systèmes de production soumis à la contrainte de "no-wait", nous pouvons trouver également des scénarios intermédiaires entre la production sans attente et celle avec attente illimitée, car il existe des situations où les jobs peuvent attendre un certain temps, mais pas au-delà d'une limite donnée. Par exemple, dans une chaîne de froid, les aliments congelés peuvent attendre jusqu'à 10 minutes afin de contrôler la contamination microbienne. Dans ce cas, nous définissons la contrainte de délai maximal appelée dans la littérature time lag maximal ou limited waiting time. Cela signifie que les durées d'attente entre deux opérations consécutives ne peuvent

pas être supérieures à une limite donnée de temps (figure 3.2). Certains auteurs de la littérature utilisent cette contrainte sous le nom de time lag positif : lag^+ [Valérie, 2000].
D'autre part, nous trouvons une autre situation où les opérations suivantes ne peuvent pas commencer tant qu'un délai minimum n'a pas expiré. Par exemple, les opérations de peinture qui nécessitent un temps de séchage minimum avant que des opérations ultérieures puissent être effectuées.
Un autre exemple concerne les produits qui nécessitent un certain type de cuisson. Dans ce cas, un temps de refroidissement est nécessaire pour pouvoir les manipuler. Cette situation est définie par la contrainte de délai minimal, appelée dans la littérature time lag minimal : lag^-.
Dans notre cas d'étude, nous nous intéressons au time lag maximal. Tous les travaux cités dans cette section étudient cette contrainte.

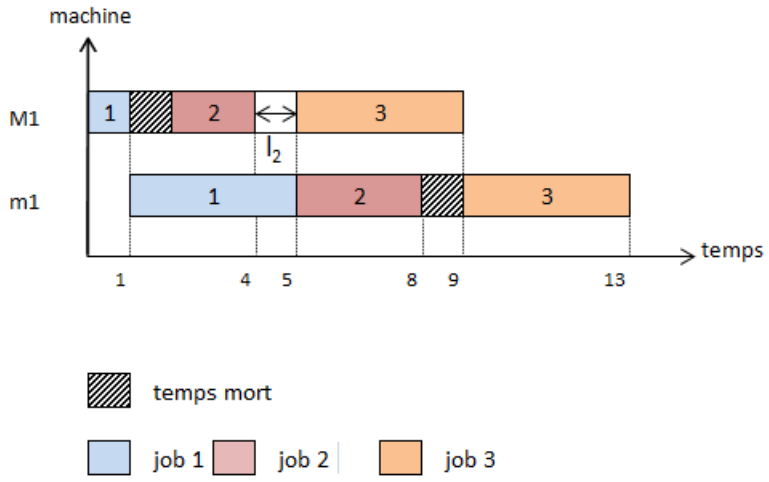


FIGURE 3.2 – Diagramme de Gantt d'une situation "Time lag"

Dans un premier temps, la complexité du flow shop à k étages avec une seule machine par étage soumis à la contrainte de délai limité a été étudiée par [Yang and Chern, 1995]. Les auteurs ont montré qu'il s'agit bien d'un problème NP-difficile.
Comme dans le cas du FH No-Wait, une revue de la littérature ([Chen and Yang, 2006]) révèle également qu'il y a peu de recherches se rapportant au FH avec time lag maximale.
Parmi les travaux rencontrés dans la littérature, on peut citer celui de [Su, 2003] qui ont considéré un FH2 (Pm,1)$|l_i|C_{max}$. Les auteurs ont proposé une heuristique basée sur la minimisation de la durée totale de traitement (Total Processing Time : TPT) et un modèle linéaire en nombre entier. Les tests ont été effectués sur des instances aléatoires. Les résultats des méthodes sont comparés à une borne inferieure obtenue par le solveur LINGO. L'heuristique réussit à trouver des solutions optimales dans 96% des instances.
Par ailleurs, [Li and Li, 2007] ont présenté une heuristique (CBH : Constructive Backtracking Heuristic) pour minimiser le makepan d'un flow shop hybride à k étages soumis au lag^+. L'heuristique proposée prend en considération la contrainte de délai maximal. En effet, les jobs passent d'un étage au suivant et chacun est affecté à la première machine disponible. Dans le cas où toutes les machines sont occupées, il y aura un backtracking sur les dates de début de traitement sur l'étage précédent. Les jeux de tests sont aléatoirement générés. Les solutions produites sont bonnes (avec

un gap de 3% par rapport à la borne inferieure proposée par [Santos et al., 1995]).
Nous trouvons aussi, le travail de [Liu et al., 2008] qui résout le même problème FHk(Pm)| $l_i|C_{max}$. Les auteurs ont proposé une méthode hybride basée sur la recherche tabou et l'heuristique (CBH). Les résultats sur les instances de [Li and Li, 2007] ont été améliorés de 10%.
Quant à [Attar et al., 2014], ils ont travaillé sur un FHm (Pm)|ST_{sd}, $l_i|C_{max}$, L_{max}. Les auteurs ont considéré deux objectifs : le makespan et le retard algébrique. Pour cela, les auteurs ont proposé dans un premier temps, un modèle mathématique bi-objectif. Dans un second temps, ils ont appliqué deux méthodes approchées, basées sur le principe de Pareto, soit la PSO et l'algorithme évolutionnaire. Les tests ont été menés sur un nouveau jeu de données. En analysant des métriques calculées, les performances des algorithmes proposés ont été comparées. PSO est avérée la méthode la plus prometteuse à résoudre ce type de problème.

Comparaison avec le problème traité

En se basant sur cette revue, on constate que plusieurs travaux ont traité la contrainte no-wait. D'autres travaux ont traité la contrainte de « time-lag positif » qui est représenté par un intervalle de temps [a-b] où a et b sont strictement positifs.
Une première différence importante porte sur la considération simultanée de la contrainte « time lag maximal » avec une autre contrainte dans un FH.
Parmi tous ces travaux, nous allons nous inspirer des approches qui utilisent l'heuristique CBH ([Li and Li, 2007], [Liu et al., 2008]) pour traiter le time-lag maximal. Le principe de l'heuristique est expliqué dans le paragraphe précédent.
D'autre part, dans le cas de no-wait, tous les travaux ont utilisé la technique FIFO étant donné que c'est la seule solution faisable pour ce type de contrainte.

3.3.3 FH avec setup

Le flow shop hybride qui intègre la contrainte de setup est connu dans la littérature en tant que FH avec "temps de setup dépendant de la séquence" (sequence dependent setup : SDST). [Gupta and Tunc, 1998] ont montré que le problème de FH avec SDST et avec seulement deux machines par étage est NP-difficile au sens fort. Dans certains problèmes d'ordonnancement, le temps de setup est négligé ou considéré comme une partie du temps d'exécution ([Ulungu et al., 1999]). Il convient de signaler que dans beaucoup de problèmes réels, cette hypothèse doit être affinée car la durée de setup n'est pas négligeable. Il existe trois catégories de setup :

1. séquence indépendante : le temps de configuration (setup) dépend uniquement du job à traiter,
2. séquence dépendente : le temps de configuration dépend non seulement de job en position actuelle mais aussi de celui qui le précède dans la séquence de traitement, on parle donc de temps de setup dépendant de la séquence : ST_{sd} ;
3. Temps de setup dépendant de la famille de job (sequence dependent family setup : SDFST), appelé aussi dans la littérature "Sequence Dependent Group setup (SDGS)" : le temps de configuration ne dépend pas du job lui-même mais il dépend de son groupe. En effet, les jobs sont classés en groupes (un groupe correspond à une sous-famille dans notre cas d'études) et

chaque groupe à un temps de configuration bien déterminé. Une configuration majeure est nécessaire lorsqu'une machine commence à traiter un nouveau groupe de jobs (car les jobs d'un même groupe s'enchainent sans temps de setup) : $ST_{g,sd}$

Nous avons constaté que la revue la plus exhaustive et le plus complète relative au problème d'ordonnancement avec setup a été publiée en 2015 par [Allahverdi, 2015]. L'auteur a fait appel à plus de 500 articles scientifiques publiés entre 2006 et fin 2014. Dans ce qui suit, nous présentons les travaux les plus pertinents qui traitent ST_{sd} et $ST_{g,sd}$

Temps de setup dépendant de la séquence

La plupart des articles font appel aux MIP et des méta-heuristiques pour résoudre le FH avec SDST. Dans ce cadre, [Zandieh et al., 2006] ont utilisé un algorithme immunitaire (IA : immune algorithm) (initialement proposé par [Mori et al., 1993]) pour un $FHk(Pm)^k|ST_{sd}|$ C_{max}. Des améliorations qui arrive jusqu'à de 30% sont apportées sur 95% des résultats de [Kurz and Askin, 2004].

[Ruiz and Maroto, 2006] ont proposé un AG, dans lequel ils ont introduit quatre opérateurs de croisement : SJOX, S2JOX, SBOX, S2BOX pour $FHk(RM)^k|ST_{sd}$, Mj$|C_{max}$. Les opérateurs proposés permettent de traiter la contrainte de setup où les nouveaux descendants héritent les caractéristiques communes de leurs parents. Pour évaluer la méthode de résolution, les auteurs ont adapté plusieurs autres méta-heuristiques au problème étudié : recuit simulé [Osman and Potts, 1989], recherche tabou [Widmer and Hertz, 1989], un AG proposé par [Reeves, 1995, Aldowaisan and Allahverdi, 2003]. Sur un jeu de données de 1320 instances, l'algorithme propose une marge d'amélioration entre 53% et 135 % par rapport aux autres méthodes testées. La méthode offre aussi des meilleures solutions aux instances réelles par rapport aux solutions manuelles du chef de production avec un taux d'amélioration entre 2.32% et 16.95%.

Dans le travail de [Yaurima et al., 2007], un AG a été proposé pour un problème d'ordonnancement réel dans une usine qui fabrique des circuits électriques : FH SDST avec des contraintes de disponibilité de machines. En suivant [Reeves, 1995], les auteurs ont utilisé un schéma de remplacement où les pires individus sont remplacés par les nouveaux meilleurs descendants. De plus, ils ont adopté une technique proposée par [Ruiz and Maroto, 2006] pour affecter les jobs aux machines parallèles à chaque étage, où la décision d'affectation se fait dans la fonction d'évaluation. La méthode proposée montre une supériorité de résolution entre 0,16% et 13,35% pour les instances de [Ruiz and Maroto, 2006].

En outre, [Lo et al., 2008] ont proposé une méthode exacte PSE et un algorithme génétique (AG) afin de résoudre un problème de flow shop flexible à deux machines en considérant le setup. Les résultats obtenus par l'AG affirment son efficacité en les comparant aux solutions optimales trouvées par la PSE proposé.

Un algorithme génétique (AG) a été proposé par [Sioud et al., 2013] pour résoudre $FHk(Pm)^k$ $|ST_{sd}|$ C_{max}. En effet, l'article a proposé une extension de l'opérateur RMPX (Random Maximal Preservative Crossover) proposé initialement par [Sioud et al., 2012]. Cet opérateur est dédié au traitement de setup. Les auteurs ont comparé l'algorithme proposé à celui de [Ruiz and Maroto, 2006] et à la recherche locale itérative de [Naderi et al., 2010]. Ils ont pu améliorer seulement 50% des résultats en réalisant un gain de 20%.

[Fattahi et al., 2014] ont travaillé sur le flow shop hybride avec opération d'assemblage. En effet les produits sont fabriqués dans les 2 premiers étages à machines parallèles, ensuite ils passent par

une phase d'assemblage sur un autre étage. Les auteurs ont utilisé un algorithme de branch-and-bound où ils ont réutilisé les bornes inférieures basé sur le SPT proposés par [Wu et al., 2012]. Récemment, [Pan et al., 2017] ont proposé des différentes heuristiques basées sur la recherche locale itérative et aussi un algorithme d'optimisation par colonie d'abeilles (Artificial Bee Colony : ABC). L'objectif était la minimisation du makespan d'un problème $FHk(Pm)^k|\ ST_{sd}|\ C_{max}$. Les auteurs ont comparé leur approche à plusieurs travaux de la littérature : AG de [Ruiz and Maroto, 2006], IA de [Zandieh et al., 2006], ILSN de [Naderi et al., 2010], le ISA de Naderi et Zandieh [Naderi et al., 2009], le AIS (artificial immune system) de Engin et Doyen [Engin and Döyen, 2004]. L'algorithme ABC a permis d'améliorer 126 résultats sur 240 instances générées aléatoirement.

Temps de setup dépendant de la famille de job (SDFST)

Dans la littérature, il y a peu de travaux qui ont considéré le FH avec SDFST. Un des premiers auteurs qui a étudié ce problème est [Huang and Li, 1998]. Ces auteurs ont étudié le problème de FH à deux étages où les jobs sont groupés en familles. Les auteurs ont proposé deux heuristiques utilisant des règles de séquencement basées sur les règles de priorité SPT, LPT. Les tests ont été menés sur 800 instances aléatoires. L'efficacité des heuristiques proposées a été mesurée par l'écart moyen à une borne inférieure (LB)) proposé par les auteurs (LB= somme des durées opératoires des jobs de chaque groupe en considérant un seul setup par groupe). Les solutions obtenues sont de bonne qualité (un gap faible 6%).
[França et al., 2005] ont travaillé sur cette contrainte dans un flow shop de permutation (PFS). Ils ont proposé deux algorithmes évolutionnaires : un algorithme génétique et un algorithme mimétique hybridé avec une recherche locale. Sur des instances de [Schaller et al., 2000], la méthode proposée améliore tous les résultats de 50%.
Pour le même problème, [Salmasi et al., 2010] ont proposé un modèle mathématique, un algorithme de recherche tabou (TS) et un algorithme de colonie de fourmis hybride (HACO). De plus, ils ont proposé une borne inférieure et ils ont pu évaluer ainsi la qualité des algorithmes proposés. Les résultats sur les instances de [Salmasi, 2005] montrent que HACO est plus performant que les autres méthodes qu'il a proposées en réalisant un écart par rapport à la borne inférieure ne dépassant pas 5.8 %.
Puis, [Shahvari et al., 2012] ont proposé un MIP pour un $FHk(Pm)^k|ST_{f,sd}\ |\ C_{max}$. Ensuite, ils ont développé des algorithmes basés sur la TS. Les auteurs ont comparé leur travail à celui de [Logendran et al., 2006] en utilisant les mêmes instances de test. La méthode proposée a amélioré de 5% les résultats des instances de moyenne et grande taille.
Nous pouvons citer aussi, [Ebrahimi et al., 2014] qui ont proposé deux méta-heuristiques hybrides avec un AG afin de minimiser le makespan et la somme des retards pondérés pour un $FHk(Pm)^k|ST_{f,sd}\ |\ C_{max}, TWT$. Les auteurs ont prouvé l'efficacité de l'approche proposée en la comparant à celle de [Karimi et al., 2010] en réalisant une amélioration de 2.5%.

Comparaison avec le problème traité

Cette revue nous a permis de constater que la contrainte de setup (SDST), bien qu'elle soit fréquemment utilisée dans les ateliers de type FH, le setup par groupe de jobs (SDGST) a été, à notre connaissance, beaucoup moins traité. Étant donné que notre problème porte sur la ca-

ractéristique SDGST, il semble important de s'inspirer du travail des auteurs qui prennent en considération cette caractéristique de deux points de vue : le nombre de groupes de jobs et la durée de setup comme le travail de [Ebrahimi et al., 2014].
Par ailleurs, certains auteurs ([Logendran et al., 2006] ont proposé des heuristiques dédiées au traitement de la contrainte SDGST en satisfaisant l'objectif C_{max}. L'idée était de minimiser le nombre de setup en construisant des ordonnancements par groupes de jobs. En effet, les jobs d'un même groupe s'enchainent sans temps de setup. Cette proposition est intéressante dans notre cas d'étude et sera réutilisée dans le chapitre 5.
Aussi, l'AG proposé par [Ruiz and Maroto, 2006] pour résoudre $FHk(Rm)^k)|ST_{sd}|C_{max}$ semble intéressant bien que les machines parallèles ne soient pas identiques. En effet, les opérateurs de croisement proposés traitent la contrainte de setup. Pour cela, un opérateur de ce travail sera réutilisé dans l'AG du chapitre 5.

3.4 FH avec machines dédiées

Le flow shop hybride avec machines dédiées est un cas particulier de flow shop hybride. En effet, chaque job, selon son type, doit être exécuté sur une machine fixée dans l'étage donné.
Le FH à 2 étages avec des machines dédiées a été considéré en premier par [Herrmann and Lee, 1992], dans lequel deux machines dédiées sont considérées dans le deuxième étage. Les auteurs ont montré qu'il est fortement NP-difficile avec l'objectif de minimisation de C_{max} ou l'objectif de minimisation du nombre de jobs en retard. De plus, [Oguz et al., 1997] ont montré que FH avec des machines dédiées sur le premier étage est NP-difficile.
Suite à ces travaux, d'autres études avec des caractéristiques diverses ont été rapportées dans la littérature. Selon la configuration des machines dédiées, les recherches existantes peuvent être classées en deux catégories : deux machines dédiées (m=2) et plusieurs machines dédiées (m $\geq$2).
Nous citons d'abord, [Riane et al., 2002] qui ont considéré FH2(1,PD2)$||C_{max}$. Ils ont étudié dans un premier temps la complexité du problème. Ensuite, les auteurs ont développé des heuristiques et un modèle de programmation dynamique (DP : Dynamic Program). Afin d'étudier l'efficacité de heuristiques, les solutions obtenues des instances de petite taille sont comparées au DP. Pour le reste des instances, les auteurs ont adapté et reprogrammé la procédure de [Campbell et al., 1970]. Cette heuristique est conçue à l'origine pour un flow shop de permutation et qui s'appuie sur la règle de Johnson [Johnson, 1954]. Les résultats par les heuristiques proposées se révèlent plus prometteurs que celle de [Campbell et al., 1970].
Nous pouvons citer aussi, [Chikhi and Abbas, 2012] qui ont proposé un MIP pour un FH2(PD_m,1)| $|C_{max}$. Ensuite, ils ont développé des bornes inférieures et deux heuristiques. Des solutions proches de l'optimum sont trouvées qui prouvent l'efficacité de l'approche proposée. Le même problème a été étudié par [Chikhi et al., 2014]. Les auteurs ont proposé deux MIPs pour résoudre des instances de petite taille et une heuristique pour les instances de grande taille. Les solutions trouvées étaient de bonne qualité.
Plus récemment, Wang et Liu [Wang and Liu, 2013b] ont traité FH2(1,PD_m)$||C_{max}$ avec des machines dédiées au deuxième étage. Les auteurs ont présenté en premier lieu trois LB basées sur une relaxation de machines sur l'étage 2. Ensuite, une méthode heuristique basée sur une structure de PSE a été proposée. La méthode proposée n'est pas une méthode exacte bien qu'elle soit construite sur la structure de PSE. En effet, les auteurs ont réduit l'espace de recherche en partant d'un

sous-ensemble de job fixé (m jobs fixés) dans le premier niveau au lieu de fixé un seul job. D'autre part, ils ont suivi [Yang, 2011] et [Gupta et al., 1997] pour générer des instances aléatoires. Les problèmes de petite taille sont résolus à l'optimum et ceux de grande taille sont résolus avec un gap inférieur à 3% par rapport à des bornes inférieures présentées.
On trouve aussi, [Hadda et al., 2014] qui ont proposé une méthode exacte pour résoudre FH2(1, PDm)||C_{max}. Les auteurs ont proposé des règles d'élimination et ont développé deux heuristiques pour le calcul de la borne supérieure initiale. Les auteurs ont montré l'efficacité des règles utilisés qui ont permis d'écarter 50% des nœuds. Les résultats obtenus montrent aussi que la méthode proposée est prometteuse car les instances utilisées sont résolues avec un écart faible par rapport aux bornes inférieures ($< 2\%$).
D'autre part, [Yang, 2015b] a étudié ce problème avec 2 machines dédiées sur le premier étage et une seule machine sur l'étage 2. Il a étudié la complexité des cas particuliers du problème où il a démontré que FH2 ($PD2$,1) || C_{max} est polynomial dans le cas où les jobs sur les machines dédiées ont les mêmes durées de traitement. En deuxième lieu, il a proposé deux heuristiques. La première est basée sur une règles simple : SPT et la deuxième est inspirée de l'algorithme Avide (appelé aussi algorithme Glouton qui cherche des solutions optimales localement sans rechercher dans le voisinage).
De même, l'auteur de [Yang, 2011], traite le même problème avec une autre configuration FH2 (1,$PD2$) | |C_{max}. Il a proposé des bornes inférieures qui supposent que chaque job passe le plus tôt possible sur l'étage 2. Ensuite, il a introduit 2 heuristiques basées sur une règles simple : SPT. Des instances aléatoires sont résolues avec un écart faible par rapport aux bornes inférieures.
[Hajji et al., 2015] se sont intéressés à résoudre le FH2(1, PDm)|r_j,q_j|C_{max}. Les auteurs ont proposé des bornes inférieurs, des heuristiques basiques ainsi qu'une heuristique de recherche tabu. Les calculs montrent que les solutions obtenues par certaines méthodes proposées sont de bonne qualité. En effet, les résultats indiquent que la déviation moyenne par rapport aux bornes inférieures ne dépasse pas 0,35%.
Plus récemment, [Hwang and Lin, 2018] ont publié une revue relative aux flow shop flexible avec des machines dédiées. Les auteurs ont présenté les différentes configurations traitées dans la littérature. Ils ont rappelé la complexité de chaque problème et les cas résolus à l'optimalité.
D'autre auteurs, tels que Mosheivo and Sarig [Mosheiov and Sarig, 2010], [Lin and Liao, 2003], [Lin, 2015], [Yang, 2013] se sont intéressés à traiter l'ordonnancement dans un atelier avec des machines dédiées, mais traitant d'autres objectifs tel la minimisation du retard algébrique maximum L_{max}, et le retard absolu maximum $\sum T_j$ ou traitant plusieurs machines dédiées.

Comparaison avec le problème traité

Les ateliers de type FH avec des machines dédiées est similaire au contexte de notre étude. Plusieurs travaux ont traité FH2 avec deux machines dédiées et ont proposé des données de test où ils ont déterminé un partage de charges diversifié sur les machines dédiées (partage équitable, partage non équitable) ([Yang, 2015a], [Wang and Liu, 2013b]). Ce point sera retenu dans notre travail lors de génération d'instances, car nous considérons sur l'étage 1 des machines dédiées sur lesquelles il est nécessaire de partager les charges.
La différence dans cette section porte sur la nature des machines utilisées dans notre atelier de travail : des machines dédiées et des machines parallèles identiques. Il convient de signaler que les travaux consultés dans la littérature n'ont pas considéré une telle configuration.

3.5 Résumé des travaux de la littérature

A notre connaissance, il n'existe pas d'autres travaux où les contraintes additionnelles sont considérées simultanément dans un même atelier. Nous avons aussi constaté que la plupart des recherches dans l'atelier de flow shop hybride à 2 étages avec une contrainte additionnelle se concentrent sur le développement d'heuristiques ou de solutions optimales pour des cas particuliers, ce qui explique l'absence de l'emploi d'une méthode exacte tel que la PSE à résoudre FH2 (PD2, Pm)| $SD_{f,st}$, l_i| C_{max}.
De plus, nous avons remarqué qu'il n'y a pas de benchmark commun et que la plupart des auteurs ont utilisé leurs propres instances. Ceci est dû aux différentes configurations étudiées du FH. Dans le tableau 3.2, nous récapitulons les caractéristiques des problèmes rencontrés dans la littérature ainsi que les méthodes utilisées. Nous nous limitons aux articles ayant traité l'objectif de minimisation C_{max} car c'est l'objectif principal qui nous intéresse dans cette thèse.

TABLE 3.2 – Résumé des travaux de la littérature

Année	**auteurs**	PD	ST_{sd}	$ST_{g,sd}$	l_i	no-wait	**Méthode de résolution**
2006	[Haouari et al., 2006]						**LB, PSE**
2006	[Ruiz and Maroto, 2006]		•				**AG**
2007	[Li and Li, 2007]				•		**heuristique**
2008	[Liu et al., 2008]				•		**heuristique, TS**
2008	[Yaurima et al., 2007]		•				**AG**
2012	[Shahvari et al., 2012]			•			**MIP, TS**
2012	[Chikhi and Abbas, 2012]	•					**LB, heuristique**
2013	[Wang and Liu, 2013a]					•	**AG**
2014	[Wang et al., 2015]					•	**PSE**
2014	[Ebrahimi et al., 2014]			•			**méta-heuristique, AG**
2014	[Attar et al., 2014]		•		•		**méta-heuristique, AG**
2014	[Chikhi et al., 2014]	•					**MIP, LB**
2015	[Yang, 2015a]	•					**LB, heuristique**
2017	[Pan et al., 2017]		•				**heuristiques**
2017	Notre problème	•		•	•	•	**MIP, AG, Heuristiques**

Nous signalons que le nombre de machines dédiées considéré dans cette thèse est PD = 2.

3.6 Contributions scientifiques

En parcourant la revue de la littérature, nous avons bien constaté que le problème étudié dans cette thèse peut être décomposé en sous-problèmes. Chaque sous-problème est déjà traité par des chercheurs et des solutions exactes ou approchées ont été proposées. Pour cela, nous nous sommes inspirés de ces travaux pour résoudre notre problème. Dans ce qui suit, nous présentons un résumé des solutions scientifiques de la littérature que nous avons adaptées ou enrichies par nos contributions :

— [Kahraman et al., 2008] : MIP
- Ré-utiliser les variables de décision
- Ajouter les contraintes d'affectation qui gèrent l'affectation d'un seul job à une seule machine parallèle et qui imposent l'affectation aux machines dédiées
- Ajouter les contraintes qui assurent la comptabilisation de la durée de setup selon le groupe de deux jobs successifs
- Ajouter la contrainte qui autorise une durée d'attente aux jobs

— [Yalaoui et al., 2013] : MIP
- Adapter les contraintes de pré-affectation des jobs aux machines dédiées : remplacer une variable de décision binaire par une matrice de données
- Ajouter les contraintes qui assurent la comptabilisation de durée de setup selon le groupe de deux jobs successifs
- Ajouter la contrainte qui autorise une durée d'attente aux jobs

— [Logendran et al., 2006] : heuristique
- S'inspirer de l'idée générale : assembler les jobs suivant le groupe
- Ordonnancer les jobs d'un seul groupe suivant LPT
- Appliquer la recherche locale à l'intérieur de groupe

— [Ruiz and Maroto, 2006] : opérateur de croisement : SJOX
- Adapter cet opérateur conçu à l'origine pour traiter SDST afin qu'il traite SDFST

— [Sioud et al., 2012] : opérateur de croisement : RMPX
- Adapter cet opérateur conçu à l'origine pour traiter SDST afin qu'il traite SDFST

— [Li and Li, 2007], [Liu et al., 2008] : Heuristique CB
- Traiter le time lag maximal

— [Wang and Liu, 2013b], [Yang, 2015a] : Génération des instances
- Utiliser les proportions de partage des charges entre les machines dédiées : 50%, 60% et 70%

— [Morizawa, 2014] : Heuristique arborescente
- Réutiliser une règle d'exploration proposée : ne pas explorer des nœuds ayant une LB inférieure à $(1 + \alpha)$LB0

Il faut signaler que le travail de [Morizawa, 2014] ne figure pas dans l'état de l'art car il ne porte pas sur le flow shop hybride ni sur une contrainte étudiée. Cependant, nous nous inspirons d'un détail de son travail dans le chapitre 6.
C'est vrai que nous nous sommes appuyés sur la richesse de la littérature pour résoudre le problème étudié et nous avons bien travaillé sur l'adaptation des méthodes déjà proposées aux exigences du problème et ses contraintes simultanées, cependant, nous avons proposé des solutions scientifiques adéquates pour résoudre ce problème (présentées dans le tableau 3.3).

TABLE 3.3 – Propositions scientifiques

Contribution	Spécificité
2 MIPs	Considère des différentes contraintes temporelles simultanément
2 AGs	Deux codages d'individus différents
	étudier différentes versions des AGs (sans et avec hybridation avec d'autres heuristiques)
Heuristique arborescente	Considère des solutions de permutation et non-permutation
	Exploration sélectionnée (ne pas explorer des nœuds ayant une borne inférieure $< (1 + \alpha)$LB0)
Étudier un nouveau objectif : minimiser le nombre d'arrêts de machines ayant une longueur maximale	

3.7 Conclusion

La revue de la littérature détaillée dans ce chapitre nous conduit à plusieurs conclusions. Premièrement, la résolution des problèmes de flow shop hybride est difficile. Les algorithmes exacts offrent des solutions optimales pour des problèmes de taille limitée qui ne dépasse pas une vingtaine de jobs en une heure de calcul. Les méthodes approchées, dont les algorithmes génétiques, ont été largement utilisés dans la littérature vu son efficacité à résoudre des problèmes proches du notre. Ceci permet la résolution des instances plus larges dans un temps de calcul très raisonnable.
Deuxièment, la contrainte de time lag maximal n'a pas retenu beaucoup d'attention dans la recherche. Quant à la contrainte de setup par groupe de jobs qui augmente la difficulté de la résolution, elle n'a pas été utilisée simultanément avec les autres contraintes.
Troisièmement, puisque aucun problème de la littérature ne correspond au problème traité dans cette thèse, nous proposons dans le chapitre qui suit deux MIPs pour résoudre FH2 (PD2, Pm)$|ST_{f,sd}, l_i|C_{max}$. Nous allons étendre la formulation proposée par [Yalaoui et al., 2013]. L'aspect relatif au time lag maximal sera pris en considération. La notion de « group setup » sera intégrée également.
Enfin, cette synthèse de revue, dans laquelle la plupart des auteurs ont montré la performance des méta-heuristiques, nous encourage à proposer aussi une méta-heuristique pour résoudre ce problème industriel. Cependant, puisqu'il était très difficile de trouver un benchmark commun, nous avons généré nos propres instances en s'inspirant d'un cas réel et des benchmarks académiques.

Chapitre 4

Modélisation mathématique et résolution

Sommaire

Pour résoudre le problème considéré, nous proposons tout d'abord de le modéliser par des modèles linéaires en nombres entiers. Deux modèles sont proposés pour être comparés.
Nous allons lancer la résolution à l'aide du solveur CPLEX où une borne inférieure sera intégrée afin d'accélérer la résolution. Nous allons pour cela tester cinq bornes : la première est intuitive et les autres sont une extension des bornes de la littérature.

4.1 Modélisation mathématique

Dans cette section, nous proposons 2 versions de PLNE inspirées de [Kahraman et al., 2008] et [Yalaoui et al., 2013]. Comme mentionné dans la section 3.6 du chapitre 3, nous avons ré-utilisé les mêmes variables de décision proposées par [Kahraman et al., 2008] et la matrice binaire proposée par [Yalaoui et al., 2013]. De plus, nous avons ajouté deux précisions concernant le setup par groupe et le time lag. Ceci est apporté dans les contraintes 10 et 12 du MIP 1 et les contraintes 22 et 23 du MIP 2.

- Variables de décision

$\mathbf{t}_{i,k}$: date de début du traitement du job i sur l'étage k
C_{max} : Makespan ou date de fin de traitement de tous les jobs

$x_{ij}^{m,k} = \begin{cases} 1 \text{ si job j est exécuté immédiatement après le job i sur la machine m de l'étage k} \\ 0 \text{ sinon} \end{cases}$

$a_{i}^{m,k} = \begin{cases} 1 \text{ si job i est affecté à la machine m de l'étage k} \\ 0 \text{ sinon} \end{cases}$

Il convient de signaler que si k=1, $a_i^{m,1}$ est fixé pour respecter la pré-affectation aux machines dédiées. Dans ce qui suit, nous présentons deux versions de modèles mathématiques pour minimiser le makespan. Ces modèles sont différents au niveau de certaines contraintes. L'objectif est de choisir le meilleur. Dans ces modèles, on ajoute deux jobs fictifs : le job "0" et le job "n+1"

4.1.1 1erModèle mathématique pour minimiser C_{max}

$$Min \quad C_{max}$$

$$S.C$$

$$\sum_{i=1}^{n+1} x_{i0}^{m,k} = 0, \qquad m \in M_k, k \in \{1,2\}, \tag{1}$$

$$\sum_{i=0}^{n} x_{n+1\ i}^{m,k} = 0, \qquad m \in M_k, k \in \{1,2\}, \tag{2}$$

$$x_{ij}^{m,k} + x_{ji}^{m,k} \leq a_i^{m,k} i = 1..n, j = 1..n, m \in M_k, k \in \{1,2\}, i \neq j \quad (3)$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij}^{m,k} = a_j^{m,k} \qquad \forall j = 1...n+1, m \in M_k, k \in \{1,2\} \quad (4)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n+1} x_{ij}^{m,k} = a_i^{m,k}, \qquad \forall i = 0...n, m \in M_k, k \in \{1,2\} \quad (5)$$

$$a_0^{m,k} = 1, \qquad \forall m \in M_k, k \in \{1,2\} \quad (6)$$

$$a_{n+1}^{m,k} = 1, \qquad \forall m \in M_k, k \in \{1,2\} \quad (7)$$

$$a_i^{m,1} = 1, \qquad \forall i = 1..n, m = A_i \quad (8)$$

$$\sum_{m=1}^{m_2} a_i^{m,2} = 1 \qquad i = 1...n, \quad (9)$$

$$t_{j,k} \geq t_{i,k} + P_{i,k} - M(1 - x_{ij}^{m,k}) + S_{j,k}.B_{ij}, \quad i = 0..n, j = 1..n, m \in M_k, k \in \{1,2\}, i \neq j \quad (10)$$

$$t_{i,2} \geq t_{i,1} + P_{i,1}, \qquad i = 1...n \quad (11)$$

$$t_{i,2} \leq t_{i,1} + P_{i,1} + l_{i,1}, \qquad i = 1...n \quad (12)$$

$$C_{max} \geq t_{i,2} + P_{i,2}, \qquad i = 1...n \quad (13)$$

$$x_{ij}^{m,k} \in \{0,1\}, m \in M_k, i = 1...n, j = 1...n, k \in \{1,2\}, i \neq j \quad (14)$$
$$a_i^{m,k} \in \{0,1\}, m \in M_k, i = 1...n, k \in \{1,2\} \quad (15)$$
$$C_{max}, t_{i,k} \geqslant 0, i = 1...n, k \in \{1,2\} \quad (16)$$

M : un nombre très grand, nous avons fixé M à $\sum_{i=1}^{n}(P_{i,1} + P_{i,2})$

La contrainte (1) impose un job fictif 0 qui sera le premier job à exécuter et aucun job ne pourra le précéder.

La contrainte (2) qui impose aussi un dernier job fictif $n+1$ et aucun job ne pourra lui succéder.

La contrainte (3) exige qu'un job ne peut pas être un prédécesseur et un successeur à la fois et ceci pour chaque machine et pour chaque étage.

Les contraintes (4) et (5) expriment qu'un job i affecté à une machine m de l'étage k a un seul successeur et un seul prédécesseur. S'il n'est pas affecté à cette machine alors on ne parle plus d'un prédécesseur et successeur car $a_i^{m,k}$=0.

Les contraintes (6), (7), (8) et (9) expriment les contraintes d'affectation qui permettent de déterminer l'affectation de chaque job aux machines de l'étage 1 et l'étage 2. En effet, la variable $a_i^{m,k}$ ne vaut 1 que si un job i est affecté à la machine m de l'étage k.
La contrainte (6) affecte un premier job fictif "0" à chaque machine de chaque étage.
La contrainte (7) affecte un le job fictif "n+1" à chaque machine de chaque étage.
La contrainte (8) exprime l'affectation des jobs aux machines dédiées de l'étage 1. La variable $a_i^{m,1}$ détermine la pré-affectation de job i à la machine m de l'étage 1.
La contrainte (9) affecte un job i à une seule machine de l'étage 2.
La contrainte (10) est relative aux dates de début de traitement. Un job affecté à une machine ne peut commencer son traitement que si le job qui le précède sur la même machine termine son traitement.
Les contraintes (11), (12) présentent les contraintes technologiques. Elles décrivent le déroule- ment des opérations sur les machines de chaque étage.
La contrainte (11) indique que les opérations sur l'étage 2 ne peuvent débuter que lorsque les opérations de premier étage ont terminé leurs traitements.
La contrainte (12) indique que les jobs sont autorisées à attendre un délai maximal avant de commencer le traitement sur le second étage.
La dernière contrainte (13) représente la fonction objectif C_{max} qui doit être supérieure ou égal à la fin de traitement de tous les jobs.
Finalement, la contrainte (14) et (15) déclarent que $x_{ij}^{m,k}$ et $a_i^{m,k}$ sont binaires. La contrainte (16) exprime que $C_{max}, t_{i,k}$ sont toujours positifs.

4.1.2 2ème Modèle mathématique pour minimiser C_{max}

Dans cette partie, nous présentons un second modèle mathématique inspiré du travail de [Yalaoui et al., 2013]. Dans ce modèle, les contraintes (17), (18), (19), (20) et (21) remplacent respectivement les contraintes (1), (2), (3), (4), (5), (6), (7), (8) et (9).

$Min \quad C_{max}$

$S.C$

$$\sum_{i=1}^{n} x_{i0}^{m,k} \leq 1, \qquad m \in M_k, k \in \{1,2\} \tag{17}$$

$$\sum_{i=1}^{n} x_{0i}^{m,k} \leq 1 \qquad m \in M_k, k \in \{1,2\} \tag{18}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{m=1}^{m_k} x_{ij}^{m,k}.Y_j^{m,k} = 1, \qquad j = 1...n, k \in \{1,2\} \tag{19}$$

$$\sum_{\substack{j=0 \\ j \neq i}}^{n} \sum_{m=1}^{m_k} x_{ij}^{m,k}.Y_i^{m,k} = 1, \qquad i = 1...n, k \in \{1,2\} \tag{20}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij}^{m,k}.Y_j^{m,k} - \sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ji}^{m,k}.Y_i^{m,k} = 0, \qquad j = 1...n, m \in M_k, k \in \{1,2\} \tag{21}$$

$$t_{j,k} \geq t_{i,k} + P_{i,k} - M(1 - x_{ij}^{m,k}) + S_{j,k}.B_{ij}, \quad i = 0..n, j = 1..n, m \in M_k, k \in \{1,2\}, i \neq j \tag{22}$$

$$t_{i,2} \leq t_{i,1} + P_{i,1} + l_{i,1}, \qquad i = 1...n \tag{23}$$

$$t_{i,2} \geq t_{i,1} + P_{i,1}, \qquad i = 1...n \tag{24}$$

$$C_{max} \geq t_{i,2} + P_{i,2}, \qquad i = 1...n \tag{25}$$

$$x_{ij}^{m,k} \in \{0,1\}, m \in M_k, i = 1...n, j = 1...n, k \in \{1,2\}, i \neq j \tag{26}$$

$$C_{max}, t_{i,k} \geqslant 0, i = 1...n, k \in \{1,2\} \tag{27}$$

M : un nombre très grand, nous avons fixé M à $\sum_{i=1}^{n}(P_{i,1} + P_{i,2})$

Comme dans le premier modèle, les contraintes (17), (18), (19), (20) et (21) sont des contraintes disjonctives qui vont gérer les conflits entre les jobs en respectant à la fois les contraintes temporelles et les contraintes des ressources.

Les contraintes (17) et (18) assurent qu'au plus un seul job passe juste avant et après le job fictif 0.

Les contraintes (19) et (20) assurent que les machines n'exécutent pas des tâches simultanément et doivent passer sur des fenêtres de temps disjointes (soit un job i ensuite un job j ou un job j ensuite un job j).

La contrainte (21) assure que chaque job doit avoir au plus un seul prédécesseur et un seul successeur.

Dans ce modèle, nous gardons les mêmes contraintes que le premier modèle (22), (23), (24), (25), (26) et (27).

4.2 Bornes inférieures

4.2.1 Présentation des bornes

Dans cette partie, nous examinons deux bornes classiques connues de la littérature [Gupta et al., 1997] dans le cas de FH standard. Ensuite, nous les adapterons à notre problème.

1. ***LBs intuitives***
   - Borne inférieure 1 :

     LB_1= Max($\sum_{i\in J1} P_{i,1}, \sum_{i\in J2} P_{i,1}$)

   - Borne inférieure 2 :

     LB_2= Max($\sum_{i\in J1} P_{i,1}, \sum_{i\in J2} P_{i,1}$)+Min($\underbrace{P_{i,2}}_{i\in J}$)

     La première borne ignore l'étage 2. Nous avons constaté qu'elle est très faible. La deuxième borne est expliquée par une relaxation de la capacité des cabines de séchage, autrement dit, inexistence d'attente.

2. ***LBs améliorées***

Nous pouvons toujours améliorer la 2ième borne en ajoutant la somme des durées de setup des groupes présents sur chaque machine, comme si les jobs de même groupe passaient successivement et ne nécessitent qu'un seul setup à chaque fois. En effet, la durée de setup d'un groupe quelconque n'est calculée que, si et seulement si, il existe au moins un job appartenant à ce groupe.

- Borne inférieure 3 : Dans un premier temps, on peut commencer par écrire la LB pour chaque machine dédiée comme suit :

  LB_{311} : ($\sum_{j\in J1} P_{j,1} + \sum_{i=1}^{u1} \mathcal{S}_{i,1}.\delta_i$), tel que δ_i=1 ssi il $\exists$ au moins un job de groupe i

  LB_{312} : ($\sum_{j\in J2} P_{j,1} + \sum_{i=1}^{u2} \mathcal{S}_{i,2}.\delta_i$), tel que δ_i=1 ssi il $\exists$ au moins un job de groupe i

  LB_{31}= max (LB_{311},LB_{312}), Il est clair que LB_{31} est supérieure à LB_1.

  D'autre part, et comme dans la 1ère situation on considère la durée de séchage la plus courte sur étage 2 : Min ($P_{i,2}$)

  De cette manière, $LB_3 = LB_{31} + \mathbf{Min}\ (\underbrace{P_{i,2}}_{i\in J})$

- Borne inférieure 4 : Puisque chaque famille de job doit terminer son traitement sur les 2 étages, on propose LB_{41} et LB_{42} comme suit :

LB_{41}= LB_{311}+ Min ($P_{i,2}$), $\forall\, i \in J1$ (i passant uniquement sur M1)

LB_{42}= LB_{312}+ Min ($P_{i,2}$), $\forall\, i \in J2$ (i passe uniquement sur M2)

LB_4= **Max** (LB_{41},LB_{42})

— Borne inférieure 5 :

$$LB_5 = \text{Min } (\underbrace{P_{j,1} + S_{j,1}}_{j \in J}) + 1/\text{m} \sum_{j \in J} P_{j,2}$$

La borne LB_5 est connue dans la littérature. En effet, la durée moyenne des traitements est une borne inférieure pour les machines parallèles identiques ([Gupta et al., 1997]). A cette borne, nous avons ajouté la durée minimale des durées de traitement sur les machines dédiées.

4.2.2 Comparaison des bornes inférieures

Dans cette section, nous proposons de comparer les moyennes des LBs dans les 2 catégories de données :

- Dans la première catégorie 1 de donnée (tableau 4.1)

TABLE 4.1 – Comparaison des bornes LB_3, LB_4 et LB_5 dans la catégorie 1

n	**Moy(LB3)**	**Moy(LB4)**	**Moy(LB5)**	**Nbr_LB3 meilleur**	**Nbr_LB4 meilleur**	**Nbr_LB5 meilleur**
10	902,1	938,1	624,0	0	16	0
20	1484,8	1507,2	1100,0	0	14	0
50	3141,6	3146,8	2514,9	0	20	0
100	5792,0	5802,2	4942,8	0	17	0
200	11075,9	11077,8	9758,0	0	11	0

Selon le tableau 4.1, LB_4 soit elle domine LB_3 soit les deux bornes sont égaux. Ceci est signalé dans la colonne Nbr_LB_4_meilleur. De plus, il est clair que LB_4 domine LB_5.

Dans la deuxième catégorie 2 de données (tableau 4.2, 4.3 et 4.5)

Dans cette catégorie de données, les moyennes des LB sont calculées pour chaque famille de problème (n, n1) de chaque classe. Les résultats sont présentés dans les tableaux 4.2, 4.3 et 4.5.

TABLE 4.2 – Comparaison des bornes LB_3, LB_4 et LB_5 dans la classe 1

Classe 1							
n1=	**n**	**Moy(LB3)**	**Moy(LB4)**	**Moy(LB5)**	**Nbr_LB3 meilleur**	**Nbr_LB4 meilleur**	**Nbr_LB5 meilleur**
0,5n	10	186,70	207,67	123,7	0	23	0
	20	326,1	337,8	209,7	0	19	0
	50	637,5	641,3	529,5	0	18	0
	100	1221,7	1224,4	1045,5	0	14	0
	200	2261,2	2261,8	2061,0	0	4	0
0,6n	10	194,5	207,5	124,1	0	21	0
	20	346,5	358,0	229,2	0	16	0
	50	724,0	727,0	531,3	0	13	0
	100	1378,5	1380,0	1045,7	0	12	0
	200	2610,9	2611,8	2071,8	0	10	0
0,7n	10	218,0	232,7	123,7	0	17	0
	20	389,2	394,0	223,1	0	8	0
	50	843,7	845,9	537,2	0	6	0
	100	1575,7	1576,5	1047,1	0	8	0
	200	3048,4	3048,8	2072,0	0	5	0

Dans la classe 1 de données (tableau 4.2), LB_4 est supérieure ou égale à LB_3. De plus, LB_4 surpasse LB_5. On peut conclure que LB_4 est dominante aussi dans cette classe.

TABLE 4.3 – Comparaison des bornes LB_3, LB_4 et LB_5 dans la classe 2

Classe 2							
n1=	**n**	**Moy(LB3)**	**Moy(LB4)**	**Moy(LB5)**	**Nbr_LB3 meilleur**	**Nbr_LB4 meilleur**	**Nbr_LB5 meilleur**
0,5n	10	572,9	606,3	396,5	0	20	0
	20	907,7	912,9	689,1	0	16	0
	50	1839,9	1842,4	1587,8	0	12	0
	100	3405,6	3407,1	3094,8	0	11	0
	200	6405,8	6406,9	6075,2	0	9	0
0,6n	10	631,6	639,1	393,5	0	11	0
	20	1008,6	1018,2	695,2	0	14	0
	50	2106,6	2108,7	1589,8	0	9	0
	100	3958,3	3958,9	3086,6	0	7	0
	200	7578,8	7579,3	6085,0	0	10	0
0,7n	10	691,0	694,4	400,9	0	7	0
	20	1150,5	1156,0	693,9	0	12	0
	50	2412,9	2414,5	1589,2	0	8	0
	100	4577,2	4578,0	3111,4	0	10	0
	200	8768,0	8768,3	6078,0	0	5	0

Dans la classe 2 de données (tableau 4.3), nous observons le même résultat de la classe précédente. LB_4 est dominante aussi dans cette classe.

TABLE 4.4 – Comparaison des bornes LB_3, LB_4 et LB_5 dans la classe 3

Classe 3							
n1=	**n**	**Moy(LB3)**	**Moy(LB4)**	**Moy(LB5)**	**Nbr_LB3 meilleur**	**Nbr_LB4 meilleur**	**Nbr_LB5 meilleur**
0,5n	10	921,8	942,0	675,8	0	16	0
	20	1399,8	1414,0	1179,9	0	19	0
	50	5067,2	5069,9	2668,9	0	20	0
	100	5067,2	5069,9	5179,1	0	1	29
	200	9581,5	9582,8	10143,0	0	0	30
0,6n	10	1021,6	1032,0	669,6	0	12	0
	20	1569,8	1577,9	1165,8	0	12	0
	50	3212,8	3215,0	2680,5	0	13	0
	100	5953,4	5954,4	5188,5	0	12	0
	200	11346,5	11347,0	10182,5	0	8	0
0,7n	10	1099,5	1108,1	678,7	0	11	0
	20	1763,3	1764,5	1178,8	0	7	0
	50	3661,2	3662,7	2665,4	0	6	0
	100	6835,3	6835,9	5151,3	0	9	0
	200	13157,9	13158,5	10177,6	0	10	0

Dans la classe 3 de données (tableau 4.4), LB_4 est supérieure ou égale à LB_3. De plus, LB_4 surpasse LB_5 dans toutes les instances sauf pour les instances de 100 et 200 jobs où il y a un équilibre des charges entre les machines dédiées. Dans cette classe, on ne peut pas confirmer que LB_4 st dominante.

4.3 Évaluation de la meilleure borne

En se basant sur l'analyse précédente, nous allons nommer la meilleure borne LB_{best}.
LB_{best} = max (LB_3,LB_4, LB_5)

Pour étudier d'avantage la qualité LB_{best}, nous avons mené une large comparaison avec la borne obtenue par le solveur CPLEX à la racine. Dans ce qui suit, nous allons comparer cette borne par rapport à la borne générée par le solveur Cplex appelé LB_{Cplex} au démarrage (à la racine). Pour cela, nous allons calculer l'écart entre les deux bornes (E_{LB}) pour chaque instance de donnée puis l'écart moyen (pour chaque famille d'instances)

$$E_{LB} = \frac{(LB_{best} - LB_{Cplex})}{LB_{Cplex}} \times 100$$

$$M_{E_{LB}} = \frac{\sum E_{LB}}{30}$$

4.3.1 Comparaison de LB_{best} et LB_{Cplex} pour la catégorie 1 de jeu de données

Le tableau ci-dessous (tableau 4.5) présente la moyenne des écarts ($M_{E_{LB}}$) entre les LB_{Cplex} et LB_{best}, le minimum et le maximum des écarts pour chaque famille d'instances de la première catégorie de jeux de données.

TABLE 4.5 – Comparaison des LBs

n	$M_{E_{LB}}$	**max** (E_{LB})	**min** (E_{LB})
10	8,84	47,37	-12,06
20	70,86	114,57	38,19
50	247,19	327,34	203,64
100	*	*	*
200	*	*	*

Dans ce premier tableau (tableau 4.5), les valeurs négatives apparaissent dans les instances de 10 jobs, ce qui signifie que le solveur offre une borne inférieure meilleure que LB_{best}. Pour les instances de 20 et 50 jobs, LB_{best} surpasse LB_{Cplex} avec un écart qui dépasse 70%. Cependant, le solveur n'arrive pas à trouver une borne à la racine pour les instances de 100 et 200 jobs.
Il convient de signaler que nous avons arrêté les MIPs après une heure d'exécution.

4.3.2 Comparaison de LB_{best} et LB_{Cplex} pour la catégorie 2 de jeu de données

Les tableaux ci-dessous (tableaux 4.6, 4.7 et 4.8) présentent respectivement la moyenne des écarts ($M_{E_{LB}}$) entre les LB_{Cplex} et LB_{best}, le minimum et le maximum des écarts pour chaque classe d'instances de la catégorie 2 de jeux de données.

Classe 1 : $P_{i,1} \in [1, 40]; P_{i,2} \in [5, 200]$

TABLE 4.6 – Comparaison des LBs pour la classe 1 des instances

		Classe 1		
n1=	**n**	$M_{E_{LB}}$	**max** (E_{LB})	**min** (E_{LB})
0,5n	10	-6,04	39,19	-39,36
	20	46,76	86,94	25,10
	50	166,26	206,28	122,75
	100	*	*	*
	200	*	*	*
0,6n	10	-8,29	21,97	-32,38
	20	55,70	107,33	23,87
	50	203,82	271,62	140,00
	100	*	*	*
	200	*	*	*
0,7n	10	5,48	50,75	-31,69
	20	70,47	107,69	37,45
	50	255,14	330,51	208,26
	100	*	*	*
	200	*	*	*

Dans le tableau (tableau 4.6), les valeurs négatives apparaissent dans les instances de 10 jobs où n1= 0.5 n ; n1= 0.6 n ; n1= 0.7 n, ce qui signifie que le solveur offre une borne inférieure meilleure que LB_{best}. Pour les instances de 20 et 50 jobs, LB_{best} surpasse LB_{Cplex} avec un écart qui dépasse 50%. Cependant, le solveur n'arrive pas à trouver une borne à la racine pour les instances de 100 et 200 jobs.

Classe 2 : $P_{i,1} \in [40, 80]; P_{i,2} \in [200, 400]$

TABLE 4.7 – Comparaison des LBs pour la classe 2 des instances

		Classe 2		
n1=	**n**	$M_{E_{LB}}$	**max** (E_{LB})	**min** (E_{LB})
0,5n	10	0,8	45,3	13,2
	20	10,3	111,28	72,80
	50	282,0	326,94	262,32
	100	*	*	*
	200	*	*	*
0,6n	10	36,97	53,57	14,44
	20	116,73	139,63	97,52
	50	336,90	365,34	316,02
	100	*	*	*
	200	*	*	*
0,7n	10	48,22	66,04	31,81
	20	142,50	169,03	121,54
	50	400,81	437,11	377,27
	100	*	*	*
	200	*	*	*

Dans la classe 2 (tableau 4.7), nous observons un écart positif entre LB_{Cplex}et LB_{best}. Cela est expliqué par la faiblesse de LB_{Cplex}. Nous remarquons que le écart est moyennement faible pour les instances de 10 jobs alors qu'il dépasse 100% pour les instances de 20 et 50 jobs.

Classe 3 : $P_{i,1} \in [80, 100]; P_{i,2} \in [400, 600]$

TABLE 4.8 – Comparaison des LBs pour la classe 3 des instances

		Classe 3		
n1=	**n**	$M_{E_{LB}}$	**max** (E_{LB})	**min** (E_{LB})
0,5n	10	34,83	43,02	28,08
	20	100,61	113,70	82,33
	50	289,79	305,84	261,98
	100	*	*	*
	200	*	*	*
0,6n	10	50,34	97,88	31,72
	20	123,56	139,97	108,43
	50	350,19	364,45	316,18
	100	*	*	*
	200	*	*	*
0,7n	10	58,12	70,93	40,63
	20	150,76	164,55	137,74
	50	408,58	431,86	375,29
	100	*	*	*
	200	*	*	*

Le tableau 4.8 montre la supériorité de LB_{best} vis-à-vis LB_{Cplex} dans la classe 3 des instances. L'écart obtenu est moyennement grand pour les instances de 10 jobs tandis qu'il très grand (plus que 100%) pour les instances de 20 et 50 jobs. Ces observations confirment notre hypothèse que, d'une part, LB_{best} est une bonne borne inférieure. D'autre part, le solveur est incapable de fournir une borne de qualité au démarrage et ça va provoquer un ralentissement de l'exécution des MIPS.

4.4 Expérimentations numériques

Dans cette partie, nous allons tout d'abord comparer la taille des deux modèles proposés auparavant. Ensuite, nous allons présenter les résultats obtenus et les analyser.

4.4.1 Comparaison de la taille des deux modèles mathématiques

Le tableau ci-dessous (4.9) résume les tailles des deux modèles. Le modèle 1 a le même nombre des variables continues que le modèle 2. Cependant le modèle 2 a moins de variables binaires et moins de contraintes.

TABLE 4.9 – Comparaison de la tailles des MIPs

Modèle	Variables continues	Variables binaires	Nombre de contraintes
Modèle 1	$t_{i,k} : n \times 2$ C_{max}	$x_{ij}^{m,k} = \begin{cases} x_{ij}^{m,1} : n \times n \times \mid (M_1) \mid \\ x_{ij}^{m,2} : n \times n \times \mid (M_2) \mid \end{cases}$ $a_i^{m,k} : n \times \mid M_2 \mid$	(6+ $2n^2$ + n) $\times$ ($\mid M_1 \mid + \mid M_2 \mid$) + 5n
Modèle 2	$t_{i,k} : n \times 2$ C_{max}	$x_{ij}^{m,k} = \begin{cases} x_{ij}^{m,1} : n \times n \times \mid (M_1) \mid \\ x_{ij}^{m,2} : n \times n \times \mid (M_2) \mid \end{cases}$	(2+n+ n^2) $\times$ ($\mid M_1 \mid + \mid M_2 \mid$)+ 7n

Dans ce qui suit, nous comparons les bornes de relaxation continues afin d'évaluer la qualités des deux modèles présentées auparavant.

4.4.2 Comparaison des bornes de relaxation

Les tableaux 4.10 et 4.11 présentent les bornes inférieures trouvés par MIP1 et MIP2 relaxés (MIPR 1 et MIPR 2) dans les deux catégorie de données.

Dans la catégorie 1 de données

TABLE 4.10 – Les bornes inférieures des MIP 1 et MIP 2 relaxés dans la catégorie 1

n	**Moy(LB)**	**durée**	**Moy(LB)**	**durée**	**nbre fois LB MIPR 1 meilleur**	**nbre fois LB MIPR 2 meilleur**
10	839,4	0,0	839,5	0,0	0	1
20	863,7	0,0	863,7	0,0	0	0
50	890,2	1,0	890,4	0,0	0	2

Dans le tableau 4.10, les bornes trouvées par les deux modèles relaxés sont quasiment les mêmes avec une très faible supériorité de MIP2. En effet, MIPR 2 a trouvé seulement 3 bornes meilleures que MIPR 1.

Dans la catégorie 2 de données

Dans le tableau ci-dessous (4.11), les bornes trouvées par les deux modèles relaxés (MIPR) sont :
- Les mêmes pour la classe 1
- Les mêmes pour la classe 2 sauf pour les instances de 50 jobs où on trouve que le MIP 2 est meilleur que MIP 1 dans une seule instance.
- Dans la classe 3, le MIP 2 surpasse le MIP 1 dans 6 instances pour lesquelles il génère des bornes meilleures. En observant le temps d'exécution, il est clair que le MIP 2 est légèrement rapide que le MIP 1 dans la génération des bornes de relaxation.

TABLE 4.11 – Les bornes inférieures des MIP 1 et MIP 2 relaxés dans la catégorie 2

Classe 1							
n1=	**n**	**Moy(LB)**	**durée**	**Moy(LB)**	**durée**	**nbre fois LB MIPR 1 meilleur**	**nbre fois LB MIPR 2 meilleur**
0,5n	10	207,5	0,1	207,5	0,1	0	0
	20	216,0	2,8	216,0	1,0	0	0
	50	226,4	15,3	226,4	7,0	0	0
0,6n	10	212,0	0,0	212,0	0,0	0	0
	20	215,2	1,1	215,2	0,2	0	0
	50	225,0	10,7	225,0	8,2	0	0
0,7n	10	205,9	0,0	205,9	0,0	0	0
	20	216,2	1,1	216,2	0,6	0	0
	50	224,2	12,0	224,2	6,5	0	0
Classe 2							
n1=	**n**	**Moy(LB)**	**durée**	**Moy(LB)**	**durée**	**nbre fois LB MIPR 1 meilleur**	**nbre fois LB MIPR 2 meilleur**
0,5n	10	446,1	0,6	446,1	0,0	0	0
	20	458,8	1,1	458,8	0,6	0	0
	50	464,4	10,1	464,4	6,5	0	0
0,6n	10	452,0	0,0	452,0	0,0	0	0
	20	452,9	1,0	452,9	0,1	0	0
	50	464,5	10,8	464,5	7,1	0	0
0,7n	10	450,3	0,1	450,3	0,0	0	0
	20	458,0	1,1	458,0	0,6	0	0
	50	463,9	10,0	464,0	6,4	0	1
Classe 3							
n1=	**n**	**Moy(LB)**	**durée**	**Moy(LB)**	**durée**	**nbre fois LB MIPR 1 meilleur**	**nbre fois LB MIPR 2 meilleur**
0,5n	10	675,6	0,0	675,7	0,0	0	1
	20	683,8	1,1	683,8	0,7	0	0
	50	690,5	9,4	690,6	6,4	0	1
0,6n	10	668,0	0,0	668,0	0,0	0	0
	20	682,9	1,0	682,9	1,1	0	0
	50	690,2	10,3	690,3	8,0	0	1
0,7n	10	674,0	0,0	674,0	0,0	0	0
	20	683,9	1,0	684,0	1,3	0	1
	50	689,9	11,7	690,1	6,7	0	2

4.4.3 Résultats numériques des MIPs : Catégorie 1

Dans cette section, nous présentons des tableaux comparatifs des MIPs et LBs pour les 2 catégories des problèmes. Comme mentionné dans l'introduction, nous avons intégré une borne inférieure pour accélérer le MIP qui est LB_{best}. Au cours de sa progression, le solveur génère une borne inférieure (LB_{cplex}) et une borne supérieure (UB_{cplex}) jusqu'à ce qu'il arrive à une solution optimale. La borne LB_{cplex} peut rester la même que LB_{best} comme elle peut être meilleure que LB_{best}. Dans le cas où le MIP ne génère pas une solution optimale, nous l'arrêtons après 3600 sec et nous calculons l'écart relatif ($ER\%$: relative deviation) pour chaque famille de problème comme suit :

$$ER(\%) = \frac{UB_{cplex} - LB_{cplex}}{LB_{cplex}} \times 100$$

Dans le tableau 4.12, la colonne 1 présente la famille d'instance, la colonne 2 et 3 présentent respectivement la moyenne des ER (MER) et la durée moyenne d'exécution (MT).
Pour cette première catégorie de problèmes, les résultats générés par les 2 MIPs (voir tableau 4.12) sont les mêmes pour les instances de 10 jobs où l'écart trouvé est nul. Ceci montre que la résolution est optimale. D'autre part, le temps d'exécution est quasiment le même. Cependant, le MIP 2 surpasse le MIP 1 de presque 10% dans la résolution des instances de 20 jobs. Le MIP 2 a réussi à trouver des solutions de bonne qualité avec un écart faible qui ne dépasse pas 6%. Les instances de 50 jobs et plus présentent des problèmes difficiles à résoudre, ce qui explique l'absence de la résolution après une heure d'exécution.

TABLE 4.12 – La moyenne de $ER(\%)$

n	MER MIP1	MT CPLEX	MER MIP2	MT CPLEX
10	0.00	134.83	0.00	43.87
20	15.12	3600.03	5.63	3601.33

TABLE 4.13 – Nombre d'instances résolues à l'optimalité par les deux modèles dans la catégorie 1

Nombre d'instances résolues à l'optimalité		
n	MIP 1	MIP 2
10	30	30
20	2	2

Comme le montre le tableau 4.13 , les deux modèles sont capables de résoudre toutes les instances de 10 jobs à l'optimalité. Cependant, ils trouvent seulement deux solutions optimales pour les instance de 20 jobs.

4.4.4 Résultats numériques des MIPs : Catégorie 2

En examinant les résultats trouvés dans le tableau 4.14, nous avons constaté que les deux modèles mathématiques donnent des résultats identiques pour les instances de petite taille (10

job) en marquant un écart nul, autrement dit la résolution est optimale.
D'autre part, nous remarquons que les résultats générés par les 2 MIPs sont proches en ce qui concerne la 1ère classe. Cependant, le MIP 1 surpasse le MIP 2 dans la résolution des instances de 20 jobs où n1=0.5n et n1=0.6n.
En ce qui concerne la 2ème et la 3ème classe, le MIP 2 surpasse le MIP 1 en réalisant des écarts faibles (inférieur à 6%).
En outre, nous pouvons comparer aussi la performance des MIPs en examinant 2 facteurs à savoir : la rapidité de résolution déterminée par le temps d'exécution et le écart trouvé. De ce fait, le deuxième MIP est considéré plus performant car il a mis moins de temps que le premier MIP à résoudre la plupart des instances, tout en gardant un écart plus faible.
Néanmoins, en augmentant la taille de problèmes, nous trouvons que les 2 MIPs sont incapables de résoudre les grandes instances de 50 jobs et plus, ce qui est conforme avec la littérature.

TABLE 4.14 – Comparaison des résultats numériques trouvés par MIP1 et MIP2

Class1					
n1=	**n**	**MER MIP1**	**MT**	**MER MIP2**	**MT**
0,5n	10	0,00	21.14	0,00	19,17
	20	10,32	3611,92	12,70	3601,00
0,6n	10	0,00	55,63	0,00	34,97
	20	10,04	3411,17	12,84	3326,80
0,7n	10	0,00	144,70	0,00	20,37
	20	14,40	3608,80	13,26	3602,10
Classe 2					
n1=	**n**	**MER MIP1**	**MT**	**MER MIP2**	**MT**
0,5n	10	0,00	18,53	0,00	11,51
	20	12,05	3681,23	3,18	3210,09
0,6n	10	0,00	4,87	0,00	5,33
	20	5,42	3576,83	3,89	3103,81
0,7n	10	0,00	13,93	0,00	26,13
	20	7,48	3693,83	5,88	3590,57
Classe 3					
n1=	**n**	**MER MIP1**	**MT**	**MER MIP2**	**MT**
0,5n	10	0,00	0,67	0,00	0,73
	20	11,89	3605,00	4,39	3585,77
0,6n	10	0,00	1,23	0,00	1,47
	20	3,84	3605,00	1,85	3276,66
0,7n	10	0,01	13,33	0,00	4,00
	20	4,72	3605,00	3,23	3486,66

Le tableau 4.15 détermine le nombre d'instances résolues à l'optimalité par chaque modèle. Nous remarquons que les deux modèles sont incapables de résoudre un grand nombre d'instances à l'optimalité sauf les instances de petite taille, celles de 10 jobs. D'autre part, nous remarquons que les deux modèles ont résolu le même nombre d'instances dans la classe 1. Cependant, le modèle 2 surpasse le modèle 1 dans la classe 2 et 3 où nous trouvons un nombre plus grand des solutions optimales. Ceci confirme les résultats trouvés dans le tableau 4.14 où nous avons remarqué que le modèle 1 surpasse le modèle 2 en générant des solutions de meilleure qualité pour la classe 1 (où n1=0.5 n et n1=0.6 n).

TABLE 4.15 – Nombre d'instance résolues à l'optimalité par les deux modèles

Nombre d'instances resolues à l'optimalité							
n1=	**n**	**Classe 1**		**Classe 2**		**Classe 3**	
		MIP 1	MIP 2	MIP 1	MIP 2	MIP 1	MIP 2
0,5n	10	30	30	30	30	30	30
	20	0	0	0	2	0	1
0,6n	10	30	30	30	30	30	30
	20	2	2	1	2	0	3
0,7n	10	30	30	30	30	30	30
	20	0	0	0	1	0	1

4.5 Conclusion

Dans ce chapitre, nous avons présenté des bornes inférieures et 2 modélisations mathématiques pour le problème étudié. Nous avons étudié la performance de la borne retenue par rapport à la borne de solveur. Par la suite, nous avons testé les 2 modèles linéaires avec le solveur CPLEX. Les 2 MIPs ont offert des solutions réalisables dans une heure de calcul pour les instances de 10 et 20 jobs. En ce qui concerne les instances de 50 jobs, les 2 MIPs donnent des solutions de mauvaise qualité avec un écart très grand (par rapport à la borne inférieure initiale LB_{best} qui dépasse 80 %, pour cela, nous ne l'avons pas présenté dans le chapitre).
D'autre part, les 2 MIPs étaient incapables de trouver des solutions pour les problèmes de 50 jobs et plus. De ce fait, nous avons présenté uniquement les résultats des MIPs pour les instances de 10 et 20 jobs. Ce chapitre a montré les limites de la première méthode de résolution et nous encourage à entamer une deuxième voie de recherche.

Chapitre 5

Méthodes approchées : algorithmes évolutionnistes

Sommaire

Après avoir présenté deux modélisations mathématiques qui permettent de résoudre, de manière exacte, le problème étudié, nous proposons ici une méthode de résolution approchée de type algorithme génétique. Ce choix est motivé par la complexité du problème étudié. En effet, les méthodes exactes ne permettent pas de résoudre des instances de taille importante dans des durées raisonnables. Ce chapitre vise alors à résoudre les instances de grande taille et obtenir des solutions de bonne qualité dans un temps de calcul acceptable.

5.1 Introduction

Dans la littérature, plusieurs méta-heuristiques ont été proposées dans le but de résoudre les problèmes de flow shop hybride. Dans ce travail, nous avons choisi les algorithmes génétiques (AGs) pour deux raisons principales. La première, les algorithmes génétiques se sont avérés efficaces avec plusieurs problèmes proches tel que présenté dans l'état de l'art. En effet, nous trouvons plusieurs auteurs affirmant l'efficacité des AGs dans la résolution du FH avec des différentes contraintes, par exemple [Reeves, 1995], [Ruiz and Maroto, 2006], [Kahraman et al., 2008], [Jolai et al., 2009], [BESBES et al., 2010], [Sioud et al., 2012, Sioud et al., 2013], [Wang and Liu, 2013a], [Jun and Park, 2015]. Ceci nous a motivé à choisir cette approche de résolution.
La seconde raison, les AGs comme les autres méta-heuristiques ne sont soumis à aucune hypothèse vis-à-vis la nature des fonctions à optimiser et peuvent donc théoriquement traiter tous types de problèmes.
Les AGs ont été introduits pour la première fois par [Holland, 1975] en 1975, puis par [Goldberg and Deb, 1991]. Ils sont inspirés de la théorie de l'évolution naturelle développée par Darwin.
L'algorithme génétique fait partie des algorithmes évolutionnaires qui se basent sur un ensemble de plusieurs solutions simultanément. Il fait évoluer une population de solutions progressivement par un processus de reproduction constitué des opérateurs de sélection, de croisement et de mutation pour créer de nouvelles populations avec de meilleurs individus. Ce processus est répété à travers les générations jusqu'à obtenir une population finale avec des individus d'une meilleure qualité et qui s'approchent au mieux de la solution optimale. La figure 5.1 présente le fonctionnement général d'un algorithme génétique.

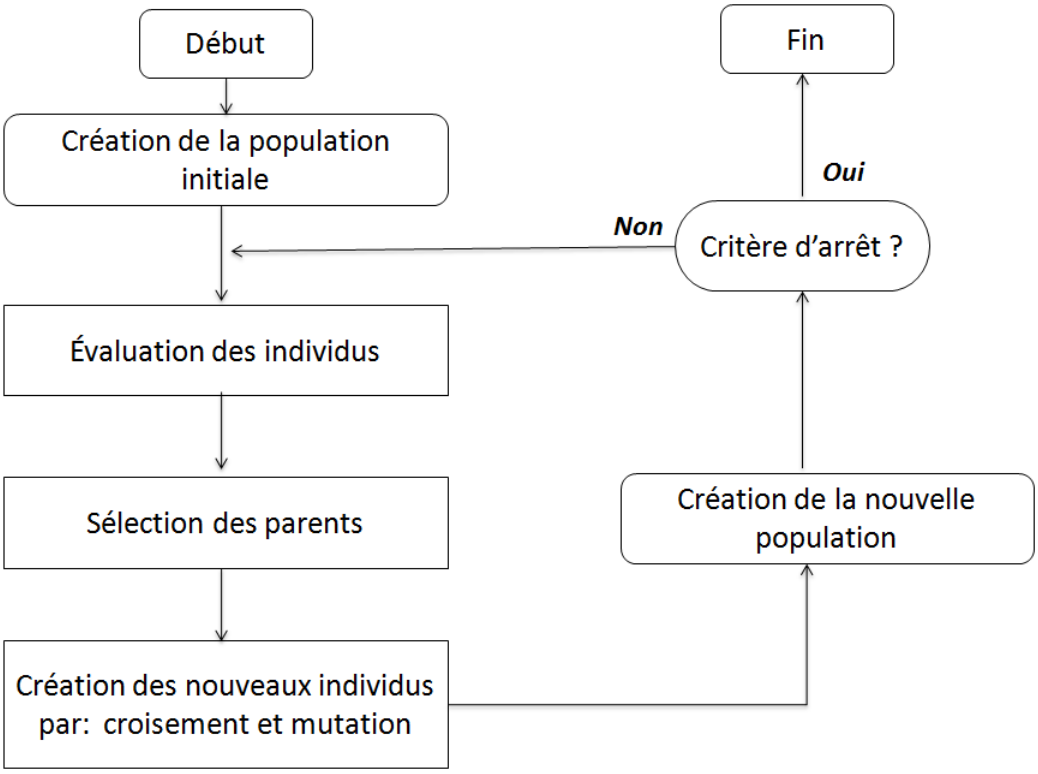


FIGURE 5.1 – Fonctionnement général de l'algorithme génétique

Dans ce chapitre, nous présentons tout d'abord un premier algorithme génétique (AG1) où un individu est représenté par une seule permutation qui définit un ordre des jobs sur le deuxième étage (E2). Puis un deuxième algorithme génétique (AG2) où chaque individu est représenté par une permutation qui définit un ordre de jobs sur le premier étage (E1). Dans les deux algorithmes, nous proposons une procédure pour la détermination de la fonction objectif.
Pour chaque AG, nous décrivons, tout d'abord, le codage de chaque individu, la construction de la population initiale, la fonction d'évaluation, la sélection des individus parents pour la reproduction, puis les opérateurs de reproduction (croisement et mutation). Ensuite, nous présentons une procédure de recherche locale afin d'explorer un espace de recherche plus large.

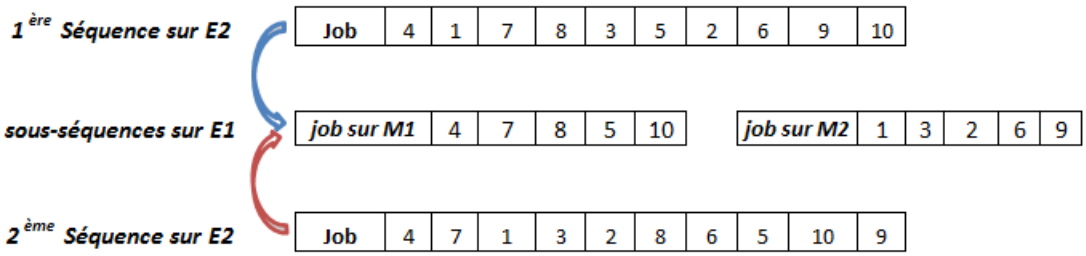
5.2 Algorithme génétique I

5.2.1 Codage de la solution

Dans ce premier algorithme, chaque individu est constitué d'une permutation de n gènes(n étant le nombre de jobs) qui représente la séquence de jobs sur l'étage 2 suivant un ordre non décroissant des dates de début de traitement sur cet étage. Nous avons adapté la technique de codage en suivant la proposition de [Haouari et al., 2006] " *This permutation is simply obtained by ranking the jobs according to the non-decreasing order of their starting times*". Ce type de codage signifie qu'on va automatiquement utiliser First Available Machine (FAM). Ceci est justifié par le fait que les machines parallèles sont identiques. Dans le cas contraire, si les machines ne sont pas identiques ou bien ne sont pas toutes disponibles alors on devrait tester toutes les machines

parallèles [Sioud et al., 2013].
La figure 5.2 représente le codage d'une solution composée de 10 jobs. Elle montre que pour deux permutations différentes sur l'E2 correspond une seule permutation sur l'E1.

FIGURE 5.2 – AG1 : Codage de solution : exemple



5.2.2 Génération de la population initiale

La performance de l'algorithme génétique est, entre autres, conditionnée par le choix de la population initiale. L'intégration de solutions de bonne qualité donne plus de chance à l'algorithme de converger rapidement vers des solutions meilleures. Ainsi, faire intégrer des individus générés par des heuristiques présente une voie prometteuse afin d'améliorer la qualité de la population initiale. Pour ce faire, nous avons recouru à une population initiale basée sur 3 groupes d'individus. Chaque groupe est généré différemment comme décrit ci-dessous :

1. **Groupe 1** : ce groupe est composé de 6 individus résultants des règles de priorité

   — SPTS1 : "Smallest Processing Time on stage 1" : les jobs sont ordonnés selon la plus courte durée de traitement sur l'E1.

   — SPTS2 : "Smallest Processing Time on stage 2" : les jobs sont ordonnés selon la plus courte durée de traitement sur l'E2.

   — LPTS1 : "Longuest Processing Time on stage 1" : les jobs sont ordonnés selon la plus longue durée de traitement sur l'E1.

   — LPTS2 : "Longuest Processing Time on stage 2" : les jobs sont ordonnés selon la plus longue durée de traitement sur l'E2.

   — ITPT : "Increasing Total Processing Time " : les jobs sont ordonnés selon la durée totale de traitement sur l'E1 et l'E2 (ordre ascendant).

   — DTPT : "Decreasing Total Processing Time " : les jobs sont ordonnés selon la durée totale de traitement sur l'E1 et l'E2 (ordre descendant).

2. **Groupe 2** : ce groupe est basé sur la minimisation du nombre des setup à l'E1. Le principe est d'utiliser différentes séquences de groupes pour générer les permutations des jobs (voir **Algorithme 1**). En effet, on va lancer tous les jobs du même groupe passant sur la même machine de façon consécutive afin de minimiser le temps de setup. Le choix de cette

sous-population est motivé par les observations de la structure des solutions optimales générées par les MIPs.

Algorithme 1 Heuristique H1

```
1: π_job : tableau d'entier /* une permutation de job initialement vide */
2: π_g : tableau d'entier /* une permutation de groupe */
3: n : entier        /* nombre de jobs */
4: j : entier        /* numéro du job */
5: i : entier        /* indice du groupe */
6:
   Début
7:     Pour (i de 0 à longueur(π_g)) faire
8:       Pour j de 0 à n faire
9:         Si (le job j ∈ groupe π_g[i]) alors
10:          ajouter le job j à π_job
11:        Fin si
12:      Fin Pour
13:    Fin Pour
   Fin
14:
```

3. **Groupe 3** : ce groupe d'individu est basé sur une génération aléatoire des individus.

5.2.3 Évaluation : calcul de Fitness

La fitness est la fonction d'évaluation de l'individu. Nous nous intéressons dans ce chapitre à la minimisation de makespan C_{max}. Dans ce premier algorithme C_{max} est calculé en considérant l'affectation aux machines parallèles. En effet, nous affectons les jobs dans le même ordre de leur apparition dans la permutation sur l'étage 2 (c'est la définition même de la permutation à l'E2). Ensuite, on ajuste le temps de début de traitement sur chaque étage. Cet ajustement est basé sur le fait qu'un job, ayant terminé son traitement sur E1, ne peut commencer son traitement sur E2 que si une machine parallèle est disponible (en considérant le time lag). Sinon, son lancement sur E1 est retardé.

5.2.4 Opérateurs génétiques : sélection, croisement et mutation

L'application des opérateurs génétiques (la sélection, le croisement et la mutation) nous permet de construire la nouvelle génération comme le montre la figure 5.1. Cependant, la qualité des nouveaux descendants dépend du choix de type d'opérateur.

1. **La sélection et le remplacement**
   Cet opérateur permet de se déplacer dans l'espace de recherche et de choisir des individus. Dans ce cadre, nous faisons la différence entre la sélection de reproduction et le remplacement qui représente une autre forme de sélection.

— **La sélection de reproduction** : deux individus parents sont choisis pour être croisés. Il existe différents opérateurs de sélection dont nous présentons les deux utilisés dans ce mémoire :

— aléatoire : ce type de sélection se base sur le hasard. Les parents sont choisis aléatoirement ;

— par roulette de casino : c'est un mécanisme proportionnel à la fitness (un individu qui a une meilleure fitness a plus de chance d'être sélectionné ;

— **Le remplacement** : Cette opération permet de faire évoluer la population d'une génération à une autre. Des individus sont sélectionnés pour être intégrés à la nouvelle génération.

Le remplacement est une opération élitiste. Nous avons fixé la taille de la population initiale à $taille_{pop}$. Ensuite, 50% des meilleures solutions seront transférées d'une génération à une autre. Il convient de signaler que les 50% d' individus sont sélectionnés de l'ensemble de la population des parents et des enfants.

2. **Croisement**

Le croisement est le principal opérateur qui produit les nouveaux individus et qui favorise l'intensification de la population. Classiquement, les opérateurs de croisement génèrent deux enfants à partir d'une paire de parents. L'idée principale consiste à échanger des parties complémentaires de gênes des parents dans le but de donner des nouveaux individus (fils) portant des propriétés héritées des deux parents. La création d'une paire d'enfants à partir d'une paire de parents est effectuée avec une probabilité de croisement P_{crois} qui sera présentée dans le paragraphe (5.5).
En outre, et en se basant sur le chapitre État de l'art, nous avons choisi de nous appuyer sur les travaux de [Ruiz and Maroto, 2006, Sioud et al., 2012] et nous avons testé les deux techniques : SJOX (Similar Job Order Crossover) proposé par [Ruiz and Maroto, 2006] et RMPX (Random Maximal Preservative Crossover) proposé par [Sioud et al., 2012]. Les étapes des deux opérateurs sont comme suit :

— Le fonctionnement de SJOX :
La première étape consiste à copier les jobs communs (même position) des deux parents chez les fils et à la même position. Par la suite, il faut créer un point de croisement qui ne doit pas dépasser le nombre total de jobs. Les positions vides de chaque fils jusqu'au point de croisement seront hérités directement du parent correspondant. L'étape suivante lance une recherche des éléments manquants chez les fils et seront copiés dans l'ordre relatif de leur apparition chez l'autre parent. Il faut signaler que s'il n'y a pas de jobs similaires en même position, l'opérateur de croisement proposé va se comporter comme un opérateur de croisement à 1 point (croisement classique). Les figures 5.3, 5.4 et 5.5 illustrent le fonctionnement de SJOX.

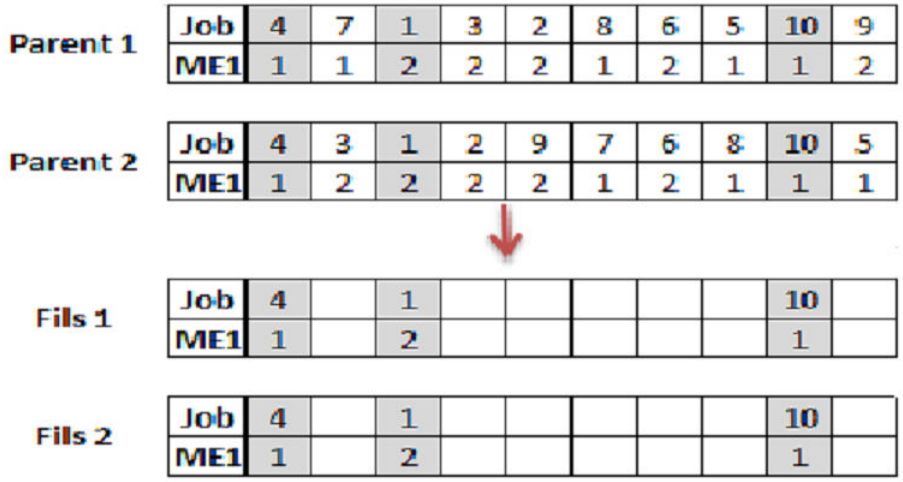


FIGURE 5.3 – SJOX : étape 1

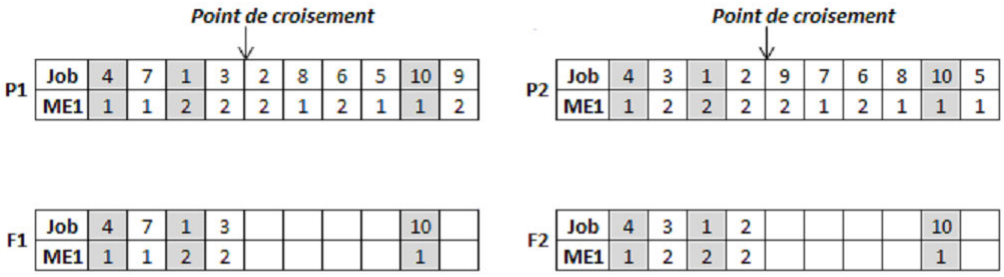


FIGURE 5.4 – SJOX : étape 2

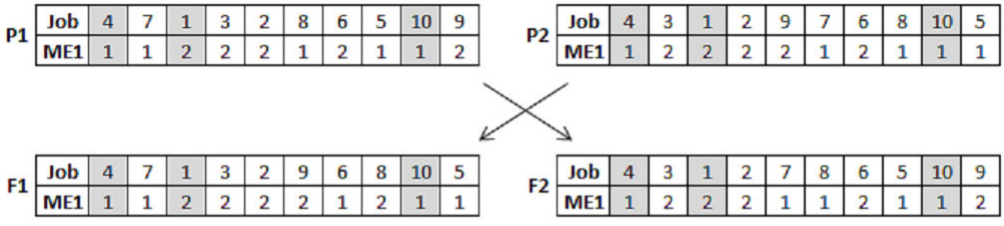


FIGURE 5.5 – SJOX : étape 3

— Le fonctionnement de RMPX :
Cette technique consiste à générer aléatoirement 2 points de croisement C1 et C2, ainsi

qu'un point d'insertion p_i de l'intervalle [1, n - (C2 - C1)]. La partie [C1-C2] de premier parent sera inséré au point p_i de fils1. Le reste des jobs seront hérités du parent 2. D'autre part, et afin de créer le deuxième descendant, le même processus est répété en renversant les rôles. La figure 5.6 illustre le fonctionnement de RMPX.

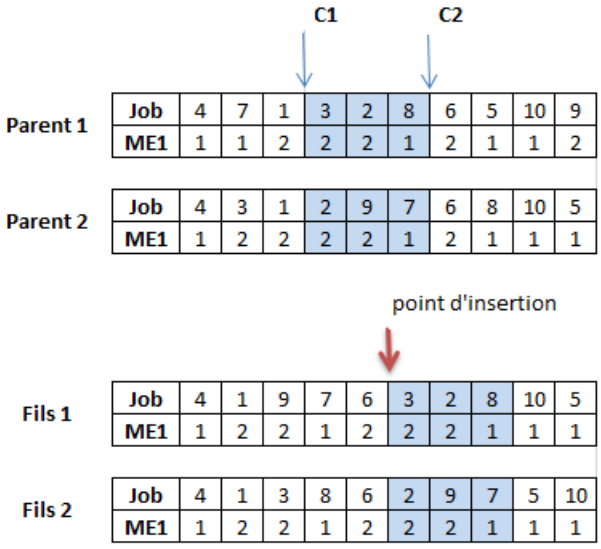


FIGURE 5.6 – Exemple de croisement par RMPX

3. **Mutation des jobs**
La mutation est un outil qui permet d'apporter une diversification à l'algorithme génétique afin d'explorer efficacement l'espace de recherche et sortir d'une convergence prématurée. Elle permet d'atteindre les sous-espaces de solutions réalisables en cherchant dans leur voisinage. Elle consiste généralement à muter au hasard un gène dans l'individu. Cet opérateur est appliqué avec une probabilité P_m (appelée probabilité de mutation). Les probabilités de mutation sont présentées dans le paragraphe (5.5.1). Dans notre étude, nous avons choisi 2 techniques utilisées fréquemment dans la littérature à savoir : Swap et Insert (figure 5.7)
   — Swap : il s'agit de choisir aléatoirement deux jobs i et j et les permuter.
   — Insert : il s'agit de choisir aléatoirement un job i et une position p (différente de celle du job i) et ensuite réinsérer i dans la position p.

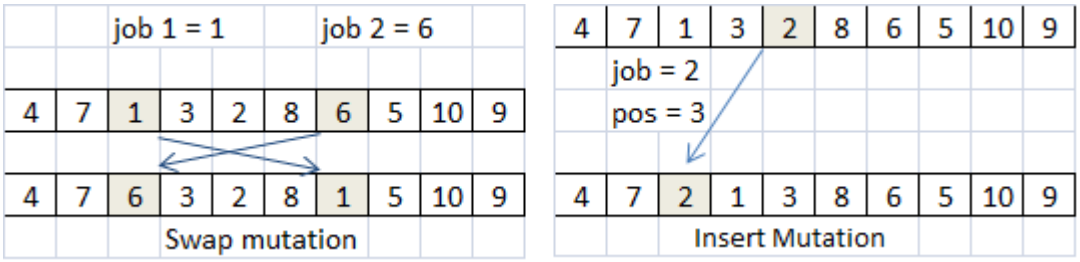
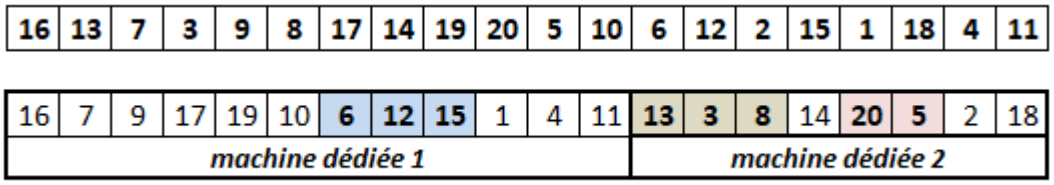


FIGURE 5.7 – Swap et Insert mutation

5.3 Algorithme génétique II

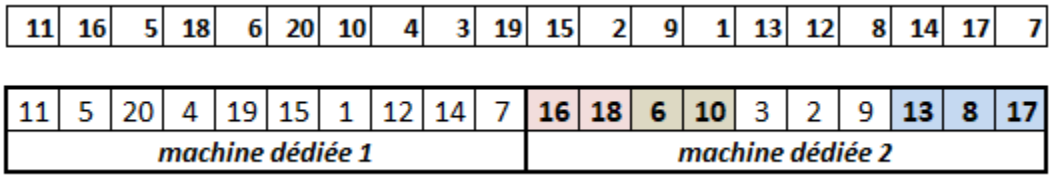
Le deuxième algorithme génétique explore l'espace de solutions d'une manière différente. Ce choix est motivé par les observations des solutions générées par les MIPs.
Dans ce qui suit, nous analysons d'abord la structure des solutions optimales obtenues par les MIPs. Nous considérons 4 instances de différentes classes. Pour chaque instance, nous présentons la permutation optimale, ainsi que la séquence des jobs passant sur la première machine dédiée suivie par la séquence des jobs sur la deuxième machine dédiée.
   — instance 1 : n=20 ; n1 = 0.6 n ; $p_{i,1} \in [1, 40]$ et $p_{i,1} \in [5, 200]$

   C_{max}= 372 , la permutation optimale est :



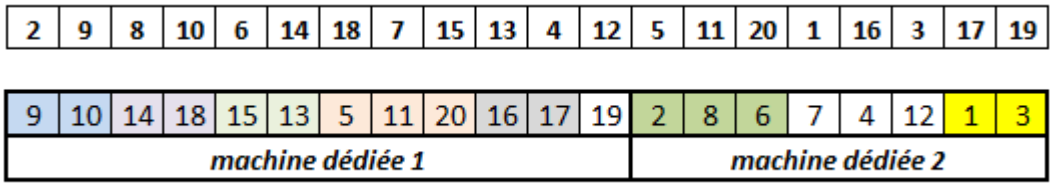
   — instance 2 : n=20 ; n1 = 0.5 n ; $p_{i,1} \in [40, 80]$ et $p_{i,1} \in [200, 400]$

   C_{max}= 954 , la permutation optimale est :



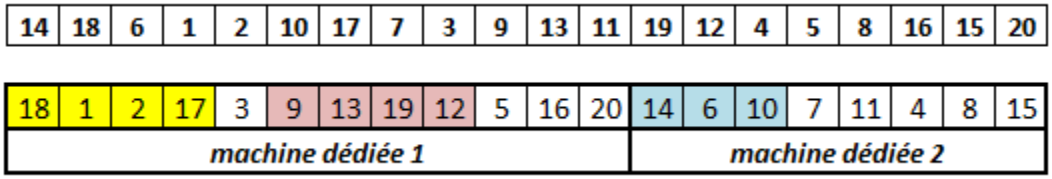
— instance 3 : n=20 ; n1=0.6 n ; $p_{i,1} \in [40, 80]$ et $p_{i,1} \in [200, 400]$

C_{max}= 1012 , la permutation optimale est :



— instance 4 : n=20 ; n1=0.6 n ; $p_{i,1} \in [80, 100]$ et $p_{i,1} \in [400, 600]$

C_{max}= 1594 , la permutation optimale est :



En observant les 4 solutions optimales, nous remarquons que chaque solution ordonnance au moins un ensemble des jobs par groupe (les jobs ayant la même couleur appartiennent au même groupe). De plus, nous avons révisé la technique d'affectation utilisée dans ces 4 exemples, nous avons noté que les jobs qui finissent les premiers passent en premier sur l'étage 2 en passant sur la première machine disponible. Compte tenu de ces faits, nous proposons une deuxième version d'AG qui utilise la procédure ECT-FAM.
En effet, la procédure de FAM est appliquée sur une permutation π_2. π_2 est trouvée après avoir appliqué un tri d'ordre croissant des dates de fin de traitement des jobs de π_1 sur l'E1. Ceci est différent de la procédure FAM appliquée dans AG1 où elle est appliquée directement sur la permutation des job π_1. L'exemple ci dessous (figure 5.8) illustre l'utilisation de FAM dans AG2 et les étapes suivies :

Etape 1 : commencer par une séquence de job sur l'étage 1

Etape 2 : calculer date fin d'exécution de chaque job sur l'étage 1

Job	4	7	1	3	2	8	6	5	10	9
Machine E1	1	2	2	1	1	1	2	2	1	2
Date fin E1	87	52	83	132	180	268	125	175	230	270

Etape 3 : donner une priorité à chaque job selon sa date fin d'exécution la plus tôt (ECT).

Etape 4 : calculer date prévisionnelle de disponibilité de machine de l'E2 ensuite affecter la première disponible au premier job en attente.

Job	4	7	1	3	2	8	6	5	10	9
Machine E1	1	2	2	1	1	1	2	2	1	2
Priorité	***3***	***1***	***2***	***5***	***7***	***9***	***4***	***6***	***8***	***10***

job	7	1	4	6	3	5	2	10	8	9
Machine E1	1	2	2	1	1	1	2	2	1	2
Machine E2	1	2	3	4	2	3	1	1	4	3
Date disponibilité Machine E2	178	128	165	277	390	401	241	515	590	620

Liste_mach_dispo : {1,2, 3, 4}→{2, 3, 4}→{3, 4}→{4}→{2}→{3}→{1}→{1}→{4}→{3}

Etape 5 : réparer la solution (date de début et fin de traitement sur étage 1) en considérant les date de disponibilité des machines de l'E2 et le time lag de chaque job

FIGURE 5.8 – Exemple d'affectation des jobs aux machines parallèles suivant AG2

D'autre part, cette décision est également appuyée par la littérature. En effet, dans la littérature, plusieurs auteurs ont prouvé l'efficacité de FAM. La technique d'affectation FAM est une règle simple et efficace qui a souvent été utilisée à affecter les jobs dans les ateliers flow shop hybride [Valérie, 2000]. D'autre part, la règle ECT-FAM (Earliest Completion Time- First Available Machine) est une technique d'affectation efficace dans le cas des machines parallèles identiques sans setup ou autre contrainte [Naderi et al., 2010], [Ruiz and Maroto, 2006]. Dans le cas de flow shop hybride avec une seule machine à l'étage

1 et soumis à la contrainte de $no-wait$, cette procédure est la seule qui génère des permutations admissibles [Wang et al., 2015].
Cependant, dans notre cas d'étude, la décision d'appliquer la règle FAM ne dépend pas seulement de la nature des machines parallèles (identiques) mais aussi de la spécificité des jobs sur l'étage 1 où ils sont soumis à contrainte de time lag maximal. Plusieurs auteurs ont adapté la technique FAM [Liu et al., 2008, Li and Li, 2007] pour ce type de problème vu que la durée d'attente maximale se présente courte par rapport à la durée totale de traitement sur les machines parallèles.

5.3.1 Codage de la solution

— *Séquence des jobs sur l'étage 1* :
Dans ce deuxième algorithme, chaque individu est constitué d'une seule séquence de n gènes représentés par une simple permutation des jobs sur l'E1. En effet, chaque job est affecté sur l'E1 ensuite sur l'E2 suivant son ordre d'apparition dans la permutation.

— *Construction de la solution correspondante (affectation des jobs aux machines parallèles sur l'étage 2)* :
L'affectation des jobs est faite lors de calcul de fitness (C_{max}). Nous avons adapté la technique ECT-FAM "Earliest Completion- First Available machine Time" à notre problème étudié, c.à.d. le premier job qui termine son traitement sur E1 est le premier qui passe sur E2, sur la première machine disponible.

5.3.2 Génération de la population initiale

Cette étape est réalisée de la même façon que le premier algorithme génétique sauf qu'une permutation de AG1 ne représente pas nécessairement la même solution pour AG2.

5.3.3 Évaluation : Calcul de la Fitness

Le makespan(C_{max}) est calculé en considérant l'affectation aux machines parallèles. Nous avons adapté la procédure ECT-FAM() en ajoutant le time lag pour affecter les jobs aux machines parallèles. En effet, nous retardons le début de traitement sur l'étage 1 s'il n'y a pas de machine disponible avant la fin de time lag maximal (Cette situation est décrite dans la littérature sous le terme forced idle-time, comme expliqué dans le chapitre 2).
Cette technique est utilisée fréquemment dans la littérature ([Li and Li, 2007], [Attar et al., 2014]) sous le nom de "constructive backtracking heuristic (CBH)" pour traiter la contrainte de time lag. Il faut signaler que cette technique est utilisée dans les 2 AGs.
L'**algorithme 2** présente le pseudo code qui correspond à cette technique (CBH).

Algorithme 2 CBH : Constructive backtracking()

```
1: m : entier          /* première machine parallèle diponible */
2: disp_m : entier     /* date de disponibilité de la machine m */
3: CT_{j,1} : entier   /* date fin de traitement du job j à l'étage 1 */
   Début
4:   Si (disp_m) ≤ (CT_{j,1}) alors
5:   affecter le job j à la machine m ;
6:   Sinon
7:     Si (disp_m > CT_{j,1}) et (disp_m ≤ (CT_{j,1} + l_j)) alors
8:       ne pas retarder le lancement du job j à l'étage 1
9:       affecter le job j à la machine m
10:    Sinon
11:      retarder le lancement du job j à l'étage 1
12:      CT_{j,1} ⟵ disp_m
13:      affecter le job j à la machine m
14:    Fin si
15:  Fin si
   Fin
16:
```

5.3.4 Opérateurs génétiques

Puisque le codage des individus dans les deux AGs est une permutation de jobs, nous gardons les mêmes opérateurs de croisement et mutation utilisés auparavant dans AG1.

Le schéma global d'AG1 et AG2 est représenté par l'**Algorithme 3**. Le schéma donne l'impression que les 2 AGs sont les mêmes, alors qu'en réalité ils sont différents puisque le codage d'une solution est différent.

Algorithme 3 Pseudo code général de AG1 et AG2

P_c : réel /* *probabilité de croisement* */
P_m : réel /* *probabilité de mutation* */
pop_{taille} : entier /* *taille de population* */
nb : entier /* nombre des individus élites */
Début
  Étape 1 : Génération population initiale de taille pop_{taille}
    Générer 6 individus en utilisant les règles de priorité :
    SPT (étage1), SPT (étage2),
    LPT (étage1), LPT (étage2),
    TPT (E1 +E2) dans l'ordre ascendant et TPT (E1 +E2) descendant
    Générer une sous-population par séquences de groupes en appliquant l'algorithme ***Algorithme 1***
    Générer le reste de la population aléatoirement
  Étape 2 : évaluer chaque individu de la nouvelle population
    Calculer C_{max} et trier la population
  Étape 3 : appliquer les opérateurs génétiques
    Tant que nouvelle population n'est pas construite **faire**
      Appliquer la sélection : sélectionner parent 1 et parent 2
      Croiser les deux parents sélectionnés avec probabilité P_c
      Intégrer les deux fils dans la nouvelle population
    Fin tant que
  Étape 4 : appliquer la phase de mutation
    Appliquer la mutation avec une probabilité P_m
  Étape 5 : évaluer chaque individu de la nouvelle population
    Calculer C_{max}
  Étape 6 : créer la nouvelle génération
    Appliquer la sélection par élitisme (nb% d'individu)
  Étape 7 : arrêter l'algorithme
    Si condition d'arrêt est satisfait **alors**
      Sortir
    Sinon Retourner à l'étape 3
Fin

5.4 Comparaison des AGs

Afin d'évaluer l'approche mise en œuvre, nous proposons de comparer les 2 algorithmes génétiques. Avant de mener la comparaison, nous rappelons dans un tableau récapitulatif (tableau 5.1), les différences entre les 2 algorithmes :

TABLE 5.1 – Les différences entre AG1 et AG2

Étape	**AG1**	**AG2**
Codage de la solution	une permutation suivant l'ordre non décroissant des dates de début de traitement sur l'E2	une permutation sur l'E1 définie suivant l'ordre d'affectation à la machine dédiée correspondante
Affectation aux machines parallèles	FAM (le premier qui apparait dans la permutation passe le premier sur la première machine disponible de l'E2)	ECT-FAM (le job qui finit son travail le premier sur l'E1 est affecté le premier sur la première machine disponible de l'E2)

Nous présentons ci-dessous un exemple illustratif qui montre la différence entre les 2 AGs. Supposons que nous avons la situation suivante représentée par la figure (5.9) où les jobs 1 et 2 sont dédiés à la machine 1 (M1) de l'E1 et les jobs 3 et 4 sont dédiés à la machine 2 (M2) de l'E1. Nous supposons que :

- les jobs 1 et 3 sont déjà ordonnancés sur E1 et sur E2
- les deux jobs 2 et 4 ne sont pas encore affectés à l'E2 ;
- $P_{2,1}$= 4, $P_{4,1}$= 5 représentent respectivement les durées de traitement de job 2 et 4 sur l'E1 ;
- $P_{2,2}$= 6, $P_{4,2}$= 8 représentent respectivement les durées de traitement de job 2 et 4 sur l'E2 ;
- l_2= 2, l_4= 3 représentent respectivement les durées de time lag maximal de job 2 et 4 ;
- $CT_{2,1}$= 7, $CT_{4,1}$= 9, $CT_{2,2}$, $CT_{4,2}$ représentent respectivement les dates fin de traitement de job 2 et 4 sur l'E1 et l'E2 ;
- $AVm1$= 8 ; $AVm2$= 11 représentent respectivement les dates de disponibilité de machine m1 et m2 de l'E2 ;

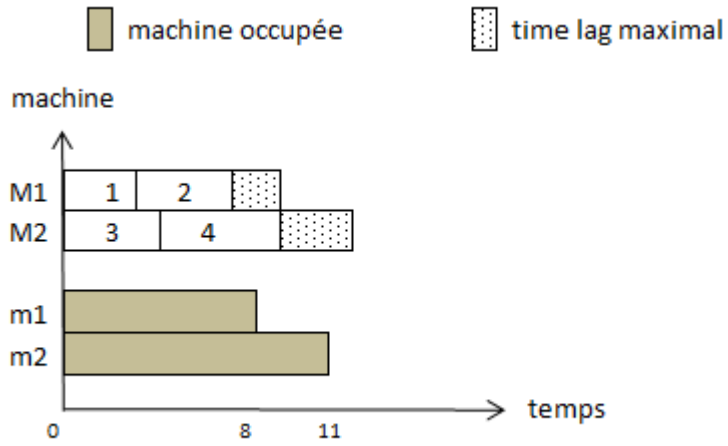


FIGURE 5.9 – Exemple d'une situation

La permutation 1-3-4-2 est considérée par AG1 comme une séquence sur l'E2 et il fait passer les jobs dans le même ordre de leur apparition dans la permutation, donc il fait passer le job

4 sur m1 et le job 2 sur m2 (figure 5.10). Par conséquent, nous aurons :
C_{max} = Max($CT_{2,2}, CT_{4,2}$)
= Max(max($AVm2$, $CT_{2,1}$)+$P_{2,2}$, max($AVm1$,$CT_{4,1}$)+$P_{4,2}$)
= Max max(11, 7)+6, max(8,9)+8= 17
C_{max}= 17

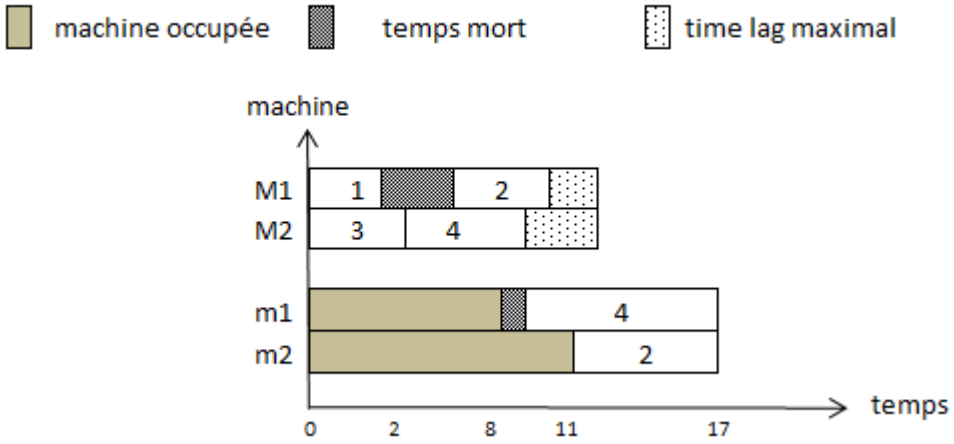


FIGURE 5.10 – Comportement d'une solution par AG1

Cependant, AG2 utilise la règle ECT-FAM et fait passer le premier job qui finit son traitement sur l'étage 1 : soit le job 2 sur la première machine disponible m1 de l'E2, ensuite il fait passer le job 4 sur la machine m2 (figure 5.11). De ce fait, nous aurons :
considère la permutation. C_{max} = Max($CT_{2,2}, CT_{4,2}$)
= Max(max($AVm1$, $CT_{2,1}$)+$P_{2,2}$, max($AVm2$,$CT_{4,1}$)+$P_{4,2}$)
= Max max(8, 7)+6, max(11, 9)+8= 19
C_{max}= 19

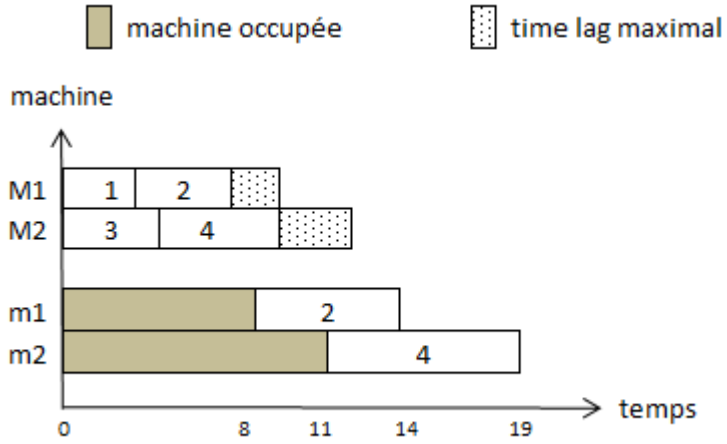


FIGURE 5.11 – Comportement d'une solution par AG2

Il faut signaler que les 2 permutations 1-3-4-2 et 1-3-2-4 correspondent à la même permutation de l'étage 1 car la date de début de traitement n'est pas importante pour la définition de la permutation puisque les machines sont dédiées. De ce fait, nous concluons que l'AG1 et l'AG2 explorent l'espace de recherche de deux manières différentes.

5.5 Paramétrage des algorithmes

La performance d'un AG dépend des choix des opérateurs et d'autres paramètres. Pour choisir les meilleurs opérateurs et paramètres à appliquer aux AGs, nous avons effectué des tests préliminaires.
Dans ce qui suit, nous présentons les paramètres testés ainsi que l'approche utilisée pour choisir les meilleurs.

5.5.1 Les paramètres testés

Le tableau ci-dessous (tableau 5.2) présente les paramètres des AGs, à savoir, la taille de population, les taux ainsi que les types de mutation et de croisement. On adoptera par la suite la meilleure combinaison des opérateurs génétiques.

TABLE 5.2 – Paramètres des algorithmes génétiques

Paramètre	**Valeur**
taille de la population	100,150,200
nombre de génération	100,200,500
sélection	aléatoire, roulette de casino
opérateur génétique	**Valeur**
croisement	SJOX, RMPX
probabilité de croisement	0,7 ; 0,9 ; 1,0
mutation	insert, swap
probabilité de mutation	0,2 ; 0,5

Le nombre total de combinaisons est 432 ($3\times3\times2\times3\times2\times2\times2$). Pour chaque couple (n, n1), nous avons généré deux instances de test et nous l'avons exécuté avec toutes les combinaisons possibles. Il faut noter que les paramètres testés et spécifiquement les ratios de mutation et croisement proposés sont choisis en se référant à la littérature. D'autre part, les taux élevés de mutation sont expliqués par le fait que la nature des machines dédiées sur l'étage 1 contribue à avoir des permutations de jobs équivalentes et avoir des descendants identiques. D'où la nécessité d'augmenter le taux de mutation. Pour bien expliquer, nous présentons un exemple de 6 jobs. Les jobs 1, 2 et 3 passent sur M1 de l'E1. Les jobs 3,4 et 5 passent sur M2 de l'E1. Considérant la permutation $\pi1$ = 1-4-2-5-3-6, $\pi2$= 4-1-5-2-6-3. $\pi1$ et $\pi2$ sont identiques selon AG2 et résultent la même valeur de C_{max}.

5.5.2 Choix des paramètres

Pour paramétrer l'algorithme génétique, nous avons choisi deux instances de chaque famille de problèmes que nous avons testés avec toutes les combinaisons des paramètres.
Il convient de rappeler que l'évaluation de chaque instance est basée sur la détermination de l'écart du C_{max} par rapport à la borne inférieure (RD). Étant donné l'aspect aléatoire de l'AG, on n'obtient pas nécessairement le même résultat en exécutant la même instance plusieurs fois, même si on garde les mêmes paramètres. Sur cette base, l'objectif est de connaitre le paramétrage susceptible de produire le plus de résultats avec des écarts faibles et pas nécessairement le meilleur résultat.
La méthode utilisée pour paramétrer l'AG est basée sur trois étapes :

— **La première étape** : consiste à regrouper les résultats en des classes qui permettent de qualifier les résultats.

— **La deuxième étape** : consiste à appliquer un algorithme de classification, pour déterminer l' opérateur génétique (le croisement ou la mutation) qui affecte le plus le résultat.

— **La troisième étape** : consiste à utiliser le tableau croisé dynamique d'Excel pour déterminer les moyennes des RD en se basant sur le croisement de différents opérateurs en fonction de celui qui affecte le plus le résultat.

Nous présentons dans ce qui suit les résultats pour la famille des problèmes à 20 jobs, les autres seront présentés en annexe A.

1- Regroupement des résultats

Nous avons commencé par ranger les résultats en fonction de l'écart moyen par rapport à la best LB (soit la RD) comme indiqué ci-dessous.

RD	0	0 < RD ≤1	> 1	> 2	> 4	> 6	> 8	> 10	> 14
Classe	Ega 0	Sup 0	Sup 1	Sup 2	Sup 4	Sup 6	Sup 8	Sup 10	Sup 14

2- Opérateur qui affecte le plus les résultats

La deuxième étape consiste à appliquer un algorithme de classification, pour déterminer l'opérateur qui affecte le plus le résultat. Nous avons utilisé le logiciel SIPINA [Sip,] qui a permis de conclure que l'opérateur de croisement permet de mieux différencier entre les résultats (voir Tableau 5.3).

TABLE 5.3 – Taux d'affectation d'opérateur

	Goodness of split	Accept
Croisement	0.0174	O
Taille	0.0023	N
Génération	0.0000	N
Sélection	0.0000	N

Dans ce cadre, RMPX s'est avéré supérieur à SJOX (faibles valeurs de RD). La figure 5.12 illustre le résultat de l'application de SIPINA. Le fait que l'opérateur de croisement (en tant

qu'attribut) se trouve au premier niveau de l'arbre signifie qu'il permet de différencier entre les valeurs des RD mieux que les autres opérateurs (mutation, nombre de générations, taille de la population).

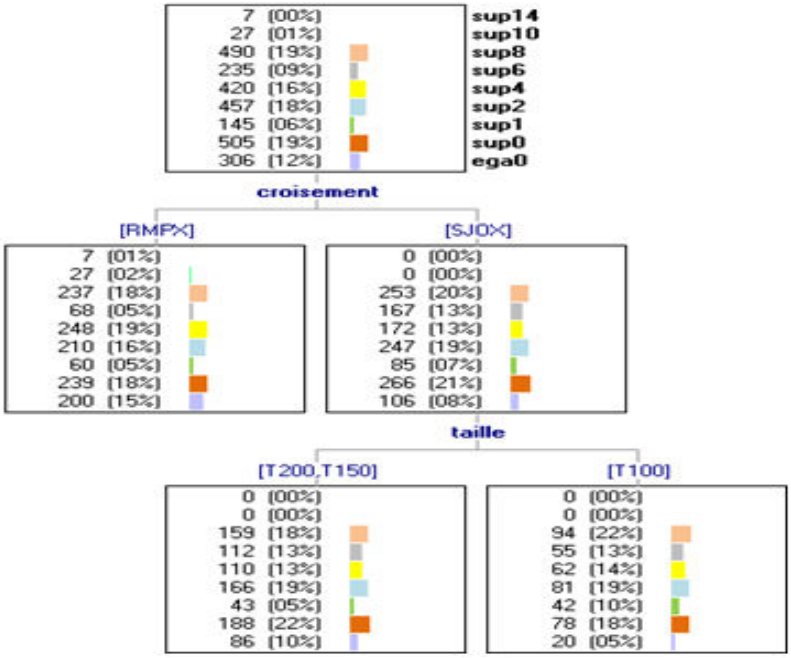


FIGURE 5.12 – SIPINA output

3- Croisement des résultats

La troisième étape consiste à déterminer les moyennes des RD pour les autres opérateurs. Les tableaux 5.4, 5.5 et 5.6 présentent les résultats en fonction de l'opérateur de mutation, de la taille de la population et du nombre de générations. Ces résultats sont générés à l'aide de l'outil tableau croisé dynamique d'Excel.

TABLE 5.4 – Opérateur de croisement × Opérateur de mutation (20 jobs)

	ins 0,2	ins 0,5	Insert	sw 0,2	sw 0,5	Swap	Total
RMPX	3,859	3,839	3,849	3,7	3,854	3,777	3,813
0.7	4,15	4,315	4,233	4,248	4,336	4,292	4,262
0.9	3,794	3,7	3,747	3,494	3,828	3,661	3,704
1.0	3,634	3,502	3,568	**3,358**	3,397	**3,377**	3,473
SJOX	3,943	4,031	3,987	4,019	3,907	3,963	3,975
0.7	4,093	4,132	4,113	4,295	4,141	4,218	4,165
0.9	3,891	4,084	3,988	3,865	3,811	3,838	3,913
1.0	3,847	3,876	3,861	3,897	3,768	3,833	3,847
Total	3,901	3,935	3,918	3,859	3,88	3,87	3,894

Il est clair qu'un opérateur de croisement de type RMPX combiné à un opérateur de mutation SWAP produit de meilleurs résultats pour les problèmes à 20 jobs.

TABLE 5.5 – Opérateur taille de la population

	T100	T150	T200	Total
RMPX	4,131	3,796	3,512	3,813
insert	4,097	3,899	3,551	3,849
swap	4,164	3,693	**3,473**	3,777
SJOX	4,208	3,952	3,766	3,975
insert	4,213	3,941	3,808	3,987
swap	4,203	3,963	3,723	3,963
Total	4,169	3,874	3,639	3,894

La combinaison de l'opérateur de croisement RMPX avec l'opérateur de mutation SWAP et une population de taille 200 produit des écarts meilleurs que les autres combinaisons dans le cas des problèmes à 20 jobs.

TABLE 5.6 – Opérateur » nombre de générations

	G100	G200	G500	Général
RMPX	3,934	3,894	3,611	3,813
insert	3,911	3,96	3,677	3,849
swap	3,958	3,828	3,545	3,777
SJOX	3,997	3,964	3,964	3,975
insert	3,982	3,975	4,004	3,987
swap	4,011	3,953	3,925	3,963
Général	3,965	3,929	3,788	3,894

Bien que les résultats sont meilleurs en testant 500 générations, il est nécessaire de remarquer que le temps d'exécution enregistré est élevé. Pour cela, nous nous sommes arrêtés à G200 et non G500.
Sur cette base, nous avons adopté, pour l'AG, les paramètres indiqués dans le tableau suivant(tableau 5.7).

TABLE 5.7 – Les paramètres retenus

Opérateur	Type
Croisement	RMPX (1.0)
Mutation	SWAP (0.05)
Générations	200
Taille population	200

5.6 Combinaison des AGs avec une recherche locale itérative

Bien que les opérateurs de croisement et mutation garantissent à la fois la diversification et l'intensification de la recherche, leur application ne garantit pas toujours l'obtention des solutions avec la qualité souhaitée. L'introduction d'une recherche locale (RL) permet d'élargir d'avantage l'espace de recherche et par conséquent obtenir des meilleures solutions.
Selon ([Papadimitriou and Steiglitz, 1998]) : cette méthode utilise une solution initiale, puis, elle explore son voisinage en parcourant l'espace de recherche itérativement. Dès qu'un meilleur voisin est trouvé, il est sélectionné et il devient la nouvelle solution courante.
Dans cette étape, nous hybridons les algorithmes avec une procédure de recherche locale itérative (RLI). Cette procédure est appliquée à la fin des AGs. Nous appliquons la recherche locale itérative aux meilleurs $nb\%$ de la dernière génération (les individus élites). La RLI est appliquée aussi bien à AG1 qu'à AG2.

5.6.1 Fonctionnement de la recherche locale

Dans ce qui suit, nous présentons deux procédures de RL. La première est une recherche locale simple (RL) et la deuxième est une recherche locale itérative (RLI).
La RL consiste à insérer un job de la solution courante dans une autre position différente de la position actuelle. Si la nouvelle solution trouvée présente une solution meilleure alors une mise à jour de la meilleure solution actuelle sera effectuée (voir Algorithme 4).

Algorithme 4 Procédure de la recherche locale

```
1: S0, Scourante, Smeilleure : tableau d'entier /* une permutation de jobs */
2: /* S0 : une solution initiale */
3: /* Scourante : la solution courante */
4: /* Smeilleure : la meilleure solution trouvée */
5: n : entier          /* nombre de jobs dans la solution */
6: pos : entier        /* indice de la position du job */
7: i : entier          /* indice du job */
8: j : entier          /* compteur des postions générées aléatoirement */
9: Smeilleure ← S0     /* on initialise la meilleure solution */
10: i ← 0              /* indice du premier job */
11:
   Début
12:     Tant que (i < n) faire
13:       Pour (j de 0 à n/10) faire
14:         pos ← une position générée aléatoirement
15:         /* pos est différente de la position actuelle de job i */
16:         Scourante ← séquence trouvée âpres avoir déplacé le job i à la position pos
17:         Si (f(Scourante) < f(Smeilleure)) alors
18:           Smeilleure ← Scourante
19:         Fin si
20:       Fin Pour
21:     Fin tant que
   Fin
22:
```

5.6.2 Fonctionnement de la recherche locale itérative

La recherche locale itérative consiste à itérer la méthode de descente jusqu'à ce que le critère d'arrêt soit atteint. Le critère d'arrêt choisi est un nombre de no-amélioration trouvé Nbr_NA. Nbr_NA est expérimentalement fixé à 10. L'algorithme 5 explique le fonctionnement de la RLI.

Algorithme 5 Procédure de la recherche locale itérative

```
1: S_0, S_courante, S_meilleure : tableau d'entier /* une permutation de jobs */
2: /* S_0 : une solution initiale */
3: /* S_courante : la solution courante */
4: /* S_meilleure : la meilleure solution trouvée */
5: Ctr_NA, Nbr_NA : entier
6: /* Ctr_NA : compteur de non-amélioration */
7: /* Nbr_NA : nombre fixé de non-amélioration */
8: S_meilleure ← S_0          /* on initialise la meilleure solution */
9:
   Début
10:    Tant que (critère d'arrêt n'est pas satisfait) faire
11:       S_courante ← recherche_locale (S_0)
12:       Si (f(S_courante) < f(S_meilleure)) alors
13:       S_meilleure ← S_courante          /* on met à jour la meilleure solution
14:       Sinon
15:       Ctr_NA ← Ctr_NA + 1
16:       Fin si
17:
18:       Si (Ctr_NA < Nbr_NA) alors
19:       S_0 ← fct_perturabtion ()
20:       /* on redémarre avec une nouvelle solution trouvée par AG
21:       Fin si
22:    Fin tant que
   Fin
23:
```

5.7 Algorithme mémétique basé sur la recherche locale itérative

L'algorithme mémétique (AM) est une méta-heuristique à base de population de solutions de type hybride. Elle combine un algorithme génétique et une méthode de recherche locale afin d'améliorer la qualité des solutions. Cet algorithme est basé sur des opérateurs génétiques et incorpore de la recherche locale à chaque génération. Dès la création de la population initiale, une recherche locale est appliquée sur chaque nouvel individu. Le but de l'algorithme mémétique est de composer une population avec uniquement d'optima locaux. Dans l'approche proposée dans la section précédente, à savoir, l'algorithme génétique renforcé par une RLI, nous avons utilisé une recherche locale itérée, ce qui a permis d'élargir la recherche au voisinage de meilleurs individus et de visiter davantage d'optima locaux.
Afin de mener une comparaison juste entre les deux méthodes approchées, nous avons remplacé la recherche locale simple dans l'AM classique par une RLI utilisée dans les AGs (figure 5.12). Cependant, l'application d'une RLI à chaque fils dans un AM augmentera nécessairement le temps d'exécution. La solution évidente est d'appliquer une recherche locale plus légère. Cette décision

sera discutée d'avantage dans la section des résultats. Nous signalons que nous avons gardé les mêmes opérateurs génétiques que les AGs présentés auparavant, à savoir : le croisement et la mutation ainsi que les paramètres de lancement.

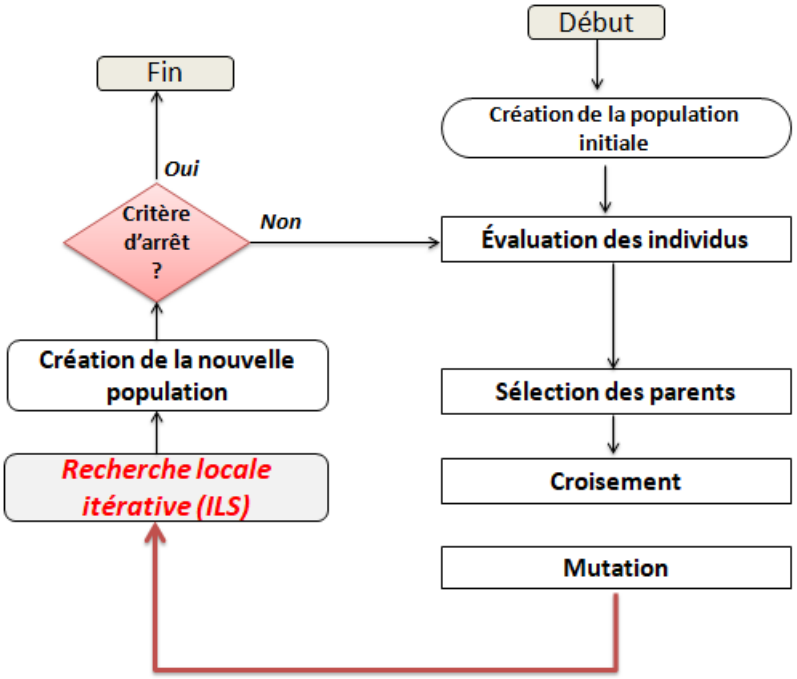


FIGURE 5.13 – Algorithme mémétique

5.8 Comparaison empirique des AGs

Dans ce qui suit, nous allons tester l'effet des opérateurs génétiques sur la solution initiale, ensuite nous allons étudier la qualité de la solution finale. Afin de tester l'effet des opérateurs, nous allons tout d'abord mesurer l'écart E entre le meilleur ordonnancement trouvé et le meilleur ordonnancement de la population initiale. Ensuite, pour mesurer la qualité de la solution, nous allons comparer la solution finale à la borne inférieure LB_{best} présentée dans le chapitre 4.
Nous définissons donc l'écart E, l'écart moyen Em, aussi l'écart E^* et Em^* comme suit :

— E : l'écart entre la meilleure solution de la population initiale ($best_{s0}$) et la meilleure solution trouvée après n générations ($best_{sf}$) :

$$E = \frac{best_{s0} - best_{sf}}{best_{sf}} \times 100$$

— E_m : la moyenne des écarts E $\qquad E_m = \frac{\sum E}{totald'instances}$

— E^* : l'écart entre la meilleure solution $best_{sf}$ et la borne inférieure LB_{best} :

$$E^* = \frac{best_{sf} - LB_{best}}{LB_{best}} \times 100$$

— E_m^* : la moyenne des écarts E^* :

$$E_m^* = \frac{\sum E^*}{totald'instances}$$

— T_m : la durée moyenne d'exécution de l'algorithme

Nous rappelons que la solution optimale est connue pour toutes les instances de 10 jobs et quelques instances de 20 jobs (trouvée par les 2 MIPs cités dans le chapitre 4). Dans ce qui suit, nous allons utiliser la bibliothèque d'instances proposée dans la section 2.4 du chapitre 2.
Il faut signaler que l'étude de l'écart E_m permet d'analyser l'efficacité des opérateurs génétiques fixés et appliqués. En effet, l'écart E_m va refléter le taux d'amélioration de la solution initiale après avoir appliqué les opérateurs génétiques. Donc, plus le E_m est élevé plus l'AG a amélioré la solution initiale. Si cet écart est faible, ceci peut avoir différentes explications :
- soit que la population initiale contient de bonnes solutions ;
- soit les opérateurs de croisement et mutation ne sont pas efficaces.
Dans ce qui suit, nous présentons les résultats numériques trouvés par chaque AG. Nous allons étudier aussi l'influence de la recherche locale itérative sur la convergence des AGs et l'effet de la recherche locale itérative sur la qualité de la solution.
Les manipulations ont été effectuées sur une machine avec un processeur i5 2.33 GHz et possédant 64 GB de RAM.

5.8.1 Comportement des AGs suivant le critère d'arrêt : nombre d'itérations

Dans cette partie, nous allons examiner toutes les versions des deux AGs (AGs purs et hybrides) en fixant un nombre d'itérations limité comme un critère d'arrêt.
Comportement des AG dans la 1ère catégorie de données

Dans cette section, nous allons étudier la variation des écarts E_m et E_m^* en étudiant l'influence de l'intégration de l'heuristique qui ordonnace les jobs par groupe (H1) dans la population initiale ainsi la combinaison des AGs avec la RLI.

- **Comportement de E_m obtenu par les 2 AGs purs**

Le tableau 5.8 compare les écarts E_m et le T_m trouvés par chaque AG pur.

TABLE 5.8 – Effet des AGs sur la solution initiale

n	AG1 pur		AG2 pur	
	Em	Tm	Em	Tm
10	3,67	0,00	3,30	0,00
20	12,09	0,00	10,79	0,00
50	10,36	1,00	11,32	1,00
100	15,50	2,00	13,82	3,00
200	13,58	4,00	11,14	8,00

Comme le montre le tableau 5.8, les écarts E_m trouvés par les 2 AGs purs dépassent 10%

sauf pour les instances résolues. Ceci montre que les opérateurs génétiques améliorent la solution initiale surtout lorsque cette dernière est basée principalement sur l'aléatoire. En ce qui concerne le temps moyen de calcul, nous remarquons que AG2 pur a mis plus de temps que AG1 pur. Ceci est expliqué par le fait que AG2 utilise une procédure de tri pour déterminer les dates prévues de début de traitement des jobs sur l'E2, autrement le tri sert à préciser quels jobs finissent en premier leurs opérations sur l'E1 avant de les affecter aux machines parallèles, tandis que AG1 place directement les jobs sur les machines parallèles.
Dans ce qui suit, nous allons étudier l'avantage des opérateurs de croisement et de mutation à promouvoir la solution initiale. Pour ce faire, nous proposons de comparer les écarts moyens E_m^* trouvés par les 2 AGs purs avec ceux trouvés par un simple algorithme aléatoire (Alg-alétoire). D'autre part, les écarts moyens E_m^* nous permet de déterminer la mesure dans laquelle les opérateurs génétiques appliqués réussissent à trouver une solution finale de bonne qualité. Dans le but de mener une comparaison équitable, il faut signaler que nous avons attribué la même durée d'exécution à l'algorithme aléatoire que les autres algorithmes.

TABLE 5.9 – E_m^* trouvés par AG1 pur, AG2 pur et l'algorithme aléatoire

n	**AG1 pur**	**Alg-aléatoire**	**Tm**	**AG2 pur**	**Alg-aléatoire**	**Tm**
10	0,00	1,30	0,00	0,00	0,97	0,00
20	3,36	10,36	0,55	3,14	9,03	1,00
50	5,68	13,94	1,00	4,89	11,94	2,00
100	7,46	18,07	3,00	6,47	16,46	4,00
200	9,33	23,78	4,00	8,21	25,11	8,00

Dans le tableau 5.9, on remarque, en premier lieu, que AG1 pur et AG2 pur surpassent clairement un algorithme aléatoire en réalisant des E_m^* beaucoup plus faibles. Ceci montre bien que les opérateurs génétiques utilisés agissent efficacement sur les solutions initiales surtout que les AGs purs démarrent d'une population aléatoire. On note en deuxième lieu, que les deux AGs purs résolvent à l'optimum les instances de 10 jobs. En plus, les E_m^* trouvés par les deux AGs sont moyennement faibles ne dépassant pas les 6% pour les instances de taille moyenne (20 et 50 jobs), et légèrement élevés (des écarts $> 8\%$) pour les instances de grande taille (100 et 200 jobs). Ceci montre clairement que la conception des deux AGs est un élément important assurant une plus grande efficience pour résoudre ce problème difficile. Dans le but d'améliorer d'avantage la qualité des solutions trouvées et baisser encore les écarts E_m^*, nous proposons :

1. Introduire l'heuristique H1 dans la population initiale
2. Hybrider les 2 AGs avec une recherche locale itérative (RLI)

• Évaluation de la qualité de la solution : Comportement de E_m^* obtenu par les 2 AGs renforcés par une heuristique (H1) et hybridés avec RLI

Nous allons étudier dans les tableaux 5.9 et 5.10 la variation de E_m^* dans chaque version des 2 AGs : AG pur, AG renforcé par une heuristique (H1) et finalement AG hybridé avec RLI.

TABLE 5.10 – E_m^* obtenu par AG1 pur, renforcé par une heuristique et hybridé avec RLI

n	AG1 pur		AG1 +heur		AG1 + Heurs+RLI	
	Em*	**Tm**	**Em***	**Tm**	**Em***	**Tm**
10	0,18	0,00	0,00	0,00	0,00	0,00
20	3,36	0,00	2,47	1,00	1,90	1,00
50	5,68	1,00	3,70	2,00	1,96	5,00
100	7,46	2,00	4,58	4,00	2,40	28,00
200	9,33	4,00	6,61	9,00	3,54	210,00

Comme le montre les deux tableaux 5.10 et 5.11, les E_m^* trouvés par les 2 AGs purs sont légèrement faibles pour les instances de 10 et 20 jobs alors qu'ils varient entre 5% et 10% pour les autres instances. Dans cette situation, il est clair que les 2 AGs purs sont capables de trouver des solutions de qualité moyenne qui s'approchent de LB_{best}.
Ensuite, l'amélioration de la qualité de la population initiale en introduisant l'heuristique a amélioré les écarts E_m^*. Ce qui explique la baisse des écarts trouvés par AG1 renforcé par l'heuristique H1 (tableau 5.10) d'au moins 30% pour toutes les instances sauf les instances résolues. L'AG2 renforcé par H1 (tableau 5.11) améliore aussi la qualité des individus en réalisant E_m^* plus faible que le premier AG. En plus, il faut remarquer que l'écart par rapport à la solution optimale dans le cas où n = 10 est nul.

TABLE 5.11 – E_m^* obtenu par AG2 pur, renforcé par une heuristique et hybridé avec RLI

n	AG2 pur		AG2 +heur		AG2 + Heurs+RLI	
	Em*	**Tm**	**Em***	**Tm**	**Em***	**Tm**
10	0,20	0,00	0,00	0,00	0,00	0,00
20	3,14	0,00	2,67	1,00	2.26	0,00
50	4,89	1,00	3,13	2,00	1,46	5,00
100	6,47	3,00	3,63	6,00	1,86	39,00
200	8,21	8,00	4,40	16,00	2,24	283,00

D'autre part, il est clair que l'application de la RLI améliore d'avantage les résultats en apportant un gain d'au moins 50% pour les instances de 50, 100 et 200 jobs. Les solutions obtenues par les AGs + Heurs + RLI sont de bonne qualité et s'approchent de l'optimum.
Il convient de signaler que AG2 + Heurs + RLI surpasse AG1 + Heurs + RLI en réalisant des $E_m^* \leq 2.2\%$ dans toutes les instances. Bien que AG1 améliore plus le résultat, AG2 semble de meilleure qualité dans toutes les instances sauf celles de 20 jobs.

Comportement des AGs dans la 2ème catégorie de données

Comme signalé dans la section précédente, l'introduction de l'heuristique H1 et la combinaison des AGs avec RLI améliorent significativement les E_m et E_m^*. Dans cette deuxième catégorie de

données, nous allons tester les 2 AGs purs, ensuite les 2 AGs renforcés par l'heuristique H1 et les 2 AGS hybridés avec RLI. Nous proposons d'évaluer les 2 AGs suivant la classe de données, le nombre de jobs (n), et suivant l'équilibre des charges des machines dédiées en termes de nombre de jobs (n1).

- **Résultats des deux AG purs**

Le tableau ci-dessous (5.12) révèle les écarts E_m trouvés par AG1 pur et AG2 pur. Comme le montre le tableau 5.12, les écarts E_m trouvés par AG1 pur varient entre 11 et 33 % dans les 3 classes d'instances où il y a un équilibre des charges entre les machines dédiées sauf les instances de 10 jobs. En ce qui concerne le reste des instances où n1= 0.6 n et n1= 0.7 n, nous trouvons les écarts E_m varient entre 6 et 23 %. Ceci confirme la conclusion précédente : AG1 est performant du point de vue opérateurs génétiques et paramètres fixés.
Nous remarquons également que les deux AGs n'améliorent pas les instances de la même classe de la même manière que pour la classe 1 et 2.

TABLE 5.12 – Évaluation des E_m trouvés par AG1 pur et AG2 pur

n1=	**n**	**AG1 pur**			**AG2 pur**		
		Classe 1	**Classe 2**	**Classe 3**	**Classe 1**	**Classe 2**	**Classe 3**
		Em	**Em**	**Em**	**Em**	**Em**	**Em**
0,5n	10	2,01	0,78	0,70	1,23	0,57	0,70
	20	18,48	18,16	11,60	10,88	18,16	9,67
	50	17,63	22,07	20,05	15,32	19,50	10,08
	100	24,16	21,44	33,65	31,64	18,38	11,19
	200	15,84	33,24	20,39	27,53	15,88	16,70
0,6n	10	0,36	1,63	0,00	0,64	1,63	1,30
	20	13,49	9,10	7,94	12,80	16,39	10,37
	50	20,64	11,10	8,83	20,07	17,33	11,08
	100	18,90	10,41	9,69	16,41	12,84	16,43
	200	14,13	11,81	14,56	15,57	14,01	9,55
0,7n	10	0,00	0,66	0,80	0,00	1,89	0,00
	20	13,68	6,86	9,63	14,25	10,02	11,63
	50	22,68	10,31	8,66	19,07	13,78	9,66
	100	18,07	9,44	9,03	15,55	9,84	9,03
	200	13,57	11,90	11,43	11,84	11,02	11,43

- **Comparaison des AGs purs à un algorithme aléatoire**

Dans ce qui suit, nous allons évaluer la qualité de la solution finale trouvée par les 2 AGs purs en la comparant à un des meilleurs individus générés aléatoirement.

TABLE 5.13 – E_m^* trouvés par AG1 pur et l'algorithme aléatoire

n1=	**n**	**Classe 1**			**Classe 2**			**Classe 3**		
		AG1 pur	**Alg aléatoire**	**Tm**	**AG1 pur**	**Alg aléatoire**	**Tm**	**AG1 pur**	**Alg aléatoire**	**Tm**
0,5n	10	0,00	0,81	0,00	0,00	0,01	0,00	0,00	0,00	0,00
	20	6,70	15,97	1,00	1,63	6,04	1,00	3,64	7,01	1,00
	50	9,35	39,46	2,00	7,84	14,95	2,00	10,39	15,63	2,00
	100	13,50	46,01	3,00	10,96	19,84	3,00	13,91	20,81	3,00
	200	16,01	57,86	7,00	12,98	25,14	7,00	15,51	25,61	7,00
0,6n	10	0,00	0,41	0,00	0,30	0,10	0,00	0,00	0,10	0,00
	20	5,53	16,26	1,00	0,52	4,13	1,00	0,11	2,39	0,00
	50	10,23	36,01	1,00	2,15	11,66	1,00	2,38	6,23	1,00
	100	10,99	46,12	3,00	5,97	15,46	3,00	6,25	11,24	3,00
	200	12,98	54,84	7,00	8,60	18,62	6,00	7,51	13,84	7,00
0,7n	10	0,00	0,00	0,00	0,20	0,00	0,00	0,00	0,00	0,00
	20	6,00	19,49	1,00	0,59	5,07	0,00	0,47	2,86	1,00
	50	8,28	37,32	1,00	3,00	12,05	2,00	1,69	6,69	2,00
	100	9,14	43,54	3,00	3,24	16,01	3,00	2,42	10,56	3,00
	200	12,01	51,50	6,00	5,21	18,74	7,00	4,36	12,63	7,00

Dans le tableau 5.13, on remarque, en premier lieu, que AG1 pur surpasse clairement un algorithme aléatoire en réalisant des E_m^* beaucoup plus faibles. Ceci montre bien que les opérateurs génétiques utilisés agissent efficacement sur les solutions initiales surtout que les 2 AGs purs démarrent d'une population aléatoire. On note en deuxième lieu, que les E_m^* trouvés sont élevés pour toutes les instances dans les 3 classes sauf celles de 10 et 20 jobs. On peut conclure que malgré l'efficacité des opérateurs génétiques à améliorer la solution initiale, la qualité de la solution finale nécessite d'avantage une amélioration.

TABLE 5.14 – E_m^* trouvés par AG2 pur et l'algorithme aléatoire

n1=	**n**	**Classe 1**			**Classe 2**			**Classe 3**		
		AG2 pur	**Alg aléatoire**	**Tm**	**AG2 pur**	**Alg aléatoire**	**Tm**	**AG2 pur**	**Alg aléatoire**	**Tm**
0,5n	10	0,00	0,52	0,00	0,00	0,12	0,00	0,00	0,10	0,00
	20	7,26	11,54	0,00	1,75	3,35	0,00	3,35	4,70	0,00
	50	12,27	34,23	1,00	6,82	12,63	1,00	8,95	13,58	1,00
	100	13,05	42,71	3,00	10,99	17,68	3,00	12,53	18,97	3,00
	200	15,53	54,05	9,00	12,79	22,81	8,00	11,73	23,33	9,00
0,6n	10	0,00	0,41	0,00		0,00	0,00		0,10	0,00
	20	6,58	11,65	0,00	0,76	1,58	0,00	0,63	1,04	0,00
	50	11,35	31,05	1,00	3,87	9,53	1,00	5,90	6,61	1,00
	100	10,29	42,36	3,00	6,62	14,10	3,00	9,62	10,10	3,00
	200	12,46	51,47	9,00	7,26	17,61	9,00	12,82	12,85	10,00
0,7n	10	0,00	0,00	0,00		0,21	0,00		0,00	0,00
	20	6,71	13,86	0,00	0,94	2,64	0,00	0,59	1,40	0,00
	50	9,66	32,80	1,00	4,95	10,20	1,00	5,25	6,46	1,00
	100	12,11	43,98	3,00	8,26	14,85	3,00	8,65	9,92	3,00
	200	14,40	51,59	9,00	10,47	17,70	9,00	11,70	13,97	10,00

Dans le tableau 5.14, les mêmes remarques signalées précédemment valent aussi pour le deuxième AG pur. Suite à cette observation, nous pouvons affirmer la mauvaise qualité de la solution trouvée par n'importe quel AG pur dans cette catégorie de données. Pour remédier à ce problème, nous allons tester les deux AGs renforcés par l'heuristique H1, ensuite les hybrider avec RLI.

Résultats des deux AG^{+heur} renforcés par RLI

Dans ce qui suit, nous proposons d'évaluer les deux AG^{+heur} sans et avec RLI. Le tableau 5.15 présente les résultats trouvés par $AG1^{+heur}$ sans et avec RLI.

TABLE 5.15 – Évaluation des E_m^* trouvés par AG1 sans et avec RLI

n1=	**n**	**Classe 1**				**Classe 2**				**Classe 3**			
		AG1		**AG1+ RLI**		**AG1**		**AG1+ RLI**		**AG1**		**AG1+ RLI**	
		Em*	**Tm**	**Em***	**Tm**	**Em***	**Tm**	**Em***	**Tm**	**Em***	**Tm**	**Em***	**Tm**
0,5n	10	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	20	6.37	0.0	5.35	1.0	1.51	0.0	0.86	0.0	2.92	1.0	2.08	1.0
	50	10.11	1.0	6.22	6.0	3.79	1.0	2.18	6.0	8.6	2.0	6.65	7.0
	100	9,4	3.0	5.12	42.0	5.77	3.0	3.65	43.0	12.06	3.0	8.88	39.0
	200	12,2	7.0	6.92	319.0	9.5	7.0	5.64	314.0	12.06	7.0	5.81	353.0
0,6n	10	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	20	5.42	1.0	4.43	1.0	0.49	1.0	0.4	1.0	0,9	0.0	0.13	1.0
	50	5.89	1.0	3.1	7.0	0.51	1.0	0.19	7.0	0.6	1.0	0.2	7.0
	100	4.31	3.0	1.84	45.0	0.55	3.0	0.22	44.0	1.35	3.0	0.5	46.0
	200	4.7	7.0	1.7	288.0	1.46	6.0	0.54	284.0	2.92	7.0	1.47	281.0
0,7n	10	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	20	5.78	1.0	4.89	1.0	0.47	0.0	0.45	1.0	0.47	1.0	0.16	0.0
	50	4.24	1.0	2.27	7.0	0.24	2.0	0.22	6.0	0.11	2.0	0.53	6.0
	100	2.61	3.0	1.16	44.0	0.23	3.0	0.28	43.0	0.15	3.0	1.34	43.0
	200	2.24	6.0	0.83	310,0	0.36	7.0	0,20	308.0	0.57	7.0	2.23	310,0

Dans un premier temps, nous remarquons qu'AG1 est plus performant dans la classe 2 des données. L'écart E_m^* est inférieur à 1% dans 9/15 familles d'instances. Ceci est aussi vrai pour la classe 3. Dans le cas de la classe 1 ; si on ignore les problèmes à 10 jobs, les écarts sont supérieur à 2%. Pour cette raison, l'effet de la recherche itérative n'est pas très important avec la classe 2 et la classe 3. Les écarts les plus élevés, pour les classes 2 et 3, se sont produits avec les instances où le nombre des jobs est n=100 et n=200. Ceci n'est pas confirmé dans le cas de la classe 1.
En termes de temps d'exécution, AG1 sans RLI est rapide, où il a pu produire des solutions de bonne qualité en courte durée même pour les instances de grande taille (200 jobs dans les 3 classes). Ceci montre qu'AG1 converge rapidement vers la meilleure solution. Cependant, faire combiner l'AG1 avec la recherche locale itérative augmente la durée d'exécution sans apporter une grande amélioration (notamment les instances des classes 2 et 3 où les jobs ne sont pas distribués en équilibre sur le premier étage) mais cette durée reste raisonnable. Cela est expliqué par le fait que

la recherche locale est en train de visiter plusieurs solutions symétriques.
Le tableau 5.16 présente les résultats trouvés par $AG2^{+heur}$ sans et avec RLI

TABLE 5.16 – Évaluation des E_m^* trouvés par AG2 sans et avec RLI

		Classe 1				**Classe 2**				**Classe 3**			
n1=	**n**	**AG2**		**AG2 +RLI**		**AG2**		**AG2 +RLI**		**AG2**		**AG2 +RLI**	
		Em*	**Tm**	**Em***	**Tm**	**Em***	**Tm**	**Em***	**Tm**	**Em***	**Tm**	**Em***	**Tm**
0,5n	10	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	20	7.11	0.0	5.46	0.0	1.8	0.0	0.61	1.0	3.36	0.0	1.97	1.0
	50	11.54	1.0	5.15	4.0	3.76	1.0	1.94	8.0	7.74	1.0	6.16	13.0
	100	8.77	3.0	4.34	45.0	4.15	3.0	2.0	55.0	9.81	3.0	8.27	136.0
	200	9.87	9.0	5.13	401,0	5.99	8.0	3.53	368.0	8.40	9.0	5.58	401.0
0,6n	10	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	20	6.08	0.0	4.76	2.0	0.71	0.0	0.4	0.0	0.85	0.0	0.18	1.0
	50	7.17	1.0	3.38	15.0	1.51	1.0	0.33	8.0	4.23	1.0	1.62	11.0
	100	5.85	3.0	1.99	129.0	1.99	3.0	0.29	53.0	6.33	3.0	3.57	95.0
	200	6.39	9.0	1.62	332,0	3.39	9.0	0.88	390.0	8.07	10.0	5.83	419.0
0,7n	10	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
	20	7.13	0.0	5.05	0.0	0.61	0.0	0.47	0.0	0.51	0.0	0.19	0.0
	50	6.22	1.0	2.23	5.0	1.04	1.0	0.25	7.0	2.15	1.0	0.67	5.0
	100	4.43	3.0	1.24	47.0	1.52	3.0	0.2	55.0	3.59	3.0	1.72	48.0
	200	3.76	9.0	0.81	311,0	1.94	9.0	0.33	367.0	4.9	10.0	3.43	427.0

Nous remarquons qu'AG2 est plus performant aussi dans la classe 2 de données. L'écart E_m^* est inférieur à 1 % dans 9/15 familles d'instances. Ceci est vrai pour seulement 3/15 familles d'instances de la Classe 3 (en ignorant les instances de 10 jobs).
Dans le cas des classes 1 et 3 ; si on ignore les problèmes à 10 jobs, les écarts sont supérieurs à 2%. Pour cette raison, l'application de la recherche itérative a amélioré tous les écarts qui sont supérieurs à 2% dans les 3 classes. Les écarts les plus élevés sont trouvés seulement dans la classe 3 avec les instances de grande taille (n=100 et n=200), en plus les jobs sont distribués en équilibre sur les 2 machines dédiées (n1 = 0.5n). Ceci n'est pas confirmé dans le cas des classes 2 et 3.
En termes de temps d'exécution, AG2 sans RLI est rapide. Il a pu produire des solutions de bonne qualité dans un temps raisonnable (le E_m^* maximal trouvé est de l'ordre de 9.5% qui correspond aux instances de 200 jobs). Ceci montre qu'AG2 converge rapidement vers une bonne solution. Cependant, faire combiner l'AG2 avec la recherche locale itérative augmente considérablement le temps d'exécution en apportant une amélioration significative (toutes les instances).

5.8.2 Comportement des AGs suivant le critère d'arrêt : durée d'exécution

Dans la section précédente, la recherche locale itérative a été appliquée à la population de la dernière génération de l'AG. La RLI a amélioré la solution mais a pris une durée d'exécution non

négligeable
La question est, si on laisse l'AG tourner pour la même durée, est-ce qu'il aurait généré des résultats comparables ?

Résultats pour la première catégorie de données

A partir du tableau 5.17 , nous notons que les 2 AGs sans RLI n'ont pas donné des résultats meilleurs que ceux trouvés par les AGs avec RLI malgré la prolongation de la durée d'exécution des AG. Toutefois, attribuer un temps d'exécution plus grand aux AGs renforcés par l'heuristique H1 a diminué clairement les valeurs E_m^* (comparaison entre tableau 5.17 avec tableau 5.9 et 5.10). Ceci s'explique par la recherche locale itérative qui cherche de nouvelles solutions dans le voisinage d'individu et par conséquent elle sort de minima locaux.

TABLE 5.17 – Comportement de AG1 et AG2 sans et avec RLI avec la même durée d'exécution

n	**AG1**	**AG1+ RLI**	**Tm**	**AG2**	**AG2+ RLI**	**Tm**
	Em*	**Em***		**Em***	**Em***	
10	0,00	0,00	0,00	0,00	0,00	0,00
20	2,38	1,90	1,00	2,50	2,07	1,00
50	3,70	1,96	5,00	2,69	1,48	5,00
100	4,30	2,40	28,00	3,39	1,86	39,00
200	5,96	3,54	210,00	4,23	2,24	283,00

Résultats pour la deuxième catégorie de données

Les observations effectuées plus haut sont aussi valables pour la deuxième catégorie. En autorisant plus de temps aux deux AGs sans RLI, les valeurs E_m^*, dans les tableaux 5.18 et 5.19, sont améliorées par rapport à celles trouvées par AG1 et AG2 arrêtés suivant le nombre d'itérations (tableaux 5.15 et 5.16). Cependant les nouveaux résultats trouvés ne dépassent pas ceux trouvés par les 2 AGs combinés avec RLI.

TABLE 5.18 – Comportement de AG1 sans et avec RLI avec la même durée d'exécution

n1=	n	Classe 1			Classe 2			Classe 3		
		AG1	AG1+ RLI	Tm	AG1	AG1+ RLI	Tm	AG1	AG1+ RLI	Tm
		Em*	Em*		Em*	Em*		Em*	Em*	
0,5n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	6,17	5,35	1,00	1,14	0,86	1,00	2,70	2,08	1,00
	50	10,11	6,22	6,00	3,69	2,18	6,00	8,49	6,65	7,00
	100	9,39	5,56	42,00	5,68	3,65	43,00	12,38	9,26	39,00
	200	12,32	6,92	319,00	9,55	5,64	314,00	15,81	10,91	353,00
0,6n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	5,34	4,43	1,00	0,44	0,40	1,00	0,16	0,13	1,00
	50	5,72	3,10	7,00	0,35	0,19	7,00	0,52	0,20	7,00
	100	4,25	1,84	45,00	0,42	0,22	44,00	1,26	0,50	46,00
	200	4,73	1,70	288,00	1,44	0,54	284,00	2,98	1,47	281,00
0,7n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	5,46	3,41	1,00	0,47	0,47	1,00	0,16	0,47	0,00
	50	4,11	2,27	7,00	0,18	0,22	6,00	0,06	0,53	6,00
	100	2,60	1,16	44,00	0,19	0,28	43,00	0,12	1,34	43,00
	200	2,03	0,83	310,00	0,36	0,20	308,00	0,57	2,23	310,00

TABLE 5.19 – Comportement de AG2 sans et avec RLI avec la même durée d'exécution

n1=	n	Classe 1			Classe 2			Classe 3		
		AG2	AG2 +RLI	Tm	AG2	AG2 +RLI	Tm	AG2	AG2 +RLI	Tm
		Em*	Em*		Em*	Em*		Em*	Em*	
0,5n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	6,18	5,46	0,00	1,25	0,61	1,00	2,92	1,97	1,00
	50	9,51	5,15	4,00	2,66	1,49	8,00	7,24	6,16	13,00
	100	7,57	4,34	45,00	3,63	2,00	55,00	10,14	8,89	136,00
	200	9,25	5,13	401,00	5,91	3,53	368,00	12,12	9,51	401,00
0,6n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	5,37	4,76	2,00	0,47	0,40	0,00	0,35	0,18	1,00
	50	5,60	3,38	15,00	0,86	0,33	8,00	3,64	1,62	11,00
	100	4,50	1,99	129,00	1,68	0,29	53,00	5,86	3,57	95,00
	200	5,62	1,62	332,00	3,17	0,88	390,00	8,01	5,83	419,00
0,7n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	5,42	5,05	0,00	0,48	0,47	0,00	0,22	0,19	0,00
	50	4,87	2,23	5,00	0,50	0,25	7,00	1,99	0,67	5,00
	100	3,30	1,24	47,00	1,03	0,20	55,00	3,28	1,72	48,00
	200	3,11	0,81	311,00	1,87	0,33	367,00	4,82	3,43	427,00

5.9 Analyse des résultats dans les deux catégories de données

En observant les écarts trouvés dans les 2 catégories de données, il est clair que la méthode approchée proposée est plus efficace dans la première catégorie, ceci est expliqué par la spécificité des instances résolues où la contrainte no-wait est plus présente. Le problème étudié constitue une configuration plus simple $FH2(PD2, Pm)|no - wait|C_{max}$ et qu'une solution optimale est vite trouvée sur la base de ECT-FAM.
Dans la deuxième catégorie de donnée, les tableaux 5.15 et 5.16 montrent que les classes 1 et 3 sont plus difficiles (cas n1=0.5 n) à résoudre par l'un ou l'autre des deux AG. Cependant, on trouve que AG2+RLI est plus performant à résoudre ces deux classes (cas n1=0.5 n), sauf les instances de 20 jobs. Dans le cas où n1=0.6 et n1= 0.7 n AG1+RLI s'est avéré meilleur.
Dans la classe 1, on rappelle que les intervalles des durées de traitement sont comme suit :[1-40] sur E1 et [5-200] sur E2. Le chevauchement entre les intervalles des durées de traitement, fait qu'en général, que les machines de l'E1 sont plus chargées.
Dans la classe 3, les intervalles des durées de traitement sont comme suit :[80-100] sur E1 et [400-600] sur E2. On se trouve fréquemment, incapable d'éviter les retards de lancement des jobs sur l'E1. Cependant, l'écart important entre les durées de traitement sur les deux étages va induire généralement un dépassement de la borne inférieure.
Dans les 3 classes, nous constatons que, plus le nombre de jobs affectés à une machine dédiée est élevé (n1=0.7n), plus les solutions trouvées sont basées sur un assemblage des jobs par groupes. Ce constat explique la performance des deux AG dans les 3 classes où n1=0.7 n.

5.10 Comparaison AG+RLI avec l'algorithme mémétique

Afin d'étudier d'avantage l'efficacité de l'AG renforcé par la recherche locale itérative, nous proposons de comparer les valeurs Em^* trouvés par AG1+RLI à la version de l'AM décrite dans la section 5.7. Dans un premier temps, nous avons effectué une première expérimentation avec l'AM+RLI. Les résultats trouvés n'étaient pas satisfaisant par rapport à AG1+RLI. En analysant le déroulement de l'AM+RLI, nous avons remarqué que cet algorithme s'arrête après avoir créé un nombre limité de générations. En effet, la RLI est appliquée à chaque nouveau fils. Dans la plupart des instances, le temps fixé à l'AM est dispersé dans la recherche itérée de la meilleure solution ce qui entraine un blocage dans l'amélioration de la qualité de générations (nous sommes face à un algorithme coincé à un nombre limité de générations). Pour cette raison, nous avons décidé d'appliquer une recherche locale légère dans l'AM. Autrement, la RL est arrêtée suivant un nombre réduit d'améliorations (fixé à 5) ou si le compteur de non-amélioration est atteint (10).

TABLE 5.20 – AG1+RLI VS l'algorithme mémétique

n1=	n	Classe 1			Classe 2			Classe 3		
		AG1+ RLI	AM	Tm	AG1+ RLI	AM	Tm	AG1+ RLI	AM	Tm
		Em*	Em*		Em*	Em*		Em*	Em*	
0,5n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	5,35	5,37	1,00	0,86	2,31	1,00	2,08	3,99	1,00
	50	6,22	10,05	6,00	2,18	4,03	6,00	6,65	10,15	7,00
	100	5,56	7,41	42,00	3,65	4,89	43,00	9,26	12,55	39,00
	200	6,92	8,39	319,00	5,64	8,00	314,00	10,91	13,09	353,00
0,6n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	4,43	4,52	1,00	0,43	1,26	1,00	0,13	0,81	1,00
	50	3,10	8,30	7,00	0,19	1,30	7,00	0,20	0,75	7,00
	100	1,84	6,11	45,00	0,22	0,93	44,00	0,50	1,16	46,00
	200	1,70	3,96	288,00	0,54	2,61	284,00	1,47	2,44	281,00
0,7n	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	20	3,41	4,63	1,00	0,47	0,52	1,00	0,47	0,47	1,00
	50	2,27	4,88	7,00	0,22	0,25	6,00	0,53	0,96	6,00
	100	1,16	3,74	44,00	0,28	0,31	43,00	1,34	1,37	43,00
	200	0,83	3,62	310,00	0,20	0,64	308,00	2,23	2,56	310,00

Dans le tableau 5.20, nous notons une supériorité claire de AG1+RLI par rapport à l'AM dans les 3 classes. Cependant, nous remarquons que les deux approches génèrent des EM^* proches dans la classe 2 et 3 où n1=0.7 n.
Nous avons estimé que nous allons déduire les mêmes conclusions en comparant l'AM avec AG2+ILS. Ceci explique l'absence de la comparaison de l'AM avec AG2+RLI. Il convient de signaler que nous avons testé l'AM seulement dans la catégorie 2 qui porte des instances variées.

5.11 Conclusion

Dans ce chapitre, nous avons présenté une première méthode approchée de résolution, les algorithmes génétiques. Deux AGs ont été proposés pour s'approcher des dates optimales d'achèvement de production. La génération de la population a été renforcée par une génération spécifique (introduire les règles de priorité et une heuristique) et l'hybridation des AGs avec une recherche locale itérative.
Afin de tester l'efficacité de la méthodologie proposée, nous avons utilisé un grand nombre d'instances. D'autre part, nous avons mené une comparaison théorique et numérique entre les différentes versions d'AGs (en termes de rapidité de la convergence, de qualité de la meilleure solution trouvée et du temps de calcul).
Le renforcement des AGs par une RLI a permis de trouver des solutions de bonne qualité et meilleures que les solutions obtenues par un solveur avec un temps de calcul raisonnable. L'AG1 renforcé par une recherche locale itérative s'est avéré aussi meilleur que l'algorithme mémétique. Selon les résultats trouvés, L'AG1+ RLI a surpassé AG2+RLI dans les 3 classes de la catégorie 2 de données dans le cas où il ya un déséquilibre de charges entre les machines dédiées. Pour le reste des instances de données, AG2+RLI s'est avéré plus efficace.

Chapitre 6

Méthode arborescente approchée

Sommaire

6.1 Introduction

Dans ce chapitre, nous développons une troisième approche de résolution de type heuristique basée sur l'arborescence afin de résoudre d'une manière plus efficace le problème d'ordonnancement qui nous intéresse. Dans un premier temps, nous allons présenter le principe général de la méthode. Nous décrivons ensuite l'algorithme utilisé. Une analyse empirique suit au niveau de la section 6.4 et nous clôturons le chapitre par une conclusion.

6.2 Principe général

Les heuristiques arborescentes (HA) sont des méthodes heuristiques inspirées de la procédure de séparation&évaluation (PSE). Elles visent à obtenir des ordonnancements proches de l'optimal dans un temps de calcul raisonnable.

Pour bien décrire le principe de cette méthode, nous nous appuyons sur deux travaux de la littérature. Le premier travail est de [Wang and Liu, 2013b]. Les auteurs ont proposé une méthode heuristique basée sur l'algorithme de PSE pour résoudre un $FH2(1, PDm)||C_{max}$.
La procédure proposée est classée comme étant une heuristique pour les trois raisons ci-dessous :

— Le 1er niveau de l'arbre, comporte m branches (m étant le nombre de machines dédiées qui est généralement inférieur à n le nombre de jobs) au lieu de n branches. Donc on parcourt le risque de ne pas visiter une solution optimale où le job en position 1 correspond à un nœud non crée au niveau 1 ;

— Uniquement des solutions de permutation sont considérées. Donc si la solution optimale est une solution de non-permutation, on risque de ne pas l'explorer ;

— La durée d'exécution de l'algorithme a été limitée à 60 secondes.

Afin d'étudier l'efficacité de la procédure proposée, les auteurs ont comparé HA, dans un premier temps, à une méthode PSE classique, où n nœuds sont explorés au niveau 1 de l'arbre, et sans avoir appliqué des règles de dominances. Dans un deuxième temps, ils ont comparé la performance de l'heuristique à deux procédures basées sur le recuit simulé et la recherche tabou. Les résultats trouvés par la méthode arborescente est plus efficace.
Un deuxième travail consacré à une heuristique arborescente a été proposée par [Morizawa, 2014]. La procédure est appelée « List Based sequenzing B&B (LSQ)», la procédure insère une recherche locale à l'intérieur du « sequenzing B&B ».
L'atelier considéré est composé de deux étages. Le premier étage comporte deux machines parallèles. Le deuxième étage est une machine unique chargée de l'assemblage des jobs traités sur l'étage 1.
Les auteurs ont ajouté une règle au branchement qui élimine certains nœuds de l'arbre ayant une borne inférieure plus élevée que $(1 + \alpha)$LB0 (LB0 est la borne inférieure initiale). Dans ce contexte, les nœuds parents à un certain niveau sont sélectionnés sur la base de la plus petite LB. Ensuite un nœud ayant une LB inférieure à $(1 + \alpha)$LB0 n'est pas sélectionné. α est choisi entre 0 et 1. La procédure se termine lorsque un ordonnancement complet est construit (une feuille de l'arbre est atteinte). Ce qui caractérise cet algorithme qu'il n'ya pas un bactracking. Ceci réduit sa complexité. Contrairement au premier travail de wang, cette procédure ne considère que les bons ordonnancements de permutation.

6.3 Description du l'algorithme HA

Dans ce qui suit, nous présentons tout d'abord des notations utilisées dans l'heuristique HA, ensuite nous présentons sa structure générale.

6.3.1 Les notations propres à l'heuristique HA

Il convient de rappeler que les notations d'ordre général relatives au problème ont été présentées dans la section 2.3 du chapitre 2.

— $N0$: racine de l'arbre où aucun job n'est fixé

— $LB0$: borne inférieure initiale

— $UB0$: borne supérieure initiale correspond à N0

— $\overline{\delta}$: ensemble de tous les jobs non encore fixés

— $\overline{\delta_i}$: ensemble des jobs non encore fixés passant sur A_i

— LJ : liste des jobs suivant l'ordre des groupes

— σ : ordonnancement partiel dans lequel les jobs ne sont pas tous fixés

— UB^* : meilleure borne supérieure enregistrée

— σ^* : ordonnancement correspondant à UB^*

— Np : le nœud-parent du nœud N

— $LB(N)$: borne inférieure de l'ordonnancement partiel σ

— $UB(N)$: borne supérieure calculée au niveau du nœud N

— $Niv(N)$: niveau du nœud N

— Q : la pile des nœuds qui sera alimentée à chaque exploration

Dans ce chapitre, l'algorithme proposé se base sur 3 étapes :

1. **Étape 1** : création de la liste des jobs (LJ) suivant l'ordre chronologique des groupes.
   - créer deux sous-listes L1 et L2. L1 correspond au jobs qui passent sur M1 et qui sont assemblés suivant le groupe. L2 correspond aux jobs qui passent sur M2 et qui sont assemblés suivant le groupe.
   - créer une liste LJ qui combine L1 et L2. Il faut signaler que les jobs de LJ sont sélectionnés des deux listes L1 et L2 alternativement, c.à.d. le premier job de LJ est copié de L1, le deuxième job est copié de L2, le troisième est de L1 ainsi de suite jusqu'à copier tous les jobs.
2. **Étape 2** : création et exploration de l'arbre :
   1. Initialisation : créer le nœud racine (N0)
   2. Créer les nœuds de premier niveau, dans chaque nœud, on fixe un job de la liste LJ
   3. Calculer les bornes inférieures des nœuds du premier niveau pour choisir le nœud à séparer ;
   4. Séparation du nœud ayant la borne inférieure minimale ;
   5. Arrêter l'exploration de l'arbre si condition d'arrêt est satisfaite ;
3. **Étape 3** : appliquer une recherche locale légère sur la solution finale (arrêt après 3 améliorations) présentée dans le chapitre 5 ;
4. **Étape 4** : Explorer partiellement des solutions de non permutation :
   - parcourir la permutation finale, si deux jobs i et j sont deux jobs successifs sur la même machine dédiée ayant $l_i > p_{i,1}$, alors inverser l'ordre de i et j sur l'E1 sans modifier l'ordre sur l'E2.
   - Vérifier si la solution modifiée est meilleure que la solution de permutation. Cette instruction est basée sur l'observation de quelques solutions optimales trouvées par les deux MIPs. En effet, nous avons observé, que lorsque le time lag est supérieur à la durée de traitement d'un job, une solution de non-permutation peut dominer une solution de permutation.

6.3.2 Schéma de branchement

Nous avons implémenté un schéma de branchement classique connu dans la littérature comme suit : nous partons d'un nœud-racine où aucun job n'est fixé. Le premier niveau de l'arbre contient n branches (n est le nombre de jobs). La première branche séquencera le premier job de la liste LJ dans la première position disponible, la deuxième branche séquence le deuxième job de LJ dans la première position, ainsi de suite jusqu'à fixer tous les jobs dans l'ordre. Nous avons opté pour une exploration en profondeur. A chaque exploration, on fixe un job de LJ.
Pour gérer les nœuds à séparer, nous avons mis en place une pile Q. A chaque branche visitée, les nœuds explorables sont ajoutés à Q suivant l'ordre croissant des LB. La tête de Q sera toujours le nœud à visiter. La procédure de Backtraking est gérée automatiquement par la pile Q. Une fois qu'un nœud est exploré, il sera supprimé de la pile Q.

6.3.3 Estimation des bornes au niveau des nœuds

Dans ce qui suit, nous présentons les bornes inférieures et supérieures calculées au niveau de chaque nœud de l'arbre afin de réduire l'espace de recherche.

Borne supérieure au niveau du nœud racine

La borne supérieure est appelée en premier temps au niveau du nœud-racine « root ». Elle est calculée à partir des deux algorithmes génétiques présentés dans le chapitre précédent AG1 et AG2. Ainsi, la borne supérieure initiale UB0 est donnée par la meilleure valeur de Cmax trouvée par les deux AGs : UB0 = min { Cmax(AG1+ RLI) ;Cmax(AG2+RLI) }.

Calcul des bornes supérieures à l'intérieure de l'arbre

Une borne supérieure est calculée si un nœud est une feuille. Autrement, un ordonnancement complet est construit en passant par tous les niveaux de l'arbre.

Calcul des bornes inférieures à l'intérieur de l'arbre

Nous signalons tout d'abord que la détermination de la borne inférieure au niveau de la racine a été présentée au chapitre 4 (lorsqu'aucun job n'est fixé). Nous proposons quatre bornes inférieures (LBs), afin d'évaluer la faisabilité des nœuds explorés. A chaque exploration, nous calculons les LBs.

1. 1ère borne inférieure :
   Cette borne est une extension de LB4 (à la racine), présentée à la section 4.2.

$$LB1 = Max \begin{cases} CT^{M1} + \sum\limits_{j \in \overline{\delta_1}} P_{j,1} + \sum\limits_{i=1}^{u1} \mathcal{S}_{i,1}.\beta_i + \underbrace{Min \quad P_{j,2}}_{j \in \overline{\delta_1}}, \\ CT^{M2} + \sum\limits_{j \in \overline{\delta_2}} P_{j,1} + \sum\limits_{i=1}^{u2} \mathcal{S}_{i,2}.\beta_i + \underbrace{Min \quad P_{j,2}}_{j \in \overline{\delta_2}} \end{cases}$$

tel que β_i *=1 si il* $\exists$ *au moins un job* $\in \delta_1$ *ou* $\in \delta_2$ *du groupe i*

En effet, la fin de traitement sur chaque machine d'E1 ne peut pas être inférieure à la disponibilité des machines sur E1, plus la durée totale de traitement des jobs non encore affectée et la durée minimale des setups. Au niveau de l'E2, on considère que le dernier jobs ordonnancé est celui ayant la durée de traitement minimale sur cet étage.

2. 2ème borne inférieure :
   On sait que la durée moyenne des traitements est une borne inférieure pour les machines parallèles identiques ([Gupta et al., 1997]). On considère les durées de fin de traitement des jobs ordonnancés sur E1 et la durée moyenne de traitement des jobs non ordonnancés sur E2.Cette borne est une extension de LB5 présentée dans le chapitre 4

$$LB2 = Min \left\{ \begin{array}{l} CT^{M1} + \underbrace{Min \quad P_{j,1}}_{j \in \overline{\delta_1}} \\ CT^{M2} + \underbrace{Min \quad P_{j,1}}_{j \in \overline{\delta_2}} \end{array} \right\} . + 1/m \sum_{j \in \overline{\delta}} P_{j,2}$$

3. 3ème borne inférieure
   Cette borne suppose que suite aux jobs ordonnancés, la disponibilité des machines parallèles à l'E2 ne peut pas être inférieure à la première machine disponible sur E2 (inspiré de [Wang et al., 2015])

$$LB3 = \underbrace{Min \quad AVi}_{i \in M_2} + 1/m \sum_{j \in \overline{\delta}} P_{j,2}$$

4. 4ème borne inférieure
   On peut améliorer LB3 si on assimile les AVi (i=1..10) à des durées de traitement de 10 jobs fictifs. Dans ce cas :

$$LB4 = 1/m \sum_{i=1}^{m} AVi + 1/m \sum_{j \in \overline{\delta}} P_{j,2}$$

   Au niveau de chaque nœud « N », on détermine LB1, LB2 et LB4 (étant donné que LB4 domine toujours LB3) ensuite on choisit la meilleure.

6.3.4 Synthèse de l'algorithme HA

L' algorithme ci-dessous (algorithme 6) ne présente pas les étapes 3 et 4 de la procédure proposée dans ce chapitre.

Algorithme 6 Pseudo code de HA

Début

Étape 1 : initialisation : créer la liste LJ ; créer le nœud N0,
niv(N0)=0, UB(N0), LB0 , Np=N0 ;
UB^* = UB(N0) ;
Étape 2 : séparation du nœud Np
Pour chaque j ∈ LJ faire
Création du nœud N descendant du Np
niv(N)= niv(Np)+1 ;
Si niv(N) = n-1 alors N est une feuille
Si (UB(feuille)< UB* **alors** MAJ UB*
Calculer LB (N)
Si (LB (N) > LB0 (1+ α)) **alors**
    ne pas explorer ce nœud
Si (LB (N) ≥ UB*) **alors**
    ce nœud ne garantit pas une solution meilleure et donc supprimer le nœud N
Si (LB (N) < UB*) **alors**
    ajouter le nœud N à la pile Q
Étape 3 : sélection du nœud à séparer
Si Q est vide **alors** sortir (aller à l'étape 4)
Sinon sélectionner le nœud N qui attend en premier (la tête de la pile)
$Np \leftarrow N(Q.tete)$
aller à l'étape 2
Étape 4 : arrêter l'exploration de l'arbre
    Si UB* = LB0
    Si la durée d'exécution fixée est atteinte (60 secondes pour n=20 jobs, 150 secondes pour n = 50 jobs, 300 secondes pour n = 100 jobs, 500 secondes pour n = 200 jobs)

Fin

6.4 Analyse empirique

Dans ce qui suit, nous allons utiliser la bibliothèque d'instances proposée dans la section 2.4 du chapitre 2.
Il convient de rappeler que nous testons 30 instances par problème. Nous commençons par choisir expérimentalement la valeur de α. Ensuite, nous proposons d'évaluer la qualité des solutions générées par HA. Finalement, nous proposons à la fois de valider l'optimalité des solutions générées par HA et d'améliorer d'autres solutions en se basant sur l'outil MIPstart de CPLEX.

6.4.1 Évaluation de α

Afin de bien évaluer l'heuristique HA, nous avons testé des différentes valeurs de α et nous avons calculé le nombre des solutions optimales trouvées pour chaque valeur. Dans le tableau ci-dessous (tableau 6.1), nous présentons la capacité de HA à résoudre à l'optimalité un échantillon

de test. En effet, nous avons choisi 3 instances de chaque famille d'instances (de 10, 20, 50, 100 et 200 jobs) des deux catégorie de données.

TABLE 6.1 – Évaluation de α

	Taux de résolution à l'optimalité (%)			
	Catégorie 1	**Catégorie 2**		
α		**Classe 1**	**Classe 2**	**Classe 3**
0,01	[60-80]	[42-65]	[55-70]	[29-58]
0,02	[30-45]	[10-40]	[12-46]	[7-42]
0,03	[0-5]	[0-8]	[0-10]	0
0,05	0	0	[0-2]	0

Les résultats trouvés dans le tableau 6.1 confirme que la capacité de HA à trouver des solutions optimales diminue en augmentant α. Une meilleure résolution est trouvée avec $\alpha = 0.01$. Ceci montre clairement l'augmentation de α élimine des nœuds prometteurs qui aboutissent à des solutions optimales. D'autre part, introduire une faible valeur de α permet d'élargir la recherche dans l'arbre en évitant d'explorer des nœuds ayant la même valeur de LB.
Dans la suite de cette analyse, nous allons mener toutes les expérimentations numériques avec $\alpha = 0.01$.

6.4.2 Évaluation de HA

Pour cette section, nous proposons tout d'abord de mesurer la performance de l'heuristique HA en mesurant l'écart de la meilleure solution trouvée par rapport à LB0. En effet, pour chaque instance nous mesurons E%, comme suit :

$$\text{E (\%)} = (UB^*\text{-LB0})/\text{LB0} * 100.$$

Ensuite nous déterminons la EM% qui est la moyenne des écarts sur toutes les instances concernées. Il convient de rappeler que nous avons fixé le temps d'exécution de HA à 60 secondes pour les instances de 20 jobs, 150 secondes pour les instances de 50 jobs, 300 secondes pour les instances de 100 jobs et finalement 500 secondes pour les instances de 200 jobs.

Comparaison du HA avec le meilleur AG

Nous comparons l'écart EM% des solutions trouvées par HA à EM% de Max (AG1+RLI, AG2+RLI) puisque nous avons démarré notre algorithme à partir d'une UB0 = (AG1+RLI, AG2+RLI). Les tableaux présentés ci-dessous permettent également d'évaluer les améliorations apportées par HA. Dans les tableaux présentés ci-dessous, les colonnes présentent :

— n : nombre de jobs

— n1 : distribution des jobs entre les machines dédiées de l'E1 (pour la catégorie 2 de données)

— N-opt : nombre des instances résolues à l'optimum

— EM : la moyenne de écarts pour les instances non résolues
— Min-E : l'écart minimal trouvé parmi les instances non résolues
— Max-E : l'écart maximal parmi les instances non résolues
— Tm : la durée moyenne d'exécution

Résultats : 1ère catégorie de données

Le tableau ci-dessous (6.2) présente les résultats des instances inspirées du cas réel. Dans ce tableau, ainsi que les tableaux 6.3 à 6.5, la colonne N-opt donne le nombre de solutions reconnues comme étant optimales par l'algorithme arborescent. Nous verrons plus tard qu'il existe d'autres solutions générées par HA, qui ne sont pas reconnues comme optimales par HA. L'optimalité de ces solutions est confirmée par le MIP start.

TABLE 6.2 – Résultats HA : instances inspirées du cas réel

n	**max (AG1,AG2)**	**HA**				
	EM	**EM**	**N-opt**	**Min-E**	**Max-E**	**Tm**
10	0,00	0,00	30	0,00	0,00	0,00
20	1,90	0,84	25	0,46	1,22	60,00
50	1,48	0,99	15	0,22	2,33	300,00
100	1,86	1,12	8	0,39	2,71	300,00
200	2,24	1,86	6	0,62	3,04	900,00

Comme nous remarquons, le HA est plus efficace que les AGs dans cette catégorie. En effet, nous remarquons que toutes les instances de 10 jobs et plus que 50% des instances de 20 et 50 jobs sont résolues à l'optimum.
En ce qui concerne les instances de grande taille, HA a réussi à trouver des solutions optimales à 27% des instances de 100 jobs et à 20% des instances de 200 jobs. Pour le reste des instances, HA a généré des solutions proches de l'optimal car le EM trouvé des instances non résolues ne dépasse pas 1.9%.

Résultats : 2ème catégorie de données

Les tableaux suivants (6.3 à 6.5) représentent les résultats du HA pour les 3 classes de données.

Résultats de la classe 1

Le tableau 6.3 présente les résultats trouvés dans la classe 1.

TABLE 6.3 – Résultats HA : classe 1

n1=	n	Classe 1						
		max (AG1,AG2)		HA				
		EM	Tm	EM	N-opt	Min-E	Max-E	Tm
0,5n	10	0,05	0,00	0,00	30	0,00	0,00	0,00
	20	5,46	1,00	4,73	10	0,30	13,29	60,00
	50	5,15	6,00	3,82	8	1,24	7,36	300,00
	100	4,34	42,00	4,20	5	1,90	13,42	300,00
	200	5,13	319,00	4,09	4	2,20	10,13	900,00
0,6n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	4,43	1,00	3,26	10	2,31	6,29	60,00
	50	3,10	7,00	2,86	9	1,87	5,33	300,00
	100	1,84	45,00	1,01	8	0,94	2,94	300,00
	200	1,62	288,00	0,96	6	0,88	2,61	900,00
0,7n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	3,41	1,00	2,67	18	0,93	3,78	60,00
	50	2,23	7,00	1,89	12	1,10	4,20	300,00
	100	1,16	44,00	0,99	10	0,81	1,90	300,00
	200	0,81	310,00	0,78	8	0,67	2,33	900,00

Selon le tableau 6.3, le HA est plus efficace que le meilleur des deux AGs. Dans cette classe, le nombre des instances résolues à l'optimum n'est pas élevé.
Dans le cas où n1= 0.5, 0.6 n, les pourcentages des instances résolues à l'optimum sont comme suit :
- toutes les instances de 10 jobs ;
- entre 26 % et 34% des instances de 20 et 50 jobs ;
- entre 13 % et 27 % des instances de 100 et 200 jobs.

Dans le cas où n1= 0.7 n, on remarque qu'il y a plus des instances résolues à l'optimum :
- toutes les instances de 10 jobs ;
- entre 40% et 60% des instances de 20 et 50 jobs ;
- entre 26% et 34 % des instances de 100 et 200 jobs.

Dans le cas des instances non résolues à l'optimum, on note que les EM trouvés sont plus élevés quand il s'agit d'une balance des charges entre les machines dédiées. Cependant, ces écarts ne dépassent pas les 4.8%.
Quand il s'agit d'un déséquilibre des charges, l'heuristique HA permet de générer des solutions quasi-optimales car les EM trouvés sont faibles et ne dépassant pas 3.26% . Nous déduisons que HA permet une exploration relativement meilleure qu'une exploration classique.

Résultats de la classe 2

Le tableau 6.4 résume les résultats trouvés pour la 2ème classe de données. Les résultats présentés montre que l'heuristique HA est plus efficace que le meilleur des deux AGs. Dans cette classe, le nombre des instances résolues à l'optimum s'est élevé par rapport à la classe 1.
Dans le cas où n1= 0.5, 0.6 n, les pourcentages des instances résolues à l'optimum sont comme suit :
- toutes les instances de 10 jobs ;
- entre 40% et 60% des instances de 20 et 50 jobs ;
- entre 10 % et 44 % des instances de 100 et 200 jobs.

Dans le cas où n1= 0.7 n, HA à trouvé des solutions optimales pour plus que 83% des instances. Pour le reste des instances non résolues à l'optimalité, HA génère des solutions quasi-optimales. Cette déduction est confirmée avec le faible EM trouvé $\forall$ la distribution des charges sur les machines dédiées : inférieur à 1% sauf pour les instances de 100 et 200 jobs où il ya un équilibre de charges. Pour ces instances, les solutions trouvées sont de bonne qualité avec des écarts moyens ne dépassant pas 2.45 %
Dans cette classe de données, HA est plus efficace. L'efficacité de l'heuristique HA est expliquée par son exploration non classique de l'arbre en suivant la liste LJ. Ceci a permis de visiter un espace de recherche plus large.

TABLE 6.4 – Résultats HA : classe 2

n1=	**n**	**Classe 2**						
		max (AG1,AG2)		**HA**				
		EM	**Tm**	**EM**	**N-opt**	**Min-E**	**Max-E**	**Tm**
0,5n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	0,61	0,00	0,45	15	0,10	1,34	60,00
	50	1,49	6,00	0,98	12	0,33	2,06	300,00
	100	2,00	43,00	1,52	5	0,96	3,17	300,00
	200	3,53	314,00	2,44	3	1,75	4,96	900,00
0,6n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	0,40	1,00	0,39	18	0,12	0,86	60,00
	50	0,19	7,00	0,19	15	0,09	0,68	300,00
	100	0,22	44,00	0,22	13	0,13	0,77	300,00
	200	0,54	284,00	0,46	4	0,26	1,09	900,00
0,7n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	0,47	1,00	0,40	28	0,31	0,49	60,00
	50	0,22	6,00	0,20	27	0,09	0,35	300,00
	100	0,28	43,00	0,24	28	0,17	0,31	300,00
	200	0,20	308,00	0,18	25	0,02	0,52	900,00

Résultats de la classe 3

Le tableau 6.5 résume les résultats trouvés pour la 3ème classe de données. Dans cette classe, le nombre des instances résolues à l'optimum est faible quand il s'agit d'un équilibre de charges (sauf les instances de 10 jobs) :
- toutes les instances de 10 jobs ;
- entre 26% et 34% des instances de 20 et 50 jobs ;
- entre 3 % et 17 % des instances de 100 et 200 jobs.

Dans le cas où n1= 0.6 et n1=0.7n, les pourcentages des instances résolues à l'optimum est moyennement élevé :
- toutes les instances de 10 jobs ;
- entre 56% et 94% des instances de 20 et 50 jobs ;
- entre 13 % et 40 % des instances de 100 et 200 jobs.

Pour le reste des instances non résolues à l'optimalité, HA génère des solutions quasi-optimales pour les instances ayant une balance des charges entre les machines dédiées. En effet, nous avons obtenu un EM inférieur à 1.75%.
Dans le cas où n1=0.5 n, les instances sont les plus difficiles à résoudre ce qui explique les EM élevés trouvés entre 4.9 % et 7.1 % (sauf les instances de 20 jobs).
Dans cette classe de données, HA est moyennement efficace. Ceci est expliqué par la difficulté des instances due à la spécificité des intervalles des durées de traitements sur les 2 étages.

TABLE 6.5 – Résultats HA : classe 3

n1=	**n**	**Classe 3**						
		max (AG1,AG2)		**HA**				
		EM	**Tm**	**EM**	**N-opt**	**Min-E**	**Max-E**	**Tm**
0,5n	10	0,05	0,00	0,00	30	0,00	0,00	0,00
	20	1,97	1,00	1,56	10	1,32	4,08	60,00
	50	6,16	7,00	4,93	8	1,05	7,31	300,00
	100	8,89	39,00	6,84	5	2,66	8,37	300,00
	200	9,51	353,00	7,08	1	5,33	10,48	900,00
0,6n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	0,13	1,00	0,10	16	0,10	0,10	60,00
	50	0,20	7,00	0,16	10	0,11	0,36	300,00
	100	0,50	46,00	0,42	4	0,27	1,60	300,00
	200	1,47	281,00	1,13	7	0,36	1,38	900,00
0,7n	10	0,00	0,00	0,00	30	0,00	0,00	0,00
	20	0,47	0,00	0,18	28	0,14	0,23	60,00
	50	0,53	6,00	0,49	17	0,24	0,72	300,00
	100	1,34	43,00	1,26	12	0,55	2,97	300,00
	200	2,23	310,00	1,74	8	0,88	3,78	900,00

6.4.3 Validation et résolution par MIP start de CPLEX

Dans cette section, nous introduisions une technique proposée par IBM Cplex appelé MIP start ou démarrage à chaud. En effet, dans le but d'évaluer l'optimalité des solutions trouvées par l'algorithme HA, nous proposons de relancer le MIP 2 avec ces solutions. Le fonctionnement de MIP start est comme suit :
1- lecture de la solution ;
2- Le MIP cherche la solution fournie dans tout l'espace de recherche ;
3- Le MIP branche sur la solution trouvée et continue l'exploration ;
Le temps de calcul fixé à 2 heures. Le MIP va générer soit une solution optimale soit confirmer l'optimalité de la solution fournie initialement, soit l'améliorer. Nous rappelons que les MIPs sont limités à résoudre les instances de grande taille à cause de la saturation de la mémoire. Pour contourner ce problème, nous avons mener une étape de rélage de CPLEX et augmenter la taille mémoire utilisée. Dans ce qui suit, nous présentons les paramètres testés et retenues.

6.4.4 Réglage des paramètres CPLEX

La stratégie de résolution de CPLEX est contrôlée par divers paramètres, leur réglage pour un modèle constitue un moyen efficace pour améliorer les performances du processus de résolution. Des expériences préliminaires ont été menées à l'aide de l'outil de réglage de CPLEX afin de déterminer les valeurs des paramètres susceptibles de fournir les meilleures performances en tant qu'alternative possible au paramétrage par défaut. Toutes les expériences de CPLEX dans ce chapitre sont conduites avec une limite de temps de calcul de 2 heures. De plus, pour éviter une défaillance due à un manque de mémoire dans certaines instances de grande taille, nous avons défini une limite de mémoire de l'arbre à 1500 mégaoctets.
Les valeurs et les significations des paramètres pour chaque paramètre CPLEX sont indiquées dans les tableaux 6.6, 6.7 et 6.8, ainsi que la valeur par défaut. Ces informations sont collectées directement du site IBM [IBM,]. Le premier paramètre, le plus critique, CPLEX est le paramètre «MIPEmphasis», qui contrôle les compromis entre faisabilité, optimalité et rapidité de résolution des MIPs (tableau 6.6). Le deuxième paramètre important de CPLEX, «Probe», définit l'étendue du sondage à effectuer sur les variables avant le branchement du MIP (tableau 6.7). Un autre paramètre CPLEX "NodeSel" définit les règles de sélection du prochain nœud à traiter lors du retour en arrière (tableau 6.8).

Table 6.6 – Description du paramètre de "MIPEmphasis"

Valeur	Symbole	Signification
0	CPX_MIPEMPHASIS_BALANCED	Équilibrer l'optimalité et la faisabilité ; par défaut
1	CPX_MIPEMPHASIS_FEASIBILITY	Privilégier la faisabilité par rapport à l'optimalité
2	CPX_MIPEMPHASIS_OPTIMALITY	Privilégier l'optimalité par rapport à la faisabilité
3	CPX_MIPEMPHASIS_BESTBOUND	Privilégier le déplacement de la meilleure borne
4	CPX_MIPEMPHASIS_HIDDENFEAS	Privilégier la recherche de solutions réalisables cachées

TABLE 6.7 – Description du paramètre “Probe”

Valeur	Signification
-1	Pas de sondage
0	Automatique : laisser CPLEX choisir ; par défaut
1	Niveau de sondage modéré
2	Niveau de sondage agressif
3	Niveau de sondage très agressif

TABLE 6.8 – Description du paramètre “NodeSel”

Valeur	Symbole	Signification
0	CPX_NODESEL_DFS	Recherche avec parcours en profondeur
1	CPX_NODESEL_BESTBOUND	Recherche de meilleure borne ; valeur par défaut
2	CPX_NODESEL_BESTEST	Recherche de la meilleure estimation
3	CPX_NODESEL_BESTEST_ALT	Recherche de la meilleure estimation alternative

Nous avons mené différentes expériences pour ajuster les paramètres CPLEX avec les instances de différentes tailles. Les expériences avec les différents paramètres ”MIPEmphasis”, ”Probe” et“NodeSel” prédéfinis peuvent affecter les solutions, élargir les espaces de recherches différents et chercher intelligemment la bonne solution. Selon nos expériences, nous avons trouvé que dans la plupart des cas, fixer le paramètre ”MIPEmphasis” à 2 (accent mis sur l’optimalité par rapport à la faisabilité) permet d’obtenir des bornes supérieures et des bornes inférieures concurrentielles avec des écarts plus faibles. En ce qui concerne les expériences avec le paramètre ”Probe”, ”MIPEmphasis” a été préréglé sur 2.
Le niveau de sondage très agressif (valeur 3 du paramètre) a amélioré les solutions ou a confirmé l’optimalité de la solution fournie initialement. Des expériences similaires avec le paramètre “NodeSel”, fixant “MIPEmphasis” à 2 ont montré que la valeur par défaut du paramètre “NodeSel” (1) (recherche bestbound) surpassait les autres valeurs de “NodeSel”.
En résumé, les valeurs suivantes des paramètres CPLEX ont été retenues dans nos expériences ultérieures (tableau 6.9)

TABLE 6.9 – Les valeurs retenues des paramètres

Paramètre	valeur retenue
MIPEmphasis	2 : Privilégier l’optimalité par rapport à la faisabilité
Probe	3 : Niveau de sondage très agressif
NodeSel	1 : Recherche de meilleure borne

6.4.5 Comparaison des instances résolues à l'optimalité par HA et MIP start

Dans cette section, nous proposons de valider les solutions optimales trouvées par HA (voir les tableaux 6.2 à 6.5). Pour le reste des instances, nous avons également utilisé les solutions générées par HA pour le démarrage à chaud de CPLEX. Nous montrons que certaines solutions se sont avérées optimales. Pour ceux qui ne se sont pas avérées optimales, le MIP start à réussi à améliorer un nombre limité d'entre elles. Dans ce qui suit, nous présentons des tableaux qui déterminent le nombre exacte des solutions optimales trouvées par chaque méthode.

Résultats 1ère catégorie de données

Au niveau du tableau 6.10, la colonne 2 présente les solutions confirmées optimales par le MIP start. Nous constatons que toutes les solutions optimales selon notre algorithme HA se sont confirmées. Ceci est une confirmation de la validité de HA. La colonne 3 présente le nombre de solutions générées par HA et qui se sont avérées optimales grâce au MIP start. En effet, nous avons démarré le MIP start avec la solution HA, après exécution, ce dernier est sorti avec la même solution en la déclarant optimale. Il convient de signaler que pour cette catégorie 1, le MIP start n'a pas réussi à améliorer les autres instances non résolues à l'optimalité. Selon le tableau 6.10, le nombre des solutions optimales trouvées par HA est supérieur à celui trouvé par MIPstart. Ceci montre clairement que la conception de l'heuristique est efficace.
Le nombre des instances résolues à l'optimum par HA est comme suit :
- toutes les instances de 10 jobs ;
- entre 50% et 84% des instances de 20 et 50 jobs ;
- entre 20 % et 27 % des instances de 100 et 200 jobs.

TABLE 6.10 – Nombre des instances résolues à l'optimalité par HA et MIPstart

n	**HA**	**MIPstart**
10	30	*
20	25	3
50	15	10
100	8	7
200	6	5

Résultats 2ème catégorie de données

Pour cette 2ème catégorie de données, nous constatons que le MIP start à aussi confirmé les résultats trouvés par HA, mais à également a amélioré d'autres résultats. Ceci s'est produit avec les instances de la classe 2 (n1= 0,7 n), soit les instances les plus faciles. Ceci est aussi vraie pour les instances de la classe 3, n=20 et n1=0,7n.

TABLE 6.11 – Nombre des instances résolues à l'optimalité par HA et MIPstart

n1=	n	Nombre total des instances résolues à l'optimum					
		Classe 1		Classe 2		Classe 3	
		HA	MIPstart	HA	MIPstart	HA	MIPstart
0,5n	10	30	*	30	*	30	*
	20	10	5	15	3	10	6
	50	8	4	12	4	8	3
	100	5	6	5	5	5	2
	200	4	3	3	6	1	4
0,6n	10	30	*	30	*	30	*
	20	10	6	18	2	16	13
	50	9	6	15	2	10	7
	100	8	4	13	3	4	12
	200	6	3	4	7	7	3
0,7n	10	30	*	30	*	30	*
	20	18	3	28	2	28	2
	50	12	5	27	3	17	10
	100	10	6	28	2	12	12
	200	8	8	25	5	8	10

Dans les 3 classes, le nombre des instances résolues à l'optimum par HA est comme suit :
Dans le cas où n1= 0.5 n :
- toutes les instances de 10 jobs ;
- entre 26% et 50% des instances de 20 et 50 jobs ;
- entre 4 % et 17 % des instances de 100 et 200 jobs.

Dans le cas où n1= 0.6 :
- toutes les instances de 10 jobs ;
- entre 30% et 60% des instances de 20 et 50 jobs ;
- entre 13 % et 43 % des instances de 100 et 200 jobs.

Dans le cas où n1= 0.7 :
- toutes les instances de 10 jobs ;
- entre 40% et 94% des instances de 20 et 50 jobs ;
- entre 26 % et 94 % des instances de 100 et 200 jobs.

6.5 Conclusion

Dans ce chapitre, nous avons proposé une heuristique arborescente permettant une résolution efficace de FH2(PD2,Pm)$|ST_{f,sd}, l_i|C_{max}$. Des bornes inférieures globales du chapitre 4 ont été réutilisées ainsi que des bornes inférieures conçues pour les ordonnancements partiels ont été développés. Les solutions trouvées par les AGs du chapitre 5 ont été utilisées comme des solutions initiales.

Étant donné que l'heuristique HA est connue comme une méthode de résolution d'énumération, il était essentiel de chercher un branchement efficace qui se base sur une liste des jobs bien choisie afin d'explorer le plus grand nombre possible de nœuds. L'heuristique HA a montré une grande efficacité en termes de résolution à l'optimalité qui a dépassé 50% dans la catégorie 1 de données (sauf les instances de 200 jobs) tandis que ce taux de résolution a varié entre 16% et 94% dans la catégorie 2. La validation de l'optimalité de toutes les solutions est réalisée par CPLEX.

Chapitre 7

Gestion de pertes de matières

Sommaire

Dans ce chapitre, nous nous intéressons au second objectif, soit la minimisation de perte des matières non recyclables. Après avoir présenté l'introduction dans la section 7.1, nous décrivons le problème industriel ainsi que les travaux liés respectivement dans les sections 7.2 et 7.3. Ensuite, une modélisation mathématique et un algorithme génétique hybride adapté des chapitres précédents seront présentés dans les sections 7.4 et 7.5. Les sections 7.6 et 7.7 seront consacrées à l'analyse empirique des résultats. Nous clôturons le chapitre par une conclusion (section 7.8).

7.1 Introduction

Dans plusieurs systèmes de production, il est complètement nécessaire que les ressources d'un système de production continuent à effectuer les opérations de manière consécutive. En effet,

dans certaines situations où des machines coûteuses sont utilisées ou dans certaines circonstances techniques ou économiques, l'arrêt des machines entre deux opérations successives pourrait résulter des pertes de matières.

7.2 Présentation du problème

Le problème traité dans ce chapitre concerne un des problèmes que rencontre l'entreprise et qui a été mentionné dans le chapitre 1, à savoir la minimisation des pertes.
En effet, nous rappelons qu'il s'agit d'un atelier de production de type flow shop hybride à deux étages. Le premier étage contient deux machines parallèles dédiées et le second étage comporte dix machines identiques parallèles. Il y a deux séries de jobs : J1 = 1..n1, J2 = 1..n2. A chaque fois qu'il y a un changement de produit ; il y a un changement de moule. Cela nécessite un temps de setup. De plus, un time lag maximal est autorisé entre les deux étages.
D'autre part, les ouvriers doivent nettoyer les machines sur le premier étage. Dans ce cas, où la production est interrompue pour plus de α= 30 min, le résidu de pâtes coincé à l'intérieur de la machine ne pouvait plus être recyclé. Ce résidu représente une quantité non négligeable. Il y a environ 50 kilogrammes de produit perdu à chaque arrêt de machine. Par conséquent, l'objectif est de minimiser les déchets de produits en minimisant les occurrences de temps d'inactivité (idle-time) qui dépasse les 30 minutes. En suivant la classification $\alpha|\beta|\gamma$ [Graham et al., 1979], ce problème pourrait être noté comme FH2 (PD2,Pm) | $ST_{g,sd}$, l_i | $N_{idle-time-max}$.
Il convient de signaler que nous ne traiterons pas un problème bi-objectif. Nous nous intéressons à chaque objectif à part. Pour chaque solution générée dans le cadre de la minimisation des nombres de temps morts dépassant α , nous calculons le C_{max} correspondant et pour chaque solution générée dans le cadre de la minimisation du C_{max}, nous déterminons le nombre d'arrêts correspondant.

7.3 Travaux liés

Des nombreux chercheurs ont décrit des applications sur l'ordonnancement de permutation avec no-idle-time (no-idle permutation flow shop scheduling (NIPFS)) tels que les opérations de fonderie et de traitement des fibres en verre présenté respectivement par [Saadani et al., 2005] et [Kalczynski and Kamburowski, 2005]. Récemment [Tasgetiren et al., 2011] ont présenté de larges exemples pratiques d'ordonnancement des jobs dans des environnements no-idle-time.
La contrainte de non-inactivité (no-idle time) apparaît lorsque chaque machine doit traiter des travaux sans interruption de début du traitement du premier job dans la séquence jusqu'à l'achèvement du traitement du dernier job. Autrement, il ne doit pas y avoir des intervalles de non-inactivité entre le traitement de toutes les opérations consécutives sur chaque machine.
[Ruiz et al., 2009] et [Goncharov and Sevastyanov, 2009] ont publié des revues récentes sur le NPFSP dans lesquelles les auteurs ont confirmé qu'il y a peu de travaux considérant deux contraintes simultanées : no-wait et no-idle-time.
À notre connaissance, la contrainte no-idle-time et celle de no-wait ont été étudiées pour la première fois par [Adiri and Pohoryles, 1982] dans un PFS afin de minimiser la somme de makespan. Plus tard, [Kalczynski and Kamburowski, 2007] ont étudié aussi les mêmes contraintes simultanément. De plus, [Baraz and Mosheiov, 2008] ont présenté un algorithme de glouton hybridé avec une heuristique amélioratrice pour résoudre un NIPFS afin de minimiser le makespan. Plus récemment,

[Shao et al., 2017] a proposé un algorithme mimétique hybride pour résoudre un NIPFSP. Les auteurs ont travaillé sur le critère de makespan et ils ont comparé leurs résultats avec des algorithmes existants.
D'après la littérature, il semble que le seul travail concernant un flow shop hybride, et qui traite le no-idle time, est celui de [Yazdani and Naderi, 2016]. En effet, les auteurs ont proposé dans un premier temps un MIP pour résoudre les petites instances à l'optimalité. Dans un second temps, deux méta-heuristiques, basées sur la recherche au voisinage et les algorithmes génétiques, sont développées pour résoudre des instances les plus grandes.
En révisant la littérature, les contraintes liées aux arrêts des machines : idle time et no-idle time concernent le temps d'inactivité des ressources utilisées dans les ateliers de travail. Ceci est totalement différent de l'objectif traité dans cette thèse qui concerne le nombre de idle-time dépassant une borne bien déterminée. Autrement dit, nous ne sommes pas intéressés à minimiser la durée d'inactivité des machines de l'E1 mais plus tôt l'objectif souhaité est de minimiser le nombre d'occurrences de ces arrêts.

7.4 Adaptation du MIP au problème traité

En partant du premier modèle mathématique proposé dans le chapitre 4, nous ajoutons la donnée et les variables de décisions suivantes :

1. **Donnée** α : la durée limite qui entraine une perte non négligeable
2. **Variables de décision** $TM_{i,j}^{m,1}$: durée de temps mort entre le traitement du job i et j sur la machine m de l'étage 1

   $v_i^1 = \begin{cases} 1 \text{ s'il y a un idle-time qui dépasse } \alpha \text{ aprés le traitement du job i} \\ 0 \text{ sinon} \end{cases}$
3. **Modèle mathématique**

Dans ce modèle, nous gardons les mêmes contraintes (1),(2),(3),(4),(5), (6),(7), (8), (9), (10), (11) et(12) que le premier modèle où on minimise C_{max} comme objectif.
En premier temps, nous changeons l'objectif en minimiser le nombre des arrêts de machines de durées maximales sur l'étage 1. Ensuite, nous ajoutons les contraintes suivantes :
contrainte (13) calcule le temps mort entre 2 jobs affectés à une même machine
contrainte (14) exprime le temps mort qui ne doit pas dépasser une durée α
contrainte (15) déclare x_{ij}^{mk}, $a_i^{m,k}$ et v_i^1 sont binaires,α égale à 30 minutes
contrainte (16) contrainte de non négativité

$$\text{Min} \sum_{i=0}^{n} v_i^1$$

S.C

$$\sum_{i=1}^{n+1} x_{i0}^{m,k} = 0, \qquad m \in M_k, k \in \{1,2\}, \tag{1}$$

$$\sum_{i=0}^{n} x_{n+1i}^{m,k} = 0, \qquad m \in M_k, k \in \{1,2\}, \tag{2}$$

$$x_{ij}^{mk} + x_{ji}^{mk} \leq a_i^{m,k}, \qquad i = 1...n, j = 1...n, m \in M_k, k = \{1,2\}, i \neq j \tag{3}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij}^{m,k} = a_j^{m,k} \qquad \forall j = 1...n{+}1, m \in M_k, k \in \{1,2\} \tag{4}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n+1} x_{ij}^{m,k} = a_i^{m,k}, \qquad \forall i = 0...n, m \in M_k, k \in \{1,2\} \tag{5}$$

$$a_0^{m,k} = 1, \qquad \forall m \in M_k, k \in \{1,2\} \tag{6}$$

$$a_{n+1}^{m,k} = 1, \qquad \forall m \in M_k, k \in \{1,2\}, \tag{7}$$

$$a_i^{m,1} = 1, \qquad \forall i = 1..n, m = A_i \tag{8}$$

$$\sum_{m=1}^{m_2} a_i^{m,2} = 1 \quad i = 1...n \tag{9}$$

$$t_{j,k} \geq t_{i,k} + P_{i,k} - M(1 - x_{ij}^{mk}) + S_{j,k}.B_{ij}, \quad i = 0..n, j = 1..n, m \in M_k, k = \{1,2\}, i \neq j \tag{10}$$

$$t_{i,2} \leq t_{i,1} + P_{i,1} + l_i, \qquad i = 1...n \tag{11}$$

$$t_{i,2} \geq t_{i,1} + P_{i,1}, \qquad i = 1...n \tag{12}$$

$$TM_{ij}^{m,1} \geq t_{j,1} - (t_{i,1} + P_{i,1}) - M(1 - x_{ij}^{m,1}), \quad i = 1...n, j = 1...n, \forall i \neq j, m \in M_1, \tag{13}$$

$$TM_{ij}^{m,1} \leq M.v_i^1 + \alpha.x_{ij}^{m,1}, \quad i = 1...n, j = 1...n \forall i \neq j, m \in M_1, \alpha = 30 \tag{14}$$

$$x_{ij}^{m,k} \in \{0,1\}, m \in M_k, i = 1...n, j = 1...n, k = \{1,2\}, i \neq j \tag{15}$$

$$a_i^{m,k} \in \{0,1\}, m \in M_k, i = 1...n, k = \{1,2\} \tag{16}$$

$$v_i^1 \in \{0,1\}, i = 1...n \tag{17}$$

$$TM_{ij}^{m,1}, t_{i,k} \geqslant 0, i = 1...n, j = 1...n, m = 1..m_k, k = \{1,2\} \tag{18}$$

M : un nombre très grand

Comme mentionné auparavant, nous ne sommes pas entrain d'utiliser l'approche bi-critère quelle que soit lexicographique ou autre l'approche lexicographique. En effet, dans le cas d'un MIP, l'approche lexicographique commence par résoudre un programme mathématique 1 (prog 1) pour

le premier critère. Ensuite, on résout un deuxième programme (prog 2) pour le 2ème critère en mettant le résultat du prog 1 comme contrainte. Ce n'est pas l'approche utilisée dans cette thèse. Pour le AG, l'approche utilisée est aussi une approche mono-critère. Toutefois, elle utilise une méthode inspirée de l'approche lexicographique pour classer les individus ; dans le cadre de l'opérateur de sélection. Aucune autre technique bi-critère n'est utilisée avec les autres opérateurs.

7.5 Adaptation du AG1 au problème traité

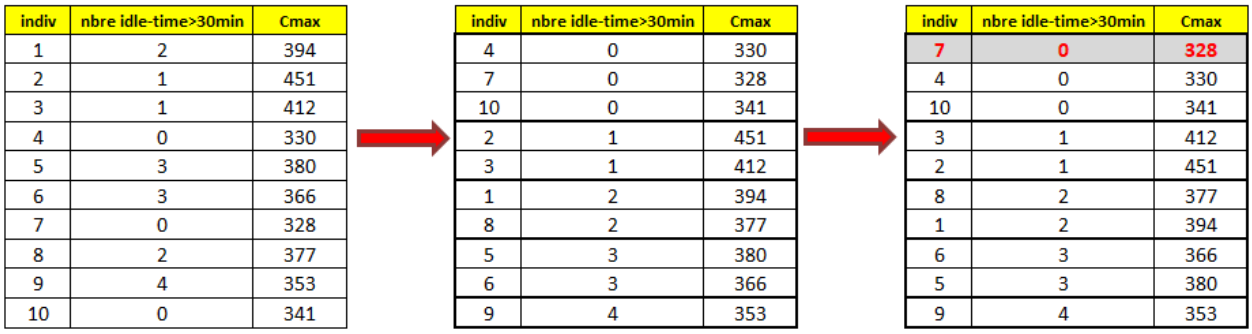
Étant donné que la résolution par le MIP est limitée et qu'elle ne peut pas couvrir les instances de grande taille, nous proposons d'adapter la structure générale de l'AG présentée dans le chapitre 4, sur lequel nous avons modifié l'objectif.
Dans un premier temps, il convient de signaler que les algorithmes évolutionnistes bi-objectif ou multi-objectif d'une manière générale affectent un score à une solution selon qu'elle est dominée ou non par d'autres solutions de la population courante. La notion de dominance est souvent combinée à des indicateurs pour la définition de valeurs de fitness cherchées.
Dans notre cas d'étude et comme mentionné avant, tenir en considération un deuxième objectif (minimisation de perte de matières) sans explicitement traiter le problème en bi-objectifs, nous a poussé à affiner la recherche de solutions par la vérification de 2 critères d'optimisation simultanément : min C_{max} et min $N_{idle-time-max}$. Cette étape est appliquée dans l'évaluation de chaque individu et le classement des solutions. Pour bien expliquer, nous rappelons d'abord les étapes de l'AG ensuite nous détaillons les modifications apportées.

7.5.1 Codage de la solution et génération de la population initiale

Cette étape est réalisée de la même façon que l'algorithme génétique du chapitre 4.

7.5.2 Classement des solutions et fonctions de fitness

Comme indiqué avant, nous avons raffiné le choix du meilleur individu en introduisant une fonction de filtrage. En effet, pour chaque individu, nous calculons en premier temps le nombre des arrêts de machines sur E1 qui dépasse α(objectif principal), ensuite nous calculons son C_{max}.
En ce qui concerne le passage d'une génération à l'autre, nous avons utilisé une fonction de classement de solution avec 2 filtres. Nous classons en premier temps les individus selon le nombre d'arrêts de machines mémorisés ensuite nous classons en ordre croissant de C_{max} les individus ayant le même nombre d'arrêts. De cette manière, la population est organisée en couches où chaque couche contient des solutions triées en ordre croissant de pertes et C_{max} (voir figure 7.1).
Étant donné que l'objectif souhaité est d'avoir des solutions avec 0 long arrêt de machine et en conséquent zéro déchets. Les solutions qui ne satisfont pas les contraintes étudiées peuvent être acceptées. Cependant, l'AG proposé préserve la primauté des solutions qui respectent les contraintes souhaitées.
Ce filtrage proposé permet ainsi un passage progressif de solutions de bonne qualité (solutions dominantes), aux moins bonnes (solutions dominées).



indiv	nbre idle-time>30min	Cmax
1	2	394
2	1	451
3	1	412
4	0	330
5	3	380
6	3	366
7	0	328
8	2	377
9	4	353
10	0	341

indiv	nbre idle-time>30min	Cmax
4	0	330
7	0	328
10	0	341
2	1	451
3	1	412
1	2	394
8	2	377
5	3	380
6	3	366
9	4	353

indiv	nbre idle-time>30min	Cmax
7	0	328
4	0	330
10	0	341
3	1	412
2	1	451
8	2	377
1	2	394
6	3	366
5	3	380
9	4	353

FIGURE 7.1 – Classement des individus par 2 critères d'optimisation

Cette méthode est inspirée de la sélection lexicographique qui se base sur les préférence pré-établit par le décideur [Mabed et al., 2000]

7.5.3 Opérateurs génétiques

Puisque le codage des individus est le même que dans le chapitre 5, nous gardons les mêmes opérateurs de croisement et mutation utilisés auparavant.

7.6 Étude de la minimisation des pertes et ses effets sur le comportement de Cmax

Dans cette section, nous allons étudier les résultats trouvés par les MIP et les AGs avec 2 objectifs différents (objectif 1 : min C_{max}, objectif 2 : min $N_{idle-time-max}$). En effet, le but est d'analyser la variation de C_{max} proportionnellement avec le nombre de idle-time maximale trouvé pour chaque famille d'instances.
Pour ce faire, nous proposons de mener une première comparaison entre les deux MIP. Le premier avec l'objectif C_{max} et le second avec l'objectif $N_{idle-time-max}$. Ensuite nous présentons une deuxième comparaison entre les 2 AG en se basant également sur l'objectif C_{max} et l'objectif $N_{idle-time-max}$. Nous appelons les différents modèles : MIP_{cmax}, MIP_{idle}, AG_{cmax} et AG_{idle}
Nous comparons aussi la quantité de pertes engendrée. Il convient de mentionner que :

$$\textbf{Déchets (kg)} = \textbf{nombre de temps d'inactivité} \times \textbf{Q}$$

Q : la quantité de matière perdue à chaque arrêt de machine dépassant une durée α.
Dans le cadre de nos expérimentations, nous allons prendre Q= 50.

7.6.1 Résultats trouvés par les deux MIP

Dans ce qui suit, nous présentons les résultats générés par les MIPs avec les 2 objectifs étudiés en ce qui concerne les 2 catégories de données.

Dans les tableaux présentés ci-dessous, la 1ère colonne désigne l'instance. Les colonnes 2 à 4 représentent respectivement ; pour le MIP_{cmax}, la moyenne des Cmax, le nombre moyen de $N_{idle-time-max}$ ainsi que la quantité moyenne des pertes.
Les colonnes 5 à 8 représentent respectivement ; pour le MIP_{idle} les mêmes données. Reste que dans le premier cas, le C_{max} est généré par le MIP_{cmax} et par conséquent la permutation correspondante. Ensuite le $N_{idle-time-max}$ qui correspond à cette permutation est calculé. Dans le deuxième cas, le$N_{idle-time-max}$ est généré par MIP_{idle} et par conséquent la permutation correspondante. Ensuite le C_{max} qui correspond à cette permutation est calculé. Nous présentons dans le tableau 7.1, les résultats pour la catégorie 1 de données. Les résultats de la catégorie 2 sont présentés dans le tableau 7.4.
Il convient de signaler que le MIP génère une seule solution pour chaque objectif, alors qu'il est possible d'avoir plusieurs solutions différentes ayant le même C_{max} (avec objectif C_{max}) ou bien le même $N_{idle-time-max}$ (avec l'objectif $N_{idle-time-max}$). Donc les observations sont effectuées par rapport aux solutions générées par le solveur. Nous verrons dans la section suivante que l'utilisation des AGs peut changer les données. Elle donne beaucoup plus de flexibilité dans la manipulation des résultats.

Résultats pour la 1ère Catégorie de données

TABLE 7.1 – Résultats de makespan et pertes par MIP

	Modèle1 : Min Cmax			**Modèle2 : Min $N_{idle-time-max}$**		
n	**Cmax**	$N_{idle-time-max}$	**Qté perdue**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**
10	**1007,00**	**0,87**	**43,33**	**1301,33**	**0,03**	**1,67**
20	**1590,67**	**2,30**	**115,00**	**2280,07**	**0,00**	**0,00**
50	**-**	**-**	**-**	**-**	**-**	**-**

Dans cette première catégorie de donnée (tableau 7.1), il est clair que leMIP_{idle} produit plus de solutions avec 0 pertes comparé au MIP_{cmax} quand le C_{max} est minimisé. Avec ce dernier, on trouve des quantités de déchet qui dépassent 100 kg dans les instances de 20 jobs. En contre partie, les C_{max} trouvés par le MIP_{idle} sont relativement élevés. Cette différence pourrait être expliquée par le fait que le deuxième modèle étend le cycle de production afin d'éviter la création des idle-times avec une durée supérieure à α. Par conséquent, il crée plusieurs arrêts sur le premier étage avec une courte durée.
Afin de bien expliquer la création de la solution avec les 2 MIP , nous présentons ci dessous la résolution d'une instance de 20 jobs de la 1ère catégorie de données. Soit π_1 et π_2 les deux permutations générées respectivement par MIP_{cmax} et MIP_{idle}. Les tableaux 7.2 et 7.3 présentent respectivement les outputs de la résolution par MIP_{cmax} et MIP_{idle} sur le premier étage.

— π_1= 13-16-8-5-2-18-19-7-20-15-6-11-17-4-3-9-1-12-14-10

— π_2= 1-19-2-16-13-5-12-15-7-4-11-18-3-6-14-9-8-20-17-10

TABLE 7.2 – Exemple de résolution par MIP_{cmax}

MIP (obj_{Cmax})							
job	**M(E1)**	**succ (job)**	**debP(job)**	**P(job)**	**finP(job)**	**debP(succ)**	**idle-time**
13	1	8	11	61	72	82	10
8	1	18	82	137	219	227	8
18	1	7	227	114	341	351	10
7	1	20	351	60	411	419	8
20	1	6	419	141	560	567	7
6	1	17	567	76	643	699	**56**
17	1	4	699	68	767	775	8
4	1	9	775	56	831	921	**90**
9	1	12	921	126	1047	1055	8
12	1	10	1055	70	1125	1135	10
16	2	5	18	81	99	116	17
5	2	2	116	77	193	205	12
2	2	19	205	57	262	282	20
19	2	15	282	135	417	528	**111**
15	2	11	528	145	673	673	0
11	2	3	673	110	783	853	**70**
3	2	1	853	69	922	942	20
1	2	14	942	94	1036	1069	**33**
$Cmax$= **1483**							

Dans ce tableau, on remarque que le nombre d'arrêt de machine ayant une durée > 30 est 5. Par conséquent la quantité de déchet= 250 kg. Dans cet exemple, le MIP (obj_{Cmax}) cherche à minimiser le $Cmax$ qui est trouvée égale à 1483 ;

TABLE 7.3 – Exemple de résolution par MIP_{idle}

MIP (obj_{idle})							
job	**M(E1)**	**succ (job)**	**debP(job)**	**P(job)**	**finP(job)**	**debP(succ)**	**idle-time**
13	1	12	441	61	502	532	**30**
12	1	7	532	70	602	612	10
7	1	4	612	60	672	695	23
4	1	18	695	56	751	781	**30**
18	1	6	781	114	895	923	28
6	1	9	923	76	999	1029	**30**
9	1	8	1029	126	1155	1155	0
8	1	20	1155	137	1292	1300	8
20	1	17	1300	141	1441	1452	11
17	1	10	1452	68	1520	1550	**30**
1	2	19	12	94	106	126	20
19	2	2	126	135	261	273	12
2	2	16	273	57	330	348	18
16	2	5	348	81	429	459	**30**
5	2	15	459	77	536	566	**30**
15	2	11	566	145	711	741	**30**
11	2	3	741	110	851	881	**30**
3	2	14	881	69	950	980	**30**
$Cmax$= **2236**							

Il est claire que le MIP_{idle}) évite les idle-time de plus que 30 minutes et crée des arrêts acceptables, ce qui explique la quantité nulle de déchets trouvée. Pour cela, le cycle de production s'étale et on trouve une valeur élevé de $Cmax$=2236.

Résultats pour la $2^{\text{ème}}$ Catégorie de données

Dans cette deuxième catégorie de donnée (tableau 7.4), nous soulignons les observations suivantes :

TABLE 7.4 – Résultats de makespan et pertes par MIP dans les 3 classes

Classe 1							
n1=	**n**	**Modèle1 : Min Cmax**			**Modèle2 : Min $N_{idle-time-max}$**		
		Cmax	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**
0,5n	10	**249,20**	**0,73**	**36,67**	**431,34**	**0,00**	0,00
	20	**374,47**	**0,63**	**31,67**	**750,47**	**0,00**	0,00
	50	-	-	-	-	-	-
0,6n	10	**249,20**	**0,70**	**35,00**	**451,05**	**0,00**	0,00
	20	**374,47**	**0,63**	**31,67**	**719,90**	**0,00**	0,00
	50	-	-	-	-	-	-
0,7n	10	**254,83**	**0,67**	**33,33**	**379,70**	**0,00**	0,00
	20	**392,37**	**1,07**	**53,33**	**761,53**	**0,00**	0,00
	50	-	-	-	-	-	-
Classe 2							
n1=	**n**	**Modèle1 : Min Cmax**			**Modèle2 : Min $N_{idle-time-max}$**		
		Cmax	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**
0,5n	10	**610,70**	**0,40**	**20,00**	**806,12**	**0,00**	**0,00**
	20	**936,27**	**0,50**	**25,00**	**1609,97**	**0,03**	**1,67**
	50	-	-	-	-	-	-
0,6n	10	**644,73**	**0,63**	**31,67**	**1063,81**	**0,00**	**0,00**
	20	**1048,13**	**2,10**	**105,00**	**1651,60**	**0,00**	**0,00**
	50	-	-	-	-	-	-
0,7n	10	**700,93**	**0,80**	**40,00**	**1030,37**	**0,00**	**0,00**
	20	**1214,80**	**2,27**	**113,33**	**1592,77**	**0,03**	**1,67**
	50	-	-	-	-	-	-
Classe 3							
n1=	**n**	**Modèle1 : Min Cmax**			**Modèle2 : Min $N_{idle-time-max}$**		
		Cmax	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**
0,5n	10	**945,17**	**0,17**	**8,33**	**1294,88**	**0,00**	**0,00**
	20	**1478,10**	**1,27**	**63,33**	**2156,50**	**0,17**	**8,33**
	50	-	-	-	-	-	-
0,6n	10	**1034,53**	**0,60**	**30,00**	**1356,07**	**0,00**	**0,00**
	20	**1606,00**	**2,20**	**110,00**	**2068,53**	**0,07**	**3,33**
	50	-	-	-	-	-	-
0,7n	10	**1112,50**	**0,63**	**31,67**	**1246,00**	**0,00**	**0,00**
	20	**1849,40**	**2,30**	**115,00**	**2009,30**	**0,23**	**11,67**
	50	-	-	-	-	-	-

— Classe 1 : pour cette classe, toutes les solutions générées par le MIP_{idle} évitent des longs arrêts de machines. En effet, il étale le cycle de production pour éviter les longs arrêts de

machines. On observe que le coût de ces "0" pertes est un doublement du C_{max}. En contre partie, les solutions trouvées par le MIP_{cmax} n'engendrent pas des pertes significatives.

— Classe 2 : dans cette classe, le MIP_{idle} résout à l'optimalité toutes les instances de 10 et 20 jobs. Toutefois, on enregistre quelques instances où la quantité de déchets $\neq 0$. Cependant le $Cmax$ dépasse celui trouvé par le MIP_{cmax} d'au moins de 30%. Les solutions trouvées par le MIP_{cmax} provoquent des pertes qui dépasse 100 kg dans le cas d'un déséquilibre de charge entre les machines (n1 =06 n, n1= 0.7 n).

— Classe 3 : les résultats trouvés dans cette classe sont pareils à la classe 2.

D'après le tableau, on peut confirmer que les solutions trouvées par leMIP_{cmax} dans les 3 classes sont plus intéressantes pour les industriels que celles trouvées par leMIP_{idle}. Ceci est expliqué par le fait que les quantités de déchets ne sont pas significatives tout en gardant une meilleure valeur de $Cmax$.
On voit que les pertes sont nuls, pour n=10 ; soit le nombre de machines parallèle. Ceci est prévu car un job n'attend jamais avant de passer à l'étage 2. Donc on a pas besoin de décaler le lancement des jobs au niveau de l'E1 (ceci évite les $idle_{time}$). Même avec n=20 jobs, il est évident que le nombre de long arrêts des machines ne soit pas aussi important. D'autre part, les MIP sont limités et ne résolvent pas les instances de grande taille, ce qui nous encourage à analyser les résultats des AGs.

7.6.2 Résultats trouvés avec AG_{Cmax} et AG_{idle}

Dans cette partie, nous allons présenter les résultats trouvés par AG_{Cmax} et AG_{idle} dans chaque catégorie de données.

Catégorie 1 de données

Contrairement au MIP, dans cette première catégorie de donnée (tableau ci dessous 7.5), le AG_{idle} réussit à obtenir des résultats de makespan quasiment les mêmes que le AG_{Cmax} tout en réalisant 0 pertes dans les instances de 10, 20 et 50 jobs. Il convient de signaler que AG_{Cmax} provoque des pertes entre 40 et 175 kg.
En ce qui concerne les instances de grande taille (100 et 200 jobs), les C_{max} trouvés par AG_{idle} dépassent ceux trouvés par AG_{Cmax} de moins 10% en gardant des quantités moyennes de déchets assez faibles (< 100 kg) comparées à celles obtenus par AG_{Cmax} (> 350 kg).
Ces résultats sont expliqués par le fait que l'espace de recherche comporte plusieurs solutions différentes (permutation différente) mais ayant même valeur de C_{max}, d'ici AG_{idle} choisit celles qui minimisent les arrêts de machines de α unité de temps en prenant en considération le makespan comme un $2^{\text{ème}}$ critère.
De point de vue industrielle, les résultats trouvés par AG_{idle} semblent plus intéressantes que ceux trouvés par AG_{Cmax} car ils satisfont les 2 objectifs souhaités.

TABLE 7.5 – Résultats de makespan et pertes par AG

	AG : Min Cmax			AG : Min $N_{idle-time-max}$		
n	**Cmax**	$N_{idle-time-max}$	**Qté perdue**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**
10	1007,00	0,77	38,33	1011,70	0,00	0,00
20	1550,20	1,60	80,00	1539,20	0,00	0,00
50	3215,93	3,47	173,33	3218,73	0,00	0,00
100	5936,63	7,03	351,67	6368,80	0,43	21,67
200	11465,93	14,63	731,67	12169,87	1,90	95,00

Pour bien expliquer ce cas, nous présentons 2 solutions π_1 et π_2 ayant la même valeur de C_{max}= 1774 trouvées respectivement par AG_{Cmax} et AG_{idle}. Ces deux solutions enregistrent une quantité de pertes différentes.

— π_1= 15-6-10-18-19-4-8-9-20-3-12-5-14-7-17-11-13-1-16-2

— π_1= 4-19-8-10-9-15-20-7-12-5-6-3-11-13-18-14-17-1-16-2

Les tableaux 7.6 et 7.7 décrivent respectivement les détails de chaque solution. Comme le montre les tableaux, dans la 1ère solution, on trouve 2 long arrêts qui provoquent des pertes de matières alors que dans la 2ème solution, il y a aucun long arrêt. Ceci est expliqué par le fait que dans l'espace de recherche, il y a plusieurs solutions optimales de même valeur de C_{max} mais ayant de différentes quantités de matières perdues.

TABLE 7.6 – Exemple de résolution par AG_{Cmax}

AG (Obj Cmax)							
job	**succ (job)**	**M(E1)**	**debP(job)**	**P(job)**	**finP(job)**	**debP(succ)**	**idle-time**
6	18	1	19	91	110	124	14
18	4	1	124	92	216	226	10
4	8	1	226	91	317	317	0
8	9	1	317	89	406	406	0
9	20	1	406	91	497	497	0
20	12	1	497	82	579	597	18
12	14	1	597	80	677	693	16
14	17	1	693	81	774	774	0
17	11	1	774	93	867	887	20
11	13	1	887	81	968	968	0
13	1	1	968	93	1061	1076	15
1	16	1	1076	81	1157	1157	0
16	2	1	1157	95	1252	1252	0
15	10	2	11	86	97	109	12
10	19	2	109	88	197	212	15
19	3	2	212	88	300	510	**210**
3	5	2	510	100	610	643	**33**
5	7	2	643	99	742	742	0
C_{max}= 1774							

TABLE 7.7 – Exemple de résolution par AG_{idle}

AG (Obj idle)							
job	**succ (job)**	**M(E1)**	**debP(job)**	**P(job)**	**finP(job)**	**debP(succ)**	**idle-time**
4	8	1	10	91	101	101	0
8	9	1	101	89	190	190	0
9	20	1	190	91	281	281	0
20	12	1	281	82	363	381	18
12	6	1	381	80	461	480	19
6	11	1	480	91	571	591	20
11	13	1	591	81	672	672	0
13	18	1	672	93	765	779	14
18	14	1	779	92	871	887	16
14	17	1	887	81	968	968	0
17	1	1	968	93	1061	1076	15
1	16	1	1076	81	1157	1157	0
16	2	1	1157	95	1252	1252	0
19	10	2	15	88	103	115	12
10	15	2	115	88	203	214	11
15	7	2	214	86	300	316	16
7	5	2	316	93	409	409	0
5	3	2	409	99	508	528	20
C_{max}= 1774							

Catégorie 2 de données

Dans la classe 1 (tableau 7.8), on observe le AG_{idle} génère des solutions avec 0 pertes pour toutes les instances de cette classe, tout en gardant des valeurs de $Cmax$ quasiment les mêmes que celles trouvées par AG_{Cmax}. D'autre part, les solutions trouvées par le AG_{Cmax} n'engendrent pas des pertes significatives.
Cependant, on peut confirmer que le AG_{idle} est plus intéressant que AG_{Cmax} de point de vue quantité de déchets signalées et $Cmax$.

TABLE 7.8 – Classe 1 : résultats de makespan et pertes par AG

Classe 1							
n1=	**n**	**AG : Min Cmax**			**AG : Min $N_{idle-time-max}$**		
		Cmax	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**
0,5n	10	249,17	0,00	0,00	249,17	0,00	0,00
	20	363,93	0,00	0,00	356,03	0,00	0,00
	50	677,90	0,00	0,00	689,53	0,00	0,00
	100	1276,37	0,07	3,33	1298,07	0,00	0,00
	200	2159,17	0,17	8,33	2427,23	0,00	0,00
0,6n	10	254,83	0,00	0,00	254,83	0,00	0,00
	20	366,77	0,03	1,67	373,37	0,00	0,00
	50	726,53	0,03	1,67	751,73	0,00	0,00
	100	1365,13	0,03	1,67	1412,77	0,00	0,00
	200	2585,70	0,07	3,33	2698,70	0,00	0,00
0,7n	10	266,57	0,00	0,00	266,57	0,00	0,00
	20	421,27	0,00	0,00	421,27	0,00	0,00
	50	868,33	0,00	0,00	868,33	0,00	0,00
	100	1601,37	0,03	1,67	1605,47	0,00	0,00
	200	3109,77	0,20	10,00	3201,57	0,00	0,00

Le tableau 7.9 présente les résultats de la classe 2. En premier temps, nous remarquons que le AG_{idle} réussit à obtenir des résultats de makespan qui sont les mêmes ou quasiment les mêmes que AG_{Cmax} dans toutes les instances de 10, 20, 50 et 100 jobs quelque soit la répartition de charges de machines.
Il convient de signaler que pour les instances de 100 jobs où n1=0.5 n, les valeurs de $Cmax$ sont plus grandes que celles trouvées par AG_{idle} d'au moins 3%.
Ainsi, le AG_{idle} dépasse le AG_{Cmax} en réalisant 0 long arrêt de machines et par conséquent des pertes nulles.
En ce qui concerne les instances de 200 jobs, la quantité moyenne de déchets trouvée par le AG_{Cmax} est d'environ 380 kg quand il s'agit d'un équilibre de charges entre les machines. Cependant, le AG_{idle} minimise clairement cette quantité en étalent le cycle de production, ce qui explique la valeur moyenne de $Cmax > 20$ %.
Dans les cas où n1=0.6 n et 0.7n, le AG_{idle} surpasse le AG_{Cmax} en diminuant les quantités de déchet de plus que 85% tout en gardant une valeur moyenne de $Cmax$ proche de celle trouvée par le AG_{Cmax} ($< 10\%$ où n1=0.6n, $< 2\%$ où n1=0.7 n).

TABLE 7.9 – Classe 2 : résultats de makespan et pertes par AG

Classe 2							
n1=	**n**	**AG : Min Cmax**			**AG : Min $N_{idle-time-max}$**		
		Cmax	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**
0,5n	10	609,20	0,00	0,00	609,20	0,00	0,00
	20	918,77	0,07	3,33	923,50	0,00	0,00
	50	1872,93	0,13	6,67	2095,77	0,00	0,00
	100	3481,00	0,23	11,67	3563,83	0,00	0,00
	200	6816,03	7,63	381,67	8538,53	0,63	31,67
0,6n	10	624,93	0,00	0,00	624,93	0,00	0,00
	20	1034,77	0,13	8,33	1059,97	0,00	0,00
	50	2115,10	0,27	13,33	2120,20	0,00	0,00
	100	3980,23	0,23	11,67	3993,73	0,00	0,00
	200	7883,43	4,80	240,00	8611,87	0,57	30,00
0,7n	10	1112,50	0,00	0,00	1112,50	0,00	0,00
	20	1162,03	0,07	3,33	1184,07	0,00	0,00
	50	2421,80	0,27	13,33	2448,30	0,00	0,00
	100	4601,50	0,43	21,67	4614,00	0,00	0,00
	200	8860,33	2,97	148,33	9023,17	0,47	23,33

Le tableau 7.10 présente les résultats de la classe 3. Cette classe 3 est la plus difficile à résoudre, ce qui explique l'incapacité des 2 AG à trouver des solutions avec 0 pertes dans les instances de grande taille.
Il est clair que le AG_{idle} surpasse le AG_{Cmax} en réalisant des pertes faibles voir nulles dans toutes les instances de 10, 20, 50 jobs quelque soit la répartition de charges de machines tout en gardant une valeur moyenne de $Cmax$ la même ou proche de celle trouvée par le AG_{Cmax} ($< 7\%$).
En ce qui concerne les instances de 100 et 200 jobs, la quantité moyenne de déchets trouvée par le AG_{Cmax} varie entre [183,1355]. Toutefois, le AG_{idle} génère des solutions avec des pertes de plus que 30% tout en gardant une valeur moyenne de $Cmax$ proche de celle trouvée par le AG_{Cmax} ($<$ 10% pour les instances de 100 jobs, $< 2\%$ pour les instances de 200 jobs).

TABLE 7.10 – Classe 3 : résultats de makespan et pertes par AG

Classe 3							
n1=	**n**	**AG : Min Cmax**			**AG : Min $N_{idle-time-max}$**		
		Cmax	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max}}$	**Qté perdue**
0,5n	10	945,17	0,00	0,00	945,17	0,00	0,00
	20	1439,73	1,03	51,67	1456,33	0,00	0,00
	50	2956,93	3,47	173,33	3153,43	0,70	35,00
	100	5516,80	9,73	486,67	6010,50	1,53	76,67
	200	10647,27	27,07	1353,33	10824,73	6,47	323,33
0,6n	10	1034,53	0,00	0,00	1034,53	0,00	0,00
	20	1579,63	0,40	20,00	1582,63	0,00	0,00
	50	3285,27	2,30	115,00	3525,90	0,10	5,00
	100	6272,10	5,03	251,67	6715,00	0,90	45,00
	200	12171,67	25,00	1250,00	12705,87	2,97	148,33
0,7n	10	1112,50	0,00	0,00	1112,50	0,00	0,00
	20	1767,73	0,30	15,00	1767,80	0,00	0,00
	50	3706,73	1,40	70,00	3828,87	0,00	0,00
	100	6984,40	3,67	183,33	8395,06	0,03	1,67
	200	13521,11	17,22	861,11	13742,20	1,93	96,72

7.7 Comparaison AG(C_{max}, $\sum$ déchets) vs AG($\sum$ déchets, C_{max})

Dans cette section, nous proposons de comparer deux approches. Dans la première, nous trions les individus selon le critère C_{max}, suivi du critère $N_{idle-time-max}$. Nous appelons $AG_{Cmax-idle}$ l'approche qui résout Lex(C_{max}, $\sum$ déchets).
Nous appelons la deuxième approche $AG_{idle-Cmax}$ qui résout Lex($\sum$ déchets, C_{max}).

7.7.1 Lex(C_{max}, $\sum$ déchets) vs Lex($\sum$ déchets, C_{max}) dans la catégorie 1

TABLE 7.11 – Résultats de makespan et pertes par $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$

	$AG_{Cmax-idle}$			$AG_{idle-Cmax}$		
n	**Cmax**	$\boldsymbol{N_{idle-time-max} > 30}$	**Q.perdue**	**Cmax**	$\boldsymbol{N_{idle-time-max} > 30}$	**Q.perdue**
10	1011,70	0,00	0,00	1011,70	0,00	0,00
20	1539,20	0,00	0,00	1539,20	0,00	0,00
50	3218,73	0,00	0,00	3218,73	0,00	0,00
100	6368,80	0,43	21,67	6368,80	0,43	21,67
200	12124,13	1,54	77,00	12169,87	1,90	95,00

Dans cette première catégorie de données, les 2 AGs (tableau 7.11) $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ génèrent les mêmes résultats pour les instances de 10, 20, 50 et 100 jobs. Les solutions trouvées ont enregistré des quantités nulles de déchets. Ceci montre que les 2 AGs sortent avec les mêmes optima locaux C_{max}. En ce qui concerne les instances de 200 jobs, $AG_{Cmax-idle}$ surpasse $AG_{idle-Cmax}$ car le premier AG a réussi à trouver des solutions meilleures qui répondent au mieux aux 2 objectifs. La quantité de déchets trouvée par $AG_{Cmax-idle}$ est diminué de 24% par rapport à celle trouvée par $AG_{idle-Cmax}$.

7.7.2 Lex(C_{max}, $\sum$ déchets) vs Lex($\sum$ déchets, C_{max}) dans la catégorie 2

Les tableaux 7.12, 7.13 et 7.14 résument les résultats trouvés dans les 3 classes de la catégorie 2 de données.

TABLE 7.12 – Résultats de makespan et pertes par $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ dans la classe 1

Classe 1							
		$AG_{Cmax-idle}$			$AG_{idle-Cmax}$		
n1=	**n**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**
0,5n	10	249,17	0,00	0,00	249,17	0,00	0,00
	20	356,03	0,00	0,00	356,03	0,00	0,00
	50	689,53	0,00	0,00	689,53	0,00	0,00
	100	1298,07	0,00	0,00	1298,07	0,00	0,00
	200	2427,23	0,00	0,00	2427,23	0,00	0,00
0,6n	10	254,83	0,00	0,00	254,83	0,00	0,00
	20	373,37	0,00	0,00	373,37	0,00	0,00
	50	751,73	0,00	0,00	751,73	0,00	0,00
	100	1401,93	0,00	0,00	1412,77	0,00	0,00
	200	2654,30	0,00	0,00	2698,70	0,00	0,00
0,7n	10	266,57	0,00	0,00	266,57	0,00	0,00
	20	421,27	0,00	0,00	421,27	0,00	0,00
	50	868,33	0,00	0,00	868,33	0,00	0,00
	100	1605,47	0,00	0,00	1605,47	0,00	0,00
	200	3201,57	0,00	0,00	3201,57	0,00	0,00

Dans la classe 1 (tableau 7.12), nous observons clairement que $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ ont généré des solutions avec des quantités nulles de déchets pour toutes les instances de cette classe. D'autre part, nous remarquons que les 2 AGs ont trouvés les mêmes valeurs de $Cmax$ pour toutes les instances sauf celles de 100 et 200 jobs où n1=0.6 n. Nous notons une baisse non significative de $Cmax$ (inférieure à 2%). Ceci est expliqué par le fait que les 2 AGs se basent sur un pool d'individus en passant d'une génération à une autre. La diversité des solutions rencontrées permet de trouver des solutions ayant des valeurs de C_{max} différentes qui engendrent des quantités de déchets nulles.

TABLE 7.13 – Résultats de makespan et pertes par $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ dans la classe 2

Classe 2							
		$AG_{Cmax-idle}$			$AG_{idle-Cmax}$		
n1=	**n**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**
0,5n	10	609,20	0,00	0,00	609,20	0,00	0,00
	20	923,50	0,00	0,00	923,50	0,00	0,00
	50	2095,77	0,00	0,00	2095,77	0,00	0,00
	100	3561,90	0,00	0,00	3563,83	0,00	0,00
	200	8623,38	0,96	48,00	8551,87	0,70	35,00
0,6n	10	624,93	0,00	0,00	624,93	0,00	0,00
	20	1059,97	0,00	0,00	1059,97	0,00	0,00
	50	2139,46	0,00	0,00	2120,20	0,00	0,00
	100	4013,09	0,00	0,00	3993,73	0,00	0,00
	200	8871,34	1,18	59,00	8611,87	0,57	30,00
0,7n	10	1112,50	0,00	0,00	1112,50	0,00	0,00
	20	1184,07	0,00	0,00	1184,07	0,00	0,00
	50	2448,30	0,00	0,00	2448,30	0,00	0,00
	100	4659,36	0,00	0,00	4614,00	0,00	0,00
	200	9109,89	1,83	91,50	9023,17	0,47	23,33

Le tableau 7.13 présente les résultats de la classe 2. En premier temps, nous remarquons que $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ réussissent à obtenir les mêmes résultats de makespan et des zéros long arrêts de machines dans toutes les instances de 10, 20, 50 jobs quelque soit la répartition de charges de machines.
Il convient de signaler que pour les instances de 100 jobs où n1=0.5 n, les valeurs de $Cmax$ trouvées par $AG_{Cmax-idle}$ sont légèrement meilleures de 0.01% . D'autre part, nous remarquons que les deux AGs n'arrivent pas à trouver des solutions avec des quantités de déchets nulles pour les instances de 200 jobs.
Dans les cas où n1=0.6 n et 0.7n, les instances de 100 et 200 jobs sont résolues par $AG_{idle-Cmax}$ plus efficacement.

TABLE 7.14 – Résultats de makespan et pertes par $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ dans la classe 3

Classe 3							
		$AG_{Cmax-idle}$			$AG_{idle-Cmax}$		
n1=	**n**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**	**Cmax**	$N_{idle-time-max}$	**Qté perdue**
0,5n	10	945,17	0,00	0,00	945,17	0,00	0,00
	20	1456,33	0,00	0,00	1456,33	0,00	0,00
	50	3325,94	1,56	78,00	3153,43	0,70	35,00
	100	6210,50	2,88	144,00	6010,50	1,53	76,67
	200	10966,07	6,93	346,50	10824,73	6,47	323,33
0,6n	10	1034,53	0,00	0,00	1034,53	0,00	0,00
	20	1582,63	0,00	00,00	1582,63	0,00	0,00
	50	3525,90	0,10	5,00	3525,90	0,10	5,00
	100	6794,00	1,12	56,00	6715,00	0,90	45,00
	200	12942,35	4,33	216,50	12705,87	2,97	148,33
0,7n	10	1112,50	0,00	0,00	1112,50	0,00	0,00
	20	1767,80	0,00	0,00	1767,80	0,00	0,00
	50	3828,87	0,00	0,00	3828,87	0,00	0,00
	100	8408,56	0,24	12,00	8395,06	0,03	1,67
	200	13743,00	1,67	83,50	13742,20	1,93	96,72

Le tableau 7.14 présente les résultats de la classe 3. Tout d'abord, nous notons que les 2 AGs réussissent à trouver les mêmes résultats de makespan et des zéros long arrêts de machines dans toutes les instances de 10, 20, 50 jobs (sauf celles de 50 jobs où n1=0.5 n). Ensuite, nous remarquons l'incapacité des 2 AGs à trouver des solutions avec 0 pertes dans les instances de grande taille. Ceci confirme notre conclusion précédente que la classe 3 est la plus difficile à résoudre
Cependant, $AG_{idle-Cmax}$ surpasse $AG_{Cmax-idle}$ en réalisant de meilleurs résultats de makespan ainsi des quantités de déchets plus faibles dans toutes les instances de 100 et 200 jobs, quelque soit la répartition de charges de machines.
En résumé, les résultats des 2 AGs : $AG_{Cmax-idle}$ et $AG_{idle-Cmax}$ sont proches dans les 3 classes de données. Les valeurs de Makespan et les quantités de déchets sont acceptables de point de vue industrielle.

7.8 Conclusion

Dans ce chapitre, nous avons introduit un nouvel objectif soit la minimisation des pertes liée à la minimisation des occurrences d'arrêts de machines. Pour cela, nous avons bien étudié une nouvelle contrainte, à savoir un idle time maximal. L'objectif étant de minimiser les déchets non recyclables.
Pour réaliser l'objectif souhaité, nous avons proposé en premier temps une re-formulation de MIP du chapitre 4. Ensuite, nous avons adapté l'AG du chapitre 5 afin qu'il réponde aux deux critères cherchés par les industriels, avoir une meilleur valeur de makespan et moins de matières perdues possible. Nous avons testé deux approches (AGs) avec Lex(C_{max}, $\sum$ déchets) et Lex($\sum$ déchets,

C_{max}). Les résultats trouvés par les AGs sont intéressants quoi que ce soit l'ordre des objectifs résolus et encourageants d'améliorer la recherche avec ce nouvel objectif.

Chapitre 8

Conclusion générale

Cette thèse est une contribution pour résoudre un système de production complexe. Le problème traité est inspiré d'un cas industriel réel rencontré dans l'industrie agro-alimentaire. Il s'agit d'un flow shop hybride à 2 étages, avec des machines dédiées sur l'étage 1 et plusieurs machines parallèles sur l'étage 2. Ce type de système est soumis à différentes contraintes temporelles. Dans ce qui suit, nous présentons les conclusions de nos travaux où une attention particulière sera donnée à nos contributions. Nous proposons également des perspectives de travail.

Dans **le chapitre 2**, nous avons présenté le problème industriel réel suivi d'une classification scientifique de ce dernier. Nous avons aussi discuté l'importance de critères d'optimisation étudiés pour l'entreprise concernée.

Dans **le chapitre 3**, un état de l'art relatif aux problèmes de la littérature les plus proches du notre a été présenté où nous avons détaillé les contraintes de setup, setup par famille, no-wait et time lag. De plus, la littérature relative au flow shop hybride avec machines dédiées a été étudié. En se basant sur une synthèse de revue nous avons constaté que le problème du flow shop hybride avec machines dédiées soumis simultanément aux contraintes de setup de groupe et de time lag maximal n'a pas été traité dans la littérature auparavant. Étudier ce problème est une contribution en soit. Étant donné que le problème est traité pour la première fois, nous avons décidé de l'approcher par des méthodes de résolution variées.

Dans **le chapitre 4**, nous avons proposé dans un premier temps, deux modélisations mathématiques différentes du problème étudié. Ces modélisations ont permis de prendre en compte les différentes nouvelles contraintes industrielles. Pour résoudre les modèles proposés nous avons fait appel au solveur CPLEX. Dans un second temps, nous avons présenté des bornes inférieures qui sont une extension des bornes de la littérature à notre problème. D'autre part, nous avons mené une large évaluation des bornes proposées afin de déterminer la meilleure. Les résultats trouvés montrent que seulement les instances de petite taille (10 jobs) sont résolues à l'optimum. Pour des instances de taille plus réaliste, le solveur est incapable de résoudre à l'optimalité les problèmes qui dépassent les 20 jobs dans les 2 catégories de données, à savoir la catégorie inspirée du cas réel et la catégorie inspirée de la littérature.

Le chapitre 5 a été dédié à une méthode de résolution approchée basée sur des algorithmes évolutionnistes. En effet, deux AGs et un AM ont été proposés pour identifier les dates optimales d'achèvement de production. Une heuristique adaptée au «SDFST» a été également proposée. Afin d'évaluer l'efficacité des opérateurs génétiques utilisés et la capacité des AGs à améliorer les solutions de la population initiale, nous avons comparé les deux AGs purs à un simple algorithme aléatoire. Les résultats trouvés ont montré une grande supériorité des AGs pur. Cependant, la version d'un AG pur n'a pas permis de trouver des solutions de bonne qualité pour les instances de grande taille de la catégorie 2 de données.
Pour remédier à ce problème, nous avons combiné la méthode proposée avec une heuristique et une recherche locale itérative. Les tests ont montré l'efficacité de l'approche proposée. En effet, des solutions optimales des instances de petite et moyenne taille sont obtenues. Ce problème a été résolu d'une manière exacte en ce qui concerne les instances de petite et moyenne taille. De plus, tout le reste des instances dans les 2 catégories de données sont résolues en moins de huit minutes environ, avec des écarts par rapport à la borne inférieure ne dépassant pas 6%. Ceci prouve non seulement la qualité des solutions fournies mais aussi la maîtrise du temps de calcul qui correspond bien aux standards de l'industrie. En revanche, il convient d'exclure de cette déduction les instances de 200 jobs de la troisième classe (qui s'est révélée la plus difficile) ayant une balance des charges entres les deux machines dédiées, car un écart légèrement élevé : 9.5% a été obtenu. Selon les résultats trouvés, l'AG1+ RLI a surpassé AG2+RLI dans les 3 classes de la catégorie 2 de données où il ya un déséquilibre de charges entre les machines dédiées. Pour le reste des instances de données, AG2+RLI s'est avéré plus efficace. De plus, et dans le but d'évaluer l'apport des AGs, nous avons comparé une version de l'AG à une version d'un algorithme mémétique. Nous avons conclu que l'AG renforcé par la recherche locale itérative est meilleur.

Le chapitre 6 a été consacré à une méthode d'énumération combinée avec une recherche locale. Nous avons appelé la méthode proposée une heuristique arborescente (HA). En effet, nous avons d'une part réutilisé des bornes inférieures (LBs) globales du chapitre 4. Et d'autre part, nous avons proposé des LBs pour les ordonnancements partiels. Pour explorer plus l'espace de recherche, nous avons réutilisé une proposition de la littérature qui permet d'accélérer l'exploration de l'arbre : ne pas visiter des nœuds ayant une marge faible par rapport à une LB initiale. Ensuite, nous avons utilisé une méthode appropriée de réparation qui consiste à appliquer une recherche locale légère et un test de vérification des solutions de non-permutation. Ceci a permis de rattraper la perte d'une solution optimale si elle n'est pas rencontrée dans l'arbre. Afin de valider l'optimalité des solutions générées, nous avons utilisé le MIPstart de CPLEX après avoir effectué un examen minutieux de ses options de réglage. Nous avons noté une grande efficacité de l'algorithme proposé ce qui explique la bonne qualité des solutions retournées. L'efficacité de cette méthode est mesurée par deux critères : (1) le nombre d'instances résolues à l'optimum ; (2) l'écart moyen des instances non résolues par rapport à la meilleure borne inférieure considérée dans la thèse. Les résultats trouvés ont montré un taux élevé de résolution à l'optimalité qui a dépassé 50% dans la catégorie 1 de données (sauf les instances de 200 jobs) tandis que ce taux a varié entre 16% et 94% dans la catégorie 2.
Pour le deuxième critère, les solutions des instances non résolues étaient proches de l'optimum car l'écart trouvé n'a pas dépassé 1.9% dans la première catégorie de données ; 4.8%, 2.5%, 7% respectivement dans la classe 1, 2 et 3 de la deuxième catégorie de données.

Le chapitre 7 a été consacré à l'étude d'un second objectif qui vise à minimiser les quantités de matières perdues. Ceci revient à minimiser les occurrences des longs arrêts des machines. Nous avons ainsi introduit une nouvelle contrainte jamais traitée dans la littérature mais qui joue un rôle très important dans l'industrie qui traite des produits périssables. Dans un premier temps, nous avons analysé quelques études trouvées dans la littérature concernant la contrainte la plus proche « no-idle time ». Ensuite, une extension d'un modèle mathématique présenté au chapitre 4 a été adaptée pour prendre en considération la nouvelle contrainte de idle-time maximal. Par ailleurs, une adaptation de l'AG1 présenté dans le chapitre 5 a également été développée. Nous avons étudié l'effet de la minimisation du C_{max} sur les pertes de matière et vice-versa. Nous avons notamment constaté qu'il ne faut pas s'arrêter à la première solution optimale ou bonne solution pour un objectif donné (C_{max} ou Idle-time), mais qu'il faut se baser sur un pool de solutions, car l'une de ces solutions peut également être bonne pour l'autre critère. Nous avons également constaté que pour certaines solutions ayant la même valeur de C_{max}, elles pouvaient répondre mieux pour le second objectif traité. Nous avons basé notre sélection sur une procédure lexicographique, sans toutefois faire appel à la programmation bi-objectif.
La modélisation mathématique proposée dans un premier temps a satisfait seulement le deuxième objectif et a fourni des solutions non acceptables du point de vue industriel car le solveur génère une seule solution pour chaque critère. La seconde approche a relevé ce défi en incluant d'abord la prise en compte de la minimisation de nombre de idle-time maximal, et ensuite le respect des dates de production maximales. Nous avons testé deux approches avec Lex(C_{max}, $\sum$ déchets) et Lex($\sum$ déchets, C_{max}). Ceci a apporté une nette amélioration des solutions : une baisse de plus de 20% de la quantité moyenne de déchets. Cette contribution correspond bien aux attentes d'industriels.

La réalisation de cette thèse a abouti à de bons résultats pour le problème industriel étudié, ce qui nous encourage à développer d'autres aspects envisageables et ouvrant de nouvelles perspectives .

- Les modèles mathématiques proposés dans cette thèse pourront être étendus en considérant un bi-objectif, à savoir, optimiser deux objectifs simultanément, par exemple minimiser le makespan tout en optimisant les quantités de déchets perdues ou considérer un autre objectif ;
- La minimisation des retards présente aussi un défi pour l'entreprise quand il s'agit d'une production sur commande. Dans cette situation, un nouveau critère d'optimisation est à explorer. En effet, l'entreprise reçoit au cours de l'année des commandes avec des caractéristiques spécifiques (poids, emballage, date de livraison...). Par conséquent, l'entreprise doit respecter les dates de livraison déjà fixées par le client afin d'établir les ordres de fabrication. Dans ce cadre, on peut proposer un ordonnancement multi-agent.
- Envisager la mise en place d'une méthode exacte : un algorithme de séparation et évaluation (PSE). Dans ce contexte, on peut envisager de résoudre le problème symétrique de notre problème étudié en considérant l'étage 1 comme étant un TSP avec time window.
- Dans les situations réelles d'ordonnancement, l'incertitude provient principalement des variables d'environnement, ceci encourage à aborder l'aspect dynamique qui est absent dans cette thèse. Il sera judicieux donc de pouvoir entamer une étude qui prend en compte les incertitudes tactiques par exemple. Dans ce cas, la prise de décision à moyen terme peut être perturbée ou changée à cause des perturbations dans l'information, telles que les flux des matières, les paramètres du marché. Il sera donc nécessaire d'inclure ces changements,

d'une manière dynamique au sein d'une solution existante et de traiter ce type d'incertitudes comme perspective scientifique.

— Tester d'autres techniques telles que la méthode de satisfaction des contraintes qui a fait ses preuves avec d'autres problèmes d'optimisation ;

— Étendre et généraliser les approches proposées à d'autres industries qui présentent les mêmes types de contraintes telles que les industries pharmaceutiques ou chimiques.

Toutes ces pistes de recherche sont très prometteuses et montrent l'importance croissante des problèmes d'ordonnancement tant sur le plan économique que sur le plan sur le plan académique.

Annexe A

Analyse des paramètres AG

Analyse pour 50 Jobs

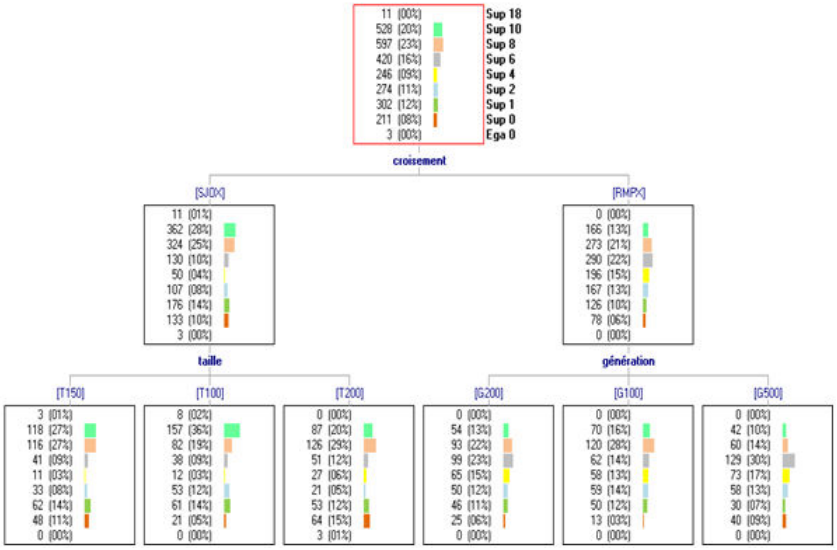


FIGURE A.1 – SIPINA output (50 jobs)

TABLE A.1 – Taux d'affectation d'opérateur(50 jobs)

	Goodness of split	Accept
Croisement	0.0374	O
Taille	0.0074	N
Génération	0.0037	N
Sélection	0.0000	N

TABLE A.2 – Opérateur de mutation (50 jobs)

	ins 0.2	ins 0.5	insert	sw 0.2	sw 0.5	swap	Total
RMPX	6,452	6,327	6,39	6,441	6,453	6,447	6,418
0.7	6,838	6,614	6,726	6,621	6,745	6,683	6,705
0.9	6,397	6,292	6,345	6,428	6,416	6,422	6,383
1.0	6,121	**6,074**	**6,098**	6,275	6,198	6,237	6,167
SJOX	7,497	7,588	7,543	7,535	7,541	7,538	7,54
0.7	7,873	7,954	7,913	7,898	7,969	7,933	7,923
0.9	7,383	7,555	7,469	7,485	7,483	7,484	7,477
1.0	7,236	7,255	7,246	7,223	7,171	7,197	7,221
Total	6,975	6,958	6,966	6,988	6,997	6,993	6,979

TABLE A.3 – Opérateur taille de la population et nombre de générations

	G100	**G200**	**G500**	**Total**
RMPX	3,934	3,894	3,611	3,813
insert	3,911	3,96	3,677	3,849
swap	3,958	3,828	3,545	3,777
SJOX	3,997	3,964	3,964	3,975
insert	3,982	3,975	4,004	3,987
swap	4,011	3,953	3,925	3,963
Total	3,965	3,929	3,788	3,894
	T100	**T150**	**T200**	**Total**
RMPX	4,131	3,796	3,512	3,813
insert	4,097	3,899	3,551	3,849
swap	4,164	3,693	3,473	3,777
SJOX	4,208	3,952	3,766	3,975
insert	4,213	3,941	3,808	3,987
swap	4,203	3,963	3,723	3,963
Total	4,169	3,874	3,639	3,894

Analyse pour 100 Jobs

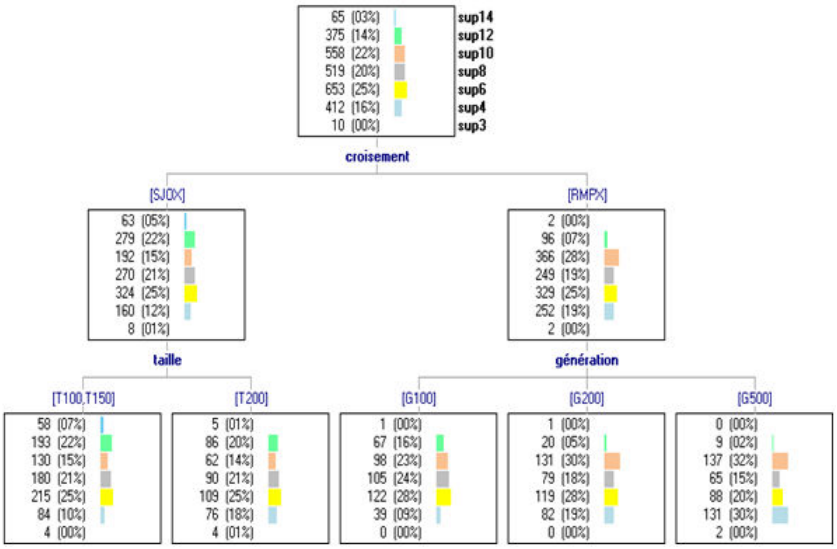


FIGURE A.2 – SIPINA output (100 jobs)

TABLE A.4 – Taux d'affectation d'opérateur(100 jobs)

	Goodness of split	Correlation	Accept
Croisement	0.0356	0.0356	O
Taille	0.0080	0.0080	N
Génération	0.0075	0.0075	N
Sélection	0.0000	0.0000	N

TABLE A.5 – Opérateur taille de la population et nombre de générations

	G100	G200	G500	Total
RMPX	10,335	9,588	9,121	9,681
insert	10,371	9,568	9,095	9,678
swap	10,299	9,607	9,147	9,684
SJOX	10,909	10,991	11	10,967
insert	10,84	10,959	10,963	10,92
swap	10,979	11,023	11,038	11,013
Total	10,622	10,289	10,061	10,324
	T100	T150	T200	Total
RMPX	10,115	9,588	9,341	9,681
insert	10,152	9,561	9,322	9,678
swap	10,078	9,616	9,359	9,684
SJOX	11,626	10,802	10,472	10,967
insert	11,59	10,785	10,386	10,92
swap	11,663	10,82	10,557	11,013
Total	10,871	10,195	9,906	10,324

Analyse pour 200 Jobs

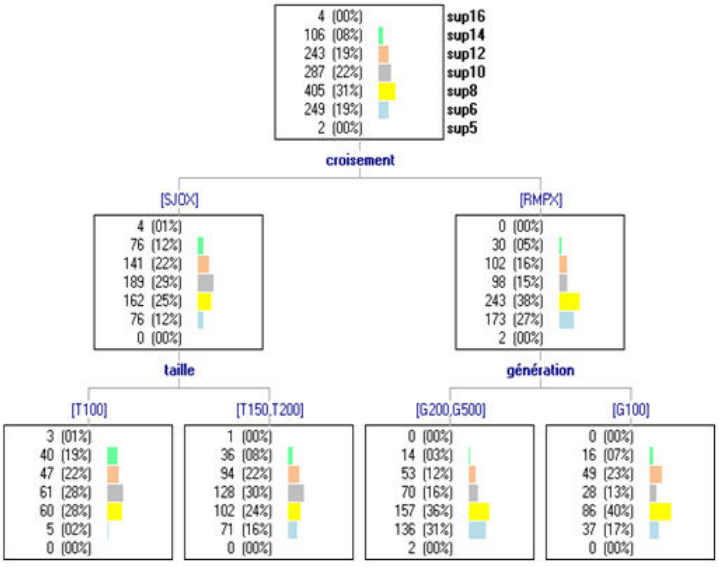


FIGURE A.3 – SIPINA output (200 jobs)

TABLE A.6 – Taux d'affectation d'opérateur(200 jobs)

	Goodness of split	Correlation	Accept
Croisement	0.0362	0.0362	O
Taille	0.0167	0.0167	0
Génération	0.0000	0.0075	N
Sélection	0.0000	0.0000	N

TABLE A.7 – Opérateur taille de la population et nombre de générations

	G100	**G200**	**G500**	**Total**
RMPX	10,335	9,588	9,121	9,681
insert	10,371	9,568	9,095	9,678
swap	10,299	9,607	9,147	9,684
SJOX	10,909	10,991	11	10,967
insert	10,84	10,959	10,963	10,92
swap	10,979	11,023	11,038	11,013
Total	10,622	10,289	10,061	10,324
	T100	**T150**	**T200**	**Total**
RMPX	10,115	9,588	9,341	9,681
insert	10,152	9,561	9,322	9,678
swap	10,078	9,616	9,359	9,684
SJOX	11,626	10,802	10,472	10,967
insert	11,59	10,785	10,386	10,92
swap	11,663	10,82	10,557	11,013
Total	10,871	10,195	9,906	10,324

Annexe B

Publications scientifiques

— Harbaoui.H, Morineau.O, and Khalfallah.S. “Scheduling a two-stage hybrid flow shop with dedicated machines, time lags and sequence-dependent family setup times.” In IEEE System, Cybernetic and Man, Budapest, Hongry, 9-13 October 2016

— Harbaoui.H, S. Khalfallah.S, and Morineau.O. ”A case study of a hybrid flow shop with no-wait and limited idle time to minimize material waste.”IEEE Intelligent Systems and Informatics (SISY), Subotica, Serbia 2017

— Harbaoui.H, Khalfallah.S, and Morineau.O. ” A novel hybrid GA for the assignment of jobs to machines in a complex hybrid flow shop problem.” Intelligent Systems Design and Applications (ISDA), Delhi, India 2017.

Bibliographie

[Sip,] Sipina - arbres de décision - data mining. `http://sipina-arbres-de-decision.blogspot.com/`.

[Adiri and Pohoryles, 1982] Adiri, I. and Pohoryles, D. (1982). Flowshop no idle or no wait scheduling to minimize the sum of completion times. *Naval Research Logistics Quarterly*, 29(3) :495–504.

[Akkoyunlu et al., 2015] Akkoyunlu, M., Engın, O., and Büyuközkan, K. (2015). A harmony search algorithm for hybrid flow shop scheduling with multiprocessor task problems. In *Modeling, Simulation, and Applied Optimization (ICMSAO), 2015 6th International Conference on*, pages 1–3. IEEE.

[Aldowaisan and Allahverdi, 2003] Aldowaisan, T. and Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers & Operations Research*, 30(8) :1219–1231.

[Allahverdi, 2015] Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2) :345–378.

[Allahverdi, 2016] Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, 255(3) :665–686.

[Attar et al., 2014] Attar, S., Mohammadi, M., Tavakkoli-Moghaddam, R., and Yaghoubi, S. (2014). Solving a new multi-objective hybrid flexible flowshop problem with limited waiting times and machine-sequence-dependent set-up time constraints. *International Journal of Computer Integrated Manufacturing*, 27(5) :450–469.

[Azizoglu and Kirca, 1998] Azizoglu, M. and Kirca, O. (1998). Tardiness minimization on parallel machines. *International Journal of Production Economics*, 55(2) :163–168.

[Baraz and Mosheiov, 2008] Baraz, D. and Mosheiov, G. (2008). A note on a greedy heuristic for flow-shop makespan minimization with no machine idle-time. *European Journal of Operational Research*, 184(2) :810 – 813.

[BESBES et al., 2010] BESBES, W., LOUKIL, T., and TEGHEM, J. (2010). A two stage flow shop with parallel dedicated machines. *8th International Conference of Modeling and Simulation, MOSIM'10,Hammamet - Tunisia*.

[Brah and Hunsucker, 1991] Brah, S. and Hunsucker, J. (1991). Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research*, 51(1) :88 – 99.

[Campbell et al., 1970] Campbell, H., Dudek, R., and Smith, M. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management Science*, 16(10) :B630–B637.

[CHAOUCH, 2015] CHAOUCH, R. (2015). Planification des chaînes de production d'usine et simulation, amélioration des lignes pâtes spéciales. *PFE, département génie industriel ENIT*.

[Chen and Yang, 2006] Chen, J. and Yang, J. (2006). Model formulations for the machine scheduling problem with limited waiting time constraints. *Journal of Information and Optimization Sciences*, 27(1) :225–240.

[Chikhi and Abbas, 2012] Chikhi, N. and Abbas, M. (2012). Makespan minimization for two-stage hybrid flow shop with dedicated machines and additional constraints. In *9th International Conference on Modeling, Optimization & SIMulation*.

[Chikhi et al., 2014] Chikhi, N., Benmansour, R., Bekrar, A., Hanafi, A., and Abbas, M. (2014). A case study of a two-stage flow shop with dedicated machines and a single robot. In *Control, Decision and Information Technologies (CoDIT), International Conference on*, pages 246–250. IEEE.

[Ebrahimi et al., 2014] Ebrahimi, M., Fatemi Ghomi, S., and Karimi, B. (2014). Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates. *Applied Mathematical Modelling*, 38, pp. 2490-2504.

[Engin et al., 2011] Engin, O., Ceran, G., and Yilmaz, M. (2011). An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Applied Soft Computing*, 11(3) :3056–3065.

[Engin and Döyen, 2004] Engin, O. and Döyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future generation computer systems*, 20(6) :1083–1095.

[Fattahi et al., 2014] Fattahi, P., Hosseini, S., Jolai, F., and Tavakkoli-Moghaddam, R. (2014). A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. *Applied Mathematical Modelling*, 38(1) :119–134.

[França et al., 2005] França, P., Gupta, J., Mendes, A., Moscato, P., and Veltink, K. (2005). Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers & Industrial Engineering*, 48(3) :491–506.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). Computers and intractability : A guide to the theory of np-completeness.

[Geem et al., 2001] Geem, Z., Kim, J., and Loganathan, G. (2001). A new heuristic optimization algorithm : harmony search. *simulation*, 76(2) :60–68.

[Goldberg and Deb, 1991] Goldberg, D. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1 :69–93.

[Goncharov and Sevastyanov, 2009] Goncharov, Y. and Sevastyanov, S. (2009). The flow shop problem with no-idle constraints : A review and approximation. *European Journal of Operational Research*, 196(2) :450 – 456.

[Graham et al., 1979] Graham, R., Lawler, E., Lenstra, J., and A, K. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, 5, pp. 287–326.

[Gupta, 1988] Gupta, J. (1988). Two-stage hybrid flow shop scheduling problem. *Journal of the Operational Research Society*, 39 :359–364.

[Gupta et al., 1997] Gupta, J., Hariri, A., and Potts, C. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Annals of Operations Research*, 69 :171–191.

[Gupta and Tunc, 1998] Gupta, J. and Tunc, E. (1998). Minimizing tardy jobs in a two-stage hybrid flowshop. *International Journal of Production Research*, 36(9) :2397–2417.

[Hadda et al., 2014] Hadda, H., Dridi, N., and Hajri-Gabouj, S. (2014). Exact resolution of the two-stage hybrid flow shop with dedicated machines. *Optimization Letters*, 8.

[Hajji et al., 2015] Hajji, M., Hadda, H., and Dridi, N. (2015). The two-stage hybrid flow shop problem with dedicated machines under release dates and delivery times. *International Journal of Advanced Operations Management*, 7(4) :300–316.

[Hall and Sriskandarajah, 1996] Hall, N. and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations research*, 44(3) :510–525.

[Haouari et al., 2006] Haouari, M., Hidri, L., and Gharbi, A. (2006). Optimal scheduling of a two-stage hybrid flow shop. *Mathematical Methods of Operations Research*, 64(1) :107–124.

[Haouari and M'Hallah, 1997] Haouari, M. and M'Hallah, R. (1997). Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations research letters*, 21(1) :43–53.

[Herrmann and Lee, 1992] Herrmann, J. and Lee, C. (1992). *Three-machine look-ahead scheduling problems*. Department of Industrial and Systems Engineering, University of Florida.

[Holland, 1975] Holland, J. (1975). Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor*.

[Hoogeveen et al., 1996] Hoogeveen, J., Lenstra, V., and Veltman, B. (1996). Preemptive scheduling in a two-stage multiprocessor flow shop is np-hard. *European Journal of Operational Research*, 89(1) :172–175.

[Huang et al., 2009] Huang, R., Yang, C., and Huang, Y. (2009). No-wait two-stage multi processor flowshop scheduling with unit setup. *International Journal of Advanced Manufacturing Technology*, 44, pp. 921-927.

[Huang and Li, 1998] Huang, W. and Li, S. (1998). A two-stage hybrid flowshop with uniform machines and setup times. *Mathematical and Computer Modelling*, 27(2) :27 – 45.

[Hwang and Lin, 2018] Hwang, F. and Lin, B. (2018). Survey and extensions of manufacturing models in two-stage flexible flow shops with dedicated machines. *Computers & Operations Research*, 98 :103–112.

[IBM,] IBM. Cplex optimizer. "https://www.ibm.com/fr-fr/analytics/cplex-optimizer/".

[Johnson, 1954] Johnson, S. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics (NRL)*, 1(1) :61–68.

[Jolai et al., 2012] Jolai, F., Rabiee, M., and Asefi, H. (2012). A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times. *International Journal of Production Research*, 50(24) :7447–7466.

[Jolai et al., 2009] Jolai, F., Sheikh, S., Rabbani, M., and Karimi, B. (2009). A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. *The International Journal of Advanced Manufacturing Technology*, 42(5) :523–532.

[Jun and Park, 2015] Jun, S. and Park, J. (2015). A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints : A case study from the transformer industry. *Expert Systems with Applications*, 42(15) :6196–6204.

[Kahraman et al., 2010] Kahraman, C., Engin, O., Kaya, I., and Elif Öztürk, R. (2010). Multiprocessor task scheduling in multistage hybrid flow-shops : A parallel greedy algorithm approach. *Applied Soft Computing*, 10(4) :1293 – 1300.

[Kahraman et al., 2008] Kahraman, C., Engin, O., Kaya, I., and Kerim Yilmaz, M. (2008). An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *International Journal of Computational Intelligence Systems*, 1(2) :134–147.

[Kalczynski and Kamburowski, 2005] Kalczynski, P. and Kamburowski, J. (2005). A heuristic for minimizing the makespan in no-idle permutation flow shops. *Computers and Industrial Engineering*, 49(1) :146 – 154.

[Kalczynski and Kamburowski, 2007] Kalczynski, P. and Kamburowski, J. (2007). On the {NEH} heuristic for minimizing the makespan in permutation flow shops. *Omega*, 35(1) :53 – 60.

[Karimi et al., 2010] Karimi, N., Zandieh, M., and Karamooz, H. (2010). Bi-objective group scheduling in hybrid flexible flowshop : a multi-phase approach. *Expert Systems with Applications*, 37(6) :4024–4032.

[Kurz and Askin, 2004] Kurz, M. and Askin, R. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, 159(1) :66–82.

[Lee and Vairaktarakis, 1994] Lee, C. and Vairaktarakis, G. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters*, 16(3) :149–158.

[Li and Li, 2007] Li, T. and Li, Y. (2007). Constructive backtracking heuristic for hybrid flowshop scheduling with limited waiting times. In *Wireless Communications, Networking and Mobile Computing, International Conference on*, pages 6671–6674. IEEE.

[Liao et al., 2012] Liao, C., Tjandradjaja, E., and Tsui-Ping Chung, T. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Applied Soft Computing*, 12(6) :1755 – 1764.

[Lin, 2015] Lin, B. (2015). Two-stage flow shop scheduling with dedicated machines. *International Journal of Production Research*, 53(4) :1094–1097.

[Lin and Liao, 2003] Lin, H. and Liao, C. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, 86(2) :133–143.

[Liu et al., 2008] Liu, S., Cui, J., and Li, Y. (2008). Heuristic-tabu algorithm for hybrid flowshop scheduling with limited waiting time. In *Computational Intelligence and Design, ISCID'08. International Symposium on*, volume 2, pages 233–237. IEEE.

[Liu et al., 2003] Liu, Z., Xie, J., Li, J., and Dong, J. (2003). A heuristic for two-stage no-wait hybrid flowshop scheduling with a single machine in either stage. *Tsinghua Science and Technology*, 8(1) :43–48.

[Lo et al., 2008] Lo, C., Chen, Shiang, C., Chang, F., et al. (2008). Two-machine flexible flow-shop scheduling with setup times. *Journal of Applied Sciences*, 8 :2217–2225.

[Logendran et al., 2006] Logendran, R., Barnard, F., et al. (2006). Sequence-dependent group scheduling problems in flexible flow shops. *International Journal of Production Economics*, 102(1) :66–86.

[Mabed et al., 2000] Mabed, M., Rahoual, M., Talbi, E., and Dhaenens, C. (2000). Algorithmes génétiques multicritères pour les problèmes de flow-shop. In *3ème Conférence Francophone de MOdélisation et de SIMulation, MOSIM*, volume 1, pages 843–849.

[Marcelo et al., 2012] Marcelo, S., Augusto, A., and Luiz, A. (2012). A new evolutionary clustering search for a no-wait flow shop problem with set-up times. *Engineering Applications of Artificial Intelligence*, 25(6) :1114 – 1120.

[Mathirajan and Sivakumar, 2006] Mathirajan, M. and Sivakumar, A. (2006). A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *The International Journal of Advanced Manufacturing Technology*, 29(9-10) :990–1001.

[Moradinasab et al., 2013] Moradinasab, N., Shafaei, R., Rabiee, M., and Ramezani, P. (2013). No-wait two stage hybrid flow shop scheduling with genetic and adaptive imperialist competitive algorithms. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(2) :207–225.

[Mori et al., 1993] Mori, K., Tsukiyama, M., and Fukuda, T. (1993). Immune algorithm with searching diversity and its application to resource allocation problem. *IEEJ Transactions on Electronics, Information and Systems*, 113(10) :872–878.

[Morizawa, 2014] Morizawa, K. (2014). A branch-and-bound based heuristic algorithm for minimizing makespan in machining-assembly flowshop scheduling. *Engineering*, 6(13) :877.

[Mosheiov and Sarig, 2010] Mosheiov, G. and Sarig, A. (2010). Minimum weighted number of tardy jobs on an m-machine flow-shop with a critical machine. *European Journal of Operational Research*, 201(2) :404–408.

[Naderi et al., 2014] Naderi, B., Khalili, M., and Khamseh, A. (2014). Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. *International Journal of Production Research*, 52(9) :2667–2681.

[Naderi et al., 2010] Naderi, B., Ruiz, R., Rubén, and Zandieh, M. (2010). Algorithms for a realistic variant of flowshop scheduling. *Computers & Operations Research*, 37(2) :236–246.

[Naderi et al., 2009] Naderi, B., Zandieh, M., Balagh, A., and Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert systems with Applications*, 36(6) :9625–9633.

[Néron et al., 2001] Néron, E., Baptiste, P., and Gupta, N. (2001). Solving hybrid flow shop problem using energetic reasoning and global operations. *Omega*, 29(6) :501 – 511.

[Oĝuz and Ercan, 2005] Oĝuz, C. and Ercan, M. (2005). A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling*, 8(4) :323–351.

[Oguz et al., 1997] Oguz, C., Lin, B., and Cheng, T. (1997). Two-stage flow shop scheduling with a common second stage machine. *Computers and Operations Research*, 24 :1169–1174.

[Osman and Potts, 1989] Osman, I. and Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6) :551–557.

[Pan et al., 2017] Pan, Q., Gao, L., Li, X., and Gao, K. (2017). Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. *Applied Mathematics and Computation*, 303 :89–112.

[Pang, 2013] Pang, K. (2013). A genetic algorithm based heuristic for two machine no-wait flowshop scheduling problems with class setup times that minimizes maximum lateness. *International Journal of Production Economics*, 141(1) :127–136.

[Papadimitriou and Steiglitz, 1998] Papadimitriou, C. and Steiglitz, K. (1998). *Combinatorial optimization : algorithms and complexity*. Courier Corporation.

[Pinedo, 2008] Pinedo, M. (2008). *Scheduling : Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition.

[Raaymakers and Hoogeveen, 2000] Raaymakers, W. and Hoogeveen, J. (2000). Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research*, 126(1) :131 – 151.

[Rabiee et al., 2012] Rabiee, M., Zandieh, M., and Jafarian, A. (2012). Scheduling of a no-wait two-machine flow shop with sequence-dependent setup times and probable rework using robust meta-heuristics. *International Journal of Production Research*, 50(24) :7428–7446.

[Rajendran, 1994] Rajendran, C. (1994). A no-wait flowshop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society*, 45(4) :472–478.

[Reeves, 1995] Reeves, C. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research*, 22(1) :5–13.

[Riane et al., 2002] Riane, F., Artiba, A., and Elmaghraby, S. (2002). Sequencing a hybrid two-stage flowshop with dedicated machines. *International Journal of Production Research*, 40(17) :4353–4380.

[Ribas et al., 2010] Ribas, I., Leisten, R., and Framiñan, J. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8) :1439–1454.

[Rios-Mercado and Bard, 1998] Rios-Mercado, R. and Bard, J. (1998). Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. *Computers & Operations Research*, 25(5) :351–366.

[Ruiz and Allahverdi, 2009] Ruiz, R. and Allahverdi, A. (2009). New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness. *International Journal of Production Research*, 47(20) :5717–5738.

[Ruiz and Maroto, 2006] Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169, pp. 781–800.

[Ruiz and Rodriguez, 2010] Ruiz, R. and Rodriguez, J. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205, pp. 1-18.

[Ruiz et al., 2009] Ruiz, R., Vallada, E., and Fernández-Martínez, C. (2009). Scheduling in flowshops with no-idle machines. pages 21–51.

[Saadani et al., 2005] Saadani, N., Guinet, A., and Moalla, M. (2005). A travelling salesman approach to solve the f/no-idle/cmax problem. *European Journal of Operational Research*, 161(1) :11 – 20. IEPM : Focus on Scheduling.

[Salmasi, 2005] Salmasi, N. (2005). *Multi-stage group scheduling problems with sequence dependent setups*. PhD thesis.

[Salmasi et al., 2010] Salmasi, N., Logendran, R., and Skandari, M. (2010). Total flow time minimization in a flowshop sequence-dependent group scheduling problem. *Computers and Operations Research*, 37, pp. 199-212.

[Santos et al., 1995] Santos, D., Hunsucker, J., and Deal, D. (1995). Global lower bounds for flow shops with multiple processors. *European Journal of Operational Research*, 80(1) :112 – 120.

[Schaller et al., 2000] Schaller, J., Gupta, J., and Vakharia, A. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*, 125(2) :324–339.

[Shafaei et al., 2011] Shafaei, M., Rabiee, M., and Mirzaeyan, M. (2011). An adaptive neuro fuzzy inference system for makespan estimation in multiprocessor no-wait two stage flow shop. *International Journal of Computer Integrated Manufacturing*, 24(10) :888–899.

[Shahvari et al., 2012] Shahvari, O., Salmasi, N., Logendran, R., and Abbasi, B. (2012). An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. *International Journal of Production Research*, 50(15) :4237–4254.

[Shao et al., 2017] Shao, W., Pi, D., and Shao, Z. (2017). Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion. *Applied Soft Computing*, 54 :164 – 182.

[Sioud et al., 2013] Sioud, A., Gravel, M., and Gagné, M. (2013). A genetic algorithm for solving a hybrid flexible flowshop with sequence dependent setup times. *IEEE Congress on Evolutionary Computation,Mexico*, 40, pp. 1064-1075.

[Sioud et al., 2012] Sioud, A., Gravel, M., and Gagné, C. (2012). A hybrid genetic algorithm for the single machine scheduling problem with sequence-dependent setup times. *Computers & Operations Research*, 39(10) :2415–2424.

[Song and Tang, 2008] Song, J. and Tang, J. (2008). Scheduling model and discrete particle swarm optimization algorithm for roller annealing. In *2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing*, pages 770–775. IEEE.

[Sriskandarajah, 1993] Sriskandarajah, C. (1993). Performance of scheduling algorithms for no-wait flowshops with parallel machines. *European Journal of Operational Research*, 70(3) :365–378.

[Sriskandarajah and Sethi, 1989] Sriskandarajah, C. and Sethi, S. (1989). Scheduling algorithms for flexible flowshops : worst and average case performance. *European Journal of Operational Research*, 43(2) :143–160.

[Su, 2003] Su, L. (2003). A hybrid two-stage flowshop with limited waiting time constraints. *Computers & Industrial Engineering*, 44(3) :409–424.

[Tasgetiren et al., 2011] Tasgetiren, M., Pan, Q., Suganthan, P., and Jin, G. (2011). A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. *International Journal of Production Research*, 49(16) :5033–5050.

[Tasgetiren et al., 2007] Tasgetiren, M., Pan, Q., Suganthan, P., and Liang, Y. (2007). A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion. In *Computational Intelligence in Scheduling, SCIS'07. IEEE Symposium on*, pages 251–258. IEEE.

[Ulungu et al., 1999] Ulungu, E., Teghem, J., Fortemps, P., and Tuyttens, D. (1999). Mosa method : a tool for solving multiobjective combinatorial optimization problems. *Journal of multi-criteria decision analysis*, 8(4) :221.

[Valérie, 2000] Valérie, B. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*, 64(1) :101 – 111.

[Vignier, 1997] Vignier, A. (1997). *Contribution à la résolution des problèmes d'ordonnancement de type monogamme, multimachine(Flow-shop hybride)*. PhD thesis.

[Wang and Liu, 2013a] Wang, S. and Liu, M. (2013a). A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. *Computers and Operations Research*, 40, pp. 1064-1075.

[Wang and Liu, 2013b] Wang, S. and Liu, M. (2013b). A heuristic method for two-stage hybrid flow shop with dedicated machines. *Computers and Operations Research*, 40, pp. 438-450.

[Wang et al., 2015] Wang, S., Liu, M., and Chu, C. (2015). A branch-and-bound algorithm for two-stage no-wait hybrid flow-shop scheduling. *International Journal of Production Research*, 53(4) :1143–1167.

[Widmer and Hertz, 1989] Widmer, M. and Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, 41(2) :186–193.

[Wu et al., 2012] Wu, C., Wu, W.-H., and Hsu, P-Hand Lai, K. (2012). A two-machine flowshop scheduling problem with a truncated sum of processing-times-based learning function. *Applied Mathematical Modelling*, 36(10) :5001–5014.

[Xie et al., 2004] Xie, J., Xing, W., Liu, Z., and Dong, J. (2004). Minimum deviation algorithm for two-stage no-wait flowshops with parallel machines. *Comput Math Appl*, 47 :1857–1863.

[Yalaoui and Chu, 2002] Yalaoui, F. and Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *International Journal of Production Economics*, 76(3) :265–279.

[Yalaoui et al., 2013] Yalaoui, N., Ouazene, Y., Yalaoui, F., Amodeo, L., and Mahdi, H. (2013). Fuzzy-metaheuristic methods to solve a hybrid flow shop scheduling problem with pre-assignment. *International Journal of Production Research*, 51(12) :3609–3624.

[Yang and Chern, 1995] Yang, D. and Chern, M. (1995). A two-machine flowshop sequencing problem with limited waiting time constraints. *Computers & Industrial Engineering*, 28(1) :63–70.

[Yang, 2011] Yang, J. (2011). Minimizing total completion time in two-stage hybrid flow shop with dedicated machines. *Computers and Operations Research*, 38(7) :1045–1053.

[Yang, 2013] Yang, J. (2013). A two-stage hybrid flow shop with dedicated machines at the first stage. *Computers & Operations Research*, 40(12) :2836–2843.

[Yang, 2015a] Yang, J. (2015a). Hybrid flow shop with parallel machines at the first stage and dedicated machines at the second stage. *Industrial Engineering and Management Systems*, 14(1) :22–31.

[Yang, 2015b] Yang, J. (2015b). Minimizing total completion time in a two-stage hybrid flow shop with dedicated machines at the first stage. *Computers and Operations Research*, 58 :1–8.

[Yaurima et al., 2007] Yaurima, V., Burtseva, L., and Tchernykh, A. (2007). Hybrid flowshop with unrelated machines, sequence dependent setup time and availability constraints : An enhanced

crossover operator for a genetic algorithm. In *International Conference on Parallel Processing and Applied Mathematics*, pages 608–617. Springer.

[Yazdani and Naderi, 2016] Yazdani, M. and Naderi, B. (2016). Modeling and scheduling no-idle hybrid flow shop problems. *Journal of Optimization in Industrial Engineering*, 10(21) :59–66.

[Zandieh et al., 2006] Zandieh, M., Fatemi, G., and Moattar Husseini, S. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Applied Mathematics and Computation*, 180 :111–127.

Titre : Ordonnancement d'un système de production industriel complexe: flow shop hybride avec des machines dédiées soumis à différentes contraintes temporelles

Mots clés : Ordonnancement, contraintes temporelles, méta-heuristique, flow shop hybride, MIP

Résumé : L'accroissement des profits, à travers l'amélioration de la productivité et la réduction des pertes de matières, représente un objectif primordial pour les entreprises industrielles. Dans cette thèse, nous nous intéressons à la résolution d'un problème industriel complexe réel avec des contraintes de temps. Nous nous sommes intéressés, tout d'abord, à un objectif principal, soit la minimisation des dates de fin de production, suivi d'un objectif secondaire qui est la minimisation des quantités de déchets non recyclables.

Dans un premier temps, nous avons modélisé le problème par des modèles mathématiques, que nous avons résolu à l'aide d'un solveur. Dans un second temps, nous avons proposé une méthode approchée en forme d'algorithmes évolutionnistes. Cette méthode est appliquée aux deux objectifs mentionnés ci-dessus séparément. Une troisième méthode est ensuite appliquée à l'objectif principal, à savoir une méthode arborescente approchée. Nous avons testé les algorithmes proposés sur des instances inspirées d'un cas réel ; issues d'une entreprise du secteur agroalimentaire et sur des instances inspirées de la littérature.

Title : Scheduling of a complex industrial production system: hybrid flow shop with dedicated machines and different time constraints

Keywords : Scheduling, temporal constraints, meta-heuristic, Hybrid flow shop, MIP

Abstract: Increasing profits, through the improvement of productivity and minimizing waste, is a primary objective for industrial companies. In this thesis, we are interested in solving a real complex industrial problem with time constraints. Firstly, we were interested in minimizing completion time (C_{max}). Secondly, we focused on minimizing of non-recyclable waste. As a first step, we formulated the problem by mathematical models, which we solved using a solver.

In a second step, we proposed an approximate method in the form of evolutionary algorithms. Both methods were applied to the two objectives mentioned above separately. Then, a third method which is a tree-search algorithm was applied only to the main objective. We tested the proposed algorithms on instances inspired from a real case; from an agri-food business, and also on instances inspired from the literature.