Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique

Guillermo Campos Ciro

Thèse
de doctorat
de l'UTT

Guillermo CAMPOS CIRO

Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique

2015TROY0035

Année 2015

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Guillermo CAMPOS CIRO

le 3 décembre 2015

Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique

JURY

M. A. MOUKRIM	PROFESSEUR DES UNIVERSITES	Président
M. S. DAUZÈRE-PÉRÈS	PROFESSEUR ENSM SAINT-ETIENNE	Rapporteur
M. F. DUGARDIN	MAITRE DE CONFERENCES	Directeur de thèse
M. A. EL MOUDNI	PROFESSEUR DES UNIVERSITES	Examinateur
M. N. REZG	PROFESSEUR DES UNIVERSITES	Rapporteur
M. F. YALAOUI	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Personnalités invitées

M. L. AMODEO	PROFESSEUR DES UNIVERSITES
M. R. KELLY	CHEF D'ENTREPRISE NORELEM

Remerciements

Les travaux présentés dans cette thèse ont été réalisés dans le cadre d'un projet thèse CIFRE entre l'Université de Technologie de Troyes (UTT) et la société NORELEM SAS à Fontaine Les Grès, sous les meilleurs auspices et sous l'encadrement de mes directeurs de thèse : Farouk YALAOUI et Frédéric DUGARDIN et le chef d'entreprise : Russell KELLY.

Je tiens à leur exprimer toute ma gratitude pour avoir accepté de diriger ma thèse. Je les remercie infiniment pour leur soutien, leur disponibilité, leurs encouragements ainsi que pour m'avoir guider autant au niveau scientifique qu'humain.

Je remercie également Mme Maïté CLEVY, ingénieure projet de NORELEM, pour toute sa participation dans le développement de cette thèse et pour m'avoir aidé et soutenu pendant ces trois années, nous avons travaillé très bien en équipe. J'exprime aussi toute ma gratitude à David MESSANT, chef d'atelier, pour sa disponibilité et excellente disposition pour travailler en équipe, à Marie-José ARAUJO, Fabien COUDRAT, Linda LOPEZ, Jerôme ROUSSELOT et Damien DUBOIS pour la bonne ambiance du travail et leur amabilité, et finalement à toute la grande famille NORELEM qui m'a accueilli de la meilleure façon en facilitant mon travail.

J'exprime toute ma reconnaissance aux membres du jury qui me font l'honneur de participer à l'évaluation de cette thèse : Monsieur Stéphane DAUZÈRE-PÉRÈS, Professeur à l'École Nationale Supérieure des Mines de Saint-Etienne et à Monsieur Nidhal REZG, Professeur à l'Université de Lorraine en leur qualités de rapporteurs, à Monsieur Abdellah EL-MOUDNI, Professeur à l'Université de Technologie de Belfort-Montbeliard et à Monsieur Aziz MOUKRIM, Professeur à l'Université de Technologie de Compiègne en leur qualité d'examinateurs.

Je tiens à remercier mes collègues de l'UTT qui m'ont soutenu pendant ce temps et en particulier à Juan Carlos RIVERA, Andrés BERNATE, Elyn SOLANO, Jorge VICTORIA, Andrés GUTIERREZ, Marie-Helene MAI, Julien AUTOURI, Matthieu LE BERRE, Birôme BA, Syrine AIT, Slim DAOUD et tous les autres membres du LOSI. Je remercie Yassine OUAZENE, Julie RUBASZEWSKI et Xixi WANG pour la bonne ambiance du bu-

reau ainsi que pour leur sens de l'humeur. Je remercie également tous les autres colombiens de l'UTT. J'exprime aussi toute ma gratitude à Véronique BANSE et Bernadette ANDRE pour l'excellent service et la gentillesse au secrétariat du pôle ROSAS ainsi qu'aux membres de l'Ecole doctorale, et particulièrement à Pascale DENIS, Isabelle LECLERCQ et Thèrese KAZARIAN pour l'amabilité et l'aide prêtée dans les procédures administratives de l'école.

Enfin, il n'y a pas de mots suffisants pour exprimer mes sentiments de remerciement à ma famille en Colombie, à ma mère Gloria pour m'encourager et me soutenir toujours, à mon père Guillermo pour me conseiller et motiver, et à mon frère Ricardo pour son soutien inconditionnel. A mes grand-parents, à tous mes oncles, tantes et cousins et à mes amis en Colombie, merci beaucoup pour m'avoir soutenu pendant ces années de thèse. Merci énormément à Adriana pour son positivisme et sa compagnie pendant tous les moments de la thèse.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La globalisation, les changements économiques et l'évolution technologique présents dans le panorama industriel actuel, exigent des entreprises plus réactives, qui répondent aux besoins du marché, et s'adaptent à l'évolution de leurs clients. La compétitivité mondiale favorise les entreprises qui ne cessent d'améliorer leurs processus en développant des techniques et des outils qui permettent d'améliorer leur efficacité, afin de conserver des parts de marché.

La planification et l'ordonnancement de la production est généralement un sujet auquel s'intéressent les sociétés qui cherchent à offrir un service de qualité. En effet, la minimisation du retard dans la fabrication et la livraison de ses produits, peut se traduire par des gains significatifs, surtout dans des secteurs où la concurrence est difficile. La prise de décisions au niveau opérationnel, liées à la minimisation des coûts et à l'augmentation de la production sont deux aspects notables qui attirent couramment l'attention des sites industriels complexes.

Dans le cadre de cette thèse, nous avons développé des outils d'aide à la décision pour la résolution du problème d'ordonnancement dans un atelier de type open-shop avec des restrictions relatives aux compétences et à l'affectation des ressources.

L'ordonnancement peut se traduire par la séquence de l'exécution des tâches dans un atelier, en respectant les relations de précédence entre les tâches (dates de début et de fin de traitement de chaque activité), en affectant les ressources nécessaires à employer et en optimisant le critère d'évaluation à considérer au moment de prendre une décision. Des définitions théoriques et exhaustives ont été établies par différents auteurs comme Baker [12], Graham *et al.* [72], Lawler *et al.* [110] et Brucker [23].

L'étude de ce problème d'ordonnancement d'un atelier de type open-shop, a permis l'identification des caractéristiques relatives à sa complexité et les méthodes les plus efficaces pour le résoudre. Dans ce problème, les ressources possèdent plusieurs compétences qui déterminent quelles tâches exécuter, les dates d'arrivée dépendent de chaque tâche, et l'objectif à minimiser dans un premier temps est le temps de séjour des produits dans le système de

production de la première opération jusqu'à la dernière.

Ce travail a conduit au développement d'une méthode de résolution pour le problème d'ordonnancement dans le type d'atelier décrit précédemment.

Un cas d'application a été identifié dans une entreprise du secteur de la production de pièces mécaniques, dédiée à la fabrication des éléments standards de précision. Ce type de produits est soumis à des contraintes de qualité et de résistance qui doivent être satisfaites le plus rapidement possible, en suivant les conditions d'un marché qui est réévalué constamment. L'enjeu de ce type d'entreprise est d'augmenter la valeur ajoutée de ces produits ainsi que la qualité et la vitesse de son service dès que la commande est prise jusqu'à la livraison de son produit (minimisation des temps d'attente des clients).

Dans ce cadre, la société Norelem s'est fixée pour objectif d'optimiser les flux de production de son atelier, afin de minimiser le retard des commandes clients et d'augmenter son niveau de qualité, déterminant dans le secteur des pièces mécaniques de précision.

Partant de ce constat, le projet collaboratif est né entre la société Norelem et le Laboratoire d'Optimisation des Systèmes Industriels (LOSI) de l'Université de Technologie de Troyes (UTT).

Différentes méthodes vont être testées afin de choisir laquelle est la plus efficace pour ce type de problème. L'idée est de proposer des systèmes d'aide à la décision applicables aux problématiques de production liées à l'atelier de l'entreprise Norelem et de contribuer à l'optimisation de ses flux de production.

Le contenu de ce mémoire est détaillé en 6 chapitres.

Le premier chapitre présente en détail le contexte de recherche, les enjeux et problématiques trouvés dans cette thèse et leur importance dans le système analysé.

Le deuxième chapitre dresse l'état de l'art sur le problème étudié et les différentes techniques pour résoudre la problématique.

Le troisième chapitre est dédié à l'optimisation par méthodes exactes de l'ordonnancement des ateliers de type open shop avec différentes contraintes de ressources et de disponibilité de tâches. Ce chapitre présente la description mathématique du problème, laquelle comporte les contraintes et l'objectif relatifs au fonctionnement de cet atelier. On présente alors les résultats relatifs à la résolution tout en observant les avantages et inconvénients relatifs de la modélisation. Les résultats trouvés et des analyses approfondies sont également présentés.

Le chapitre quatre illustre la résolution de la problématique d'ordonnancement de type

open shop en utilisant des méthodes approchées. Pour cela, les différentes approches testées sont exposées, ainsi que les résultats obtenus.

Le chapitre cinq aborde la résolution de ce problème d'ordonnancement en prenant en compte l'optimisation multi-objectif. Des méthodes de résolution approchées adaptées (Non-Dominated Sorting Genetic Algorithm - NSGA - II et NSGA - III) sont exposées, avec la description des objectifs employés. Des tests effectués sur des instances théoriques sont présentés.

Les applications industrielles sont présentées dans le chapitre six. Ce chapitre représente une étape très importante du projet car elle permet de valider les méthodes de résolution proposées et montrer notre contribution sur un contexte réel au travers des différentes études.

Enfin, la dernière partie est consacrée aux conclusions et perspectives portant sur les résultats obtenus dans cette thèse.

Chapitre 1

Introduction : Contexte de recherche

La gestion d'un site de production fait appel au traitement de diverses problématiques issues de différents domaines : les achats, les ventes, le service du client, entre autres qui permettent un bon fonctionnement du site. De nos jours, la résolution de ces problématiques est nécessaire dans un nombre croissant de sociétés, se traduisant par une intégration des solutions de manière physique ou logique afin d'améliorer la productivité. Dans ce mémoire, nous nous sommes concentrés sur la planification de la production d'un atelier avec une structure complexe.

Les travaux développés dans cette thèse ont été orientés vers l'amélioration des performances d'un atelier de production mécanique de type open shop. L'atelier en question présente des contraintes par rapport à la disponibilité des produits et des ressources avec multi-compétences lors de l'exécution d'une opération. D'autres contraintes supplémentaires sont considérées et décrites dans un prochain chapitre.

Les sections suivantes décrivent la problématique d'ordonnancement d'ateliers de type open shop. De plus, des méthodes de résolution ont été développées pour ce problème.

Le contexte de recherche est décrit plus en détail dans le chapitre 6, où le cas industriel est exposé.

1.1 Optimisation d'un atelier de production

Nous traitons ce problème en considérant la disponibilité des tâches, des machines ainsi que les ressources humaines et les outils nécessaires pour exécuter une opération. L'objectif est ici de minimiser le temps de séjour des produits dans l'atelier depuis la commande client jusqu'à la fin de traitement du produit dans l'atelier.

Les sections suivantes décrivent les principaux concepts relatifs aux tâches et aux machines, afin de mieux comprendre l'organisation et le fonctionnement de l'atelier.

1.1.1 Ordonnancement : définitions et notations

La gestion de la production dans un atelier a déjà été largement étudiée dans la littérature. Trouver l'ordre de passage des tâches sur un ensemble de serveurs (habituellement appelés machines) est un problème connu sous le nom d'ordonnancement de tâches. Ce type de problème présente une échéance à court terme, c'est-à-dire qu'il fait partie des décisions opérationnelles qui incluent également la gestion des stocks, la gestion du personnel, la conception des lignes de production, entre autres. Plusieurs définitions de l'ordonnancement ont été développées par différents auteurs, dans notre mémoire nous reprenons celle donnée par Graham *et al.* [72] qui introduit une classification en utilisant une nomenclature sur trois champs. L'ordonnancement est lié à un ensemble de tâches N qui sont exécutées sur un ensemble de machines M, en respectant certaines contraintes de disponibilité, caractéristiques des opérations, types de ressources à utiliser et surtout l'organisation et la configuration de l'atelier. La notation proposée par Graham *et al.* [72] : $\alpha/\beta/\gamma$, présente la structure du problème (α), les contraintes d'ordonnancement à respecter (β) et l'objectif à optimiser (γ).

1.1.1.1 L'environnement des machines (le champ α)

Le problème est représenté par deux sous-champs, le premier fait référence à la configuration de l'atelier :

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

La deuxième partie décrit le nombre de machines à considérer. Si cette valeur est un entier fixe, le nombre de machines est égal à ce numéro. Si la valeur est m, le nombre de machines est variable (supérieur à 1). Enfin, si la première partie est vide, la deuxième partie reste vide ou égale à 1.

1.1.1.2 Les contraintes d'ordonnancement (le champ β)

Les contraintes d'ordonnancement précisent la description des tâches et des machines, ce qui limite l'espace des solutions possibles. Nous trouvons par exemple les notations suivantes :

- $pmtn, \emptyset \rightarrow$ Les tâches peuvent être interrompues au moment de leur exécution. Pour tout autre cas, le champ reste vide.
- $res, res1, \emptyset \rightarrow$ La présence de ressources supplémentaires est comprise (*res, res1*).
- $prec, tree, \emptyset \rightarrow$ Les relations de précédence (*prec*) sont marquées, avec des précisions si elles sont nécessaires (*tree*)
- $r_i, \emptyset \rightarrow$ Les dates de disponibilité des tâches sont indépendantes pour chaque tâche.
- $p_{im} \rightarrow$ Les temps opératoires sont définis par tâche et par machine, par tâche, par machine, ou sont identiques dans tous les cas.

1.1.1.3 Les objectifs (le champ γ)

Cette dernière partie définit l'objectif à optimiser pour le problème d'ordonnancement. Il y a de nombreux objectifs à considérer et ils dépendent de l'environnement étudié. Ci-dessous, nous présentons les plus connus :

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

Dans le chapitre suivant "État de l'art", nous exposons les objectifs les plus étudiés dans le type d'atelier considéré par ce rapport.

Dans la littérature, nous pouvons trouver des définitions et classifications différentes du problème. Brucker [23] a proposé de nouvelles classifications pour certains cas qui n'étaient pas décrits par Graham *et al.* [72]. L'auteur a pris en compte les cas où les machines sont dédiées à la réalisation de certains types de tâches ainsi que le cas où une machine peut

traiter plusieurs tâches en parallèle. D'autres propriétés des problèmes d'ordonnancement ont été considérées : la production par lots, les types de précédences, les systèmes réentrants ainsi que la permutation dans le problème de flow shop. Les fonctions objectifs classiques et le comportement des données employées sont aussi exposées. Les données (caractéristiques des tâches et machines) peuvent être connues à l'avance de façon déterministe ou générer des problèmes stochastiques qui suivent des lois de probabilité.

D'autres auteurs comme Garey et Johnson [66], Lawler *et al.* [109], Papadimitriou et Steiglitz [149] proposent de classer les problèmes d'ordonnancement suivant leur complexité. La théorie de la complexité donne un cadre mathématique pour faire le classement des problèmes entre faciles et difficiles, à partir de l'analyse des temps de résolution du problème.

D'après le classement des problèmes par rapport à la complexité, différentes méthodes de résolution : exactes et heuristiques ont été proposées. Les problèmes NP-difficiles peuvent être résolus avec des procédures de séparation et d'évaluation, des méthodes de programmation dynamique ou programmation linéaire [23] qui ont des limites en termes de taille du problème à cause du temps de calcul.

Les méthodes approchées ont été développées pour proposer des solutions de bonne qualité aux problèmes de tailles grandes en réduisant les temps de calcul. Nous avons alors les heuristiques, méthodes qui utilisent des règles empiriques de priorité à partir de propriétés d'optimalité ou dominance, et les métaheuristiques qui prennent des procédures de générations itératives de solutions qui sont hybridées aux concepts heuristiques. Nous expliquerons ces concepts plus en détail dans le chapitre 2 où nous présenterons certaines métaheuristiques destinées à résoudre notre problème.

1.1.2 Les indicateurs de performance des méthodes de résolution

Selon les types de problèmes, leurs complexités et leurs tailles, nous pouvons trouver des cas où il n'est pas possible d'obtenir une solution exacte. C'est pourquoi nous utilisons des méthodes approchées, et à partir de là nous devons mettre en place des indicateurs pour mesurer la qualité de la solution obtenue.

Nous avons choisi les indicateurs suivants pour les méthodes développées.

1.1.2.1 L'écart relatif (GAP)

Le premier indicateur de performance permettant de comparer les solutions obtenues dès méthodes proposées est le GAP. Cet indicateur est utilisé souvent dans les problèmes

combinatoires. Dans ce mémoire, nous utilisons la définition suivante :

$$GAP = \begin{cases} \frac{H - S_{ME}}{H} & H > S_{ME} \\ 0 & Sinon \end{cases} \tag{1.1}$$

Le calcul du GAP pour une instance donnée est fait à partir de la différence entre la solution obtenue par la méthode approchée (H) et la méthode exacte (S_{ME}).

1.1.2.2 Temps d'exécution

Comme nous traitons des problèmes opérationnels, nous devons prendre des décisions dans un court délai, donc nous avons besoin de méthodes de résolution qui soient performantes en termes de temps de calcul pour les instances de grandes tailles.

1.2 Contexte industriel

Après la description théorique du problème d'ordonnancement, les notations utilisées dans la littérature, l'introduction aux méthodes de résolution et ses indicateurs de performance, nous présentons le problème dans son contexte industriel afin de mieux comprendre ses origines.

1.2.1 L'entreprise Norelem

Norelem est une société spécialisée dans le domaine de la mécanique de précision qui fait partie du groupe Allemand Kipp, étant sa maison mère depuis 2000. Son activité principale est : la conception, la réalisation et commercialisation d'éléments standards mécaniques, de systèmes de bridage modulaires, de systèmes modulaires en alliage léger et d'éléments pour montage de contrôle.

De la conception à la livraison, toutes les étapes de l'élaboration de l'ensemble de gammes (produits standards et spéciaux) sont développées par Norelem.

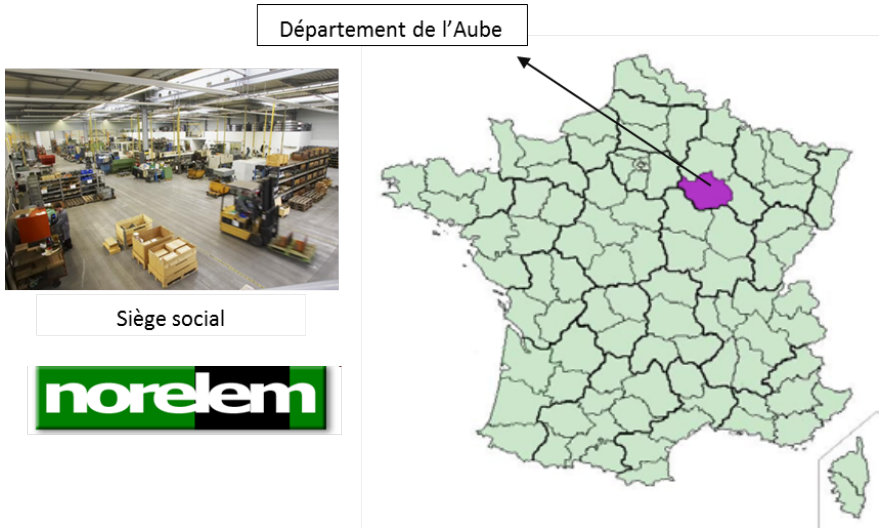


FIGURE 1.1 – Norelem

1.2.2 Sa mission, ses services et ses produits

1.2.2.1 Mission

L'entreprise Norelem recherche et développe des solutions pratiques aux nouvelles technologies de production en perfectionnant constamment ses produits et ses gammes. L'entreprise suit des projets en partenariat avec de grandes entreprises publiques et privées ou à la demande de l'Éducation Nationale.

Le réseau de vente et de distribution de la société est présent sur les cinq continents, doté de spécialistes, de moyens techniques et de contrôle, performants pour donner rapidement satisfaction à ses clients.

1.2.2.2 Services

La différence principale de Norelem avec ses concurrents est l'importance accordée aux services : une production propre et constamment contrôlée ainsi qu'une logistique fidèle aux délais. La société dispose de plusieurs services destinés à la satisfaction des besoins des clients :

- **Services et prestations spéciales aux clients :** Développements spécifiques depuis l'étude de conception, de réalisation jusqu'à la livraison pour tout client demandeur de solutions techniques.

- **Service hotline commerciale et hotline technique :** Assurer par le pôle com-

mercial de 8h30 à 17h 5/7j. Les clients peuvent suivre leurs commandes, commander catalogues et CD-rom.

- **Bibliothèque CAO Norelem :** la bibliothèque informatique permet aux clients de sélectionner et trouver les composants (en modèles 2D et 3D) les plus adaptés à leurs besoins.

- **Le site internet :** il permet d'accéder aux descriptifs techniques des produits, à leur disponibilité, au format CAO 2D et 3D et aux demandes de prix 24h/24.

- **Formation éducation nationale en matériel didactique pour fabrication de :** skate-board, arroseur carré, tournevis à cliquet.

1.2.2.3 Produits

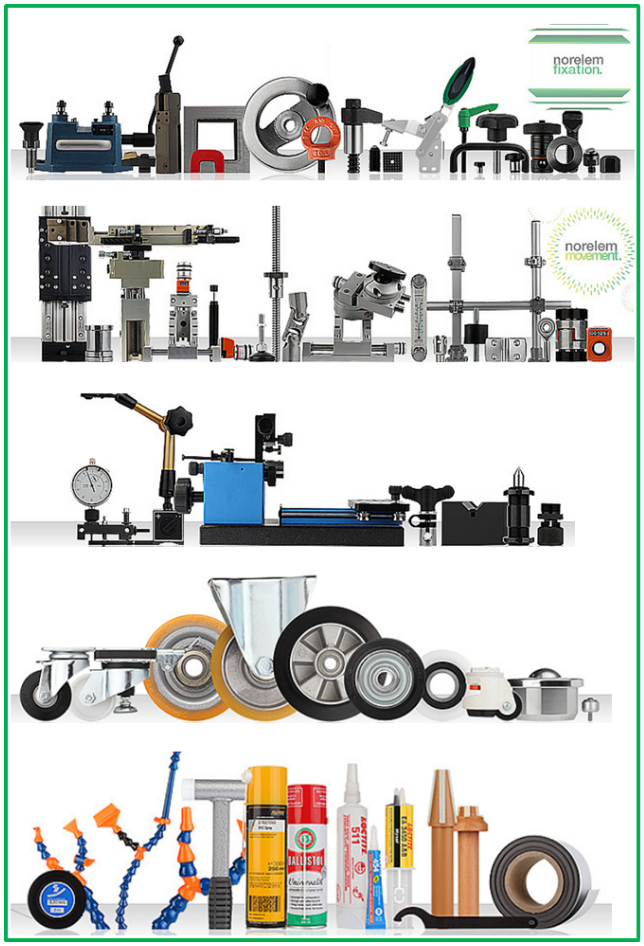
L'entreprise Norelem fait la conception, la réalisation et la commercialisation de plusieurs groupes de produits qui seront présentés ci-après :

FIGURE 1.2 – Catalogue principal Norelem "Big Green Book" contenant plus de 36000 références sur 1400 pages

Ses éléments standards sont présents dans toute l'industrie depuis les panneaux solaires des satellites jusque dans les portes de cuisines, en passant par les vélos, les trains, les airbus ainsi que le matériel médical (figure 1.3).

Kit de bridage pour mesure par analyse d'image

Le kit bridage Norelem pour mesure sans contact s'adapte à toutes les situations, les pièces et les machines. Trois versions sont proposées pour faire face à plusieurs situations (figure 1.4) : petites machines(a), pièces cylindriques(b) et des mesures en série(c).



Eléments standards mécanique

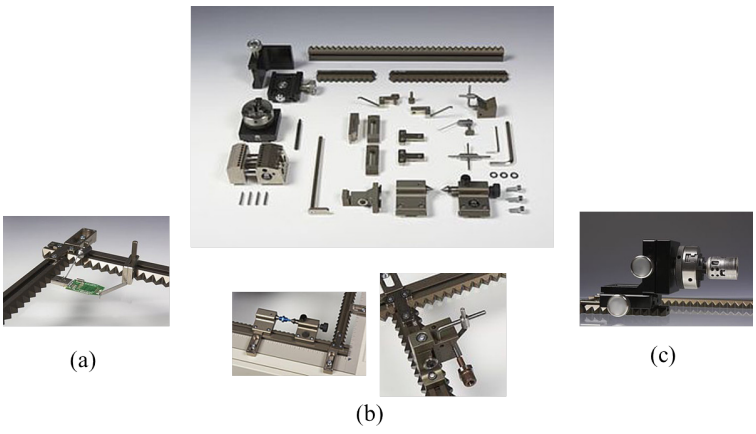
Systèmes et composants pour la construction de machines et d'installations

Éléments de mesure et de contrôle

Éléments de transport et manutention

TechnoShop norelem

FIGURE 1.3 – Produits du "Big Green Book"



(a) (b) (c)

FIGURE 1.4 – Kid de bridage Norelem

Nouvelle technologie de serrage pour machines 5 axes

L'étau 5 axes Norelem est un outil de productivité pour les fraisages à 5 axes. Il répond aux opérations d'usinage qui sont faites sans démontage où une grande rigidité et une très

bonne répétabilité sont des critères prépondérants dans le choix d'un moyen de serrage adapté aux caractéristiques des machines.

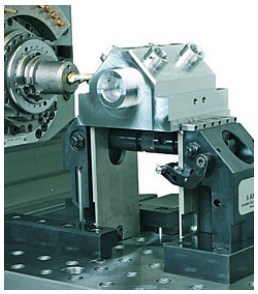


FIGURE 1.5 – Serrage pour machines 5 axes

Tables de soudage, éléments de serrage modulaire

Norelem propose un système de table de soudage et d'éléments modulaires qui sont utilisés principalement comme supports de travail, de préparation, de mesure et de marquage. L'optimisation de l'ergonomie des postes de travail augmente la productivité et améliore la qualité.

FIGURE 1.6 – Table de soudage et accessoires

1.2.3 Analyse de l'existant - état des lieux

Afin de mieux comprendre notre problématique, nous allons faire la présentation et description de l'atelier de production existant à Norelem et à partir de là, nous commencerons à identifier les différents points qui ont motivé ce travail de recherche.

Comme nous l'avons exposé précédemment, le site se trouve à Fontaines Les Grès (près de Troyes, Aube) et la superficie des bâtiments est estimée à $3100m^2$. Il dispose d'un atelier de production de pièces mécaniques de précision, et où les ressources sont groupées par nature d'activité raison pour laquelle le cheminement du produit est discontinu. Cet atelier est dédié à la production en petites et moyennes séries de produits très divers.

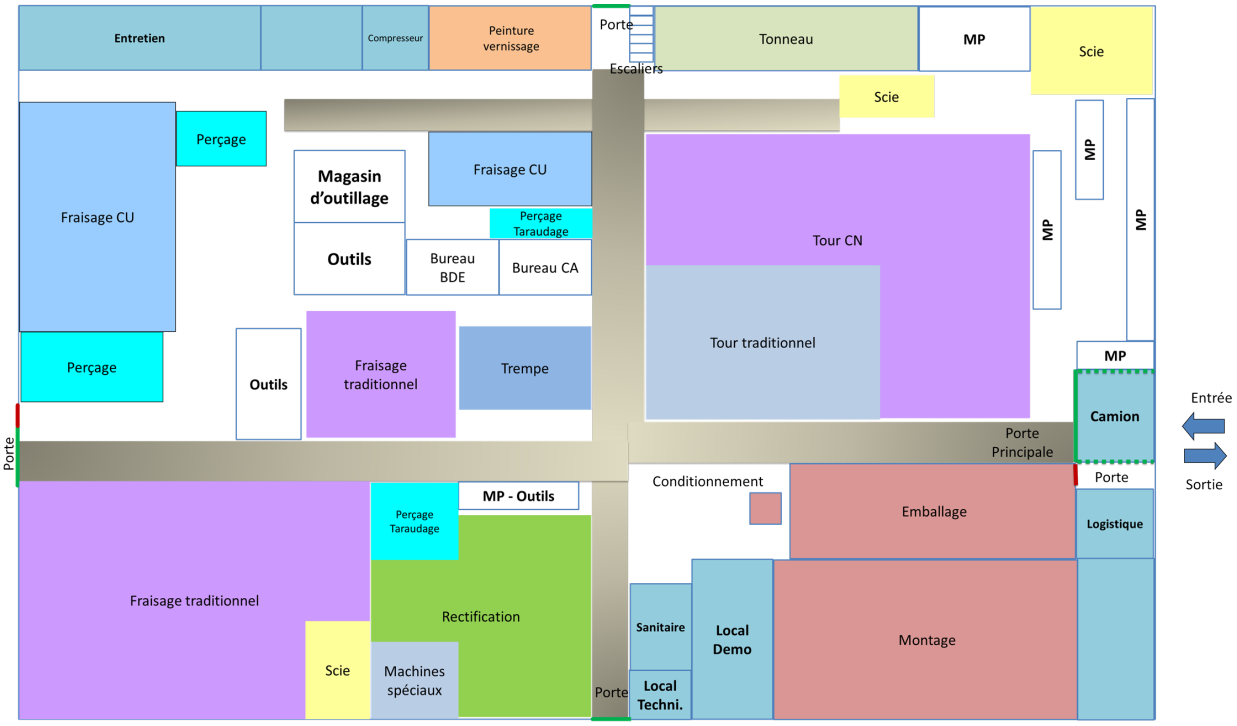


FIGURE 1.7 – Plan atelier Norelem par zone opératoires

La société Norelem a un atelier de production mécanique qui travaille en partenariat avec sa maison mère (Kipp) en Allemagne. Norelem est client dans certains cas et également fournisseur de Kipp dans d'autres cas. Chaque semaine, les pièces réalisées dans l'atelier sont envoyées à son centre logistique européen à Sulz (Bad Wurtemberg) deux fois par semaine d'où elles seront ensuite dispatchées à tous les clients du groupe dans le monde.

La fabrication est conditionnée par l'apparition d'un besoin du client - commande client de Kipp, qui génère une alerte de réapprovisionnement de stock. La gestion de l'atelier est gérée par un ERP allemand Sage Bäurer Industrie 5.1 (appelé B2 industrie) couplé avec un ERP commercial Pikadeli. L'ERP SAP sera prochainement employée (projet en cours).

Un autre logiciel de gestion des temps de production nommé BMS est aussi utilisé. Celui-ci permet vérifier l'ordre de fabrication (OF), l'état d'avancement des opérations, entre autres. Cette base de données fonctionne avec un système de douchette de lecture des code-barres de chaque opération d'un OF.

La production passe sans interruption par les opérations définies dans une gamme, selon la disponibilité des machines et des matières premières. Certaines opérations, selon le type de produit, peuvent être sous-traitées à l'extérieur, comme par exemple les opérations de traitement de surface et traitement thermique.

L'estimation du nombre de références produites sur le site en un an est de 5000 références,

et l'étendu des tailles de série de 1 à 5000 pièces, avec en moyenne 100 à 500 sur commande numérique (CN) et de plusieurs dizaines sur les autres machines traditionnelles. De plus, le temps de cycle des produits ne justifie pas un investissement en robotique.

A partir de cette première notion, nous pouvons dire que l'atelier Norelem est flexible, c'est-à-dire, qu'il a une structure avec de nombreux avantages :

- Adaptation rapide aux variations du marché, aussi bien en quantité qu'en nature.
- Satisfaction du client, grâce à la livraison au bon moment du produit voulu.
- Réduction des stocks et des en-cours.
- Meilleur contrôle de la production : mise à disposition d'informations en temps réel.

De plus, la société ne gère pas de stock de produits finis au sein de Norelem France, tous les produits sont expédiés au stock central en Allemagne, hormis quelques exceptions dédiées à des clients directs et du stock de composants intermédiaires qui permettent la réalisation des montages de produits finis.

FIGURE 1.8 – Atelier Norelem

A propos du parc machine de la société, la flexibilité des équipements de fabrication (ressources humaines comprises) est primordiale afin de s'adapter à la demande des clients. Cet atelier est destiné à la production de pièces variées du point de vue des formes et des dimensions, des matières. Divers types de pièces d'une famille de produits peuvent avoir différents processus de fabrication. Cette diversité de pièces, ainsi que les possibilités de combinaison qui en résultent pour l'affectation des pièces aux machines de l'atelier supposent un pilotage en temps réel de l'atelier. Actuellement, dès qu'une machine est disponible, l'opérateur recherche une opération d'un ordre de fabrication, parmi celles qui sont en attente, qui doit passer sur cette machine pour continuer la fabrication d'un produit et ainsi minimiser

les temps d'arrêt machine. De la même façon, en cas de panne d'une machine, un mode de production équivalent et/ou dégradé est généralement possible avec les machines restantes.

Par rapport au fonctionnement des machines, le temps d'ouverture d'une machine CN théorique est de 14 heures et parfois plus, en sachant qu'elle peut fonctionner pendant la nuit en temps masqué. Le temps d'ouverture d'une machine traditionnelle est de 7 heures, ce qui est le temps moyen de présence des opérateurs.

L'atelier Norelem est composé de centres d'usinage à commandes numériques, de tours à commandes numériques, de tours traditionnels, de rectifieuses à CN, de fraiseuses, de scies à ruban automatiques, de scies à ruban traditionnelles, de perceuses à colonne, de presses de montage, d'une machine à trempe partielle par induction et d'une fraiseuse conventionnelle à picots. L'atelier dispose aussi d'une zone de préréglage, d'une salle de métrologie et contrôle, d'une salle de peinture et une autre dédiée aux tests mécaniques et enfin un magasin d'outillage. Toutes ces machines permettent une production de haute qualité afin de satisfaire les demandes du marché et ainsi fournir les meilleurs produits aux clients.

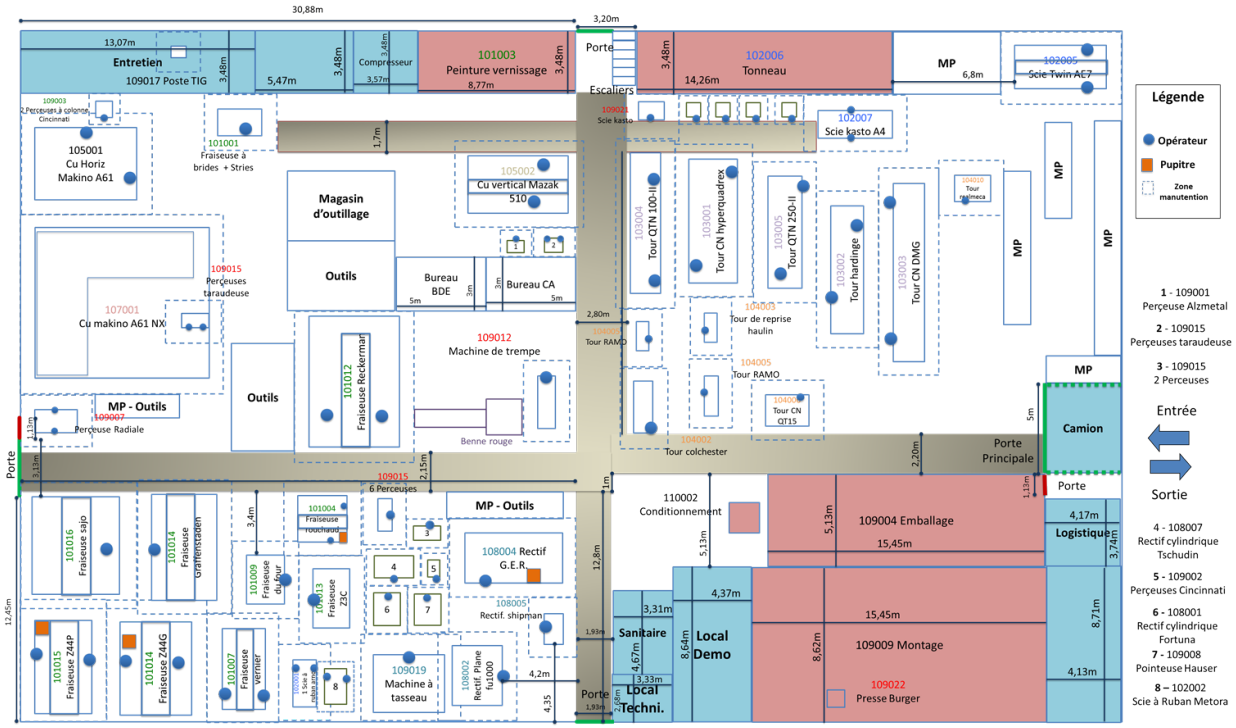


FIGURE 1.9 – Plan atelier Norelem

Maintenant que nous avons donné une description générale de l'entreprise, de ses processus, des logiciels internes utilisés, des machines et salles disponibles, il nous faut mettre l'accent sur les points clés de la situation actuelle qui ont conduit au développement de ce travail de recherche.

- Les prévisions de production ne sont pas possibles. Chaque année le panel de production est d'une grande diversité de produits finis qui changent fortement en fonction du carnet de commandes, il n'y a pas de produit dominant (best seller ou le plus demandé par les clients).
- Actuellement, il y a une forte fluctuation sur la production, les retards s'accumulent en raison de l'augmentation du carnet de commandes. En guise de solution, une notion d'urgence a été instaurée afin de donner une ordre de priorité à la production. Celle-ci évolue presque chaque jour selon les nouvelles priorités qui arrivent de la maison mère Kipp.
- Les 95% de la production sont conditionnés par la maison mère Kipp, soit pour des commandes de clients directes ou pour du réapprovisionnement de distributeurs.
- Il y a un vingtaine d'opérateurs de production qui ont des compétences différentes par rapport aux opérations existantes dans l'atelier : fraisage, tournage, débit, rectification, montage de pièces, peinture, emballage. Selon les compétences demandées par l'opération, elle peut être réalisée par l'opérateur x ou l'opérateur y.
- Un autre aspect important est le fait que les ordres de fabrication à traiter sur chaque période et machine ne sont pas séquencés, phénomène qui constitue un réservoir de réactivité permettant l'adaptabilité de l'entreprise aux aléas quotidiens, des retards d'approvisionnement, des commandes urgentes, des pannes machines, d'absence opérateur, entre autres.
- Actuellement, la planification de la production de l'entreprise est gérée par le chef d'atelier qui met au point un plan de charge d'une à plusieurs semaines, en listant les ordres de fabrication sur lesquels travaillera chaque machine ou poste de charge au cours de la semaine, ce qui est suffisant pour répartir le travail et suivre la production.

Ces points clés avaient permis l'identification d'une situation problématique existante liée au système de production. Le chef d'atelier et les opérateurs gèrent conjointement l'ordonnancement des centres de charge où ils exécutent une opération, de façon autonome, en collaboration avec les autres opérateurs des centres de charges en amont et/ou aval à leur production, en rappelant que le chef d'atelier détermine les priorités de production. Toute cette démarche est orale avec un support papier listant les OF's à réaliser sans ordonnancement prédéfini.

Cette situation montre un manque d'organisation, qui d'une part génère des temps-morts de production et d'autre part peut donner lieu à des dysfonctionnements importants lors d'une absence du chef d'atelier, et ainsi affecter la satisfaction du client.

D'après la description de la situation actuelle, une planification et un ordonnancement

optimisés sont requis pour minimiser les temps-morts et améliorer le flux de la production de l'entreprise afin de remplir plusieurs objectifs :

- D'abord la grande diversité de combinaisons de fabrication en termes de machines et de séquences des opérations, l'entreprise veut connaître quand et sur quelle machine produire pour réduire la durée globale du cycle de fabrication, agir sur la réactivité, avantager les teps d'opérations de sous-traitance et optimiser l'utilisation des ressources (matérielles et humaines).
- Avoir une interaction plus dynamique entre la partie commerciale et la production, de telle manière que les dates de livraison soient plus réalistes, et que les passages de commandes assurent un bon équilibre entre le nombre d'ordres de fabrication et la charge des machines.
- Augmenter la rentabilité financière de l'entreprise, en minimisant les délais de livraison aux clients ainsi qu'en améliorant la productivité de l'atelier.
- Développer un outil d'aide à la décision qui permette de gérer facilement et rapidement l'ordonnancement de l'atelier en prenant en compte les ressources disponibles à chaque instant.

1.3 Ordonnancement d'un atelier de type open shop

A partir des points décrits sur le comportement de l'atelier de l'entreprise Norelem dans la section précédente, des études préliminaires faites par une ingénieure projet de Norelem sur les différentes références de produits, des observations et discussions avec le chef d'atelier, nous pouvons faire les remarques suivantes :

Certains ordres de fabrication peuvent suivre un chemin déterminé selon son flux de production (flux unique - le processus d'élaboration de produits est linéaire). D'autres ordres sont conditionnés par le type de produit (flux variés). En plus, il existe des cas où un grand nombre de combinaisons de chemins est disponible pour gérer un ordre de fabrication.

C'est pour cette raison que, d'après l'analyse de données ainsi que d'après le comportement de la production, on en déduit qu'il existe une production avec des gammes ouvertes où plusieurs chemins peuvent être considérés au moment de lancer un ordre de fabrication. Ce type d'atelier est connu comme un open shop. La figure 1.10 représente 5 machines et 1 tâche, laquelle peut suivre plusieurs chemins de fabrication possibles.

Dans ce cadre, cette section présente la configuration d'atelier qui représente le mieux le système de production de l'entreprise Norelem et qui est étudié dans le chapitre 3. Nous

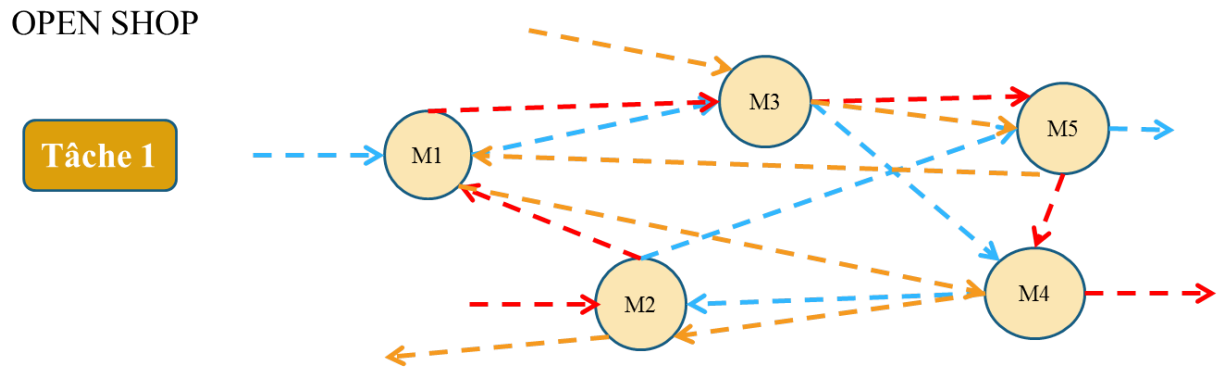


FIGURE 1.10 – Exemple d'open shop

parlons de l'ordonnancement de tâches d'un atelier de type open shop avec minimisation d'un objectif qui sera illustré dans les paragraphes suivants.

Le problème d'ordonnancement d'un ensemble de tâches avec des gammes ouvertes, est souvent étudié dans le cas d'applications industrielles où l'ordre de réalisation des opérations est sans importance tel que les situations de maintenance et les tests, qui sont les cas les plus connus.

Le problème de la gestion opérationnelle de ce type d'atelier est un problème combinatoire. Les tâches doivent être traitées sans interruption sur une machine et chaque machine doit traiter au maximum une tâche à la fois. L'objectif à minimiser est le retard total, mais selon les conditions de production de Norelem et en sachant que les produits finis sont envoyé au stock en Allemagne à la maison mère, il est plus opportun de minimiser le temps de séjour des produits dans le système de production, dès qu'il commence sa première opération jusqu'à la fin de la dernière. Cet objectif est souvent appelé, le temps total de séjour, où il y a un horizon fixe de travail et le retard n'est pas une contrainte forte (*date de fin effective de la tâche* $\ll$ *date d'échéance de la tâche*). De la même façon, la minimisation du temps total de séjour minimise indirectement le retard total.

Dans les chapitres suivants dédiés au problème d'ordonnancement de type open shop, nous fournissons une description plus détaillée de ce problème combinatoire, accompagné des modèles mathématiques et d'une série de méthodes approchées pour résoudre cette problématique d'une façon efficace.

1.4 Conclusion

Ce chapitre introduit le problème d'ordonnancement étudié dans cette thèse, la description de ses différents composants ainsi que les contraintes à prendre en compte pour sa résolution.

De la même façon, nous avons présenté le contexte général de l'étude pour donner une première idée sur les différents sujets à venir dans le développement de ce mémoire. Les travaux de cette thèse font partie des projets d'amélioration de l'ordonnancement de la société Norelem.

Nous avons aussi souligné l'intérêt de la société Norelem de mieux gérer le lancement de ses ordres de production, afin d'améliorer la qualité de service, réduire les retards et améliorer le flux de production de son atelier.

Dans le chapitre suivant, un état de l'art sur les sujets abordés dans cette thèse est présenté a fin de connaître des problématiques similaires qui ont déjà été étudiées dans la littérature.

Chapitre 2

État de l'art

Ce chapitre présente une étude bibliographique sur les différents sujets traités dans ce mémoire. Des références bibliographiques sur le problème d'ordonnancement qui a été introduit dans le chapitre précédent sont présentées afin de décrire les problématiques et les solutions proposées dans la littérature. Ces problèmes seront traités ensuite dans les chapitres 3 et 4. Cela concerne l'ordonnancement de tâches dans un atelier de type open shop avec la prise en compte des ressources humaines et de l'outillage pour trouver une séquence satisfaisante de la production ($Om/r_i/\sum F_i$).

L'ordonnancement est un sujet qui a inspiré une très grande quantité de travaux [23]. Les différents sujets étudiés proviennent de la variation des paramètres ou configurations du problème ; changements relatifs aux tâches, aux machines ou aux fonctions objectifs ainsi que la prise en compte des contraintes additionnelles de ressources. Ces paramètres sont souvent réunis dans une nomenclature à trois champs [72], [109], [111] comme nous l'avons exposé dans le chapitre 1.

Dans la littérature, aucun auteur n'a proposé des instances pour les problèmes traités dans cette thèse, en considérant toutes les contraintes exposées. Les méthodes de résolution employées dans ce domaine, pour les problèmes combinatoires, sont généralement classées en deux : les méthodes exactes et les méthodes approchées.

Ce chapitre débute par la présentation de généralités sur les problèmes d'ordonnancement ainsi que les méthodes qui sont utilisées pour résoudre des problèmes combinatoires proches de ceux décrits. Dans une première partie, un état de l'art concernant le problème d'open shop est présenté. Nous incluons, entre autres, un résumé sur les caractéristiques et variations proposées au moment d'étudier le problème, ainsi que sur les fonctions objectifs souvent employées. Dans une deuxième partie, nous présentons les méthodes utilisées dans la résolution des problèmes d'ordonnancement, en parlant des méthodes exactes et méthodes

approchées. Nous proposons un résumé pour chaque type de méthodes, avec des références adaptées aux problèmes d'ordonnancement et appliquées dans les ateliers de type open shop.

2.1 Problèmes d'ordonnancement

Les problèmes d'ordonnancement impliquent l'affectation de tâches qui doivent être exécutées avec une disponibilité limitée de ressources pour accomplir un objectif, en précisant l'ordre ou la séquence dans laquelle les tâches seront traitées. Actuellement, l'ordonnancement est un des thèmes les plus étudiés de la recherche opérationnelle. Un planning serré de la production réduit le temps requis pour finir les différentes tâches au même instant ce qui augmente la rentabilité d'une entreprise, aspect fondamental pour satisfaire les exigences et les contraintes existantes dans le marché actuel.

L'ordonnancement présente des concepts communs comme les tâches, les ressources, les contraintes et les critères qui sont rappelés par Yalaoui [179]. Un produit (job) est une tâche définie par un ensemble d'informations qui peuvent changer selon les conditions du problème : la date de disponibilité (à partir de quel moment la tâche est disponible pour un traitement), le temps opératoire, la date d'échéance et la date de fin. Généralement, une tâche est composée de plusieurs opérations consécutives qui peuvent suivre ou pas une séquence. Ensuite, les ressources sont des moyens de production (machines, outils, matières premières, ressources humaines, par exemple) disponibles pour réaliser les tâches. Dans ce cadre, les ressources peuvent être renouvelables ou non renouvelables (consommables) [41].

Nous considérons qu'une ressource est renouvelable si elle devient à nouveau disponible après son utilisation. De l'autre côté, une ressource est appelée non renouvelable si elle n'est pas disponible après son utilisation, c'est-à-dire qu'elle est consommée pendant l'opération. Nous rappelons qu'une contrainte est une condition nécessaire pour exécuter les tâches. Il existe de nombreuses contraintes relatives à ce domaine comme la préemption ou bien le "splitting".

Le critère est l'objectif à optimiser (dans le chapitre 1 - champ γ). Concernant l'ordonnancement de la production, plusieurs critères peuvent être considérés : la minimisation du retard total (somme des retards de toutes les tâches), la minimisation du makespan (temps nécessaire pour finir toutes les tâches), la minimisation des dates de fin de toutes les tâches, la minimisation du temps total de séjour (somme du temps passé par chaque tâche dans le système de production). En outre, plusieurs objectifs peuvent être optimisés en même temps, ou combinés en une somme pondérée selon les demandes et les conditions du problème.

Différents types de problèmes d'ordonnancement ont été proposés et étudiés dans la littérature. Dans les différents types d'atelier de production, par rapport à la configuration de machines, les modèles les plus connus font référence à une machine unique, machines parallèles, ateliers à cheminement unique (flow shop), à cheminements multiples (job shop) ou cheminements libres (open shop).

Dans le problème d'ordonnancement à machine unique, un ensemble de tâches est réalisé par une seule machine. Toutes les tâches demandent la même machine, c'est-à-dire qu'elles ne sont composées que d'une seule opération (figure 2.1 (a)). Plusieurs auteurs ont travaillé sur ce sujet, la minimisation de la date de fin pondérée d'un problème d'ordonnancement avec des dates de disponibilité de tâches a été étudié par Goemans *et al.* [68], la minimisation des pénalités du retard sur une machine en utilisant des métaheuristiques par Feldmann *et al.* [63] ainsi que la minimisation du makespan d'un problème à une machine avec temps de montage a été proposé par Koulamas *et al.* [101]. Ce problème d'apparence académique peut avoir des applications dans la réalité lorsqu'une machine limite le système.

Par rapport au problème d'ordonnancement d'ateliers à machines parallèles, nous disposons d'un ensemble de machines pour exécuter les tâches. Ces dernières sont composées d'une seule opération et une tâche exige une seule machine. Ce type d'ordonnancement se fait en deux phases : la première qui consiste à affecter les tâches aux machines et la deuxième à établir la séquence d'exécution sur chaque machine (figure 2.1 (b)). Les travaux proposés par Gabrel [65] présentent des heuristiques pour résoudre un problème de machines parallèles identiques où la préemption de tâches est interdite. Ouazene et al. [145] considèrent un problème d'équilibrage de charge en même temps que l'ordonnancement sur des machines parallèles et aussi d'autres développements sont présentés par Jouglet et Savourey [89] pour la minimisation du retard pondéré avec machines parallèles.

L'atelier à cheminement unique ou flow-shop est un processus de production appelé "linéaire", les étapes de transformation sont identiques pour tous les produits. Donc le flux de la production suit le même chemin et l'objectif est de trouver une séquence de tâches qui respecte les différentes contraintes et minimise le temps total de production (figure 2.2 (a)). Des problèmes de ce type ont été traités par Johnson [88] pour minimiser le makespan, Gupta [77] a présenté un flow shop hybride où au moins deux équipements sont disponibles pour faire une opération ; de la même façon un ensemble d'heuristiques a été étudié pour minimiser les retards dans un flow shop hybride à deux étages par Choi *et al.* [40].

Nous pouvons également trouver les ateliers à cheminements multiples appelés job shop où les produits sont traités de façons différentes, et plusieurs types de machines sont demandés dans des séquences variées. Dans cette configuration, chaque produit (ou tâche) suit

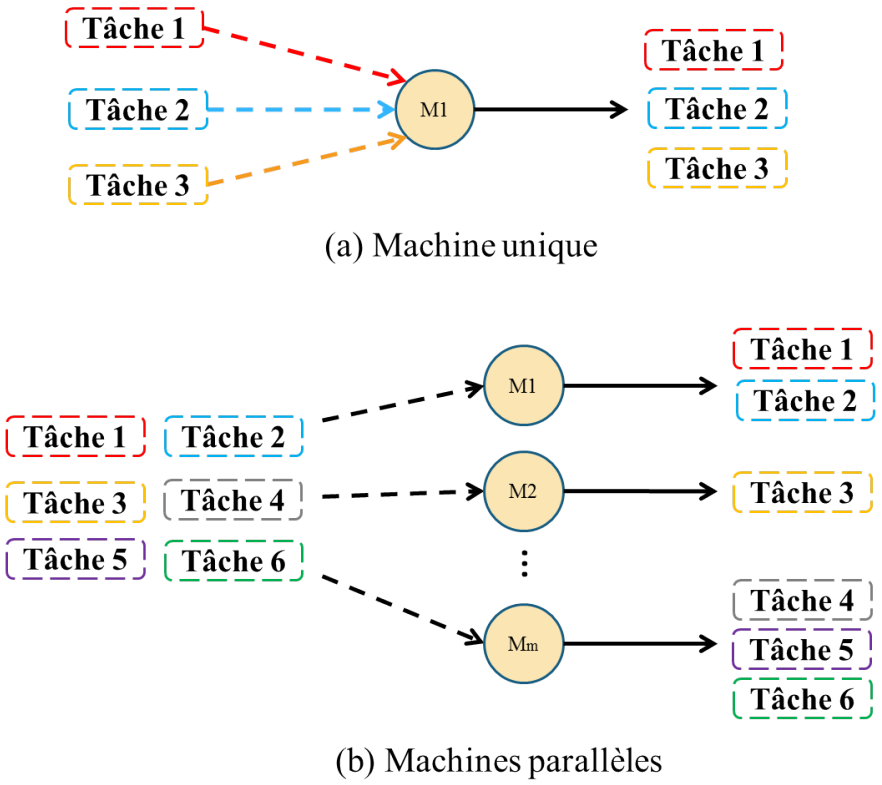


FIGURE 2.1 – Diagramme d'une configuration à machine unique (a) et à machines parallèles (b)

un chemin unique en passant par les différentes machines, l'objectif est alors de trouver une séquence de tâches sur les machines afin de minimiser le temps total de production (figure 2.2 (b)). Le problème de job shop a été fortement étudié, Blazewicz *et al.* [15] présentent des méthodes pour minimiser le makespan, des algorithmes évolutifs ont été proposés pour résoudre ce type de configurations dans [131], une formulation mathématique pour représenter un job shop avec des temps opératoires incertains a été exposé par Shafia *et al.* [163].

Enfin, on peut mentionner les ateliers à chemin libre encore désignés par open shop, lequel sera expliqué en détail dans la section suivante, car c'est notre configuration d'étude comme nous l'avons précisé dans le chapitre 1.

2.2 Ordonnancement de type open shop

Un problème de type open shop diffère d'un job shop et d'un flow shop car il n'y a pas de relation de précédence entre les opérations, il existe alors un espace de solution plus large car les solutions faisables sont plus nombreuses. On peut avoir comme exemple des sociétés spécialisées dans des essais où les unités ne sont pas réalisées dans un ordre pré-déterminé,

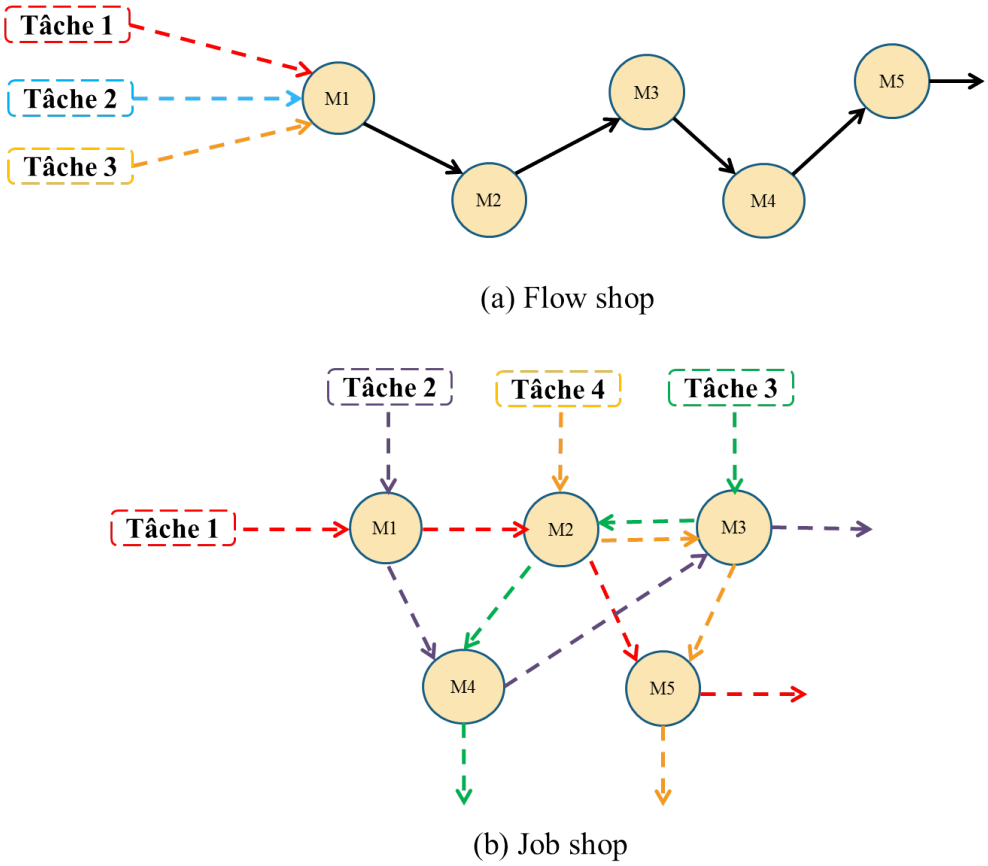


FIGURE 2.2 – Diagramme d'un flow shop et d'un job shop

mais aussi plusieurs situations de maintenance où la séquence pour exécuter les opérations n'est pas importante.

Le problème d'ordonnancement d'atelier de type open shop est un sujet qui a été étudié par plusieurs auteurs comme Kyparisis et Koulamas [105], Liaw *et al.* [118], Liaw [116], Blazewicz *et al.* [16], Naderi *et al.* [137] qui ont fait référence aux différents objectifs à optimiser, le nombre de machines impliquées dans le problème, et quelques cas particuliers qui peuvent être trouvés. Anand *et al.* [7] ont présenté une revue de littérature sur les différentes configurations étudiées concernant les problèmes de type open shop.

Dans ce cadre, le niveau de complexité du problème est aussi un point important qui peut varier selon la configuration des paramètres de la problématique. En 1976, Gonzalez *et al.* [70] ont expliqué qu'un problème d'open shop où la préemption est possible peut être résolu dans un temps polynomial pour un nombre arbitraire de machines. Kubiak *et al.* [103] ont étudié la complexité des problèmes de type open shop résolus jusqu'en 1990. De la même façon Giaro [67] a démontré la complexité NP-difficile d'un problème d'ordonnancement compact dans des ateliers de type open shop et flow shop simplifiés. De la même manière,

Brucker *et al.* [26] ont exposé des résultats sur la complexité d'un problème d'open shop avec retard de transport. Roemer [156] a présenté aussi une note sur la complexité d'un problème d'open shop, où les opérations d'une tâche peuvent être réalisées en parallèle (sur machines spécifiques). Il est prouvé que même le problème, plus simple, pour deux machines où l'objectif est de minimiser la somme de dates de fin des tâches ($\sum C_i$), est un problème NP-difficile. De plus, les généralisations de la littérature montrent que les problèmes d'open shop avec deux machines avec préemption interdite sont aussi des problèmes NP-difficiles.

Nous proposons maintenant une revue de la littérature associéé à l'ordonnancement d'open shop en classant les travaux par objectif à minimiser. Nous commençons par la minimisation du makespan (C_{max}), qui est la valeur maximale des dates de fin de toutes les tâches ; Kubale et Nadolski [102] ont considéré l'étude de la complexité d'un open shop cyclique et aussi d'un open shop de structure compacte cyclique. Akker *et al.* [174] ont étudié un open shop à deux machines, où il existe une solution faisable pour deux valeurs données (D1 et D2) où la première ou deuxième machine exécute les tâches pendant les intervalles [0,D1] et [0,D2]. Brasel *et al.* [20] ont décrit les séquences qui ne sont plus réductibles dans un open shop en montrant qu'il y a au moins une solution optimale dans un ensemble de séquences de ce type. Un modèle mathématique proposé par Masuda *et al.* [129] montre qu'il existe une solution optimale quand deux objectifs (minimiser le makespan et le retard maximal) sont pris en compte pour un open shop à deux machines. Schuurman et Woeginger [159] ont traité un open shop hybride où chaque étage possède des machines parallèles identiques avec la minimisation du makespan. Kis *et al.* [93] ont étudié un problème d'open shop avec préemption des tâches et plusieurs machines, l'ensemble de machines est divisé en groupes afin d'avoir une solution polynomiale.

Un deuxième objectif très étudié est la minimisation de la somme de dates de fin de tâches ($\sum C_i$). Tautenhahn [170] a développé un algorithme avec un temps calcul polynomial pour résoudre un open shop avec des temps opératoires unitaires et des dates d'échéance imposées pour chaque tâche. Tang et Bai [169] ont travaillé sur une heuristique (SPTB - Shortest Process Time Block) pour minimiser une situation où le nombre de tâches est un multiple du nombre de machines. Brasel *et al.* [19] ont décrit un algorithme qui divise le système en sous-problèmes pour minimiser la somme de dates de fin d'un problème open shop avec un nombre arbitraire de machines, temps opératoires unitaires et des précédences out-tree entre les tâches ($O|t_{ij} = 1, Outtree|\sum C_i$).

Pour continuer l'énumération des objectifs, nous mentionnons la minimisation du temps total de séjour ($F_i = \sum C_i$) qui fait référence au temps passé par la tâche dans le système de production depuis sa date de disponibilité jusqu'à la date de fin de sa dernière opération.

Achugbue et Chin [2] ont démontré que minimiser le temps total de séjour d'un problème open shop à deux machines sans préemption est NP-complet. Brasel *et al.* [18] ont proposé différentes heuristiques constructives pour générer des ordonnancements actifs et non-actifs afin de minimiser le temps de séjour. De la même manière, un recuit simulé et un algorithme génétique ont été proposés par [8] où la date de disponibilité des tâches dépend de la séquence de production de la période souhaitée.

La minimisation de la somme pondérée des dates de fin de toutes les tâches ($\sum w_i C_i$), où chacune de ces dernières possèdent un niveau d'importance ou priorité (w_i) pour être réalisée, a été mentionné par [153] dans lequel Queyranne *et al.* ont développé la formulation d'un modèle de programmation linéaire avec des indices des intervalles pour résoudre un open shop général avec préemption. Doulabi *et al.* [55] ont proposé un modèle de programmation mixte en nombres entiers pour minimiser le critère $\sum w_i C_i$ ainsi que le coût intermédiaire de stockage dans un problème d'open shop. Ensuite, nous avons des auteurs qui ont abordé la minimisation du retard total des tâches ($\sum T_i$). Liu *et al.* [120] ont étudié un open shop avec des temps opératoires identiques (temps unitaires), ils ont développé deux algorithmes pour minimiser le retard total et le nombre des tâches en retard. Naderi *et al.* [138] ont exposé quatre modèles de programmation linéaire mixte en nombres entiers. Ils ont proposé une recherche de voisinage variable avec deux structures de recherche basées sur l'insertion de voisins.

Dans la littérature, nous trouvons aussi l'étude de la minimisation de la somme pondérée des retards ($\sum w_i T_i$) par des auteurs comme Hossein *et al.* [54] qui ont proposé un modèle de programmation linéaire en nombres entiers pour traiter un open shop. Dans l'article [16], Blazewicz *et al.* ont exposé un autre modèle linéaire pour minimiser le retard total pondéré dans un problème d'open shop avec préemption. Andresen *et al.* [9] présentent l'application d'un recuit simulé dans un problème d'open shop pour minimiser le retard pondéré en prenant en compte des dates de disponibilité des tâches. La minimisation de la somme pondérée du nombre de tâches en retard ($\sum w_i U_i$) a été traité dans un problème de type open shop avec temps opératoires 0-1 et une date d'échéance commune *"d"* [141], un algorithme approché a été proposé pour le résoudre mais ce type de problème n'est pas très commun dans la réalité.

De la même manière, différentes approches du problème ont été abordées entre les années 1990 et 2015 : un problème à deux machines où la préemption est permise a été étudié par Liaw [115], d'autres travaux avec deux machines avec prise en compte du retard sont expliqués par Munier *et al.* [134], Kononov *et al.* [98] cherchent à minimiser le makespan d'un open shop à deux machines avec un algorithme de temps linéaire. Rebaine *et al.* [155] considèrent la minimisation du makespan d'un open shop sans préemption de n tâches à deux machines avec

des temps de transport. Un autre problème de minimisation du makespan d'un open shop de n tâches et deux machines avec du retard entre la finalisation d'une opération et le début d'une autre a été étudié dans [154]. Des problèmes de type open shop à deux machines avec deux critères à optimiser ont été exposés par Gupta et Werner [76], et Kyparasis et Koulamas [106]. Le problème à deux machines avec une contrainte de disponibilité de machines, c'est-à-dire que les machines ne sont pas toujours disponibles pour exécuter une tâche, a été examiné par Briet *et al.* [21], [122] et [22]. Kubzin et Strusevich [104] ont abordé un problème d'open shop à deux machines pour minimiser la somme des dates de fin où chaque machine doit passer par une action de maintenance pendant la période de production. Des problèmes d'open shop en considérant trois machines où l'objectif est de minimiser le makespan ont été décrits par [35] et [57]. Panneerselvam [148] a développé une heuristique pour minimiser le makespan d'un open shop à n tâches et à m machines. Murugesan *et al.* [135] ont créé une heuristique pour n tâches et m machines dans un problème d'open shop afin d'identifier le rang minimal d'une séquence optimale.

Des approches avec des temps opératoires contrôlables ont été étudiées par [38]. Goharch *et al.* [69] ont proposé une approche sur une simulation d'optimisation qui considère des temps opératoires aléatoires. Des problèmes avec des machines parallèles qui ont été fortement étudiés précédemment dans les ateliers de type flow et job shop, sont maintenant aussi liés avec l'open shop par [162], [137], [37].

La minimisation du makespan dans un problème d'open shop avec temps opératoires stochastiques a été étudiée par Alcaide *et al.* [6] qui ont développé une heuristique qui résout ce problème en considérant des pannes aléatoires pour les machines ; Gupta *et al.* [75] ont conçu une autre heuristique pour n tâches quand les délais de transport sont donnés.

A partir de là, selon les objectifs à optimiser et les approches abordées, différentes méthodes de résolution ont été proposées au cours des dernières années pour traiter des problèmes de type open shop. Nous pouvons trouver des méthodes exactes : un algorithme de branch & bound basé sur une formulation de graphe disjonctif qui minimise le makespan d'un open shop sans préemption décrit par Brucker *et al.* en 1997 [24] ; une autre technique qui améliore la méthode de branch & bound proposée par Brucker *et al.* [25] fait les parcours des solutions dans les branches avec une procédure de retour en arrière intelligent dans [73]. De plus, un algorithme de branch & bound amélioré pour le problème d'open shop avec préemption est présenté par Liaw en 2013 [117] afin de minimiser le temps total de fin effective des tâches. Dror [58] a travaillé sur un algorithme optimal pour la minimisation du makespan et le temps total de séjour d'un problème d'open shop avec temps opératoires qui dépendent de la machine ; au sujet du makespan, l'algorithme a une complexité de $O(mn), n \geq m$, qui

devient NP-difficile si $n < m$ mais ≥ 3, alors qu'en termes de temps total de séjour pour résoudre un problème à deux machines, la complexité est de $O(n)$. Cheng et Shakhlevich [38] ont étudié un problème d'open shop à deux machines où les temps opératoires sont contrôlables, impliquant certains coûts, ils ont alors développé un algorithme avec une complexité $O(nlog(n))$ qui fournit un ordonnancement avec un makespan optimal. Cependant, malgré la qualité de ces méthodes, elles ont des limites en termes de taille de problèmes à cause du temps de calcul. C'est pour cette raison que plusieurs heuristiques ont été développées pour fournir des solutions de bonne qualité dans un temps de calcul raisonnable : un algorithme génétique par Senthilkumar et Shahabudeen [161], une recherche tabou par Liaw [112], Low et Yeh [126], une version améliorée de la recherche tabou dans l'article [114], l'optimisation par essaims d'abeilles [82] a été utilisée dans la résolution de ce type de problème d'affectation et d'ordonnancement des tâches pour minimiser le retard total, la date de fin effective des tâches et le makespan dans un problème d'ordonnancement de type open shop. Aguirre-Solis [4] a utilisé une recherche tabou qui génère une solution initiale faisable à partir d'une règle de dispatching pour minimiser le makespan d'un open shop avec $"n"$ tâches, $"m"$ machines et temps de montage dépendants. Un algorithme de recuit simulé est proposé dans [157] pour minimiser le makespan d'un problème d'open shop sans préemption, où les temps de montage de tâches dépendent des prédécesseurs sur chaque machine. Zhang et Velde [186] ont présenté un algorithme glouton pour minimiser le makespan d'un open shop à deux machines, ainsi que Liaw [119] qui a construit un autre algorithme itératif pour un cas à m machines où la préemption est interdite. Une programmation optimale par contraintes a été développée par Malapre *et al.* [128] afin de minimiser le makespan dans un open shop.

Dans ce cadre, des méthodes hybridées ont aussi occupé une place importante dans l'étude d'un problème d'ordonnancement de type open shop dans la première décennie du $XXI^{ème}$ siècle : un algorithme génétique hybride qui utilise les opérateurs génétiques et intègre une procédure d'amélioration locale basée sur la recherche tabou a été proposé par Liaw [113] avec pour objectif la minimisation du makespan en prenant en compte les cas sans préemption de tâches ; Lui et Ong [121] proposent une meta-heuristique pour résoudre un problème de job, open et mixte (combinaison des deux) shop en utilisant le recuit simulé, le seuil d'acceptation et la recherche tabou. Kokosinski *et al.* [94] ont décrit un algorithme génétique hybride séquentiel et parallèle pour réduire le makespan dans un problème d'open shop de m machines et n tâches. Modarres *et al.* [132] ont présenté un algorithme hybride qui comprend une optimisation par colonie de fourmis, un beam search et une recherche locale pour résoudre un open shop cyclique généralisé. Ahmadizar *et al.* [5] ont présenté un autre algorithme génétique hybride pour la minimisation du makespan où un opérateur de croisement et une stratégie sont appliqués pour éviter des solutions redondantes au moment

de faire la mutation. Ensuite, en sachant que le facteur principal d'un algorithme à colonie de fourmis est le mécanisme de construction des solutions d'une façon probabiliste (méthode de recherche en arbre), une méthode hybride entre la beam search et un algorithme à colonie de fourmis a été développé dans [17]. Panahi *et al.* [147] ont exposé une méthode hybride qui cherche à minimiser deux objectifs : le makespan et le retard total en utilisant le recuit similé et une colonie de fourmis pour résoudre le problème.

De plus durant cette thèse, ce travail de recherche s'est concentré sur un problème qui prête une attention particulière aux contraintes de ressources au moment de faire l'ordonnancement des tâches. Cette étude nous a conduit à considérer les problèmes d'ordonnancement avec des contraintes de ressources ou "Resource Constraints Scheduling Problem (RCSP)" en anglais, où l'objectif est de trouver en même temps la meilleure planification possible avec la meilleure utilisation de ressources, bien sûr en considérant sa disponibilité et son type. Ce type de problèmes a été traité par des auteurs comme Coelho *et al.* en 2011 [41] qui parlent de deux types de ressources, lesquelles sont nommées par rapport à leur capacité d'être réutilisées ; elles sont décrites comme des ressources renouvelables (outils, main d'œuvre, etc.) et non-renouvelables (combustible, argent, etc.) dans la résolution d'un projet d'ordonnancement pour minimiser le makespan. Dauzère-Pérès *et al.* [45] ont aussi traité un problème d'ordonnancement avec ressources flexibles pour minimiser la date de fin maximum.

En considérant les ressources dans le problème d'open shop, plusieurs références qui traitent les problèmes d'ordonnancement avec cette contrainte peuvent être trouvées dans la littérature. En 1979, Slowinski [164] a étudié un problème d'open shop avec tâches indépendantes avec des dates de disponibilité et dates d'échéances prévues sous contraintes de ressources (renouvelables et non-renouvelables) pour minimiser le makespan. Également, dans [47] la disponibilité d'une ressource unique non-renouvelable pour construire un ordonnancement a été développé afin de minimiser la date de fin effective des tâches. L'approche suivante a été proposée par De Werra *et al.* [46] qui ont considéré qu'une seule unité d'une ressource est disponible pour exécuter une opération, et ont également étudié des cas avec des ressources renouvelables et non-renouvelables pour obtenir la date totale de fin effective minimale. Jurisch et Kubiak [90] ont développé un algorithme pour résoudre un open shop à deux machines avec une seule ressource renouvelable et sans préemption. Un autre travail [97] fourni une version avec logique floue du modèle présenté précédemment par De Werra *et al.* où deux objectifs sont optimisés : ce premier est le niveau de satisfaction minimal en relation avec les intervalles pour réaliser les tâches et le deuxième est la quantité de ressources employée dans les intervalles de production.

D'autres travaux prennent aussi en compte les ressources, comme Nonobe *et al.* [142]

qui proposent des algorithmes classiques tel que la recherche tabou pour obtenir une solution faisable ou un algorithme hybride présenté dans [125], ainsi que [78] qui étude des ressources plus spécifiques en parlant des ressources humaines (opérateurs), de la même façon qu'Agnetis *et al.* [3] qui exposent un problème de job shop avec deux ressources différentes : les machines et les opérateurs. Dans ce cadre, il est très important de mentionner, que dans notre situation, chaque opérateur peut maîtriser différents types de compétences qui définissent les capacités pour réaliser une opération. A partir de cette notion, notre problème implique en même temps l'affectation de personnel avec multi-compétences, problème déjà été étudié dans un projet d'ordonnancement par Kazemipoor *et al.* [91] où le terme de "membres de personnel (staff members)" a été employé pour décrire les ressources qui ont été classées comme renouvelables, de grande valeur et qui sont discrètes avec de multiples compétences.

En continuant, nous avons trouvé l'inclusion des dates de disponibilité de tâches dans la résolution d'un open shop dans le travail de Chen *et al.* [36] et dans la minimisation du makespan par Bai *et al.* [11], en sachant que la date de disponibilité de tâches est un paramètre qui détermine si la tâche est prête à être exécutée. Cho et Sahni [39] ont aussi étudié des problèmes de flow, job et open shop avec tâches indépendantes et des dates de disponibilité et d'échéance. Lu et Posner [127] ont examiné la complexité d'un problème d'open shop à deux machines avec dates de disponibilité différentes,et qui cherche à minimiser le makespan. Kononov *et al.* [99] ont expliqué qu'un open shop de m machines avec un schéma d'approximation du temps polynomial où les dates de disponibilités sont données et la préemption n'est pas possible. Sedeno-Noda *et al.* [160] considèrent la minimisation du makespan d'un open shop avec préemption avec fenêtres de temps, où, pour chaque tâche, il y a une fenêtre de temps qui est définie par ses dates de disponibilité et d'échéance.

Enfin, nous avons présenté un résumé sur les différents aspects qui ont été étudiés sur les problèmes d'ordonnancement de type open shop dans la littérature jusqu'à ce jour. Plusieurs objectifs à optimiser ont été étudiés et développés : minimisation du makespan (C_{max}), minimisation de la somme de dates de fin de tâches ($\sum C_i$), la minimisation du temps total de séjour ($F_i = \sum C_i$), la minimisation de la somme pondérée des dates de fin de toutes le tâches ($\sum w_i C_i$), des retards ($\sum w_i T_i$) et du nombre de tâches en retard ($\sum w_i U_i$). Nous avons constaté que selon l'objectif à prendre en compte, différentes méthodes de résolution ont été proposées. Dans les méthodes exactes, les algorithmes de branch & bound ont montré des solutions de bonne qualité mais ils sont limités en termes de taille de problèmes (temps de calcul). Des méthodes approchées ont été aussi développées : algorithmes génétiques, recherches tabou, algorithmes de recuit simulé, colonie de fourmis, optimisation par essaims d'abeilles et algorithmes hybridées, afin d'obtenir des solutions de bonne qualité dans un

temps de calcul raisonnable. Également, cet état de l'art a montré l'intervention des ressources dans les problèmes d'ordonnancement de type open shop en divisant les ressources renouvelables de celles non-renouvelables, ainsi que la présentation des ressources humaines dans des problèmes d'affectation de personnel avec multi-compétences. Les dates de disponibilité des tâches ont aussi été étudiées mais malgré toutes les publications qui traitent le problème, il y a un nombre réduit d'heuristiques publiées qui s'occupent du cas général à m-machines.

Dans cette section, nous avons donné un point de départ pour le type de problème qui sera expliqué plus en détail dans les chapitres suivants.

2.3 Les méthodes de résolution exactes

Les méthodes exactes sont multiples et variées, elles sont proposées pour résoudre les problèmes combinatoires de manière optimale où son efficacité dépend en grande partie de la structure du problème. Ces méthodes sont souvent lentes et ne s'occupent que des problèmes dont l'espace de recherche est petit. Cependant, ces méthodes sont intéressantes dans le sens où un résultat sur un problème de petit taille peut donner des tendances dans les approches de résolution pour un problème de grande taille.

Les méthodes exactes sont multiples et variées telles que : la programmation mathématique et les méthodes par séparation et évaluation, etc.

2.3.1 La programmation mathématique

La programmation mathématique s'occupe de l'étude théorique des problèmes d'optimisation ainsi que de la conception et mise en œuvre des algorithmes de résolution. Dans cette catégorie, nous pouvons trouver la programmation linéaire/non linéaire, les méthodes de décomposition, la programmation en nombres entiers ou la programmation dynamique. Cette dernière part est un modèle à dimension N et est ensuite décomposé en un ensemble de modèles de plus petites dimensions. Le problème comporte N étapes qui sont résolues de manière séquentielle. Cette programmation est conditionnée par une équation récursive permettant de décrire la valeur optimale du critère (à un niveau donné du système), en fonction de la valeur du niveau antérieur.

En ce qui concerne la programmation linéaire, elle est composée des variables de décision qui sont clairement définies, d'une fonction objectif associée à un coût ou profit lié avec les variables de décision qui ont un sens d'optimisation : soit la minimisation, soit la

maximisation. Des contraintes qui limitent et représentent la structure du problème. A partir de ces concepts, nous disons qu'une solution est réalisable si elle respecte les conditions exprimées par les contraintes et qui a un coût (profit) calculé avec la fonction objectif. Différents types de programmation linéaire peuvent être employés selon les variables de décision à utiliser : programmation binaire (variables binaires), programmation en nombres entiers (variables entières) ou programmation mixte (variables entières et réelles ou MIP "Mixed Integer Problem").

La résolution d'un modèle de programmation linéaire se fait en utilisant différentes méthodes, la plus connue est l'algorithme simplex, lequel a été utilisé dans de nombreux domaines. Il y a aussi des solveurs dédiés comme CPLEX, XPRESS, GAMS, EXCEL pour les modèles linéaires et LINGO pour les non-linéaires.

Comme cela est mentionné par la plupart des auteurs, les méthodes de programmation mathématique se limitent à résoudre des instances de petites tailles à cause de ses temps de calcul.

Dans la littérature, nous trouvons que plusieurs des auteurs ont proposé des formulations mathématiques pour représenter un problème de type open shop : Masuda et Ishii [129], Kis *et al.* [93], Kyparisis et Koulamas [101], Hossein *et al.* [54], Naderi *et al.* [137], [138], entre autres.

2.3.2 Les méthodes de séparation et évaluation

Cette méthode aussi appelée Branch & Bound (B&B) consiste à décomposer le problème en sous-problèmes de plus en plus petits et faire une exploration intelligente de l'espace de solutions.

Deux composantes déterminent deux procédures indispensables dans ce type de méthodes. La première, séparation, permet d'énumérer toutes les solutions possibles, mais en même temps, elle doit faciliter l'identification de solutions partielles, afin d'appliquer la procédure d'évaluation. De cette façon, en identifiant la meilleure solution trouvée par sous-problèmes, il est possible de garantir l'optimalité de la solution trouvée pour le problème initial. Les différentes solutions et sous-problèmes associés forment une hiérarchie naturelle en forme d'arbre, souvent appelée arbre de décision. La procédure d'évaluation réduit le nombre de solutions explorées en permettant de prouver mathématiquement que l'ensemble des solutions réalisables associées à un des nœuds de l'arbre ne contient pas une solution intéressante pour le problème initial. Pour cela, la méthode la plus générale consiste à déterminer une borne inférieure ou une propriété de dominance pour confirmer qu'une branche

ne contient pas de solution optimale.

Des méthodes du type Branch & Bound ont été proposées dans la résolution de problèmes d'ordonnancement : Brucker *et al.* [25] ont montré un algorithme rapide de Branch & Bound pour résoudre un problème job shop ; un B&B proposé par Szwarc *et al.* [168] pour minimiser le retard total dans un problème avec une seule machine ; une méthode exacte à partir de la méthode Branch & Bound a été proposé par Yalaoui *et al.* [180] pour résoudre un problème de machines parallèles avec dates de disponibilité de tâches pour minimiser la date de fin totale de tâches. Artigues *et al.* [10] ont présenté un Branch & Bound itératif pour minimiser le makespan d'un problème de job shop avec des temps de montage multiples.

Concernant les problèmes de type open shop, nous rappelons les travaux proposés par Brucker *et al.* [24] qui ont développé une méthode exacte basée sur un B&B qui se base sur un graphe disjonctif afin de minimiser le makespan. Guéret et Prins [74] ont considéré la minimisation du makespan d'un open shop en proposant une borne améliorée pour un B&B qui est définie comme le makespan optimal du problème OS_k relâché (open shop où les opérations d'une tâche doivent être simultanées sauf pour une tâche k sélectionnée). Des techniques pour analyser les solutions partielles des branches dans une méthode de B&B ont été proposées par Guéret *et al.* [73] pour un open shop. De la même façon, Liaw [117] a exposé un algorithme B&B pour minimiser la date totale de fin d'un open shop avec préemption des tâches.

2.4 Les méthodes approchées

Cette section comporte une présentation de quelques méthodes approchées. Celles-ci sont connues sous le nom d'heuristiques ou de métaheuristiques. Ces méthodes sont utilisées pour résoudre des problèmes de grandes tailles qui ne peuvent pas l'être de manière exacte en des temps de calcul acceptables en sachant néanmoins qu'elles ne garantissent pas l'optimalité de la solution trouvée.

Les heuristiques sont dédiées à des problématiques spécifiques alors que les métaheuristiques sont plus génériques.

Les méthodes approchées étant efficaces sur des instances de grandes tailles, elles peuvent être implémentées dans des utilisations industrielles réelles.

Il existe de nombreuses métaheuristiques et plusieurs domaines d'application : ordonnancement, transport, problèmes d'agencement, de conception de lignes de production, des chaînes logistiques, entre autres. La suite de cette section présente quelques méthodes ap-

prochées connues pour leurs aptitudes à résoudre des problèmes NP-difficiles.

2.4.1 Les algorithmes génétiques

Les algorithmes génétiques (AG) ont été largement utilisés comme une approche fonctionnelle pour résoudre différents problèmes d'optimisation. Le concept d'algorithme génétique a été introduit par Holland *et al.* [80] comme une technique qui imite le processus de l'évolution biologique des espèces pour résoudre des problèmes combinatoires.

L'AG commence avec un ensemble de solutions qui est appelé population où chaque solution (individu) est codée selon quelques critères afin de générer un chromosome qui est lié avec une mesure d'évaluation (fitness). Suivant le développement naturel des populations, les différents individus sont croisés (combinaison des éléments des chromosomes), mutés (perturbations aléatoires sur le chromosome) et sélectionnés afin d'améliorer la population au cours des générations en gardant les meilleurs individus et la même taille de la population.

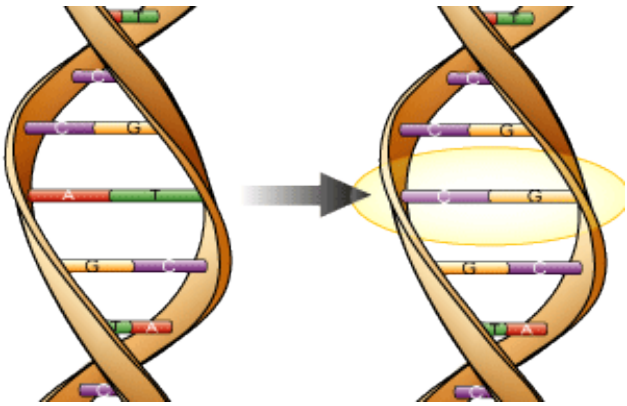


FIGURE 2.3 – Brins d'ADN [60]

Cette méthode approchée a été utilisée pour résoudre différents types de problèmes d'ordonnancement : par exemple, nous pouvons citer Jawahar *et al.*[86] qui ont décrit un AG pour générer une combinaison optimale de règles de priorité de lancement de production pour obtenir la planification d'un système d'atelier flexible. Tsai *et al.* [171] ont présenté un AG qui combine un algorithme génétique classique avec une méthode Taguchi qui sélectionne les meilleurs gènes pour réaliser le croisement. Jia *et al.*[87] ont exposé un algorithme génétique basé sur les différents modes de faire le codage des opérations pour résoudre un problème de job shop. Demir *et al.* [50] ont proposé un algorithme génétique efficace pour minimiser le makespan d'un job shop flexible.

Les problèmes de type open shop ont été résolus par des algorithmes génétiques : un AG pour résoudre un open shop avec re-ordonnancement a été proposé dans [123]. Khuri *et al.* [92]

ont fait la comparaison de trois AG's qui minimisent le makespan : un AG de permutation, un AG hybride et un auto-gène algorithme. Prins [151] a proposé plusieurs algorithmes génétiques compétitifs pour minimiser le makespan en considérant l'importance d'une bonne solution initiale pour garantir la qualité des solutions finales. Andresen *et al.* [8] ont adapté d'autres AG pour minimiser le temps total de séjour, en utilisant un graphe disjonctif comme un outil efficace pour construire le schéma de codage des différents individus. Cette technique utilise un graphe de niveau des opérations qui explore les différentes solutions en évitant la redondance. Matta [130] a proposé un AG pour résoudre un problème d'open shop avec plusieurs machines, qui est comparé à deux modèles de programmation linéaire en nombres entiers afin d'évaluer sa performance. Doulabi *et al.* [55] a présenté un AG pour minimiser la somme des dates de fin pondérées dans un open shop. Un autre AG qui minimise le retard total est exposé par Naderi *et al.* [138].

Des algorithmes génétiques hybrides ont été développés pour minimiser le makespan par Fang *et al.* [62] ; Zobolas *et al.* [191] ont combiné un AG avec une recherche de voisinages variables pour améliorer la solution, en générant la population initiale à partir de l'heuristique NEH (Nawaz-Enscore-Ham).

2.4.2 Les colonies de fourmis

La colonie de fourmis (ACO - *Ant Colony Optimization*) est une métaheuristique qui a été proposée dans les années 1990 par Dorigo [52], et Dorigo *et al.* [53], basée sur le comportement réel des fourmis. Cet algorithme est un modèle probabiliste qui inclut certains paramètres pour construire des solutions au problème étudié. Le modèle de probabilité est généralement appelé le modèle de phéromone qui est un ensemble de paramètres τ connus comme étant la trace de la phéromone (c'est la trace que laisse une fourmi chaque fois qu'elle fait le parcours entre la colonie (C) et la source d'alimentation (S), la trace de phéromone sera plus forte sur les chemins courts - figure 4.6). L'algorithme à colonie de fourmis est un processus itératif où la trace de la phéromone d'une itération sera déterminée par le comportement de la précédente. Cette étape est connue comme la mise à jour des quantité de phéromones qui cherche la génération de meilleures solutions au cours du temps, en augmentant la probabilité de visiter un bon chemin.

En d'autres termes, à chaque itération, ϕ fourmis construisent des solutions pour résoudre le problème en question. Les solutions sont générées étape par étape à partir des valeurs de paramètres de la phéromone, la désidérabilité et l'évaporation. Chaque nœud de la solution est sélectionné par une probabilité de transition σ. Les probabilités de transition sont une fonction des valeurs de la phéromone τ de l'itération et une fonction de pondération η

(information heuristique), alors le choix d'un nœud va conditionner la sélection du prochain.

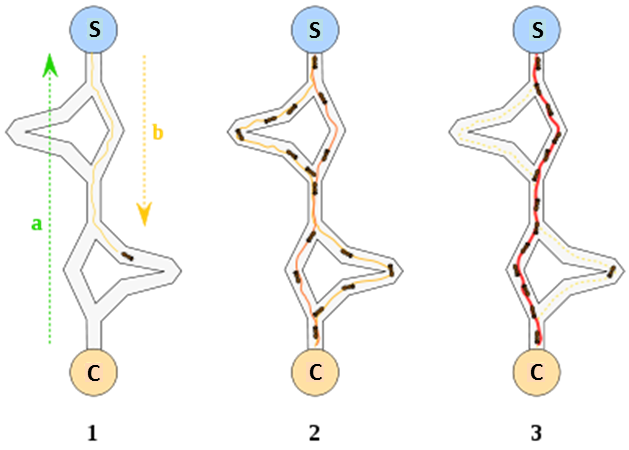


FIGURE 2.4 – Diagramme du comportement des fourmis [177]

Les algorithmes à colonie de fourmis ont eu plusieurs applications dans les problèmes d'ordonnancement ; un ACO pour minimiser le retard et le temps de séjour d'un problème job shop en utilisant des règles multi-attributs de lancement de production a été proposé par Korytkowski *et al.* [100]. Eswaramurthy *et al.* [61] ont présenté une recherche tabou hybridée avec un algorithme à colonie de fourmis pour résoudre un problème de job shop qui considère l'influence de la phéromone au moment de sélectionner les voisinages pour améliorer la solution. Un multiple algorithme à colonie de fourmis pour minimiser le makespan d'un problème de job shop a été développé par Udomsakdigool *et al.* [172]. Chang *et al.* [34] ont présenté un ACO pour obtenir la solution d'un problème de job shop multi-objectif qui prend en compte les critères quantitatifs et qualitatifs, ainsi que, Zhou *et al.* [187] ont exposé la performance d'un ACO dans un problème de job shop dynamique en testant différents environnements d'expérimentation.

De la même manière, certains auteurs ont proposé différentes applications de l'ACO dans la résolution de problèmes d'ordonnancement type open shop. Un problème d'open shop multi-objectif (minimisation du retard et du makespan) est résolu par Panahi *et al.* [147] à partir d'un algorithme à colonie de fourmis hybridé avec un recuit simulé pour obtenir des solutions de grande qualité. En outre, Blum [17] a présenté une approche beam-ACO pour minimiser le makespan, en combinant deux méthodes pour explorer l'espace de recherche : l'ACO d'un mode probabiliste qui trouve de bonnes zones de recherche à partir des expériences passées, et la méthode beam qui est déterministe faisant intervenir une fonction de pondération.

2.4.3 Les recherches locales

La recherche locale est basée sur l'évolution d'une solution et utilisée dans l'exploration des voisinages crées par des procédures qui garantissent des recherches intensifiées. Elle autorise des mouvements particuliers qui modifient la solution et sont adaptés en fonction du problème en question.

Nous définissons un voisinage comme un sous-ensemble de configurations déduites à partir des transformations données d'une solution s. La recherche locale explore les voisinages comme une méthode de diversification afin de sortir des optimaux locaux. Un optimum local est une solution avec de très bonnes performances qui empêche la génération des nouveaux voisinages. Pour appliquer une recherche locale à une population, il faut comme il est présenté dans la figure 2.5, d'abord sélectionner les solutions à modifier (a), puis appliquer la recherche locale à chaque solution (b) et finalement, remplacer la solution de base par la meilleure solution trouvée si celle-ci améliore la solution de départ (c).

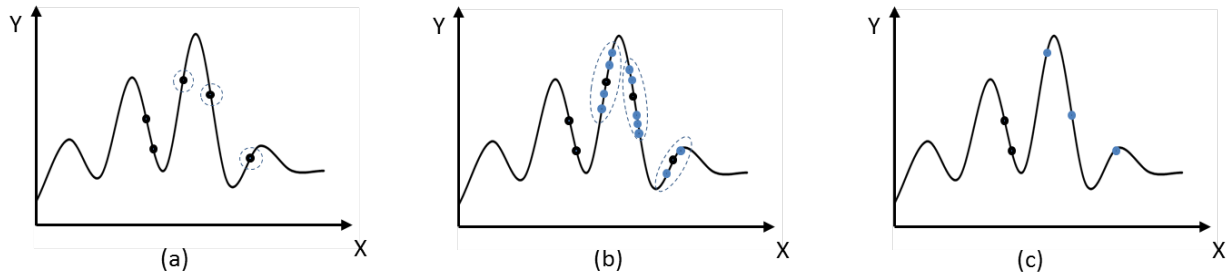


FIGURE 2.5 – Recherche locale

La recherche locale a une place importante dans la résolution des problèmes d'ordonnancement. Un problème de machines indépendantes a été traité par Persma et Dijk [150] avec un algorithme de recherche locale de voisinages efficaces. Nous pouvons trouver des algorithmes qui utilisent la recherche locale pour améliorer ses solutions, Cai *et al.* [28] ont exposé un algorithme génétique hybridé avec une recherche locale afin de minimiser le makespan d'un problème de job shop. Gonzalez-Rodriguez *et al.* [71] ont présenté une heuristique basée sur la recherche locale pour résoudre un problème d'open shop.

Plusieurs critères d'arrêt peuvent être utilisés pour une recherche locale. Par exemple, nous pouvons citer le temps CPU, le nombre d'itérations ou si une meilleure solution n'est pas améliorée après un nombre d'itérations défini.

2.5 Conclusion

Ce chapitre montre les différentes études réalisées sur les problèmes d'ordonnancement au cours des dernières années et surtout l'engouement récent de la littérature sur les problèmes de type open shop.

Nous avons repris les méthodes de résolution typiquement utilisées dans la résolution de problèmes combinatoires. D'un côté, les méthodes d'optimisation exactes qui sont très intéressantes car elles assurent l'optimalité des solutions mais sont limitées au niveau des tailles des problèmes à résoudre. Leur application aux problèmes industriels est souvent limitée.

De l'autre côté, nous trouvons les méthodes approchées comme une réponse intéressante pour obtenir des solutions de bonne qualité en un temps de calcul acceptable. En effet, il existe maintenant une pléthore de méthodes approchées pour résoudre les problèmes d'ordonnancement de type open shop qui sont inspirées par des phénomènes naturels (évolution biologique, fourmis, abeilles, entre autres).

Néanmoins, nous avons noté un manque d'étude concernant la résolution du problème d'ordonnancement de type open shop avec la prise en compte de contraintes de ressources (affectation de ressources humaines avec multi-compétences et disponibilité d'outillage). C'est pourquoi nous proposons de résoudre ce problème dans cette thèse.

Dans les chapitres suivants, des méthodes exactes et approchées utilisées pour résoudre les problèmes traités dans ce mémoire vont être présentées. L'objectif a été d'obtenir des solutions efficaces dans des temps de calcul courts, afin d'implémenter ces méthodes de résolution dans la gestion d'ordonnancement d'un atelier réel de production qui fabrique des pièces de précision mécanique.

Chapitre 3

Ordonnancement d'Open shop Mono-objectif : méthodes exactes

Dans le chapitre précédent, nous avons passé en revue les différents aspects qui ont été étudiés sur le problème d'ordonnancement de type open shop : les différentes configurations, les approches de résolution, les fonctions objectifs à optimiser ainsi que la prise en compte des ressources matérielles et humaines pour exécuter une activité ; les conditions de disponibilité des tâches et des machines dans les cas spéciaux. Tous ces éléments nous ont donné les outils nécessaires pour décrire le scénario qui correspond au problème étudié dans ce chapitre.

En premier lieu, dans ce type de problème combinatoire, toute instance ou scénario est caractérisé par le nombre de machines disponibles et la quantité de tâches à ordonnancer. En outre, il y a des informations complémentaires sur les tâches connues à l'avance : temps opératoires, temps de montage et dates de disponibilité entre autres.

Le contexte de recherche (chapitre 1) décrit un cas pratique dans lequel les critères utilisés pour gérer la fabrication de produits dans l'atelier de l'entreprise Norelem ne sont pas optimaux comme en témoigne l'existence de retards de production. Ainsi, l'objectif du problème est de minimiser ces retards et augmenter le flux de production de l'atelier.

Dans ce cadre, ce chapitre présente une description complète du problème et des méthodes exactes considérées pour sa résolution. Différents types de formulations mathématiques, linéaires et non-linéaires sont proposés. Les résultats expérimentaux sont utilisés pour évaluer le comportement de chaque formulation et faire les comparaisons pertinentes afin de déterminer la plus performante. Ces résultats seront pris en compte dans le chapitre suivant au moment de tester la performance des méthodes approchées sur le traitement des petites instances.

3.1 Introduction

Dans le monde économique actuel, la concurrence entre les différentes entreprises est chaque fois plus serrée et exigeante. Les décisions aux niveaux stratégique et tactique conditionnent les méthodes futures de production, les développements technologiques, l'expansion de marchés et d'autres activités qui prennent un temps considérable. Cependant, tous ces projets ont besoin d'un support rapide et rentable au niveau opérationnel, c'est-à-dire, une gestion efficace des ressources disponibles dans l'activité traitée, soit dans le secteur de la production, des services, du transport, du commerce, financier, etc.

Concernant les ateliers de production, l'utilisation efficace des équipements disponibles, des ressources matériels et humaines est notamment un sujet qui s'impose de plus en plus pour fixer les standards de qualité d'une entreprise industrielle. La bonne exécution des activités aura comme récompense des rétributions monétaires et la satisfaction du client.

Cette gestion de ressources demande des décisions d'ordonnancement qui consistent à définir des séquences de passage sur les machines ou, en d'autres termes, à décider quelle tâche suit quelle autre sur une machine, pour toutes les machines et toutes les tâches, en considérant des contraintes de disponibilité de ressource, d'outillages, et de main d'œuvre. Il est donc dans l'intérêt de toute société de maîtriser la gestion de la production, et ainsi de contrôler les temps et les délais relatifs au management des ressources disponibles pour réaliser les tâches de production. De ce fait, la minimisation des retards dans le système de production est une nécessité à l'égard des surcoûts qu'ils peuvent générer.

Par la suite, nous présentons une première approche de résolution pour la gestion de l'ordonnancement des ateliers de type open shop (ou chemin ouvert). Le reste de ce chapitre traite de cette problématique. Dans un premier temps, nous exposons une description complète du problème, en donnant les concepts et notations nécessaires pour la compréhension du chapitre. Ensuite, des modèles mathématiques sont décrits pour représenter le problème en optimisant l'objectif qui est mentionné dans la section suivante. Pour terminer, les résultats expérimentaux sont exposés pour évaluer et comparer les modèles mathématiques.

3.2 Présentation du problème

En reprenant les conclusions du chapitre 1, nous étudions ici un atelier de production de type open shop, sujet très intéressant mais qui selon l'état de l'art proposé dans le chapitre précédent n'a pas été assez traité notamment lorsqu'il s'agit d'impliquer des contraintes de ressources. Dans ce cadre, nous avons trouvé un cas réel de ce type de problème : les dates de

disponibilité des tâches sont différentes, les temps opératoires et de montage sont considérés de façon indépendante, des opérateurs sont qualifiés pour exécuter certaines opérations selon leurs capacités ainsi que la prise en compte des outils qui conditionnent l'exécution des opérations. De plus, des produits sont expédiés deux fois par semaine vers la maison mère. De cette façon, nous cherchons à satisfaire les besoins des clients le plus rapidement possible.

Le problème d'ordonnancement de type open shop décrit dans ce chapitre a pour objectif la minimisation du temps total de séjour et considère des dates de disponibilité de tâches différentes. Il est classé, en employant la nomenclature présenté en [72] comme $O_m|r_i|\sum F_i$. Cette problématique considère aussi des contraintes de ressources avec multi-compétences qui conditionnent l'affectation de certaines ressources humaines et la réalisation de tâches ; il y a également des contraintes d'outillage selon la demande de chaque opération.

Le problème d'open shop exposé dans ce mémoire est composé par :

- M machines ($m = 1, 2, 3, \ldots, M$).
- Un ensemble de N tâches ($i = 1, 2, 3, \ldots, N$), où chaque tâche possède au plus M opérations et la séquence de production peut avoir plusieurs chemins selon le type de produits.
- Chaque opération O_{im} doit être exécutée sur une machine avec un temps opératoire p_{im}, un temps de montage SE_{im} et en prenant en compte la disponibilité des ressources (opérateurs, compétences maîtrisées).
- La préemption n'est pas permise.
- Le temps de montage est considéré de façon indépendante et il peut commencer si la machine est disponible.
- Chaque machine ne peut traiter qu'une tâche à la fois et une tâche peut être exécutée que par une seule machine à chaque instant.
- Nous supposons que toutes les machines sont disponibles pendant la durée de l'ordonnancement et les dates de disponibilité des tâches sont déterminées par la demande du client.
- Dans ce problème d'ordonnancement, les tâches sont indépendantes : leurs paramètres ne dépendent ni des autres tâches, ni des machines.

La fonction objectif à minimiser est $\sum_{i=1}^{N} F_i = \sum_{i=1}^{N}(C_i - r_i)$, qui est connue comme le temps total de séjour, calculé à partir de la date de fin d'une tâche C_i et de sa date de disponibilité r_i. Le temps total de séjour fait référence au temps passé par une tâche dans le système de production dès sa date de disponibilité jusqu'à la fin de sa dernière opération.

En ce qui concerne les ressources, une ressource humaine (opérateur) peut être affectée seulement pour utiliser une compétence qu'elle maîtrise ; elle emploie une compétence pour

réaliser une opération dans un temps donné et tout opérateur qui a été affecté doit rester sur la machine pendant tout le temps opératoire. Les outils sont affectés selon la demande des opérations et sa disponibilité au moment de traiter une opération.

Toutes les données sont connues à l'avance et le comportement du système est stable (cas déterministe). Les notations suivantes sont utilisées pour formuler le problème :

- N_W : Nombre d'opérateurs disponibles
- N_S : Nombre de compétences existantes
- N_k : Nombre de types d'outils disponibles
- p_{im} : Temps de traitement de l'opération O_{im} (tâche i sur la machine m).
- SE_{im} : Temps de montage de l'opération O_{im} (tâche i sur la machine m). Correspond au temps requis pour faire les préparatifs pour réaliser l'opération (réglage de la machine - outils, montage d'usinage, etc.)
- r_i : Date de disponibilité de la tâche i. Aucune opération de la tâche i ne peut commencer avant cette date
- C_i : Date de fin effective de la tâche i

Les valeurs pour les paramètres N, M, N_W, N_S et N_K sont des entiers positifs. Pour les valeurs de p_{im}, SE_{im} et r_i, nous supposons aussi qu'ils prennent des valeurs entières et positives.

De cette façon, une solution complète pour une instance de la problématique décrite, identifie les N tâches à traiter en premier sur les M machines, et ensuite leurs successeurs respectifs. Les dates de début et de fin pour chaque tâche sont aussi fournies.

3.2.1 Modélisation mathématique : contraintes de ressources humaines

Comme nous l'avons mentionné antérieurement, les ressources jouent un rôle très important sur la planification de la production d'un atelier de production. Dans un premier temps, nous proposons d'aborder la modélisation mathématique du problème de type open shop en ne considérant que des contraintes qui font référence aux ressources humaines (opérateurs), c'est-à-dire que, nous gérons l'affectation des opérateurs pour exécuter une opération selon leurs compétences maitrisées. Cette partie nous amène dans un problème d'affectation de personnel avec multi-compétences, lequel sera formulé dans les deux sous-sections suivantes : un modèle non-linéaire et deux modèles linéaires sont proposés. Nous présentons un exemple graphique (figure 3.1).

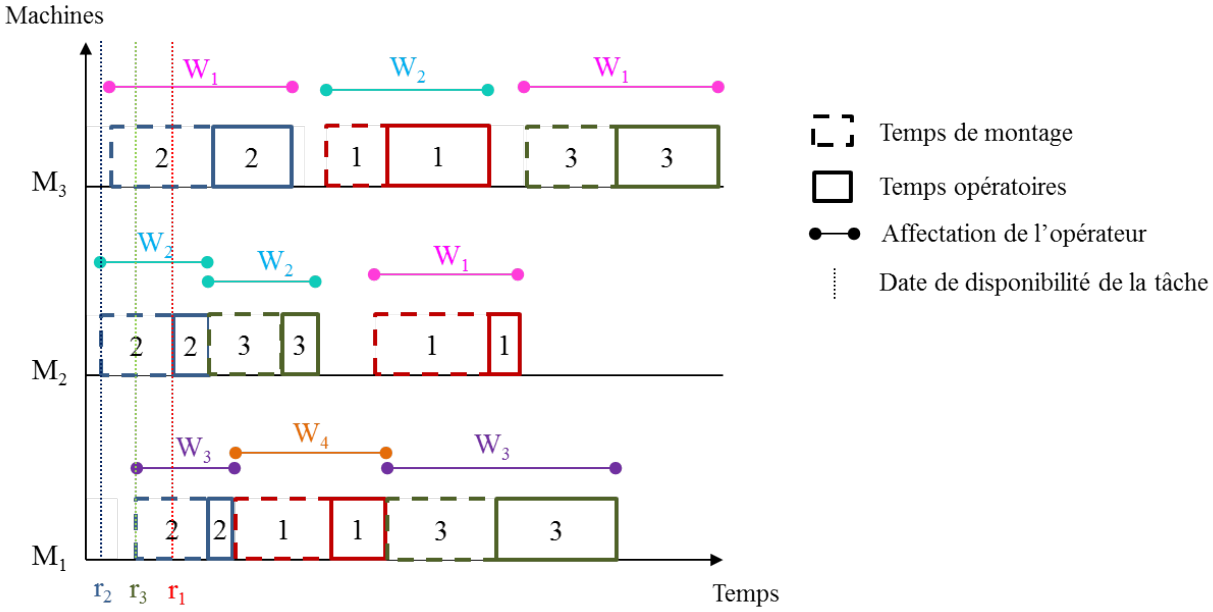


FIGURE 3.1 – Exemple d'ordonnancement sous contraintes de ressources humaines

La figure 3.1 représente l'ordonnancement de 3 tâches ($N = 3$, tâche 1 en rouge, tâche 2 en bleu et tâche 3 en vert) sur 3 machines ($M = 3$), c'est-à-dire, 9 opérations ($N \times M$), et l'affectation de 4 opérateurs ($N_w = 4$) selon leurs compétences. Les tâches ont des dates de disponibilité différentes (r_i) ainsi que des temps de montage (rectangle de ligne pointillée) et temps opératoires (rectangle de ligne continue). Dans le graphique, nous pouvons constater que l'opérateur 1 (W_1) peut exécuter la tâche 1 sur la machine 2 et les tâches 2 et 3 sur la machine 3. L'opérateur 2 (W_2) peut réaliser les tâches 2 et 3 sur la machine 2 et la tâche 1 sur la machine 3. L'opérateur 3 (W_3) exécute les tâches 2 et 3 sur la machine 1. Finalement, l'opérateur 4 (W_4) réalise la tâche 1 sur la machine 1.

Nous prenons en compte les modèles proposés par Low *et al.* [126] et Noori-Darvish [143], [144] pour nous inspirer de la construction des modèles formulés dans ce chapitre. Le premier modèle suppose les temps opératoires, de montage et de démontage d'une façon indépendante, pendant que les autres travaux résolvent un problème d'open shop à deux-objectifs avec paramètres hybridés par logique floue qui minimise le retard total et le makespan avec des temps de montage dépendants de la séquence employée.

3.2.1.1 Modélisation mathématique non-linéaire

Le modèle présenté dans cette sous-section est non-linéaire, il contient deux types de variables, donc il est considéré de type MIP (Mixed Integer Problem). Les notations, paramètres et variables de décision pour formuler ce modèle mathématique sont exposés ci-après :

- **Paramètres**

J	Ensemble des tâches $J = \{1, ..., N\}$
L	Ensemble des machines $L = \{1, ..., M\}$
W	Ensemble des opérateurs $W = \{1, ..., N_W\}$
S	Ensemble des compétences $S = \{1, ..., N_S\}$
i, j	Indices des tâches, i,j $= \{1, ..., N\}$
m, m'	Indices des machines, m,m' $= \{1, ..., M\}$
w	Indices des opérateurs, $w = \{1, 2, 3, ..., N_W\}$
s	Indices des compétences, $s = \{1, 2, 3, ..., N_S\}$
G	Grand nombre positif
SE_{im}	Temps de montage de la tâche i sur la machine m
p_{im}	Temps opératoire de la tâche i sur la machine m
r_i	Date de disponibilité de la tâche i

$$RE_{ims} = \begin{cases} 1 \text{ Si la tâche } i \text{ sur la machine } m \text{ nécessite la compétence } s \\ 0 \text{ Sinon} \end{cases}$$

$$A_{ws} = \begin{cases} 1 \text{ Si l'opérateur } w \text{ possède la compétences } s \\ 0 \text{ Sinon} \end{cases}$$

Les paramètres présentés par rapport à la liste de compétences ont été inspirés par des auteurs qui ont déjà travaillé sur les problèmes d'ordonnancement avec multi-compétences comme Montoya en 2012 [133]. Le tableau 3.1 présente les compétences requises pour chaque activité et le tableau 3.2 expose les compétences maîtrisées par chaque opérateur (la valeur 1 indique respectivement que la compétence est requise pour exécuter l'opération ou que l'opérateur la maîtrise, sinon la valeur sera 0).

- **Variables de décision**

C_{im}	Date de fin de la tâche i sur la machine m
C_i	Date de fin de la tâche $i = \underset{m \in L}{Max}\ C_{im}$

$$X_{ijm} = \begin{cases} 1 & \text{Si la tâche } i \text{ précède la tâche } j \text{ sur la machine } m \\ 0 & \text{Sinon} \end{cases}$$

$$Z_{imm'} = \begin{cases} 1 & \text{Si la tâche } i \text{ est exécutée sur la machine } m \text{ puis sur la machine } m\text{'} \\ 0 & \text{Sinon} \end{cases}$$

TABLE 3.1 – Liste RE_{ims} de compétences s demandées pour exécuter la tâche i sur la machine m

	sk_1	sk_2	...	sk_S
O_{11}	0	1	...	0
O_{12}	1	1	...	1
...				...
O_{1M}	1	0	...	0
O_{21}	0	0	...	1
O_{22}	1	1	...	0
...				...
O_{2M}	0	1	...	1
...				...
O_{NM}	0	1	...	0

TABLE 3.2 – Liste A_{ws} de compétences s maîtrisées par chaque opérateur w

	sk_1	sk_2	...	sk_S
OP_1	1	0	...	1
OP_2	1	1	...	1
OP_3	0	1	...	0
...				...
OP_W	1	0	...	1

$$Y_{wim} = \begin{cases} 1 & \text{Si l'opérateur } w \text{ exécute la tâche } i \text{ sur la machine } m \\ 0 & \text{Sinon} \end{cases}$$

La première modélisation mathématique est définie par le modèle non-linéaire suivant :

$$Minimiser \sum_{i=1}^{N} F_i = Minimiser \sum_{i=1}^{N} (C_i - r_i) = Minimiser \sum_{i=1}^{N} C_i \tag{3.1}$$

Sous contraintes :

$$C_i \geq C_{im} \qquad \forall i \in J; \forall m \in L \tag{3.2}$$

$$C_{im} - G(1 - X_{ijm}) \leq C_{jm} - SE_{jm} - p_{jm} \qquad \forall i, j \in J, i \neq j; \forall m \in L \tag{3.3}$$

$$C_{im} - G(1 - Z_{imm'}) \leq C_{im'} - p_{im'} \quad \forall i \in J; \forall m, m' \in L, m \neq m' \tag{3.4}$$

$$Y_{wim} \leq 1 - Y_{wjm'} \left[1 - \frac{max(C_{im} - C_{jm'}; 0)}{p_{im} + SE_{im}} - \frac{max(C_{jm'} - C_{im}; 0)}{p_{jm'} + SE_{jm'}}\right] \quad \forall i, j \in J, i \neq j; \forall m, m' \in L, m \neq m' \tag{3.5}$$

$$X_{ijm} + X_{jim} = 1 \quad \forall i, j \in J, i \neq j; \forall m \in L \tag{3.6}$$

$$Z_{imm'} + Z_{im'm} = 1 \quad \forall i \in J; \forall m, m' \in L, m \neq m' \tag{3.7}$$

$$X_{ijm} + X_{jim} \geq Y_{wim} + Y_{wjm} - 1 \quad \forall i, j \in J, i \neq j; \forall w \in W \tag{3.8}$$

$$Y_{wim} \leq A_{ws} \times RE_{ims} + (1 - RE_{ims}) \quad \forall i \in J; \forall m \in L; \forall w \in W; \forall s \in S \tag{3.9}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} Y_{wim} \geq 1 \quad \forall w \in W \tag{3.10}$$

$$\sum_{w=1}^{W} Y_{wim} = 1 \quad \forall i \in J; \forall m \in L \tag{3.11}$$

$$C_{im} - SE_{im} - p_{im} \geq r_i \quad \forall i \in J; \forall m \in L \tag{3.12}$$

$$X_{ijm}, Z_{imm'}, Y_{wim} \in \{0, 1\} \quad \forall i, j \in J; \quad \forall m, m' \in L; \forall w \in W \tag{3.13}$$

Dans ce modèle, l'équation (3.1) exprime l'objectif traité dans ce problème, la minimisation du temps total de séjour où la date de fin de la tâche i (C_i) est supérieure à sa date de disponibilité (r_i) selon l'équation (3.12). De ce fait, nous garantissons une valeur positive pour le temps total de séjour ($F_i \geq 0$). L'équation (3.2) définit la date de fin de la tâche. L'équation (3.3) présente la relation entre deux opérations consécutives sur la machine m. La date de début de l'opération $O_{jm'}$ doit être supérieure ou égale à la date de fin de l'opération O_{im}. L'équation (3.4) montre la relation entre deux opérations de la tâche i qui ne sont pas obligatoirement consécutives. La date de début de l'opération $O_{im'}$ doit être supérieure ou égale à la date de fin de l'opération O_{im}. L'équation (3.5) exprime la relation entre deux opérations qui sont exécutées par le même opérateur w, il ne peut pas être affecté pour exécuter d'autres opérations en même temps, c'est-à-dire, un opérateur w doit finir l'exécution d'une opération O_{im} pour réaliser l'opération $O_{jm'}$.

Ensuite, la contrainte (3.6) exprime l'ordre dans lequel deux tâches sont exécutées sur la même machine m. La contrainte (3.7) exprime l'ordre dans lequel une tâche i est effectuée sur deux machines m et m'.

Ici, nous présentons deux types de contraintes : l'équation (3.8) qui fait référence à l'affectation d'ordre des opérations qui sont exécutées par le même opérateur ; et les équations (3.9), (3.10), (3.11) qui assurent qu'un opérateur soit affecté pour réaliser une tâche i s'il maîtrise toutes les compétences demandées, un opérateur exécute au moins une opération

afin d'éviter qu'un opérateur reste inactif dans l'ordonnancement, et un seul opérateur est affecté pour réaliser une tâche i sur la machine m, respectivement. L'équation (3.12) signifie que la date de début de l'opération O_{im} est conditionnée par la date de disponibilité de chaque tâche et finalement, l'équation (3.13) définit les variables de décision binaires.

3.2.1.2 Deuxième modélisation mathématique : linéaire

Après le premier modèle proposé, nous avons décidé de le linéariser afin de présenter un modèle linéaire et réduire sa complexité de résolution au prix d'un grand nombre de variables. Dans l'état de l'art, nous avons trouvé des points intéressants sur les façons de formuler mathématiquement un problème d'ordonnancement [95], [96]. En utilisant un modèle linéaire, deux approches sont possibles : la première est la modélisation qui prend en compte les indices de temps pour donner la séquence de tâches période par période, c'est-à-dire, la discrétisation du temps pendant laquelle sera effectuée le processus de planification de la production (horizon du temps). La deuxième méthode est la modélisation selon les relations de précédence existantes entre les différentes tâches, autrement dit, à partir de la détermination de dates de début et dates de fin des activités. Cette sous-section expose les deux types de formulation ci-après :

Modèle mathématique avec discrétisation du temps

Le modèle proposé est une programmation linéaire mixte en nombres entiers (MILP - mixed integer linear programming). Il est ainsi très performant sur des problèmes avec horizon important. Il est décrit par les paramètres et variables de décisions ci-après :

- **Paramètres**

J	Ensemble des tâches $J = \{1, ..., N\}$
L	Ensemble des machines $L = \{1, ..., M\}$
W	Ensemble des opérateurs $W = \{1, ..., N_W\}$
S	Ensemble des compétences $S = \{1, ..., N_S\}$
T	Ensemble du temps $T = \{1, ..., H\}$
i, j	Indices de tâches, $i,j = \{1, ..., N\}$
m, m'	Indices de machines, $m,m' = \{1, ..., M\}$
w	Indices de opérateurs, $w = \{1, 2, 3, ..., N_W\}$
s	Indices de compétences, $s = \{1, 2, 3, ..., N_S\}$
H	Date de livraison attendue (horizon du temps)

t	Indices du temps d'ordonnancement, $t = \{1, 2, 3, ..., H\}$
G	Grand nombre positif
SE_{im}	Temps de montage de la i sur la machine m
p_{im}	Temps opératoire de la tâche i sur la machine m
r_i	Date de disponibilité de la tâche i

$$RE_{ims} = \begin{cases} 1 & \text{Si la tâche } i \text{ sur la machine } m \text{ nécessite la compétence } s \\ 0 & \text{Sinon} \end{cases}$$

$$A_{ws} = \begin{cases} 1 & \text{Si l'opérateur } w \text{ possède la compétence } s \\ 0 & \text{Sinon} \end{cases}$$

- **Variables de décision**

C_{im}	Date de fin de la tâche i sur la machine m
C_i	Date de fin de la tâche $i = \underset{m \in L}{Max}\ C_{im}$

$$X_{ijm} = \begin{cases} 1 & \text{Si la tâche } i \text{ précède la tâche } j \text{ sur la machine } m \\ 0 & \text{Sinon} \end{cases}$$

$$Z_{imm'} = \begin{cases} 1 & \text{Si la tâche } i \text{ est exécutée sur la machine } m \text{ puis sur la machine } m\text{'} \\ 0 & \text{Sinon} \end{cases}$$

$$Y_{wim} = \begin{cases} 1 & \text{Si l'opérateur } w \text{ exécute la tâche } i \text{ sur la machine } m \\ 0 & \text{Sinon} \end{cases}$$

$$OP_{wimt} = \begin{cases} 1 & \text{Si l'opérateur } w \text{ exécute la tâche } i \text{ sur la machine } m \text{ pendant la période } t \\ 0 & \text{Sinon} \end{cases}$$

$$SK_{wims} = \begin{cases} 1 & \text{Si l'opérateur } w \text{ exécute la tâche } i \text{ sur la machine } m \text{ en utilisant la} \\ & \text{compétences } s \\ 0 & \text{Otherwise} \end{cases}$$

$$\gamma_{imt} = \begin{cases} 1 & \text{Si la tâche } i \text{ est affectée à la machine } m \text{ pendant la période } t \\ 0 & \text{Sinon} \end{cases}$$

Nous présentons le modèle avec les indices de temps :

$$Minimiser \sum_{i=1}^{N} F_i = Minimiser \sum_{i=1}^{N} (C_i - r_i) = Minimiser \sum_{i=1}^{N} C_i \tag{3.14}$$

Sous contraintes :

$$C_i \geq C_{im} \quad \forall i \in J; \forall m \in L \tag{3.15}$$

$$C_i \leq H \quad \forall i \in J \tag{3.16}$$

$$C_{im} - G(1 - X_{ijm}) \leq C_{jm} - SE_{jm} - p_{jm} \quad \forall i, j \in J, i \neq j; \forall m \in L \tag{3.17}$$

$$C_{im} - G(1 - Z_{imm'}) \leq C_{im'} - p_{im'} \quad \forall i \in J; \forall m, m' \in L, m \neq m' \tag{3.18}$$

$$X_{ijm} + X_{jim} = 1 \quad \forall i, j \in J, i \neq j; \forall m \in L \tag{3.19}$$

$$Z_{imm'} + Z_{im'm} = 1 \quad \forall i \in J; \forall m, m' \in L, m \neq m' \tag{3.20}$$

$$C_{im} - SE_{im} - p_{im} = \sum_{t=0}^{H} \gamma_{imt} \times t \quad \forall i \in J; \forall m \in L \tag{3.21}$$

$$\sum_{t=0}^{H} \gamma_{imt} = 1 \quad \forall i \in J; \forall m \in L \tag{3.22}$$

$$\sum_{w=1}^{N_w} OP_{wimt} = \sum_{d}^{H} \gamma_{imd} \quad \forall i \in J; \forall m \in L; \forall t \in T \tag{3.23}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} OP_{wimt} \leq 1 \quad \forall w \in W; \forall t \in T \tag{3.24}$$

$$OP_{wimt} \leq Y_{wim} \quad \forall w \in W; \forall i \in J; \forall m \in L; \forall t \in T \tag{3.25}$$

$$\sum_{w=1}^{N_w} Y_{wim} \leq 1 \quad \forall i \in J; \forall m \in L \tag{3.26}$$

$$SK_{wims} \leq A_{ws} \quad \forall w \in W; \forall i \in J; \forall m \in L; \forall s \in S \tag{3.27}$$

$$\sum_{s=1}^{N_s} RE_{ims} \times \sum_{t=0}^{H} OP_{wimt} \geq \sum_{s=1}^{N_s} SK_{wims} \quad \forall i \in J; \forall m \in L; \forall t \in T \tag{3.28}$$

$$\sum_{w=1}^{N_s} SK_{wims} = RE_{ims} \quad \forall i \in J; \forall m \in L; \forall s \in S \tag{3.29}$$

$$C_{im} - SE_{im} - p_{im} \geq r_i \quad \forall i \in J; \forall m \in L \tag{3.30}$$

$$X_{ijm}, Z_{imm'}, Y_{wim}, \gamma_{imt}, OP_{wimt}, SK_{wims} \in \{0, 1\} \quad \forall i, j \in J; \forall m, m' \in L; \forall w \in W \tag{3.31}$$

De la même manière que dans le modèle précédent, la fonction (3.14) exprime la minimisation du temps total de séjour (la date de fin de la tâche i (C_i) est supérieure à sa date

de disponibilité (r_i) - équation (3.30)), nous garantissons une valeur positive pour le temps total de séjour ($F_i \geq 0$). L'équation (3.15) définit la date de fin de la tâche et l'équation (3.16) exprime que cette date de fin de la tâche qui doit être inférieure à l'horizon du travail défini (limiter la discrétisation du temps). L'équation (3.17) exprime la relation entre deux opérations consécutives sur la machine m. La date de début de l'opération $O_{jm'}$ doit être supérieure ou égale à la date de fin de l'opération O_{im}. L'équation (3.18) présente la relation entre deux opérations de la tâche i qui ne sont pas obligatoirement consécutives. La date de début de l'opération $O_{im'}$ doit être supérieure ou égale à la date de fin de l'opération O_{im}.

Après, la contrainte (3.19) montre l'ordre dans lequel deux tâches sont exécutées sur la même machine. La contrainte (3.20) indique l'ordre dans lequel une tâche i est effectuée sur deux machines m et m'.

Nous continuons avec des contraintes qui vont discrétiser le temps dans le modèle, l'équation (3.21) permet de créer une variable binaire (γ) à partir de la date de début d'une opération. La contrainte (3.22) garantit l'existence d'une seule date de début pour une opération i sur la machine m à la période t. Ensuite, les équations (3.23), (3.24), (3.25) et (3.26) assurent l'affectation de l'opérateur w pendant toute la période d'exécution de la tâche i sur la machine m, c'est-à-dire qu'un seul opérateur sera affecté pour exécuter l'opération O_{im}.

Les équations (3.27) et (3.28) indiquent que seul un opérateur ayant les compétences demandées peut effectuer la tâche concernée. L'équation (3.29) garantit la prise en compte de toutes les compétences requises pour exécuter chaque tâche.

Finalement, l'équation (3.30) décrit que la date de début de l'opération O_{im} est conditionnée par la date de disponibilité de chaque tâche et l'équation (3.31) définit les variables de décision binaires.

Modèle mathématique sans discrétisation du temps

Cette section présente un modèle mathématique de programmation linéaire en nombres entiers, quelques paramètres sont modifiés et ajoutés par rapport aux modèles précédents :

Variables de décision

C_{wim} Date de fin de la tâche i sur la machine m réalisée par l'opérateur w

C_i Date de fin de la tâche $i = \underset{w \in W, m \in L}{Max} C_{wim}$.

C'est-à-dire que nous modifions la date de fin de l'opération O_{im} pour la date à laquelle l'opérateur w finit l'opération.

$$V_{imjm'} = \begin{cases} 1 \text{ si la tâche } i \text{ sur la machine } m \text{ précède la tâche } j \text{ sur la machine } m' \\ 0 \text{ Sinon} \end{cases}$$

$$Minimiser \sum_{i=1}^{N} F_i = Minimiser \sum_{i=1}^{N} (C_i - r_i) = Minimiser \sum_{i=1}^{N} C_i \tag{3.32}$$

Sous contraintes :

$$C_i \geq C_{wim} \quad \forall w \in W; \forall i \in J; \forall m \in L \tag{3.33}$$

$$C_{wim} - G(1 - X_{ijm}) \leq C_{wjm} - SE_{jm} - p_{jm} \quad \forall i, j \in J, i \neq j; \forall m \in L; \forall w \in W \tag{3.34}$$

$$C_{wim} - G(1 - Z_{imm'}) \leq C_{wim'} - p_{im'} \quad \forall i \in J; \forall m, m' \in L, m \neq m'; \forall w \in W \tag{3.35}$$

$$C_{wim} - G(1 - V_{imjm'}) \leq C_{wjm'} - SE_{jm'} - p_{jm'} \quad \forall i, j \in J; \forall m, m' \in L; \forall w \in W \tag{3.36}$$

$$X_{ijm} + X_{jim} = 1 \quad \forall i, j \in J, i \neq j; \forall m \in L \tag{3.37}$$

$$Z_{imm'} + Z_{im'm} = 1 \quad \forall i \in J; \forall m, m' \in L, m \neq m' \tag{3.38}$$

$$V_{imjm'} + V_{jm'im} \leq 1 \quad \forall i, j \in J; \forall m, m' \in L \tag{3.39}$$

$$V_{imjm'} + V_{jm'im} \geq Y_{wim} + Y_{wjm'} - 1 \quad \forall i, j \in J, i \neq j; \forall m, m' \in L; \forall w \in W \tag{3.40}$$

$$V_{imim'} + V_{im'im} \geq Y_{wim} + Y_{wim'} - 1 \quad \forall i \in J; \forall m, m' \in L, m \neq m'; \forall w \in W \tag{3.41}$$

$$Y_{wim} \leq RE_{ims} \times A_{ws} + (1 - RE_{ims}) \quad \forall i \in J; \forall m \in L; \forall w \in \Gamma; \forall s \in S \tag{3.42}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} Y_{wim} \geq 1 \quad \forall w \in W \tag{3.43}$$

$$\sum_{w=1}^{W} Y_{wim} = 1 \quad \forall i \in J; \forall m \in L \tag{3.44}$$

$$C_{wim} - SE_{im} - p_{im} \geq r_i \quad \forall i \in J; \forall m \in L; \forall w \in W \tag{3.45}$$

$$X_{ijm}, Z_{imm'}, Y_{wim}, V_{imjm'} \in \{0, 1\} \quad \forall i, j \in J; \forall m, m' \in L; \forall w \in W \tag{3.46}$$

Ce modèle linéaire considère aussi la minimisation du temps total de séjour(3.32) ($F_i \geq 0$). L'équation (3.33) représente la date de fin. Les équations (3.34) et (3.35) comme il a déjà été expliqué auparavant, montrent la relation entre deux opérations consécutives qui sont exécutées sur la même machine m et la relation entre deux opérations de la tâche i qui

ne sont pas obligatoirement consécutives. L'équation (3.36) exprime la relation entre deux opérations qui sont exécutées par le même opérateur w.

Les contraintes (3.37) et (3.38) expriment l'ordre dans lequel deux tâches sont exécutées sur la même machine m et l'ordre dans lequel une tâche i est effectuée sur deux machines m et m'. La contrainte (3.39) définit l'ordre dans lequel deux opérations sont exécutées par le même opérateur w.

Ensuite, deux types de contraintes sont pris en compte, le premier type - les équations (3.40), (3.41) - exprime la méthode d'affectation d'ordre aux opérations qui sont exécutées par le même opérateur, ces contraintes garantissent l'ordre de deux opérations faites par l'opérateur w même sur deux machines différentes. Concernant, le deuxième type - les équations (3.42), (3.43), (3.44) - de la même façon que sur le premier modèle, seul un opérateur qui maîtrise toutes les compétences demandées sera affecté pour réaliser la tâche, un opérateur exécute au moins une opération et un seul opérateur est affecté pour réaliser une tâche i sur la machine m. Enfin, l'équation (3.46) définit les variables de décision binaires.

3.2.2 Modélisation mathématique : contraintes de ressources humaines et d'outillage

Dans cette sous-section nous partons de la même situation problématique mais en ajoutant une contrainte en plus par rapport à l'outillage, autrement dit, nous prenons en compte la disponibilité des différents outils qui sont demandés pour traiter une opération. Les outils peuvent varier selon les activités et les machines à utiliser (porte outil, montages d'usinage, outil coupant). Nous présentons le modèle avec la nouvelle contrainte pour le cas linéaire sans discrétisation du temps puisque selon les résultats expérimentaux exposés dans la section 3.4.2, il est le modèle le plus performant en considérant les contraintes liées aux affectations de personnel multi-compétent.

Nous reprenons le graphe de la sous-section 3.2.1, d'une solution de problème d'ordonnancement. Nous ajoutons un paramètre en plus, l'affectation des outils (figure 3.2). Nous rappelons que les dates de disponibilité sont différentes pour chaque tâche ainsi que les temps de montage et les temps opératoires.

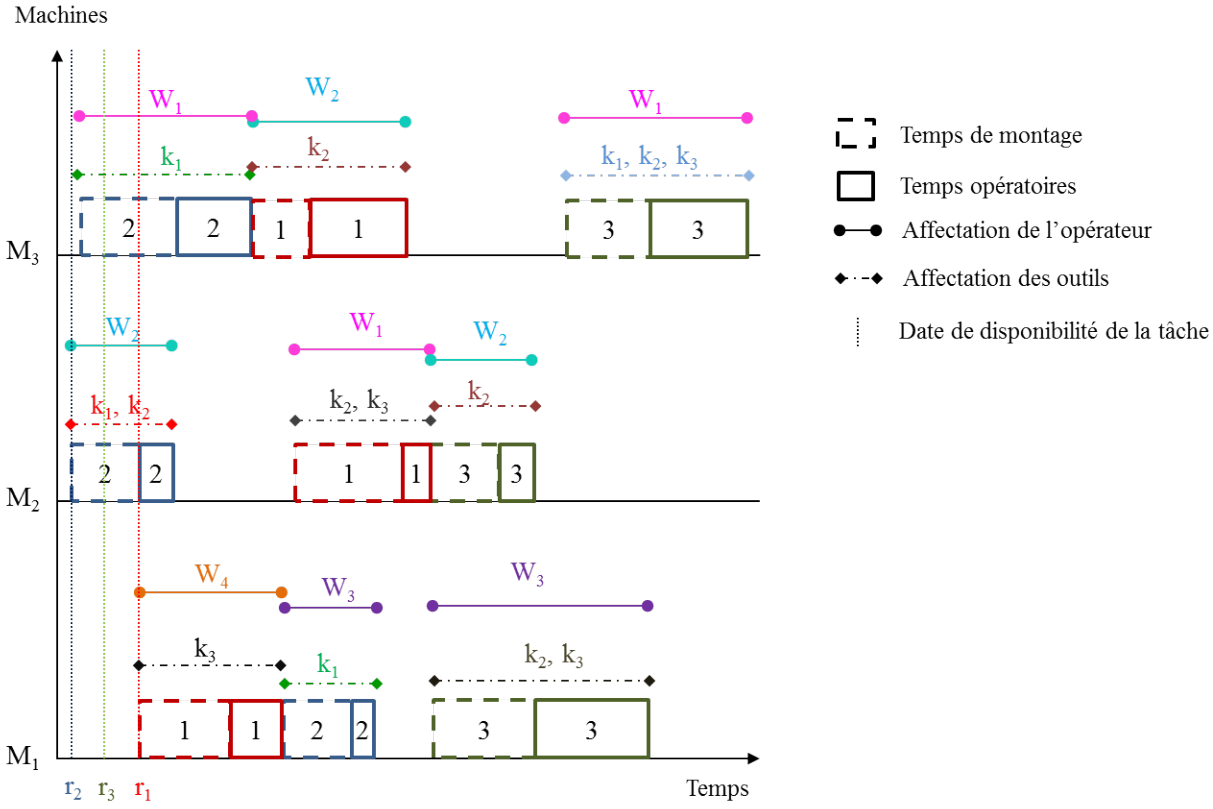


FIGURE 3.2 – Exemple d'ordonnancement sous contraintes de ressources humaines et d'outillage

- **Paramètres**

J	Ensemble des tâches $J = \{1, ..., N\}$
L	Ensemble des machines $L = \{1, ..., M\}$
W	Ensemble des opérateurs $W = \{1, ..., N_W\}$
S	Ensemble des compétences $S = \{1, ..., N_S\}$
K	Ensemble des outils $K = \{1, ..., N_K\}$
i, j	Indices de tâches, $i,j = \{1, ..., N\}$
m, m'	Indices de machines, $m,m' = \{1, ..., M\}$
w	Indices de opérateurs, $w = \{1, 2, 3, ..., N_W\}$
s	Indices de compétences, $s = \{1, 2, 3, ..., N_S\}$
k	Indices des outils, $k = \{1, 2, 3, ..., N_K\}$
G	Grand nombre positif
SE_{im}	Temps de montage de la i sur la machine m
p_{im}	Temps opératoire de la tâche i sur la machine m
r_i	Date de disponibilité de la tâche i

NBO_k Nombre des outils disponibles de type k

$$RE_{ims} = \begin{cases} 1 \text{ Si la tâche } i \text{ sur la machine } m \text{ nécessite la compétence } s \\ 0 \text{ Sinon} \end{cases}$$

$$A_{ws} = \begin{cases} 1 \text{ Si l'opérateur } w \text{ possède la compétence } s \\ 0 \text{ Sinon} \end{cases}$$

$$RO_{imk} = \begin{cases} 1 \text{ Si la tâche } i \text{ sur la machine } m \text{ demande l'outil } k \\ 0 \text{ Sinon} \end{cases}$$

- **Variables de décision**

C_{wim} Date de fin de la tâche i sur la machine m réalisé par l'opérateur w

C_i Date de fin de la tâche $i = \underset{w \in W, m \in L}{Max} C_{wim}$

$$X_{ijm} = \begin{cases} 1 \text{ Si la tâche } i \text{ précède la tâche } j \text{ sur la machine } m \\ 0 \text{ Sinon} \end{cases}$$

$$Z_{imm'} = \begin{cases} 1 \text{ Si la tâche } i \text{ est exécutée sur la machine } m \text{ puis sur la machine } m\text{'} \\ 0 \text{ Sinon} \end{cases}$$

$$Y_{wim} = \begin{cases} 1 \text{ Si l'opérateur } w \text{ exécute la tâche } i \text{ sur la machine } m \\ 0 \text{ Sinon} \end{cases}$$

$$V_{imjm'} = \begin{cases} 1 \text{ Si la tâche } i \text{ sur la machine } m \text{ précède la tâche } j \text{ sur la machine } m\text{'} \\ 0 \text{ Sinon} \end{cases}$$

$$Minimiser \sum_{i=1}^{N} F_i = Minimiser \sum_{i=1}^{N} (C_i - r_i) = Minimiser \sum_{i=1}^{N} C_i \tag{3.47}$$

Sous les contraintes suivantes :

$$C_i \geq C_{wim} \quad \forall w \in W; \forall i \in J; \forall m \in L \tag{3.48}$$

$$C_{wim} - G(1 - X_{ijm}) \leq C_{wjm} - SE_{jm} - p_{jm} \quad \forall i, j \in J, i \neq j; \forall m \in L; \forall w \in W \tag{3.49}$$

$$C_{wim} - G(1 - Z_{imm'}) \leq C_{wim'} - p_{im'} \quad \forall i \in J; \forall m, m' \in L, m \neq m'; \forall w \in W \tag{3.50}$$

$$C_{wim} - G(1 - V_{imjm'}) \leq C_{wjm'} - SE_{jm'} - p_{jm'} \quad \forall i, j \in J; \forall m, m' \in L; \forall w \in W \tag{3.51}$$

$$X_{ijm} + X_{jim} = 1 \quad \forall i, j \in J, i \neq j; \forall m \in L \tag{3.52}$$

$$Z_{imm'} + Z_{im'm} = 1 \quad \forall i \in J; \forall m, m' \in L, m \neq m' \tag{3.53}$$

$$V_{imjm'} + V_{jm'im} \leq 1 \quad \forall i, j \in J; \forall m, m' \in L \tag{3.54}$$

$$V_{imjm'} + V_{jm'im} \geq Y_{wim} + Y_{wjm'} - 1 \quad \forall i, j \in J, i \neq j; \forall m, m' \in L; \forall w \in W \tag{3.55}$$

$$V_{imim'} + V_{im'im} \geq Y_{wim} + Y_{wim'} - 1 \quad \forall i \in J; \forall m, m' \in L, m \neq m'; \forall w \in W \tag{3.56}$$

$$RO_{imk} + \sum_{\substack{j=1,\\ j\neq i}}^{N} \sum_{\substack{m'=1,\\ m'\neq m}}^{M} (RO_{imk} \times RO_{jm'k} \times (1 - V_{imjm'} - V_{jm'im})) \leq NBO_k \quad \forall i \in J; \forall m \in L; \forall k \in K \tag{3.57}$$

$$Y_{wim} \leq RE_{ims} \times A_{ws} + (1 - RE_{ims}) \quad \forall i \in J; \forall m \in L; \forall w \in \Gamma; \forall s \in S \tag{3.58}$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} Y_{wim} \geq 1 \quad \forall w \in W \tag{3.59}$$

$$\sum_{w=1}^{W} Y_{wim} = 1 \quad \forall i \in J; \forall m \in L \tag{3.60}$$

$$C_{wim} - SE_{im} - p_{im} \geq r_i \quad \forall i \in J; \forall m \in L; \forall w \in W \tag{3.61}$$

$$X_{ijm}, Z_{imm'}, Y_{wim}, V_{imjm'} \in \{0, 1\} \quad \forall i, j \in J; \forall m, m' \in L; \forall w \in W \tag{3.62}$$

Ce modèle a comme objectif la minimisation du temps total de séjour (3.47) ($F_i \geq 0$). L'équation (3.48) définit la date de fin de la tâche i.

Les équations (3.49) et (3.50), expliquées dans la sous-section 3.2.1.2, expriment la relation entre deux opérations consécutives qui sont exécutées sur la même machine m et la relation entre deux opérations de la tâche i qui ne sont pas obligatoirement consécutives. L'équation (3.51) montre la relation entre deux opérations qui sont exécutées par le même opérateur w.

Les équations (3.52) et (3.53) indiquent la séquence dans laquelle deux tâches sont exécutées sur la même machine m et l'ordre dans lequel une tâche i est traitée sur deux machines m et m'. La contrainte (3.54) définit l'ordre dans lequel deux opérations sont réalisées par le même opérateur w, il est nécessaire de fixer un ordre entre les deux opérations que si elles sont faites par l'opérateur w (même opérateur).

Par la suite, trois types de contraintes sont pris en compte, le premier type - les équations (3.55), (3.56) - précise la séquence des opérations exécutées par le même opérateur, elles

garantissent l'ordre de deux opérations faites par l'opérateur w même sur deux machines différentes. Le deuxième type - les équations (3.57) - garantit l'affectation des outils afin d'exécuter une tâche en prenant en compte sa disponibilité et la quantité d'outils de chaque type. Enfin, nous avons le troisième type, les équations (3.58), (3.59), (3.60). De la même façon que sur le modèle linéaire sans discrétisation, seul un opérateur qui maîtrise toutes les compétences demandées sera affecté pour réaliser la tâche, un opérateur exécute au minimum une opération et un seul opérateur est affecté pour traiter une tâche i sur la machine m. Enfin, l'équation (3.62) définit les variables de décision binaires.

3.3 Méthodes de résolution

Cette section présente les méthodes de résolution adaptées pour résoudre le problème d'ordonnancement d'un atelier de type open shop. D'abord, nous présentons les méthodes exactes employées : la résolution des problèmes linéaires et non-linéaires proposés. Puis dans le prochain chapitre, nous exposons l'ensemble des méthodes approchées qui ont été développées pour obtenir des solutions de bonne qualité.

Méthodes exactes

Comme pour tout problème NP-difficile, les méthodes de résolution qui utilisent la programmation mixte en nombres entiers, garantissent l'optimalité pour des instances de petites tailles, avec des temps de calcul acceptables ; cette durée augmente rapidement avec l'augmentation de la taille des instances (grandes tailles). Cette résolution par méthodes exactes permet aussi d'estimer la performance des méthodes approchées sur des instances de petites tailles (voir chapitre 4). Dans cette sous-section de méthodes exactes, les modèles mathématiques ont été résolus pour traiter le problème développé dans ce chapitre.

De cette façon, la méthode de résolution exacte est introduite et les résultats de la résolution des modèles mathématiques des sections précédentes seront ensuite présentés et expliqués.

- **Méthodes par programmation linéaire et non-linéaire**

Les modèles mathématiques proposés dans les sous-sections 3.2.1 et 3.2.2 ont été conçus avec des variables entières et binaires. Ces types de modèles sont basés sur la programmation mixte en nombres entiers (MIP). Des expérimentations sont réalisées sur des solveurs commerciaux tels que CPLEX de ILOG-IBM et LINGO de Lindo Systems Inc. Nous avons testé les modèles, et les résultats sont résumés dans la sous-section 3.4.2.

La résolution du problème $O_m|r_i|\sum F_i$ avec contraintes de ressources humaines et d'outillages, requiert la construction de M séquences de tâches. Les méthodes exactes permettent de déterminer les solutions exactes des problèmes de tailles limitées.

La résolution des instances testées dans les logiciels CPLEX et LINGO a été bornée à 1800 secondes (30 minutes). Pour les logiciels qui résolvent des problèmes combinatoires, garantir l'optimalité de certaines instances en temps raisonnables n'est pas faisable pour les problèmes classifiés comme NP-difficiles. Cependant, les logiciels peuvent fournir des solutions de bonne qualité.

3.4 Expérimentations

Cette partie présente les résultats de l'application numérique des modèles mathématiques décrits dans ce chapitre, sur des instances théoriques du problème d'ordonnancement dans un atelier de production open shop. Cette section commence avec une description sur la génération des instances numériques qui ont été testées. Ensuite, les différents tableaux de résultats sont exposés et finalement, les comparaisons respectives et conclusions entre les formulations exposées sont effectuées.

3.4.1 Génération des instances numériques

Plusieurs problèmes ont été générés de façon aléatoire pour chaque environnement de production. Dans un premier temps nous traitons les environnements qui n'incluent que les contraintes de ressources humaines (opérateurs). Ils sont détaillés ci-dessous :

- Un environnement ou scénario de production est défini par le nombre de tâches N, le nombre de machines M, le nombre de ressources humaines (opérateurs W) et le nombre de compétences S qui sont nécessaires pour réaliser les opérations. Le nombre de tâches pour les instances de petites tailles sont 3, 4, 5, 7 et le nombre de machines 2, 3, 4.
- Pour la génération numérique de ces instances, différents cas ont été proposés afin de montrer l'influence des données sur les résultats :
  Des auteurs comme [143], [144], [165] ont proposé la génération des valeurs de temps opératoires et de montages en utilisant un intervalle de distribution uniforme. D e la même façon, Loukil *et al.* [124] ont exposé que ce mécanisme est un moyen classique existant dans la littérature pour générer aléatoirement des instances pour des problèmes d'ordonnancement. Ainsi, les temps opératoires sont des entiers uniformé-

ment distribués dans $[1, \omega]$, et les temps de montage sont des variables aléatoires entre $[30, \omega]$, où ω peut prendre trois valeurs (ω_1, ω_2 et ω_3) : 50, 75 ou 100. Nous avons décidé de créer un intervalle variable avec des valeurs différentes afin de tester des instances incluant un comportement largement diversifié.

D'après le facteur considéré par Sourd [166] et Belouadah *et al.* [14], pour la date de disponibilité d'une tâche, nous avons également défini la date de disponibilité dans un intervalle aléatoire de distribution uniforme décrit par $[0, \lambda \sum_{m=1}^{M} p_{im}]$; où p_{im} est le temps opératoire d'une tâche i sur la machine m, et λ est le facteur de date de disponibilité qui peut prendre la valeur de (λ_1, λ_2 et λ_3) : 0.1, 0.3, ou 0.5. Cet aspect signifie que nous pouvons générer des combinaisons différentes de paramètres au moment de créer une instance.

Selon les références trouvées dans la littérature et comme il n'y a pas d'instances dédiées aux problèmes d'ordonnancement avec multi-compétences (MSPSP - multi-skilled project scheduling problem), nous avons suivi les conseils de Kazemipoor *et al.* [91] qui ont résolu un modèle d'ordonnancement avec multi-compétences par une recherche de dispersion et Bellenguez *et al.* [13] qui ont traité les problèmes d'ordonnancement avec multi-compétences avec un niveau hiérarchique de compétences ; pour générer quelques instances de façon raisonnable. Pour chaque scénario, le nombre de ressources humaines (opérateurs) a été déterminé par le nombre de machines (nous supposons une ressource par machine), le nombre de compétences pour exécuter les activités est 3, 5 ou 7 et les 2 matrices, l'une définissant les compétences demandées par chaque opération et l'autre définissant les compétences maîtrisées par les opérateurs qui sont générées d'une manière aléatoire (0 ou 1), où 1 implique que l'opération demande cette compétence ou que l'opérateur domine cette compétence.

Concernant les instances qui prennent en compte les contraintes d'outillage, nous avons gardé les critères déjà présentés pour générer ces paramètres mais nous ajoutons de nouveaux éléments nécessaires pour créer les environnements de production respectifs. Ceux-ci sont décrits ci-après :

- Un environnement de production de cette deuxième partie est défini de la même façon, par le nombre de tâches N, le nombre de machines M, le nombre de ressources humaines W, le nombre de compétences qui sont nécessaires pour réaliser les opérations S et le nombre de types d'outils existants K. Le nombre de tâches pour les instances de petites tailles reste identique (3, 4, 5, 7) ainsi que le nombre de machines (2, 3, 4).
- Le nombre de types d'outils et le nombre de copies de chaque type d'outils ont été générés de manière aléatoire, à partir des instances décrites dans [185]. De cette façon, le nombre de types d'outils est une valeur dans l'intervalle $[M, (N \times M)]$ pendant que

le nombre de copies par type d'outil est une valeur aléatoire dans l'intervalle [2, M]. La matrice qui définit le type d'outils demandé par chaque opération est crée de manière aléatoire (0 ou 1), où 1 implique que l'opération requiert ce type d'outil pour être exécutée.

3.4.2 Résultats expérimentaux

Dans cette sous-section, nous montrons les tableaux de données des différents tests réalisés. Tout d'abord, nous avons deux types de résultats à présenter, les premiers sont liés avec les modèles qui prennent en compte la contrainte d'affectation du personnel avec multi-compétences et les deuxièmes considèrent aussi les contraintes d'affectation d'outillage.

Pour le premier type de résultats, la notation pour chaque scénario suit la structure suivante : $N \times M\_\omega - \lambda - N_W - N_S$. Au sujet des petites instances, les tailles de tâches considérées ont été $N = 3$, 4, 5 et 7 ; des machines $M = 2$, 3 et 4 ; des nombres de ressources humaines (opérateurs), $N_W = 2$, 3 ou 4 et le nombre de compétences $N_S = 3$, 5 ou 7 ; ω et λ sont deux paramètres qui ont été mentionnés dans la sous-section précédente et qui définissent les intervalles des valeurs utilisées dans la génération des temps opératoires et de montage ainsi que dans les dates de disponibilité de tâches. Neuf types de problèmes ont été générés de façon aléatoire pour chaque scénario, dans lequel chaque problème a été lancé dix fois de manière répétitive. Le nombre de tests et de lancements a été choisi pour fournir un large échantillon de différents environnements mais ils ont été limités en taille à cause du temps de calcul. De cette façon, nous avons désigné 117 scénarios différents qui représentent 1053 instances testées (9 problèmes de chaque type de scénario). Dans chaque scénario nous disposons de 9 combinaisons possibles : ω_1 (λ_1, λ_2, λ_3), ω_2 (λ_1, λ_2, λ_3) et ω_3 (λ_1, λ_2, λ_3). Ensuite, nous présentons les tableaux de résultats, en exposant 13 ensembles d'instances (81 problèmes à l'intérieur de chacun) qui regroupent les valeurs des paramètres ω et λ après les différentes combinaisons, en utilisant la notation suivante : $N \times M - N_W - N_S$.

Le tableau 3.3 montre les résultats obtenus à partir du modèle non-linéaire proposé dans la section précédente. Il est composé de deux parties, la première qui est liée avec les valeurs des solutions optimales : *# Inst. résolues* expose le nombre des instances qui ont été résolues de façon optimale dans la limite du temps de 1800s ; la deuxième partie présente l'information par rapport au temps de calcul. Le *CPUs moyen* représente la valeur moyenne du temps de calcul pour résoudre une instance qui appartient à ce scénario, donc chaque environnement possède 81 instances. De la même façon, les valeurs *Max*, *Min* et *Médiane* sont les valeurs maximales, minimales et la médiane de chaque type d'instances. Enfin, σ_{CPUs} montre la déviation standard du temps de calcul entre les instances d'un même scénario. A partir de

ces résultats, les observations suivantes sont obtenues.

La complexité des instances augmente avec le nombre des opérations à ordonnancer, l'espace de recherche s'agrandit. Le modèle MINLP trouve des solutions optimales jusqu'aux instances des 12 opérations avec les différentes configurations de ressources et compétences. A partir de ce point le problème non-linéaire testé sur Lingo réduit considérablement le nombre de solutions optimales trouvées, il n'obtient pas de solutions optimales dans la limite du temps de calcul. Les résultats exposés montrent aussi que parfois, il y a des instances de plus de 12 opérations où la fonction objectif est près de l'optimum avant le temps limite à cause du schéma de séparation et évaluation employé par Lingo, en évitant les solutions optimales. Enfin, cette méthode n'arrive pas à trouver de solutions optimales pour 801 instances.

TABLE 3.3 – Résultats du modèle MINLP - contrainte de ressources humaines

	MINLP (Lingo)					
	Solution optimale	**CPUs**				
Problème	**# Inst. résolues**	**Moyenne**	**Max**	**Min**	**Médiane**	σ_{CPUs}
3X3-3-3	56/81	142,99	1800,00	13,61	60,16	257,77
3X3-3-5	53/81	106,66	1003,53	12,25	52,82	152,40
4X2-2-5	72/81	16,67	126,61	3,17	9,02	20,58
4X3-3-3	38/81	1316,86	3261,00	19,50	1669,04	634,25
4X3-3-5	33/81	1274,93	1800,00	62,65	1535,66	580,52
4X4-4-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
4X4-4-5	0/81	1756,19	1800,00	24,81	1800,00	277,07
5X4-4-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
5X4-4-5	0/81	1742,44	1800,00	112,73	1800,00	266,23
5X4-4-7	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-5	0/81	1696,02	1800,00	31,34	1800,00	408,13
7X3-3-7	0/81	1780,92	1800,00	665,00	1800,00	133,57

D'autre part, nous avons les résultats du premier modèle linéaire exposé qui utilise la discrétisation du temps pour trouver la séquence des tâches. Le tableau 3.4 expose les résultats obtenus d'après la résolution du modèle dans le logiciel Cplex. Nous pouvons constater que le modèle trouve des solutions optimales pour des instances avec 9 opérations ou moins. L'inclusion des indices de temps fait en sorte que la limite de temps de calcul soit atteinte depuis le premier type d'environnement de données. Les résultats sont moins efficaces en comparaison avec ceux obtenus par le modèle non-linéaire proposé. De cette façon, Cplex n'obtient pas la solution optimale dans 926 instances.

Le tableau 3.5 exprime les résultats pour le modèle linéaire qui prend en compte les

TABLE 3.4 – Résultats du modèle MILP avec discrétisation du temps - contrainte de ressources humaines

	MILP-t (Cplex)					
	Solution optimale	**CPUs**				
Problème	**# Inst. résolues**	**Moyenne**	**Max**	**Min**	**Médiane**	σ_{CPUs}
3X3-3-3	32/81	1435,10	1800,00	81,42	1800,00	533,19
3X3-3-5	37/81	1391,71	1800,00	126,69	1800,00	556,60
4X2-2-5	58/81	1068,39	1800,00	113,90	949,47	647,54
4X3-3-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
4X3-3-5	0/81	1800,00	1800,00	1800,00	1800,00	0,00
4X4-4-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
4X4-4-5	0/81	1800,00	1800,00	1800,00	1800,00	0,00
5X4-4-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
5X4-4-5	0/81	1800,00	1800,00	1800,00	1800,00	0,00
5X4-4-7	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-5	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-7	0/81	1800,00	1800,00	1800,00	1800,00	0,00

relations de précédence entre tâches (dates de début et fin) pour les ordonnancer. Ce modèle a été lancé sur Cplex et présente de bonnes performances en termes de nombre de solutions optimales obtenues, le modèle arrive à résoudre toutes les instances qui ont 16 opérations. Les résultats montrent que plus le nombre d'opérations augmente, plus le nombre des solutions optimales se réduit. Pour les scénarios avec 20 opérations ou plus (7 tâches et 3 machines), Cplex n'atteint pas la solution optimale dans 191 instances.

A partir de ces tests nous pouvons affirmer que le modèle linéaire sans discrétisation du temps lancé sur Cplex, présente la meilleure performance des trois. Il arrive à trouver plus de solutions optimales pour les différentes instances générées et testées. De la même manière, nous constatons que le modèle qui inclut l'indice de temps dans sa formulation présente les plus mauvais résultats, fait qui montre que la discrétisation du temps pose des problèmes à mesure que la taille de l'instance augmente.

A priori nous avons trouvé le modèle avec la meilleure performance pour résoudre notre problème. Néanmoins, nous avons testé les deux meilleurs modèles sur logiciels différents (Cplex et Lingo), afin de vérifier l'influence du logiciel sur l'efficience des résultats. Nous avons donc décidé de lancer le modèle MILP sans discrétisation du temps sur Lingo.

Le tableu 3.6 montre les résultats obtenus après le lancement du modèle MILP en utilisant

TABLE 3.5 – Résultats du modèle MILP sans discrétisation du temps - contrainte de ressources humaines

	MILP (Cplex)					
	Solution optimale	**CPUs**				
Problème	**# Inst. résolues**	**Moyenne**	**Max**	**Min**	**Médiane**	σ_{CPUs}
3X3-3-3	81/81	0,65	2,12	0,27	0,59	0,28
3X3-3-5	81/81	0,56	1,30	0,31	0,52	0,16
4X2-2-5	81/81	0,50	0,76	0,30	0,50	0,09
4X3-3-3	81/81	2,41	7,69	0,58	2,37	1,23
4X3-3-5	81/81	3,49	47,53	0,47	2,53	5,50
4X4-4-3	81/81	25,20	568,34	3,48	7,30	76,77
4X4-4-5	81/81	35,70	831,25	1,69	10,67	102,10
5X4-4-3	72/81	361,39	1800,00	12,76	142,09	524,42
5X4-4-5	68/81	508,83	1800,00	17,52	186,09	650,07
5X4-4-7	65/81	617,36	1800,00	9,88	264,27	653,05
7X3-3-3	30/81	1399,79	1800,00	20,20	1800,00	590,09
7X3-3-5	35/81	1290,00	1800,00	81,85	1800,00	656,01
7X3-3-7	25/81	1514,78	1800,00	100,71	1800,00	577,48

TABLE 3.6 – Résultats du modèle MILP sans discrétisation du temps (Lingo) - contrainte de ressources humaines

	MILP (Lingo)					
	Solution optimale	**CPUs**				
Problème	**# Inst. résolues**	**Moyenne**	**Max**	**Min**	**Médiane**	σ_{CPUs}
3X3-3-3	81/81	2,11	11,78	0,31	1,48	2,05
3X3-3-5	81/81	2,06	22,80	0,23	1,51	2,82
4X2-2-5	81/81	0,79	2,93	0,23	0,66	0,50
4X3-3-3	81/81	43,51	600,37	2,36	16,38	88,43
4X3-3-5	81/81	85,06	1216,13	4,62	22,03	205,64
4X4-4-3	30/81	720,49	1800,00	51,39	396,12	673,73
4X4-4-5	28/81	790,89	1800,00	49,08	427,12	710,17
5X4-4-3	0/81	1800,00	1800,00	1800,00	1800,00	0,00
5X4-4-5	0/81	1800,00	1800,00	1800,00	1800,00	0,00
5X4-4-7	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-3	0/81	1796,42	1800,00	1509,64	1800,00	32,26
7X3-3-5	0/81	1800,00	1800,00	1800,00	1800,00	0,00
7X3-3-7	0/81	1800,00	1800,00	1800,00	1800,00	0,00

le logiciel Lingo. Nous pouvons voir que le modèle linéaire reste le modèle le plus performant. Cependant, nous trouvons aussi que dans le cas du modèle MILP, le nombre de solutions optimales varie d'un logiciel à l'autre. Le modèle MILP lancé sur Lingo ne résout pas 590 instances alors que Cplex ne résout pas 191 instances.

Nous présentons le tableau 3.7 qui résume les résultats obtenus par le modèle MINLP (Lingo) et le modèle MILP sans discrétisation du temps (Cplex et Lingo).

TABLE 3.7 – Résumé de résultats du modèle MINLP et MILP

	MINLP			**MILP**					
Problème	**MINLP (Lingo)**			**MILP(Lingo)**			**MILP(Cplex)**		
	# Ins. rés.	$CPUs_A$	σ_{CPUs}	**# Ins. rés.**	$CPUs_A$	σ_{CPUs}	**# Ins. rés.**	$CPUs_A$	σ_{CPUs}
3X3-3-3	56/81	142,99	257,77	81/81	2,11	2,05	**81/81**	0,65	0,28
3X3-3-5	53/81	106,66	152,40	81/81	2,06	2,82	**81/81**	0,56	0,16
4X2-2-5	72/81	16,67	20,58	81/81	0,79	0,50	**81/81**	0,50	0,09
4X3-3-3	38/81	1321,19	634,25	81/81	43,51	88,43	**81/81**	2,41	1,23
4X3-3-5	33/81	1274,93	580,52	81/81	85,06	205,64	**81/81**	3,49	5,50
4X4-4-3	0/81	1800,00	0,00	30/81	761,96	673,73	**81/81**	20,69	76,77
4X4-4-5	0/81	1756,19	277,07	28/81	712,19	710,17	**81/81**	41,09	102,10
5X4-4-3	0/81	1800,00	0,00	0/81	1712,73	0,00	**72/81**	434,49	524,42
5X4-4-5	0/81	1800,00	0,00	0/81	1800,00	0,00	**68/81**	566,79	650,07
5X4-4-7	0/81	1800,00	0,00	0/81	1800,00	0,00	**65/81**	514,15	653,05
7X3-3-3	0/81	1800,00	0,00	0/81	1796,42	32,26	**30/81**	1399,79	590,09
7X3-3-5	0/81	1696,02	408,13	0/81	1800,00	0,00	**35/81**	1290,00	656,01
7X3-3-7	0/81	1780,92	133,57	0/81	1800,00	0,00	**25/81**	1463,30	577,48

Dans ce cadre, nous avons le tableau 3.8 qui exprime la relation trouvée entre les deux types de modèles et les logiciels Cplex et Lingo. Nous exposons le nombre de fois qu'est réduit le temps de calcul, du modèle MINLP au modèle MILP testés sur Lingo et du modèle MILP lancé sur Lingo puis sur Cplex.

La figure 3.3 représente le nombre d'instances non résolues sur Cplex pour le modèle MILP sans discrétisation (le plus performant de trois) en regardant les différents nombre de compétences testées. Selon les résultats obtenus, 72/162 (44,44%) instances n'ont pas été résolues quand le nombre de compétences prend la valeur maximale possible.

Au sujet de la variation du paramètre λ, qui est utilisé dans la génération des dates de disponibilité de tâches, dans la figure 3.4, l'impact le plus représentatif a été trouvé quand il prend la valeur de 0.3 (λ_2) - 67/351 soit 19.09% des instances n'ont pas été résolues. Par rapport à la variation du paramètre ω qui est employé dans la création du temps opératoire et de montage, la figure 3.5 montre que le nombre d'instances non résolues diminue avec l'augmentation de la valeur du paramètre. Le nombre maximum d'instances non résolues -

TABLE 3.8 – Speed up Cplex - Lingo

Problème	**Speed up MINLP/MILP (Lingo)**	**Speed up Lingo/Cplex (MILP)**	**Overall Speed up**
3X3-3-3	67,74	3,27	221,63
3X3-3-5	51,69	3,66	189,24
4X2-2-5	21,06	1,57	33,06
4X3-3-3	30,37	18,02	547,31
4X3-3-5	14,99	24,39	365,56
4X4-4-3	1,59	54,61	87,01
4X4-4-5	1,40	30,47	42,74
5X4-4-3	1,00	4,14	4,14
5X4-4-5	1,00	3,18	3,18
5X4-4-7	1,00	3,50	3,50
7X3-3-3	1,00	1,28	1,29
7X3-3-5	0,94	1,40	1,31
7X3-3-7	0,99	1,23	1,22

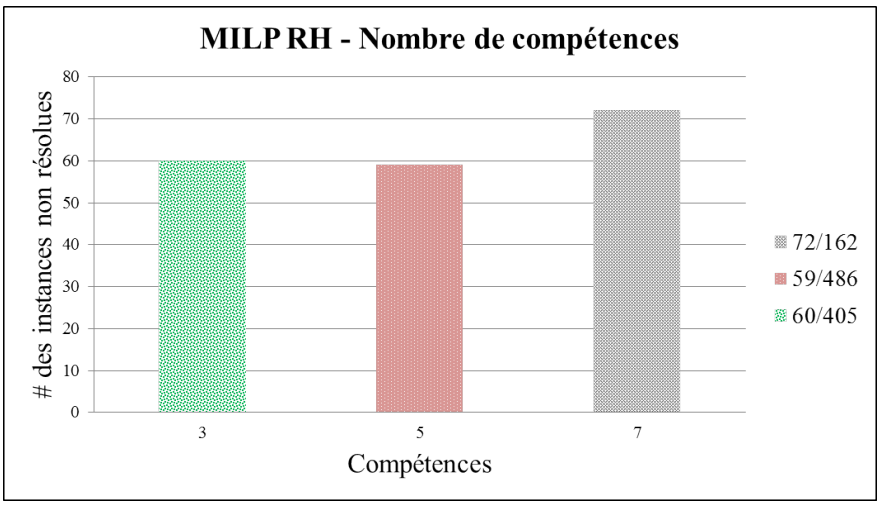


FIGURE 3.3 – Nombre des instances non résolues modèle MILP RH - selon le nombre de compétences

71/351 (20.20%) est obtenu avec la valeur de ω_1 (50)

D'après cette première partie de résultats, pour les tests qui ajoutent les contraintes de disponibilité d'outils, la notation pour chaque scénario possède la structure suivante : $N \times M\_\omega - \lambda - N_W - N_S - N_K$. Les nombres de tâches considérées demeurent identiques (3, 4, 5 et 7) ; des machines (2, 3 et 4) ; le nombre de ressources humaines (2, 3 ou 4), le nombre de compétences (3, 5 ou 7) et le nombre de types d'outils (valeur aléatoire dans l'intervalle $[M, (N \times M)]$) ; ω et λ sont encore utilisés pour générer les valeurs des temps

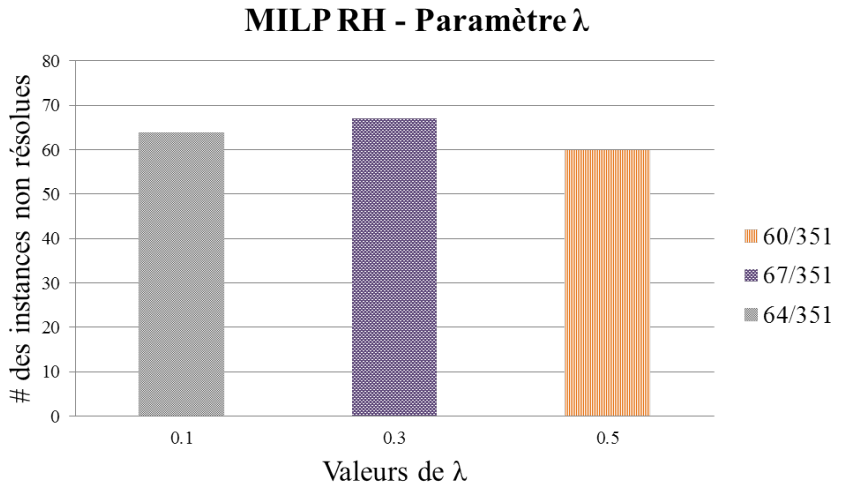


FIGURE 3.4 – Nombre des instances non résolues modèle MILP RH - selon le paramètre λ

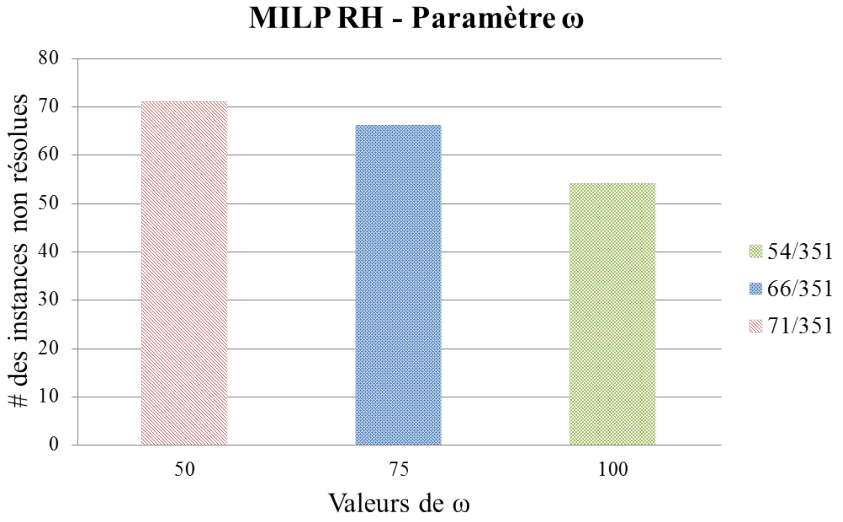


FIGURE 3.5 – Nombre des instances non résolues modèle MILP RH - Paramètre ω

opératoires, de montage et dates de disponibilité de tâches. Comme dans le cas précédent, neuf types de problèmes ont été générés de façon aléatoire pour chaque scénario, dans lequel chaque problème a été lancé dix fois de manière répétitive. Nous avons désigné aussi 117 scénarios différents (1053 instances testées et 9 problèmes de chaque type de scénario). Nous disposons des mêmes 9 combinaisons possibles pour chaque scénario : ω_1 (λ_1, λ_2, λ_3), ω_2 (λ_1, λ_2, λ_3) et ω_3 (λ_1, λ_2, λ_3). Nous exposons les tableaux de résultats où 13 ensembles d'instances regroupent les valeurs des paramètres ω et λ (81 problèmes à l'intérieur de chacun après les différentes combinaisons), de sorte que nous arrivons à la notation suivante : $N \times M - N_W - N_S - N_K$.

Le tableau 3.9 présente le comportement du modèle lancé sur le logiciel Cplex pour

résoudre un problème d'open shop avec des contraintes d'affectation de personnel avec multi-compétences en prenant en compte en même temps les contraintes de disponibilité des outils. Le modèle arrive à trouver toutes les solutions optimales pour les instances avec 16 ou moins opérations. Cplex n'atteint pas la solution optimale dans 246 instances.

TABLE 3.9 – Résultats du modèle MILP sans discrétisation du temps - contrainte de ressources humaines et d'outillage

	MILP (Cplex - outils)					
	Solution optimale	**CPUs**				
Problème	**# Inst. résolues**	**Moyenne**	**Max**	**Min**	**Médiane**	σ_{CPUs}
3X3-3-3-3	81/81	0,77	2,36	0,31	0,67	0,34
3X3-3-5-3	81/81	0,72	1,92	0,06	0,69	0,25
4X2-2-5-5	81/81	0,58	1,01	0,30	0,58	0,14
4X3-3-3-8	81/81	3,95	56,82	0,72	2,75	6,82
4X3-3-5-7	81/81	2,74	10,94	0,87	2,43	1,48
4X4-4-3-10	81/81	20,60	290,32	3,18	10,47	34,98
4X4-4-5-12	80/81	54,08	1800,00	2,68	9,22	238,37
5X4-4-3-18	76/81	409,81	1800,00	9,75	151,56	533,05
5X4-4-5-15	66/81	546,70	1800,00	9,70	172,97	698,15
5X4-4-7-11	60/81	707,76	1800,00	17,92	290,26	720,54
7X3-3-3-9	18/81	1552,02	1800,00	37,75	1800,00	522,48
7X3-3-5-17	12/81	1662,53	1800,00	263,69	1800,00	363,40
7X3-3-7-21	9/81	1687,01	1800,00	105,46	1800,00	359,47

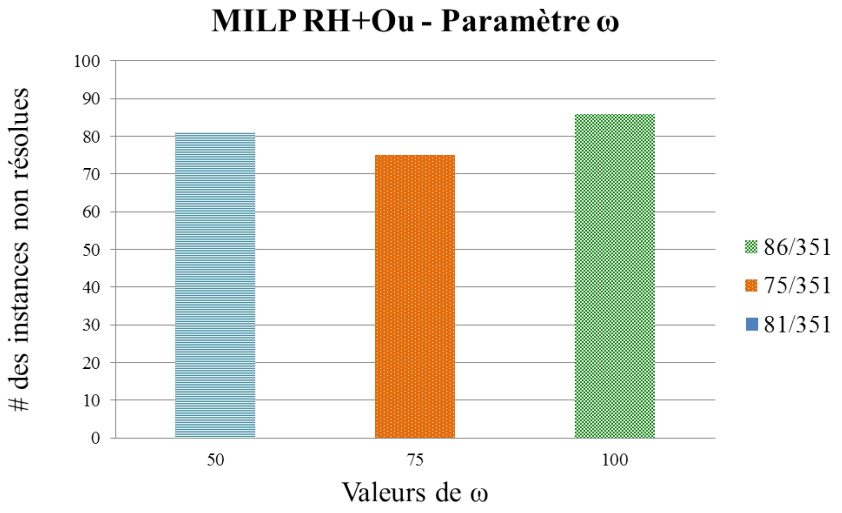


FIGURE 3.6 – Nombre des instances non résolues modèle MILP RH et outillage - selon le paramètre ω

De la même façon que pour le modèle qui implique que les contraintes d'affectation de personnel avec multi-compétences, nous étudions l'influence de la variation des paramètres pour ce modèle qui comprend les contraintes des outils. Selon le graphique 3.6, la majorité des instances non résolues on une valeur ω_3 (86/351 instances - 24.50%). Concernant la valeur de λ, nous trouvons que quand il prend la valeur de 0.1 (λ_1) au moment de générer les différentes instances, il produit le plus grand pourcentage d'instances non résolues (24.50%). De cette manière nous pouvons voir qu'en ajoutant une contrainte en plus, le nombre d'instances non résolues de façon optimale change ainsi que l'influence des paramètres pour déterminer la quantité d'instances sans solution optimale qui appartient à chaque valeur du paramètre. Cependant, nous constatons que l'augmentation du nombre de tâches et machines (nombre d'opérations) a une relation avec la difficulté du type d'environnement de production. Donc, pour les deux types de modèle (RH, et RH + outillage) les instances plus difficiles à résoudre sont celles où le nombre d'opérations est supérieur à 20.

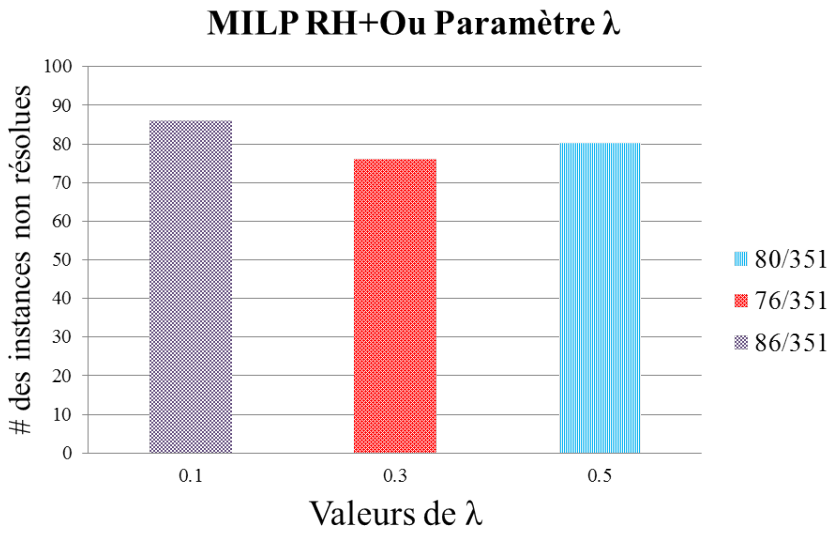


FIGURE 3.7 – Nombre des instances non résolues modèle MILP RH et outillage - selon le paramètre λ

3.5 Conclusion

Dans ce chapitre nous avons considéré le problème d'ordonnancement de type open shop $O_m|r_i|\sum F_i$ dans lequel un ensemble de tâches doivent être affectées sur un ensemble de machines en considérant des informations qui les décrivent : temps opératoires, de montage, dates de disponibilité. En plus, des contraintes existantes liées aux ressources humaines (opérateurs), compétences maîtrisées et compétences demandées pour réaliser une opération

ont été prises en compte.

Nous avons traité ce problème par des méthodes exactes : trois formulations mathématiques ont été proposées pour représenter un atelier open shop avec contraintes de ressources avec multi-compétences, et qui cherchent à minimiser le temps total de séjour des tâches dans l'atelier. Un modèle non-linéaire a été formulé et résolu en utilisant le logiciel LINGO 14. Deux modèles de programmation linéaire en nombres entiers ont été proposés et résolus à l'aide de CPLEX 12.5. Un premier modèle a considéré la discrétisation du temps pour faire l'affectation des tâches et ressources humaines aux machines, alors que le deuxième modèle est basé sur les relations de précédence des tâches et des ressources pour trouver la meilleure séquence de production.

L'ensemble des résultats a été également analysé afin d'identifier la méthode de formulation mathématique la plus performante et afin d'établir les limites des modèles développés pour cette problématique en terme de temps de calcul, en sachant que nous traitons un problème opérationnel qui demande la prise de décision dans le délai le plus court possible.

Selon les résultats obtenus, le modèle non-linéaire trouve 252/1053 solutions optimales, c'est-à-dire, 23,93% des solutions sur les différents scénarios (lancé sur LINGO). Ensuite, le premier modèle linéaire exposé qui utilise la discrétisation du temps résout 127/1053 instances de manière optimale, 12,06% du total des problèmes générés (Lancé sur CPLEX). Comme troisième modèle, nous avons le modèle linaire basé sur les relations de précédence de tâches qui a trouvé la solution optimale de 463/1053 instances, équivalent à 43,97% des problèmes (lancé sur LINGO) et 862/1053 solutions optimales, autrement dit il obtient l'optimum dans 81,86% des instances théoriques générées (lancé sur CPLEX). Concernant le temps de calcul, le modèle linéaire avec la discrétisation du temps exécuté sur CPLEX a présenté la performance la plus mauvaise, le modèle non-linéaire lancé sur LINGO a été le deuxième moins performant et finalement, le modèle linéaire de précédence a été le meilleur en sachant que CPLEX a montré les meilleurs rapports entre solutions optimales et temps de calcul.

Ces résultats et pourcentages permettent de conclure que la meilleure méthode de résolution en terme de performance et temps de calcul, pour ce problème avec contraintes d'affectation des ressources humaines est le modèle mathématique linéaire qui utilise les relations de précédence. Par rapport au logiciel d'optimisation qui donne la meilleure relation optimum-temps de calcul, CPLEX a une meilleure performance dans toutes les comparaisons faites, en réduisant jusqu'en 547 fois le temps de calcul selon le type de scénario résolu.

A partir du modèle présentant la meilleure performance (modèle linéaire testé sur CPLEX) nous avons présenté un quatrième modèle qui prend en compte les contraintes

liées aux outils, c'est-à-dire, sa disponibilité, sa quantité et les opérations qui les demandent. Les résultats montrent que le modèle obtient 807/1053 solutions optimales, 76,63% des optimums des différentes instances.

Dans le chapitre suivant, nous allons aborder les méthodes approchées qui seront destinées à résoudre des instances du cas réel industriel, en particulier les plus grandes.

Chapitre 4

Ordonnancement d'Open shop Mono-objectif : méthodes approchées

Dans le chapitre précédent, nous avons présenté le problème traité dans ce mémoire : un problème d'open shop avec contraintes de ressources humaines prenant en compte l'affectation du personnel avec multi-compétences ainsi que celle de l'outillage, c'est à dire, les outils nécessaires pour exécuter une opération. Nous avons proposé quatre modèles mathématiques pour décrire le problème et nous avons exposé les résultats obtenus par chacun. Tout d'abord, nous savons que la résolution par méthodes exactes présente des limitations par rapport à la taille des instances en termes de temps de calcul. C'est pour cette raison que dans ce chapitre nous abordons les méthodes approchées comme une alternative pour générer des solutions pour résoudre notre problématique, lesquelles sont obtenues dans un temps de calcul raisonnable et de bonnes qualités.

Nous présentons ici les méthodes développées afin de résoudre ce problème. Dans un premier temps, nous testons les méthodes sur les petites instances qui ont été générées dans le chapitre précédent afin de comparer les résultats avec la solution optimale trouvée par le modèle mathématique proposé (MILP résolu sur Cplex) et ainsi déterminer la performance de la méthode. Ensuite, nous réalisons des tests pour les instances de tailles plus grandes pour déterminer quelle méthode fournit les solutions de meilleure qualité.

De la même manière, nous testons les méthodes en ajoutant les contraintes liées à l'outillage (comparaison des résultats avec le modèle de la section 3.2.2) et nous exposons les résultats obtenus avec le lancement des instances de grandes tailles.

4.1 Introduction

Dans le cadre de notre recherche, nous nous sommes intéressés à l'étude d'une problématique d'ordonnancement, qui comme nous l'avons signalé tout au long de ce manuscrit est un sujet qui joue un rôle important dans les industries de production actuelles. Ce sujet peut donner un avantage potentiel sur les concurrents puisqu'il permet l'optimisation du système de production, en améliorant le planning de fabrication de l'atelier, l'efficacité de la production et le niveau de satisfaction du client.

Comme nous l'avons décrit dans l'introduction du chapitre précédent, la gestion des ressources d'une entreprise est un point clé pour déterminer le niveau d'efficacité et de performance de ses processus. Plus précisément, l'ordonnancement d'un atelier de production, et spécialement d'un open shop, où nous devons trouver des alternatives de solutions pour exécuter les tâches sur les machines (ordre des opérations) dans un intervalle de temps court (décision de type opérationnelle). Même si nous avons formulé le problème mathématiquement dans le chapitre précédent, il nous est impossible de résoudre ce problème sur des instances industrielles en raison de leur trop grande taille où il y a plus de 30 machines à considérer, le nombre de tâches est variable d'une semaine à une autre (minimum 30 par semaine), il y a 19 opérateurs qui ont des compétences différentes pour exécuter une opération ainsi que le nombre d'outils à utiliser change selon les types des activités à réaliser (taille, diamètre, fonction, machine, etc). Pour éviter cet écueil, nous nous intéressons dans la suite de ce chapitre à la résolution approchée de ce problème.

4.2 Méthodes de résolution approchées

Cette section présente les méthodes proposées pour résoudre le problème d'ordonnancement de type open shop avec contraintes de ressources : un algorithme génétique, un algorithme à colonie de fourmis et son hybridation avec la logique floue. Ces méthodes ont été sélectionnées en raison de leur efficacité pour résoudre différents types de problèmes d'ordonnancement de grandes tailles.

Dans la même logique que dans la formulation des modèles mathématiques du chapitre précédent, nous traitons le problème en deux étapes. En premier lieu, les méthodes approchées seront présentées pour résoudre le problème d'open shop avec contraintes de ressources humaines multi-compétences. Ensuite, des modifications sont faites pour adapter les méthodes aux contraintes d'outillage.

Toutes les méthodes sont décrites, puis comparées et discutées. En premier lieu, les parties

en commun sont expliquées, ensuite, les spécificités de chaque méthode d'optimisation sont détaillées.

4.2.1 Représentation d'une solution

La manière de coder une solution est très importante car elle possède une grande influence sur l'efficacité des méthodes d'optimisation. Pour coder chaque solution, un vecteur d'opérations est crée ($N \times M$), c'est-à-dire que chaque élément représente l'affectation d'une tâche à une machine. Ci-dessous, le tableau 4.1 montre la relation entre 3 tâches et 3 machines : l'opération 1 du vecteur fait référence à la tâche 1 traitée sur la machine 1 ; l'opération 2 à la tâche 1 exécutée sur la machine 2 et ainsi de suite jusqu'à l'opération 9 qui signifie que la tâche 3 est effectuée sur la machine 3.

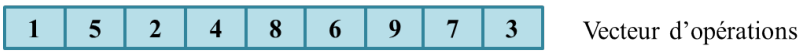


FIGURE 4.1 – Représentation d'une solution

TABLE 4.1 – Représentation de la combinaison tâche-machine

Opération	1	2	3	4	5	6	7	8	9
Tâche	1	1	1	2	2	2	3	3	3
Machine	1	2	3	1	2	3	1	2	3

4.2.2 Évaluation

Pour calculer le fitness (Z), qui est ici la minimisation du temps total de séjour ($\sum F_i$), nous devons calculer la date de fin de chaque tâche d'après la séquence de production proposée par la méthode et lui soustraire sa date de disponibilité afin d'obtenir le temps total pendant lequel la tâche est restée dans le système de production. Cette valeur détermine la qualité d'une solution qui est évaluée d'après l'affectation des ressources humaines et d'outillage.

4.2.3 Critères d'arrêt

Pour les méthodes proposées, nous avons choisi la limite de temps de calcul comme le premier critère d'arrêt pour les instances de petites tailles afin de comparer les résultats avec

le modèle. Un deuxième critère est utilisé pour les instances de grandes tailles, le nombre de générations/itérations selon la méthode.

4.2.4 Contraintes de ressources humaines

Comme nous l'avons déjà mentionné, nous allons exposer les méthodes qui ont été développées pour résoudre notre problématique en considérant les contraintes de ressources liées à l'affectation de ressources humaines. Nous présentons un algorithme génétique, un algorithme à colonie de fourmis puis ce dernier hybridé avec la logique floue.

4.2.4.1 Algorithme génétique (AG)

L'algorithme génétique est une métaheuristique qui a été très utilisée dans la résolution de divers sujets d'optimisation et notamment dans l'ordonnancement, Jawahar *et al.* [86], Tsai *et al.* [171], Demir *et al.* [50], Khuri *et al.* [92], Doulabi *et al.* [55] sont des exemples d'auteurs qui ont proposé des méthodes de résolution en employant les algorithmes génétiques. Cette méthode est basée sur la théorie de l'évolution de Darwin, où les meilleurs individus vont survivre. Dans différents champs d'application, elle a prouvé son efficacité sur de nombreux problèmes, c'est pourquoi nous nous y sommes intéressés. La particularité de cette méthode permet l'interaction entre individus d'une population afin d'améliorer la qualité de ses codes génétiques et proposer des solutions intéressantes d'un point de vue de la problématique en question.

Dans cette partie, une méthode basée sur un algorithme génétique est proposée pour obtenir des solutions du problème d'open shop. La structure générale de la méthode est exposée dans l'algorithme 1 et présentée ci-dessous :

Représentation

Comme [126] l'a exposé dans son travail, une bonne représentation du problème est nécessaire pour assurer une illustration appropriée de sa solution. Des auteurs proposent deux aspects concerné par tous les types de problème d'open shop : le premier, la détermination de la route pour chaque tâche et le deuxième, la séquence de tâches pour chaque machine.

A partir de ce principe, nous exposons la représentation du chromosome de l'algorithme génétique proposé comme un ordonnancement qui montre les combinaisons tâche-machine, où chaque chromosome est composé des opérations qui représentent les gènes. Dans cette

représentation, les opérations sont affichées suivant leur ordre d'exécution ; l'algorithme génétique a pour objectif de trouver le meilleur ordonnancement des opérations afin d'obtenir le chromosome avec la meilleure fonction fitness. Un exemple de quatre tâches et trois machines est proposé.

Le tableau 4.2 décrit l'ensemble des opérations considérées ici. Le chromosome peut alors être formé : $[12-10-1-3-4-6-8-5-7-2-9-11]$. Sachant que nous sommes dans le cas d'un open shop, il n'y a pas de relation de précédence entre les opérations. Ci-dessous la représentation du chromosome créé :

$$\begin{aligned} &Machine\ 1 : tâche\ 4 \rightarrow tâche\ 1 \rightarrow tâche\ 2 \rightarrow tâche\ 3 \\ &Machine\ 2 : tâche\ 3 \rightarrow tâche\ 2 \rightarrow tâche\ 1 \rightarrow tâche\ 4 \\ &Machine\ 3 : tâche\ 4 \rightarrow tâche\ 1 \rightarrow tâche\ 2 \rightarrow tâche\ 3 \end{aligned}$$

TABLE 4.2 – Représentation de la combinaison tâche-machine

Opération	1	2	3	4	5	6	7	8	9	10	11	12
Tâche	1	1	1	2	2	2	3	3	3	4	4	4
Machine	1	2	3	1	2	3	1	2	3	1	2	3
Date de disponibilité	3	3	3	2	2	2	0	0	0	1	1	1

Population initiale

La population initiale est générée de façon aléatoire selon le nombre de chromosomes défini. Puis, l'affectation de ressources est faite en partant des plus spécialisées, qui limite le nombre minimal des opérations au moment de débuter la construction d'un ordonnancement. Les ressources qui restent sont affectées selon les compétences requises pour exécuter l'activité et les compétences maîtrisées par chaque opérateur. Au moins une activité doit être exécutée par chaque opérateur (algorithme 2). Nous reprenons la solution exposée précédemment $[12-10-1-3-4-6-8-5-7-2-9-11]$ et nous présentons des exemples de tableaux de compétences (4.3 et 4.4) qui vont permettre de faire l'affectation des opérateurs. A la fin de cette étape nous aurons un chromosome qui sera composé pour différentes opérations à réaliser et chacune sera liée avec une ressource (opérateur - figure 4.2).

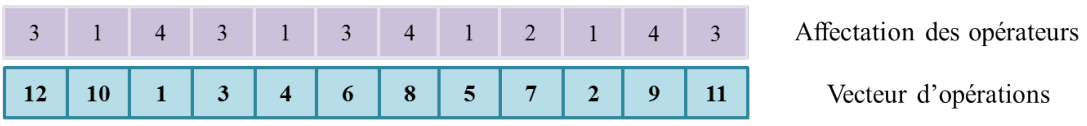


FIGURE 4.2 – Individu de la population initiale

# Opérations	Sk_1	Sk_2	Sk_3
1	1	0	0
2	0	1	0
3	1	0	0
4	1	0	0
5	0	1	0
6	0	1	0
7	0	0	1
8	1	0	0
9	1	0	0
10	0	1	0
11	0	1	0
12	0	0	1

TABLE 4.3 – Compétences demandées pour exécuter l'opération

Opérateurs	Sk_1	Sk_2	Sk_3
W_1	1	1	0
W_2	0	0	1
W_3	1	1	1
W_4	1	0	0

TABLE 4.4 – Compétences maîtrisées par l'opérateur

Algorithme 1 Algorithme Génétique

1: La population initiale est générée de façon aléatoire (n chromosomes)
2: Affectation de ressources pour exécuter les différentes tâches (*algorithme* 2)
3: **pour** $i = 1$ à *critère d'arrêt* **faire**
4: Calculer la fonction objectif Z pour chaque chromosome généré
5: Générer une nouvelle population (n chromosomes en plus) :
Sélectionner deux parents $P1$ et $P2$ de la population initiale à partir de la méthode de tournoi
Appliquer le croisement entre $P1$ et $P2$ avec une probabilité P_c
Appliquer la mutation sur le chromosome sélectionné avec une probabilité P_m
Calculer la fonction objectif Z pour chaque chromosome de la nouvelle génération
6: Faire le classement des chromosomes selon la fonction objectif
7: Garder les n premiers chromosomes pour la prochaine génération
8: **fin pour**

Algorithme 2 Affectation de ressources (opérateurs)

1: **pour** $i = 1$ à *nombre de chromosomes* **faire**
2: Affecter les ressources pour exécuter une tâche en commençant par les plus spécialisées selon ses compétences demandées et les compétences maîtrisées par les opérateurs (relation entre les matrices RE_{ims} et A_{ws})
3: Vérifier que chaque ressource est affectée pour réaliser au moins une activité
4: **fin pour**

Sélection

Le schéma de sélection détermine quelle solution de la population actuelle sera stockée pour la prochaine génération. Dans notre cas, la sélection est réalisée selon la méthode de tournoi qui implique la confrontation de deux individus de la population initiale qui sont choisis au hasard afin de stocker les individus présentant la meilleure fonction objectif (figure 4.3). La fonction fitness d'un individu est calculée à partir de (3.47).

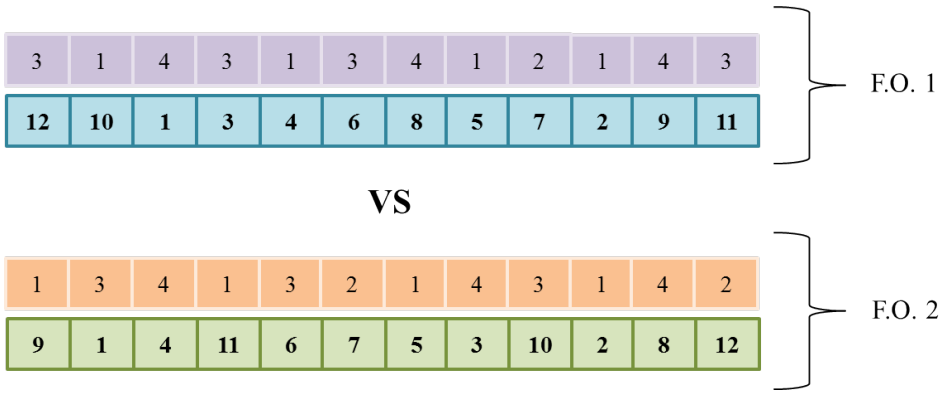


FIGURE 4.3 – Sélection des individus

Croisement

Le croisement prend deux solutions qui sont appelées parents et génère deux nouvelles solutions en les recombinant. Nous supposons que la qualité des enfants (fonction de fitness) sera déterminée par la qualité des parents. Dans cette méthode, nous utilisons un croisement à un point selon l'algorithme 3 :

Algorithme 3 AG : croisement

1: **Étape 1 :** sélectionner un point de coupure au hasard et deux sub-divisions sont générées pour chaque parent (qui est composé par deux vecteurs : séquence des opérations et affectation des opérateurs)
2: **Étape 2 :** échanger les sub-divisions de deux parents (le changement de positions des opérations est fait en gardant l'affectation de l'opérateur qui a été déjà faite)
3: **Étape 3 :** réaliser un processus de correction sur les chromosomes résultants pour éviter la répétition des éléments (opérations) comme il est montré dans la figure 4.4

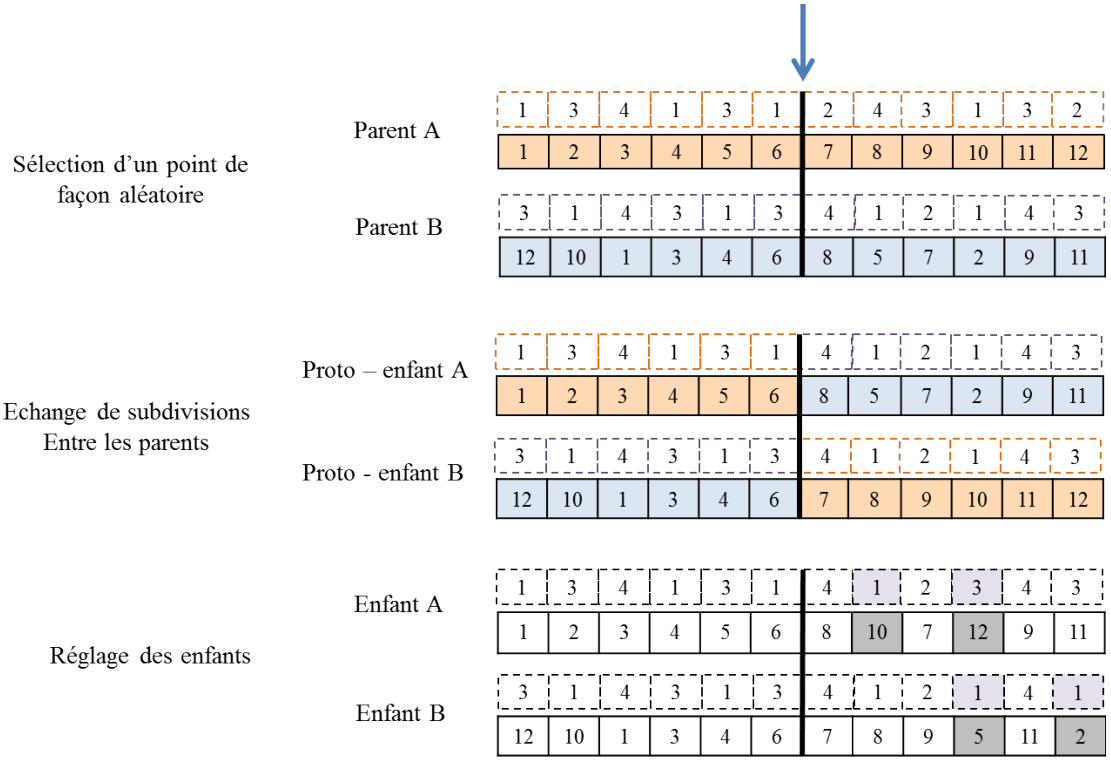


FIGURE 4.4 – Opérateur de croisement

Mutation

La mutation produit de nouvelles solutions à partir d'une modification aléatoire qui est faite sur un parent (chromosome). Ce mécanisme aide à préserver un niveau raisonnable de diversité sur la population ainsi qu'à éviter les optimum locaux. Plusieurs types d'opérateurs de mutation ont été proposés dans la littérature, tels que le déplacement ou l'échange. Nous adoptons la méthode suivante décrite par l'algorithme 4 et illustrée par la figure 4.5.

Algorithme 4 AG : mutation

1: **Étape 1 :** sélectionner deux points de coupure de façon aléatoire, donc le chromosome est divisé en trois sous-vecteurs appelés sv#1, sv#2 and sv#3
2: **Étape 2 :** échanger sv#1 et sv#3

Mise à jour de la population

Après l'utilisation des différents opérateurs génétiques et le calcul des fonctions objectifs, il faut déterminer les individus qui seront gardés d'une génération à une autre. Nous

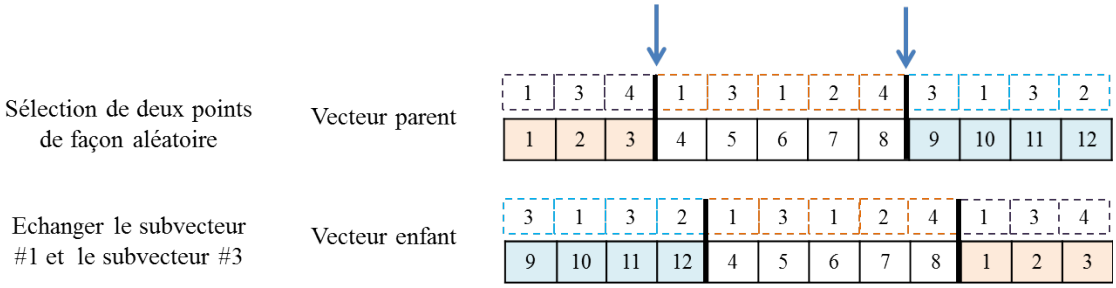


FIGURE 4.5 – Opérateur de mutation

avons parlé d'une population initiale, puis nous avons généré des enfants en employant les procédures de sélection, croisement et mutation exposées (nous avons doublé la taille de la population initiale pendant ces étapes - $2n$). Cette population finale doit être classée selon la qualité de la fonction objectif, de la meilleure jusqu'à la plus mauvaise afin de garder un nombre constant d'individus dans chaque génération (taille de notre population : n).

4.2.4.2 Algorithme à colonie de fourmis (ACO)

Cette méthode résout le problème en imitant le comportement des fourmis pour chercher leur nourriture dans la nature. Dans la littérature nous pouvons trouver un état de l'art qui expose les applications de la méthode à colonie de fourmis sur les problèmes d'ordonnancement [140]. Nous proposons un algorithme à colonie de fourmis comme une autre alternative pour résoudre le problème en question. L'ACO développé présente les mécanismes conventionnels améliorés par des changements relatifs à la problématique qui augmentent l'efficacité de la méthode.

Construction d'une solution

Pour construire une solution, nous partons du même vecteur des opérations déjà introduit par l'algorithme génétique mais à la différence près que cette méthode ne génère pas les vecteurs d'affectation des opérateurs (ressources humaines) depuis le début. Dans cette partie, nous travaillons avec une méthode en deux parties, c'est-à-dire, que nous avons une première étape où nous faisons la construction d'une séquence d'ordonnancement (vecteur des opérations) et une fois la meilleure configuration trouvée, nous générons des vecteurs d'affectation des opérateurs pour obtenir la meilleure combinaison des deux.

La principale caractéristique de cette méthode est l'utilisation d'un graphe représentant dans ce cas, les opérations et la séquence d'exécution. Cette représentation est utilisée pour construire les différents chemins parcourus par les fourmis (vecteur des opérations), de telle

façon qu'une fourmi laisse une trace de phéromone sur les arcs de ce graphe pour inciter les autres fourmis à privilégier un chemin par rapport à un autre. La figure 4.6 montre un exemple de 5 opérations où le chemin avec la trace de phéromone plus concentrée, correspond à la solution du problème. La procédure complète de la méthode est décrite ci-après.

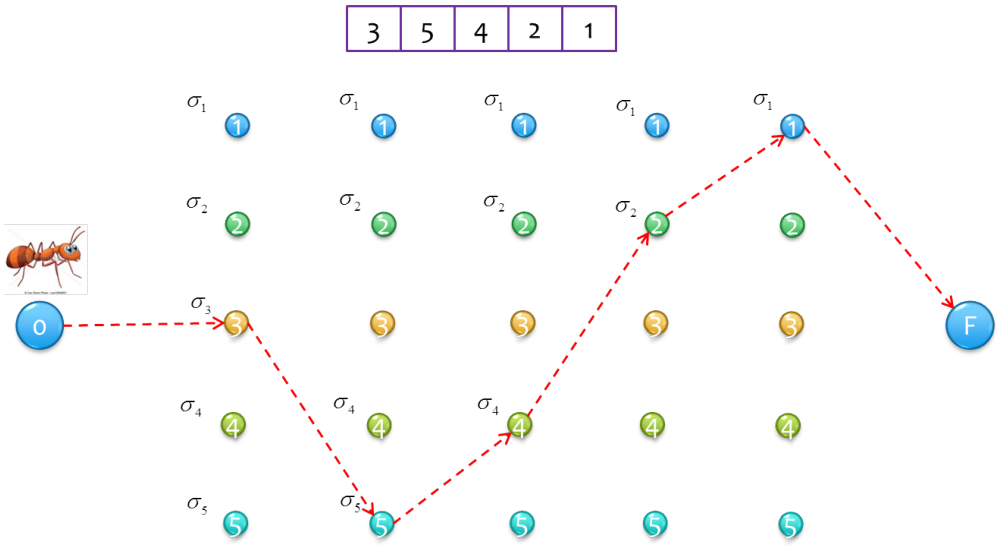


FIGURE 4.6 – Illustration du chemin parcouru par une fourmi

Paramètres d'initialisation

Dans cette partie, nous expliquons comment nous avons fait pour créer les parcours des fourmis pour chaque itération.

D'après les tests préliminaires, nous décidons de générer les parcours des fourmis de façon aléatoire pour 10% des itérations comme un mécanisme de diversification pour constituer des solutions de différentes qualités et explorer des chemins qui probablement ne seraient pas visités après avoir atteint un minimum local. Nous avons un ensemble d'opérations disponibles à visiter par une fourmi. Chaque fois, une opération est sélectionnée aléatoirement et placée comme le prochain nœud à visiter et l'ensemble des opérations disponibles est réduit. Cette procédure continue jusqu'à placer toutes les opérations sous forme de vecteur (similaire au vecteur des opérations exposées pour l'algorithme génétique). Les itérations qui restent sont générées selon la procédure de l'algorithme 5.

Nous avons besoin d'initialiser tous les paramètres de l'algorithme à colonie de fourmis pour les itérations qui ne seront pas générées de façon aléatoire. Le premier élément est la phéromone qui détermine la trace laissée par une fourmi quand elle parcourt un chemin, laquelle influe sur le choix du chemin qui sera pris pour les prochaines fourmis. Nous avons

Algorithme 5 Algorithme à colonie de fourmis

Initialisation
Définir les paramètres initiaux ($Max\Phi$, α, β, ρ)
La matrice de phéromone initiale $\tau_{u,v} = c$, c est une constante positive
pour $it = 1$ à *critère d'arrêt* **faire**
  si it $\leq 10\% \times$ *critère d'arrêt* **alors**
    Générer les chemins de fourmis de façon aléatoire
  sinon
    Sélectionner le premier nœud à visiter par la fourmi ϕ selon l'équation (4.1)
    pour $\phi = 1$ à $Max\Phi$ **faire**
      pour *opérations*$= 1$ à *nombre des opérations* **faire**
        Calculer $Z_{partiel}$
        Calculer $\eta_{u,v}$ selon l'équation (4.2)
        Calculer σ, équation (4.3)
        Choisir la prochaine opération selon la matrice de transition de probabilité
      fin pour
      Calculer la fonction objectif Z_ϕ pour chaque fourmi
      Mise à jour locale de la phéromone selon l'équation (4.5)
    fin pour
  finsi
  Mise à jour globale de la phéromone, équation (4.6)
  Affectation de ressources (opérateurs) - suivant *l'algorithme* 6
fin pour

défini les valeurs initiales de la phéromone (τ_0) en $c = 0.5$ après plusieurs tests réalisés et comme cela est suggéré par Blum [17] dans sa méthode proposée.

L'autre élément fondamental à initialiser est la matrice de désirabilité ($\eta_{0,v}$), qui décrit la désirabilité de visiter un nœud (opération). Dans ce cas, nous avons appliqué un critère différent pour sélectionner la première opération à exécuter (premier nœud visité par une fourmi). Nous avons décidé de faire ce choix basé sur les temps opératoires et de montage (p_{im}, SE_{im}) et aussi les dates de disponibilité de tâches (r_i) pour chaque fourmi ϕ comme cela est présenté dans l'équation (4.1). Ensuite, nous calculons une fonction objectif partielle ($Z_{partiel}$) afin de calculer la valeur de la désirabilité (η), en utilisant (4.2) et les valeurs de la matrice de probabilité de transition (4.3) pour placer la prochaine opération.

$$\eta_{0,v} = \frac{1}{r_v + SE_v + p_v} \; \forall v \in L_\phi \tag{4.1}$$

L_ϕ est l'ensemble de nœuds potentiellement empruntables par la fourmi ϕ.

Nous résumons ce processus dans l'algorithme 5 où l'idée est de placer chaque fourmi ϕ sur le premier nœud en utilisant (4.1), et après appliquer le critère qui est développé dans les sous-sections suivantes jusqu'à ce que toutes les opérations de l'ensemble L_ϕ soient visitées.

Les autres paramètres (l'influence de la phéromone (α), l'influence de l'information heuristique (β), le facteur d'évaporation (ρ), le nombre de fourmis (ϕ) et le nombre d'itérations) ont été établis d'après les différents tests et combinaisons. Dans la section suivante, nous détaillons les valeurs utilisées pour exécuter l'algorithme.

Matrices de désirabilité et de probabilité de transition

D'après la définition des paramètres et le choix de la première opération à réaliser par la fourmi ϕ, nous continuons à décrire la fonction objectif partielle qui sera très utile pour déterminer les prochaines valeurs des deux matrices mentionnées dans cette partie.

Nous supposons un ensemble d'opérations $L_\phi = [O_{i1}, O_{i2}, O_{h1}, O_{h2}, O_{j1}, O_{j2}, ...]$ qui sont disponibles et peuvent être exécutées par une fourmi ϕ. Dans la figure 4.7, nous présentons un exemple d'ordonnancement des trois premières opérations où deux machines (m_1 et m_2) sont disponibles à la date 0 et où les opérations sont disponibles à leur date de début au plus tôt (r_i, r_j, r_h). La figure 4.7 (a) montre la première opération sélectionnée (O_{i2}). En employant la fonction objectif partielle nous pouvons calculer la date de fin temporaire de (O_{i2}) de la tâche *i2*, notée (C_{i2}) et ensuite nous proposons le critère pour calculer les valeurs de la matrice η pour la prochaine opération.

A cette étape, nous avons placé une opération de la tâche i sur la machine *2* (Figure 4.7 (a)) et maintenant nous devons choisir quelle opération sera exécutée ensuite. Ainsi, nous disposons de plusieurs opérations qui appartiennent aux différentes tâches j et h où la première est disponible plus tôt (date de disponibilité plus petite : $r_j < r_h$). Et on prend en compte également le fait que la valeur de r_j est plus petite que la valeur de C_{i2} qui a été calculée à l'étape précédente. De plus, r_h est plus grande que C_{i2}. A partir de cette information et en rappelant que l'intérêt de cet ordonnancement est de minimiser le temps total de séjour ($\sum(C_i - ri)$), dans le problème exposé pour la majorité de cas, il est plus intéressant de commencer les activités qui sont disponibles le plus tôt possible afin d'éviter l'augmentation de la fonction objectif (rechercher des ordonnancements actifs).

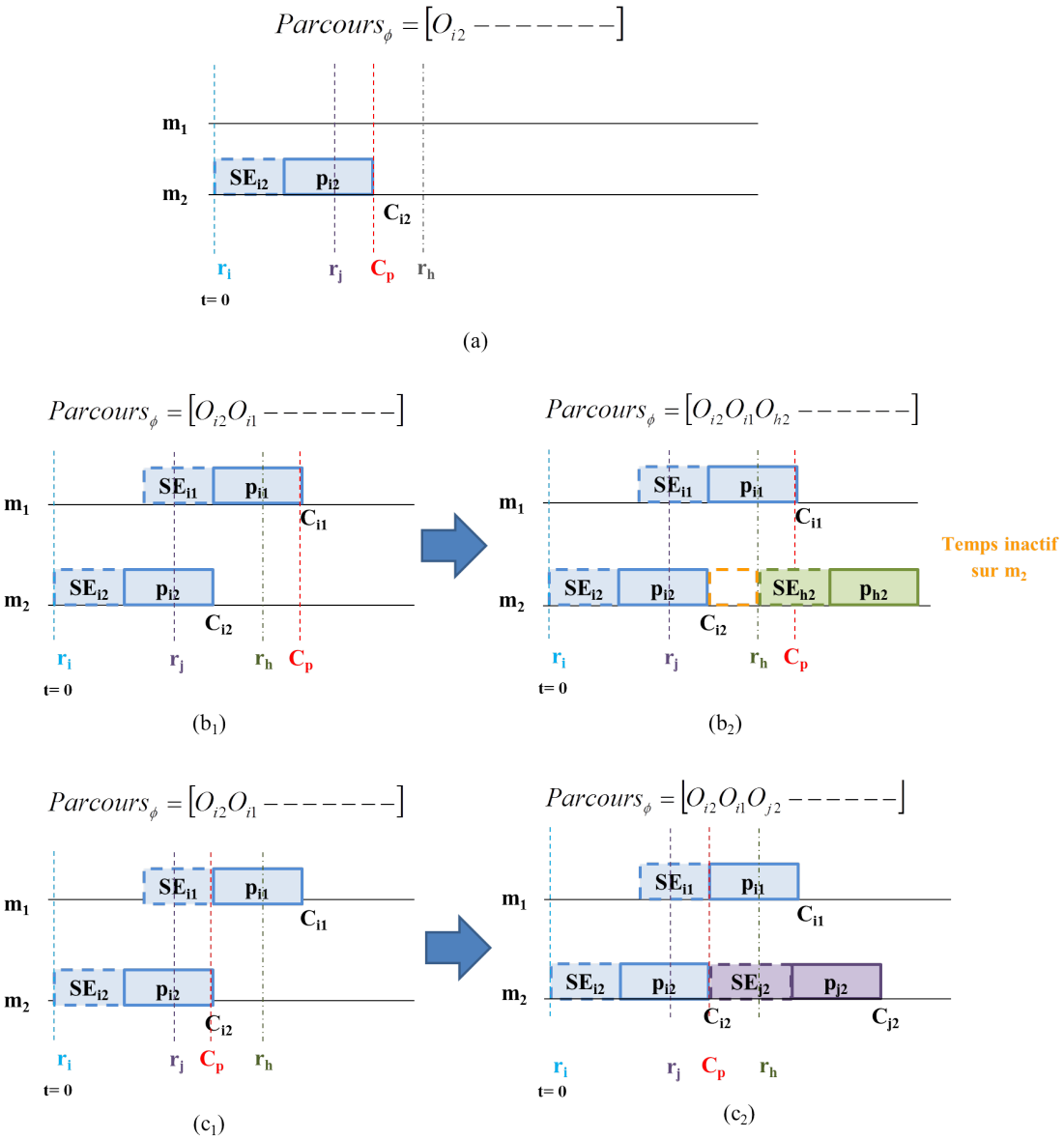


FIGURE 4.7 – Calcul de la fonction objectif partielle

Partant de ce principe, nous devons placer une opération de la tâche i ou j. Nous choisissons O_{i1} (Figure 4.7 (b_1)) et en utilisant la fonction objectif partielle, nous calculons la date de fin de la tâche (C_{i1}). A cette étape, nous avons deux points de référence : les dates de fin de C_{i2} et C_{i1} ; par conséquent nous devons choisir un point de référence, nommé C_p, qui sera le facteur clé pour placer l'opération suivante. De cette façon, nous avons deux options pour choisir la valeur de C_p.

Dans la Figure 4.7 (b_1), nous avons choisi la valeur de C_{i1} comme le nouveau C_p. A partir de ce point, nous vérifions que toutes les tâches sont déjà disponibles (r_i, r_j et r_h sont inférieures à C_{i1}), donc si on place O_{h2} (Figure 4.7 (b_2)), nous générons un temps inactif sur la machine 2 (m_2) puisque nous devons attendre jusqu'à ce que la tâche h soit disponible

pour la commencer.

Comme deuxième option, dans la Figure 4.7 (c_1)) nous avons défini C_{i2} comme la valeur de notre C_p. De cette manière, nous vérifions que à ce moment, l'unique tâche disponible est j (les opérations de la tâche i sont déjà placées et r_j est inférieure à C_{i2}). Nous choisissons O_{j2} (Figure 4.7 (c_2)) et on peut constater que le temps inactif a disparu.

A partir de cette analyse, nous sélectionnons la date de fin, la plus petite calculée jusqu'à cet instant comme la valeur de référence ($C_p = \underset{i\in J, m\in L}{Min}\ C_{im}$). Dans notre exemple, à ce moment donné $C_p = C_{i2}$, et nous continuons avec l'opération suivante.

Nous avons placé l'opération O_{j2}, donc il faut que nous établissions la nouvelle valeur de C_p. Comme énoncé précédemment, nous prenons la date de fin, la plus petite jusqu'à ce moment, c'est-à-dire, la nouvelle valeur de C_p sera C_{i2} dans le but de poursuivre avec l'attribution des opérations restantes.

Selon cette explication, nous proposons le calcul suivant pour la désirabilité.

- **Matrice de désirabilité**

  La désirabilité ($\eta_{u,v}$) est connue comme l'information heuristique du problème. Cette valeur est utilisée pour estimer la désirabilité, c'est-à-dire, le bénéfice d'aller de l'opération u vers l'opération v. Elle peut être déterminée de différentes manières selon le problème traité. Dans notre cas, en considérant l'impact que peut être généré par les paramètres du modèle, et plus particulièrement par les dates de disponibilité des tâches ; nous présentons la désirabilité dans l'équation suivante :

$$\eta_{uv} = \begin{cases} \frac{1}{(SE_v+p_v)} & \forall v \in L_\phi \qquad Si\ r_v - C_p \leq 0 \\ \\ \frac{1}{(1+r_v-C_p)(SE_v+p_v)} & \forall v \in L_\phi, \forall u \in \Omega \quad Sinon \end{cases} \tag{4.2}$$

  Où C_p est la valeur de la date de fin (C_{im}), la plus petite jusqu'à ce moment (nœud) que nous avons déjà calculé avec la fonction objectif partielle (expliquée dans la sous-section précédente). A partir de l'équation (4.2), nous observons que lorsque la date de disponibilité de la tâche de la prochaine opération est plus grande que la date de fin de l'opération précédente, la désirabilité de cette opération diminue. Dans le cas contraire, si la date de fin est plus petite, la désirabilité augmente.
  Pour résumer, nous considérons une règle de priorité d'ordonnancement, une modification du délai de fabrication le plus court (SPT - shortest processing time) pour fournir plus de désirabilité aux opérations qui ont des temps opératoires et de montage petits,

ainsi que celles qui ont aussi une date de disponibilité petite.

- **Matrice de probabilité de transition**

Pendant qu'une solution faisable est construite, chaque fourmi ϕ doit sélectionner une opération qui sera exécutée parmi un ensemble d'opérations candidates L_ϕ aussi bien que chaque fois qu'une opération est choisie, elle est ajoutée dans une liste taboue afin d'éviter la répétition des activités.
La sélection d'une opération depuis l'ensemble L_ϕ (nombre de nœuds restants que la fourmi ϕ doit encore visiter) est faite en utilisant la règle de transition d'états. Cette règle montre la probabilité avec laquelle une fourmi ϕ fait le choix d'une opération à partir de la trace de phéromone τ et la désirabilité η :

$$\sigma = \begin{cases} arg\ max_{v \in L_\phi}\{[\tau_{u,v}]^\alpha \times [\eta_{u,v}]^\beta\} & q \leq q_0 \\ \kappa \qquad Sinon \end{cases} \tag{4.3}$$

Où κ est une variable aléatoire qui est choisie en employant la distribution de probabilité suivante :

$$P_\phi(\kappa) = \frac{[\tau_{u,v}]^\alpha \times [\eta_{u,v}]^\beta}{\sum_{v \in L_\phi}[\tau_{u,v}]^\alpha \times [\eta_{u,v}]^\beta} \tag{4.4}$$

α et β sont des valeurs constantes qui indiquent l'importance relative de τ (phéromone) et η (désirabilité), respectivement. Alors, une grande valeur de α signifie que la trace de la phéromone est prépondérante et les fourmis vont prendre les chemins qui ont déjà été parcourus par d'autres fourmis. Inversement, si la valeur de β est élevée, les fourmis vont choisir le nœud plus proche (les opérations avec les temps opératoires, de montage et les dates de disponibilité plus petites)
Ensuite, q, $0 \leq q \leq 1$, est une valeur qui est générée par une distribution de probabilité uniforme et q_0, $0 \leq q_0 \leq 1$, est un paramètre de réglage qui définit l'importance relative entre l'intensification et la diversification dans la méthode. De cette façon, si $q > q_0$, le système va prendre en compte la diversification, donc de nouvelles solutions seront explorées dans l'espace de recherche. Dans le cas contraire, si $q < q_0$, le système applique l'intensification, ce qui signifie que l'algorithme va tirer avantage de l'information qui a été collectée par les itérations précédentes.
Finalement, la règle d'état de transition de (4.3) et (4.4) est appelée règle de proportion

pseudo-aléatoire.

Mise à jour de la phéromone

Dans cette partie, nous décrivons les deux mécanismes proposés pour mettre à jour la phéromone de notre algorithme.

- **Mise à jour locale de la phéromone**
  Cette mise à jour locale cherche à étudier l'influence d'une fourmi sur la route qui sera sélectionnée par les autres, cette règle est donc appliquée après la construction d'une solution complète. Ce critère utilise un mécanisme d'évaporation de phéromone de l'opération sélectionnée par la fourmi ϕ. Cette procédure est faite pour diversifier les chemins qui seront parcourus par les différentes fourmis et s'échapper de la convergence vers un optimum local. Donc, cette mise à jour modifie la phéromone des opérations sélectionnées selon l'expression suivante :

$$\tau_{u,v}(it) = (1-\rho)\tau_{u,v}(it) + \rho \times \tau_0 \qquad \forall v \in L_\phi \tag{4.5}$$

  Où τ_0 est la valeur initiale de la phéromone et ρ $(0 < \rho < 1)$ est le paramètre d'évaporation qui représente le pourcentage de la trace de phéromone qui va disparaitre pour la construction de la prochaine route à parcourir par les fourmis.

- **Mise à jour globale de la phéromone**

  Une fois que toutes les fourmis ont fini de parcourir leurs chemins, une mise à jour globale est appliquée pour augmenter la probabilité de suivre une route qui a généré une bonne solution, c'est-à-dire, que pour la prochaine itération la probabilité de la choisir sera plus élevée.
  Dans cette partie, nous appliquons la technique proposée par Bullnheimer *et al.* [27], où l'idée est de sélectionner un certain nombre de fourmis W_f qui va fournir les valeurs de la phéromone pour l'itération suivante. Tout d'abord, ces W_f fourmis doivent être classées dans l'ordre décroissant, de la meilleure fonction objectif jusqu'à la pire, où la meilleure sera la fourmi W_f. Dans ce cadre, la mise à jour globale de la phéromone sera déterminée par la contribution de ces fourmis, en sachant que la meilleure fournira plus de phéromone. L'expression pour calculer cette valeur dépend de :

$$\tau_{u,v}(it+1) = (1-\rho)\tau_{u,v}(it) + \sum_{r=1}^{W_f-1}(W_f - r)\Delta\tau^r_{u,v} + W_f\Delta\tau^{bs}_{u,v} \qquad \forall v \in L_\phi \tag{4.6}$$

$$\Delta\tau^r_{uv} = \begin{cases} \frac{1}{Z_r} & Si\ la\ r^{ème}\ fourmi\ a\ parcouru\ l'arc(u,v) \\ 0 & Sinon \end{cases} \tag{4.7}$$

$$\Delta\tau^{bs}_{uv} = \begin{cases} \frac{1}{Z_{bs}} & Si\ la\ meilleure\ fourmi\ a\ parcouru\ l'arc(u,v) \\ 0 & Sinon \end{cases} \tag{4.8}$$

Où Z_{bs} est la meilleure solution (meilleure fourmi) de l'itération, Z_r est la solution obtenue par la $r^{ème}$ fourmi d'après le classement des W_f fourmis. Les apports de $\Delta\tau^r_{uv}$ et $\Delta\tau^{bs}_{uv}$ à la matrice de phéromone de la prochaine itération ($it+1$) seront plus significatifs pour les routes qui ont montré les meilleures solutions. De cette façon, ces chemins auront une plus grande probabilité d'être repris par une fourmi dans une itération suivante. Cette mise à jour globale considère aussi ρ ($0 < \rho < 1$) qui est le paramètre d'évaporation de la phéromone qui indique le pourcentage de la trace de la phéromone qui va disparaître de l'itération présente pour la construction des prochaines routes que vont suivre les fourmis dans l'itération suivante.

Affectation de ressources humaines

L'affectation de ressources humaines est décrite dans l'algorithme 6. Dans un premier temps, la meilleure fourmi ($Z_{Best_{ant}}$) de l'itération est choisie ; ensuite l'affectation de ressources est faite à partir de cette solution d'ordonnancement (nous revenons sur le concept du vecteur d'affectation des opérateurs). L'affectation des ressources commence avec les ressources les plus spécialisées qui limitent le nombre minimal des opérations au moment de débuter la construction d'un ordonnancement. Ensuite, le processus est complété avec les ressources qui manquent selon les compétences demandées par l'opération et les compétences maitrisées par les opérateurs (relation entre les matrices RE_{ims} et A_{ws} - même principe exposé dans l'algorithme génétique). Enfin, une vérification est faite pour garantir que chaque ressource est affectée au moins une fois pour réaliser une activité.

Ce processus d'affectation de ressources est répété pendant un nombre de fois défini. Cette partie-là est aussi définie au début de l'algorithme afin de trouver la meilleure combi-

Algorithme 6 Affectation de ressources (opérateurs)

1: Sélectionner la $Z_{Best_{ant}}$ de l'itération
2: **pour** $i = 1$ à *nombre d'affectations* **faire**
3: Affecter une ressource pour exécuter une activité selon le critère établi
4: Vérifier que chaque ressource a été affectée au moins une fois pour exécuter une opération
5: Calculer Z
6: **fin pour**
7: Sélectionner la meilleure solution trouvée (Z_{Best})
8: Appliquer la recherche locale (RL)
9: Calculer Z_{rl}
10: **si** $Z_{rl} < Z_{Best}$ **alors**
11: $Z_{Best} \leftarrow Z_{rl}$
12: reprendre la ligne 2
13: **finsi**

naison possible entre la solution obtenue par la fourmi et l'affectation des opérateurs (Z_{Best}). A la fin, la meilleure solution est gardée et à ce moment, une recherche locale est employée comme décrite ci-dessous.

- **Graphe disjonctif**

  La figure 4.8 montre un graphe disjonctif d'un problème d'open shop avec $N = M = 3$. Il est composé par des opérations qui sont représentées comme sommets et les arcs entre deux opérations qui sont immédiatement consécutives avec l'objectif de définir l'ordre des opérations qui doivent être traitées sur la même machine et celles qui correspondent à la même tâche. De cette façon, nous pouvons trouver une combinaison faisable de l'ordre de machines et l'ordre de tâches si le graphe est acyclique.
  Le graphique disjonctif est usuellement employé pour déterminer le chemin le plus large d'une séquence d'ordonnancement qui est appelé le chemin critique (makespan) ; mais dans notre cas, le graphe est utilisé pour améliorer la solution du problème. Nous savons aussi que différents vecteurs d'ordonnancement des opérations peuvent générer le même graphique disjonctif, autrement dit, créer une séquence de production différente avec la même valeur de la fonction objectif. L'idée est ici d'intégrer une recherche locale qui anticipe la redondance de solutions au moment d'explorer un voisinage.

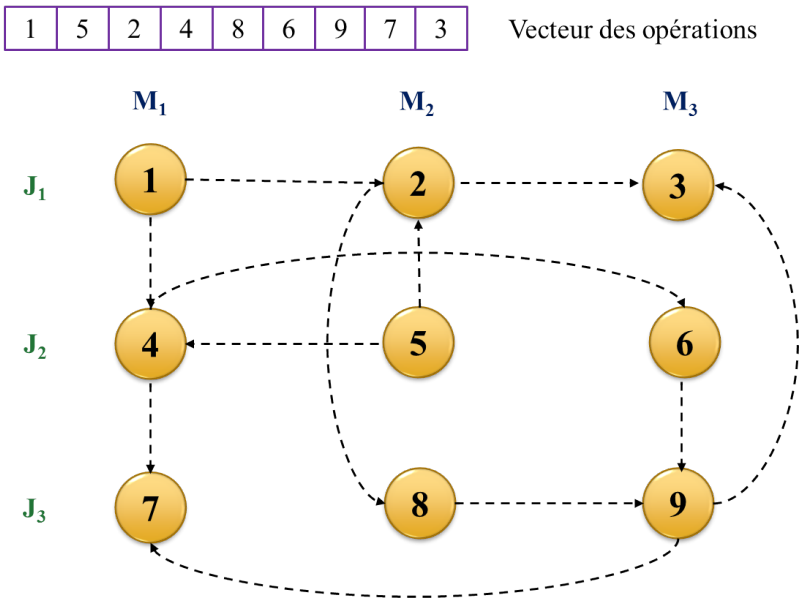


FIGURE 4.8 – Exemple d'un graphe disjonctif

- **Recherche locale**

  La recherche locale est employée pour explorer les différents voisinage du problème étudié.
  D'après la littérature, nous avons trouvé des auteurs comme Dauzère-Pérès *et al.* [45] qui ont proposé des critères à prendre en compte au moment de faire une recherche locale dans un problème d'ordonnancement afin de éviter des solutions redondantes ainsi que des mouvements répétitifs. De la même façon, quelques théorèmes ont été proposés par Naderi *et al.* [136] afin de générer la permutation des opérations et explorer les voisins qui vont créer des solutions différentes. Cela signifie que quelques changements de position peuvent générer exactement la même solution faisable sans prendre en compte un changement d'ordre pour compléter certaines activités. De cette manière et selon les 4 théorèmes qui ont été traités, nous fixons une recherche locale qui évite d'explorer la solution où les opérations qui seront changées appartiennent à des tâches différentes ou utilisent des machines différentes. Nous évitons aussi des échanges entre opérations qui demandent des ressources différentes ou qui ont une plus petite date de début au plus tôt de l'activité que l'opération actuelle.
  Ce critère utilisé dans la recherche locale est basé sur le fait que la permutation entre deux opérations produit des modifications utiles dans la fonction objectif quand la séquence des opérations crée des perturbations dans une machine ou dans la manière dont les autres activités liées avec la tâche en question seront re-ordonnancées pour respecter les différentes contraintes d'ordonnancement. L'application de ce critère sert

à éliminer en grande quantité les solutions redondantes et donc à diminuer le temps de calcul de cette phase.

En conclusion du dernier paragraphe, nous proposons une recherche locale basée sur la méthode de la meilleure insertion qui travaille en deux étapes. En considérant le vecteur avec la meilleure fonction objectif (Z_{best}) qui est le résultat d'une combinaison de l'ordonnancement fait par une fourmi et le vecteur d'affectation généré, nous exposons un vecteur final avec deux composantes comme cela est montré dans la figure 4.9 (nous supposons un problème avec 9 opérations et 3 ressources (opérateurs) où les vérifications pour affecter les différentes ressources pour réaliser chaque tâche ont été faites sur la configuration de la figure). De cette façon, la première phase consiste en l'application de la recherche locale avec la meilleure insertion au vecteur des opérations ordonnancées, en commençant par les éléments de la gauche, donc le décalage des opérations sera fait du côté gauche vers le côté droite. Par conséquent, chaque fois que nous faisons un changement dans la position de l'opération, nous gardons la ressource qui a été affectée, c'est-à-dire, nous allons créer une nouvelle séquence d'opérations mais en gardant l'affectation faite pour chaque ressource.

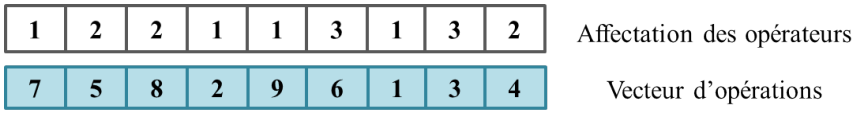


FIGURE 4.9 – Représentation d'un vecteur de solution

La deuxième étape est presque la même, une fois que nous avons fini la recherche locale de la première partie, nous gardons la meilleure solution trouvée et nous continuons avec une autre recherche locale avec meilleure insertion mais en changeant le point de départ. Cette fois, nous commençons le vecteur du côté droit vers la gauche. A la fin du processus, nous récupérons la solution qui sera le vecteur avec la meilleure solution d'ordonnancement ainsi que sa fonction objectif (Zrl). De cette façon, comme est marqué sur l'algorithme 6, si on améliore la solution, on reprend tout le processus d'affectation, sinon on garde la dernière solution obtenue.

4.2.4.3 Algorithme à colonie de fourmis hybridé avec la logique floue

Notre choix d'un algorithme à colonie de fourmis a été fait par rapport à son efficacité connue dans les différents domaines de l'optimisation. Celui-ci a de nombreux avantages telle que la résolution de problèmes de grandes tailles dans des temps d'exécution très courts. Par contre, cet algorithme présente un inconvénient, les paramètres qu'il utilise sont fixes au cours

des générations. Il faut donc choisir les bons paramètres (α, β, entre autres) pour générer des solutions avec une bonne qualité. C'est pour cette raison, que l'approche proposée à cette étape cherche à créer un ajustement dynamique au niveau des paramètres en utilisant la logique floue.

La logique floue a été introduite par Lotfi Zadeh [184]. Elle consiste à prendre en compte les incertitudes sur les états du système. Nous pouvons trouver l'application de la logique floue dans l'optimisation de problèmes d'ordonnancement dans des travaux proposés par Yalaoui *et al.* [181] et Yalaoui *et al.* [183], Dugardin [59] qui ont utilisé des contrôleurs à logique floue pour gérer les réglages de paramètres. A partir des bons résultats obtenus, nous avons décidé d'adapter cette méthode de résolution à notre problème d'ordonnancement de type open shop avec contraintes de ressources humaines, en développant un algorithme à colonie de fourmis hybridé avec la logique floue.

Cette hybridation prend en compte l'algorithme à colonie déjà expliqué mais il considère aussi quelques modifications qui seront présentées ci-dessous (algorithme 7).

Contrôleurs de logique floue

Les contrôleurs de logique floue (FLC) consistent à introduire des perturbations dans les données du problème pour générer un bon réglage de la valeur des paramètres de l'algorithme en prenant une décision à partir d'un tableau.

L'application de la logique de floue dans l'optimisation est composée de trois étapes : la fuzzyfication, la prise de décision et la défuzzyfication. Dans notre travail, les FLC sont utilisés pour définir dynamiquement les paramètres α et β de l'ACO toutes les 10 itérations.

Plusieurs informations d'entrée sont considérées par un FLC (figure 4.10), le *paramètre(it)* expose la valeur de α ou β dans l'itération *it*. Un FLC génère la valeur de $\Delta_{paramètre}(it)$ en utilisant la différence entre les valeurs de la fonction objectif $F.O._{moyen}(it-1)$ et $F.O._{moyen}(it-2)$ qui sont les valeurs moyennes des fonctions objectifs aux itérations *it-1* et *it-2*. Nous avons aussi un autre paramètre *d(it-1)* qui est la somme des distances Hamming (équation 4.9) entre les fourmis à la génération *it* - expliqué dans [107].

$$Distance = \frac{1}{\frac{Max\Phi(Max\Phi-1)}{2}} \sum_{i'=1}^{Max\Phi} \sum_{j'=1}^{Max\Phi} \sum_{k'=1}^{O} \frac{(\delta(g_{i',k'}, g_{j',k'})}{O} \tag{4.9}$$

$$\delta(g_{i',k'}, g_{j',k'}) = \begin{cases} 1 & \text{Si } g_{i',k'} = g_{j',k'} \\ 0 & \text{Sinon} \end{cases}$$

Algorithme 7 Algorithme à colonie de fourmis hybridé avec logique floue

Définir les paramètres initiaux ($Max\Phi$, $\alpha, \beta, \rho, \tau$)
pour $it = 1$ à *critère d'arrêt* **faire**
  si $it = 10$ **alors**
    $floue\_compteur = 1$
  finsi
  si $it = 10 \times floue\_compteur$ **alors**
    Appliquer les contrôleurs de logique floue
    $floue\_compteur = floue\_compteur + 1$
  finsi
  si it $\leq 10\% \times$ *critère d'arrêt* **alors**
    Générer les chemins à parcourir par les fourmis de façon aléatoire
  sinon
    Sélectionner le premier nœud à visiter par la fourmi ϕ
    pour $\phi = 1$ à $Max\Phi$ **faire**
      pour *opérations* $= 1$ à *nombre des opérations* **faire**
        Calculer $Z_{partiel}$
        Calculer $\eta_{u,v}$ (désirabilité)
        Calculer σ (probabilité)
        Choisir la prochaine opération selon la matrice de probabilité
      fin pour
      Calculer la fonction objectif Z_ϕ pour chaque fourmi
      Mise à jour locale de la phéromone
    fin pour
  finsi
  Mise à jour globale de la phéromone
  Affectation de ressources (opérateurs) *algorithme* 6
fin pour

Où $Max\Phi$ est le nombre de fourmis, O est le nombre des opérations et $g_{i',k'}$ ($g_{j',k'}$) est la valeur du $k'^{ème}$ nœud visité par la i'^{me} ($j'^{ème}$) fourmi. Ce calcul de la distance Hamming permet d'estimer la similarité des chemins générés par les fourmis afin de favoriser la diversité et éviter la construction de routes déjà parcourues.

Dans la première étape, le fuzzyfication, les valeurs d et le $F.O.(it-1) - F.O.(it-2)$ (permet d'estimer la progression de la population - fourmis d'une itération à une autre) sont traitées en employant des fonctions d'appartenance définies dans les figures 4.12 et 4.11.

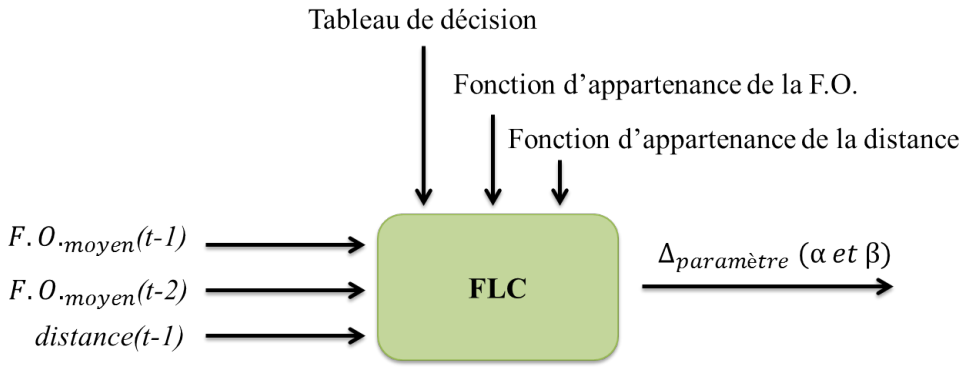


FIGURE 4.10 – Description d'un contrôleur de logique floue [181]

Un coefficient π est établi pour mettre à l'échelle la fonction objectif. Les différents termes linguistiques (NLR...PLR - expliqués dans [181]) décrivent la fonction objectif en lui donnant une valeur spécifique (proportion).

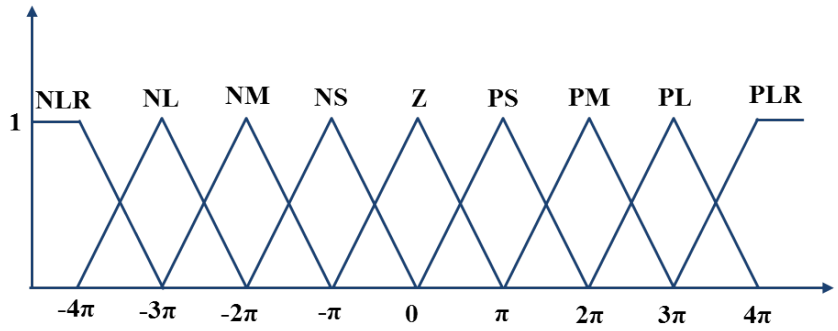


FIGURE 4.11 – Fonction d'appartenance de $F.O.(it-1) - F.O.(it-2)$, $\Delta\alpha$, $\Delta\beta$ [181]

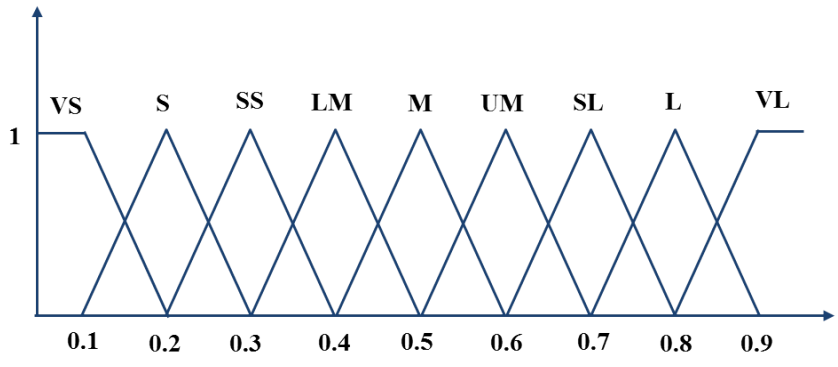


FIGURE 4.12 – Fonction d'appartenance de $d(t-1)$ [181]

La deuxième étape est la prise de décision décrite dans le tableau 4.7 pour $\Delta\alpha$ et le tableau 4.8 pour $\Delta\beta$. Dans ce cadre, différentes combinaisons sont testées en utilisant les valeurs obtenues par la fonction d'appartenance de $F.O.(it-1) - F.O.(it-2)$ et celle de

$d(it-1)$. Par exemple, la fonction d'appartenance de $F.O.(it-1)-F.O.(it-2)$ donne les deux termes du NLR et NL, et celle de $d(it-1)$ donne S et SS. Quatre combinaisons sont alors obtenues, elles seront interprétées de façon différente pour α et β selon les tableaux 4.7 et 4.8, respectivement. De cette manière, si $F.O.(it-1)-F.O.(it-2)$ est NLR et $d(it-1)$ est S alors $\Delta\alpha$ est PS.

TABLE 4.5 – Termes linguistiques de $F.O.(it-1)-F.O.(it-2)$, $[\Delta\alpha]$, $[\Delta\beta]$

Termes linguistiques	**Signification**
NLR	Negative larger
NL	Negative large
NM	Negative medium
NS	Negative small
Z	Zero
PS	Positive small
PM	Positive small
PL	Positive large
PLR	Positive larger

TABLE 4.6 – Termes linguistiques de $d(it-1)$, $[\Delta\alpha]$, $[\Delta\beta]$

Termes linguistiques	**Signification**
VS	Very small
S	Small
SS	Slightly small
LM	Lower medium
M	Medium
UM	Upper medium
SL	Slightly large
L	Large
VL	Very small

La troisième étape ou defuzzyfication commence avec la somme de la valeur maximale de chaque terme linguistique (NLR, NL, ...). Après cela, il faut calculer la valeur de X_g qui est l'abscisse de la zone du centre de gravité créée par les valeurs des différents termes (figure 4.13), à partir de l'équation suivante :

TABLE 4.7 – Tableau de décision $\Delta\alpha$ [181]

$d(t-1)$	$F.O.(t-1) - F.O.(t-2)$								
-	**NLR**	**NL**	**NM**	**NS**	**Z**	**PS**	**PM**	**PL**	**PLR**
VL	PLR	PLR	PL	PL	PM	PM	PS	PS	Z
L	PLR	PL	PL	PM	PM	PS	PS	Z	NS
SL	PL	PL	PM	PM	PS	PS	Z	NS	NS
UM	PL	PM	PM	PS	PS	Z	NS	NS	NM
M	PM	PM	PS	PS	Z	NS	NS	NM	NM
LM	PM	PS	PS	Z	NS	NS	NM	NM	NL
SS	PS	PS	Z	NS	NS	NM	NM	NL	NL
S	PS	Z	NS	NS	NM	NM	NL	NL	NLR
VS	Z	NS	NS	NM	NM	NL	NL	NLR	NLR

TABLE 4.8 – Tableau de décision $\Delta\beta$ [181]

$d(t-1)$	$F.O.(t-1) - F.O.(t-2)$								
-	**NLR**	**NL**	**NM**	**NS**	**Z**	**PS**	**PM**	**PL**	**PLR**
VL	NLR	NLR	NL	NL	NM	NM	NS	NS	Z
L	NLR	NLR	NL	NM	NM	NS	NS	Z	PS
SL	NL	NL	NM	NM	NS	NS	Z	PS	PS
UM	NL	NM	NM	NS	NS	Z	PS	PS	PM
M	NM	NM	NS	NS	Z	PS	PS	PM	PM
LM	NM	NS	NS	Z	PS	PS	PM	PM	PL
SS	NS	NS	Z	PS	PS	PM	PM	PL	PL
S	NS	Z	PS	PS	PM	PM	PL	PL	PLR
VS	Z	PS	PS	PM	PM	PL	PL	PLR	PLR

$$X_g = \int_{-5\pi}^{5\pi} xh(x)dx$$

Où : x est l'axe horizontal de la figure 4.13, $h(x)$ est la valeur maximale de chaque terme linguistique, $X_g = \Delta_{\text{paramètre}}$ ($\Delta\alpha$ pour l'exemple), π le paramètre de mise à l'échelle de la valeur de la F.O.

La même procédure est utilisée pour β en considérant un tableau de prise de décisions avec des modifications.

Pour terminer, la valeur du paramètre considéré à l'itération it est définie par l'expression ci-dessous :

$$paramètre(it) = paramètre(it-1) + \Delta_{paramètre}.$$

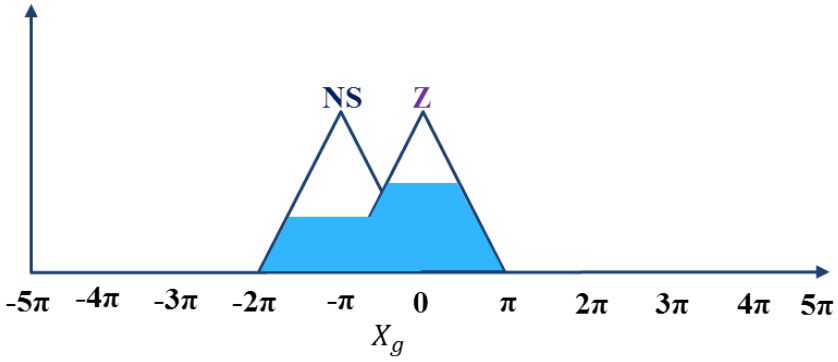


FIGURE 4.13 – Représentation graphique du centre de gravité

Nous présentons un exemple numérique sur le fonctionnement d'un contrôleur de logique floue pour trouver $\Delta\alpha$. Nous supposons l'échelle de la valeur de la F.O. = 100 ($Pi = 100$). Au moment d'appliquer la première étape du processus ou fuzzyfication, nous avons trouvé $F.O.(it-1) - F.O.(it-2) = -230$ qui donne les termes NL = 0.3 et NM = 0.7 et $d(t-1) = 0.37$, alors les termes suivants sont SS = 0.4 et LM = 0.6 (figure 4.14). A partir de ces termes, nous abordons la prise de décisions selon le tableau 4.7 où nous avons quatre combinaisons possibles : $NL(0.3) \otimes SS(0.4) \rightarrow PS(0.3)$, $NL(0.3) \otimes LM(0.6) \rightarrow PS(0.3)$, $NM(0.7) \otimes SS(0.4) \rightarrow Z(0.4)$ et finalement $NM(0.7) \otimes LM(0.6) \rightarrow PS(0.6)$.

La dernière étape, la defuzzyfication garde les valeurs plus grandes de chaque terme, c'est-à-dire, Z(0.4) et PS(0.6) et à partir de ces valeurs il faut calculer la zone du centre de gravité selon la figure 4.15.

Après le calcul de la surface du graphique, nous trouvons que dans cet exemple $X_g = 0.011$ donc $\Delta\alpha = 0.011$.

De cette manière, nous modifions la façon de générer un vecteur solution des opérations en utilisant des paramètres α et β dynamiques (le reste de la procédure reste identique).

4.2.5 Contraintes de ressources humaines et d'outillage

Dans cette sous-section nous prenons en compte une contrainte additionnelle liée à l'affectation des outils selon leurs disponibilités. Tout d'abord, nous allons présenter les éléments qui sont nécessaires pour réaliser l'affectation des outils aux opérations.

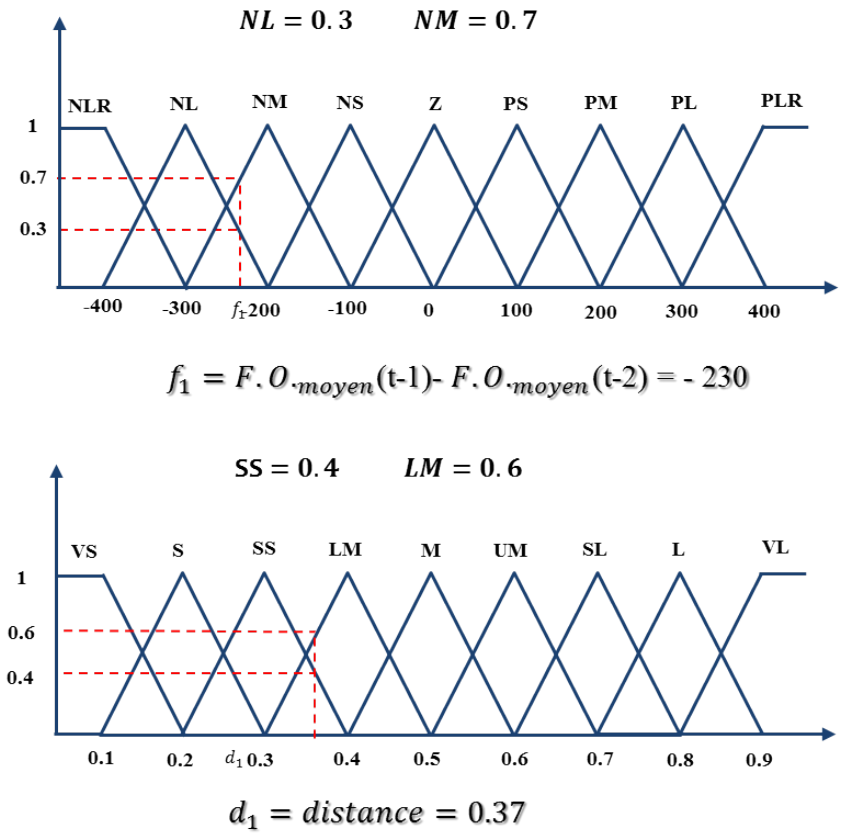


FIGURE 4.14 – Fonctions d'appartenance de $F.O.(it-1) - F.O.(it-2)$ et $d(t-1)$

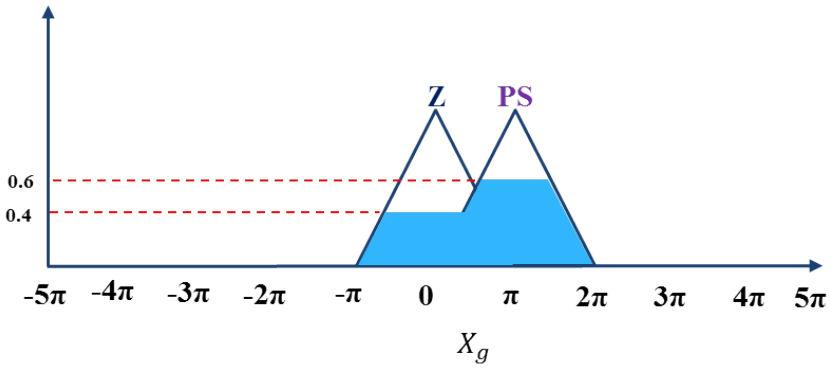


FIGURE 4.15 – Exemple du centre de gravité : termes Z et PS

Comme nous avons déjà fait pour les opérations et les ressources humaines (opérateurs), nous allons construire un vecteur d'affectation des outils qui sera déterminé par une matrice montrant les outils demandés pour exécuter une opération (tableau 4.9) ainsi que par un tableau limitant la quantité des outils disponibles de chaque type (tableau 4.10).

Si nous reprenons notre exemple de 4 tâches et 3 machines (12 opérations) en considérant les données des tableaux liés aux outils (6 types d'outils constitué 5 outils de type 1, 3 de type 2, 2 de type 3, 4 de type 4, 2 de type 5 et finalement 6 de type 6) nous aurons le vecteur

représenté par la figure 4.16.

# Opérations	Outil1	Outil2	Outil3	Outil4	Outil5	Outil6
1	0	0	1	1	1	1
2	1	0	0	0	1	0
3	0	1	1	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	1
6	0	0	1	0	0	0
7	0	1	1	1	0	0
8	1	1	0	0	0	0
9	1	0	1	1	0	1
10	1	1	0	0	0	1
11	0	0	1	1	0	0
12	0	1	0	0	1	1

TABLE 4.9 – Outils demandés pour exécuter l'opération

Type	Qté
Outil1	5
Outil2	3
Outil3	2
Outil4	4
Outil5	2
Outil6	6

TABLE 4.10 – Quantité des outils disponibles

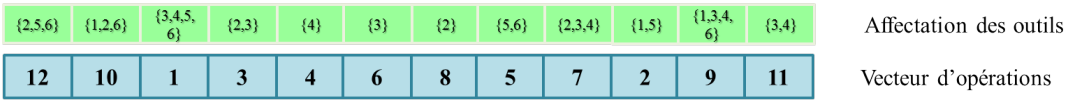


FIGURE 4.16 – Représentation d'un vecteur d'affectation des outils

Dans ce cadre, l'affectation des outils est faite suivant les requêtes de chaque opération. Les outils sont assignés aux machines, s'ils sont disponibles lorsque l'opération doit être traitée. De cette manière, nous générons un premier ordonnancement faisable en termes d'affectation des outils qui après sera amélioré par les différentes méthodes avant d'affecter les ressources humaines. Autrement dit, nous partons d'une solution initiale de la forme du vecteur de la figure 4.16 et ensuite nous appliquons la procédure décrite antérieurement selon la méthode approchée (algorithme génétique, algorithme à colonie de fourmis ou algorithme à colonie de fourmis hybridé avec la logique floue).

Après avoir appliqué la méthode de résolution en question (algorithmes antérieurement expliqués et détaillés) nous construisons une solution à trois vecteurs (séquence des opérations, outils et opérateurs affectés - figure 5.6).

Nous avons présenté dans cette section les différentes méthodes approchées pour résoudre notre problème d'open shop avec contraintes de ressources. Leurs performances sont analysées dans la section suivante.

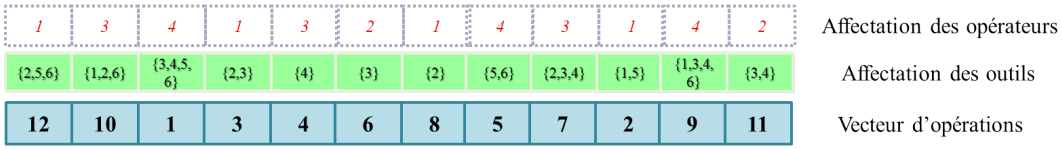


FIGURE 4.17 – Représentation d'une solution d'ordonnancement en considérant des ressources humaines et d'outillage

4.3 Expérimentations

Cette section expose les résultats de l'application numérique des méthodes approchées décrites dans ce chapitre, sur des instances théoriques du problème d'ordonnancement dans un atelier de production de type open shop. Les méthodes proposées ont été testées en employant le logiciel codeblocks 12.11, codées en langage C dans un ordinateur personnel avec CPU CORE i3 2.4 GHz, une mémoire RAM de 4GB et le système opératoire windows 7.

Cette section commence avec une description sur la génération des instances numériques qui ont été testées. Puis les différents tableaux de résultats sont présentées et à la fin, des comparaisons entre les différents algorithmes sont faites.

4.3.1 Génération des instances numériques

Pour la résolution de notre problème, nous avons proposé les tests de différents types d'instances numériques afin de réaliser les comparaisons respectives entre le modèle de formulation mathématique du chapitre précédent et les méthodes approchées proposées. De cette façon, nous avons réalisé deux divisions des instances : celles de petites tailles (qui seront comparées avec la formulation mathématique) et des instances de moyennes et grandes tailles qui seront appliquées dans la résolution de notre problème dans des environnements de production qui sont désignés d'une façon plus proche des cas réels, sachant que la comparaison est à titre indicatif car le lien entre les deux groupes d'instances n'est pas démontré. Cette démarche est souvent adoptée et permet aussi de voir les zones d'instances et leur résolution.

4.3.1.1 Instances de petites tailles

Plusieurs problèmes ont été générés de manière aléatoire pour chaque environnement de production. Dans un premier temps nous traitons les environnements qui incluent les contraintes de ressources humaines (opérateurs). Pour notre test des méthodes approchées,

nous reprenons les instances numériques déjà générées dans le chapitre 3 qui ont été testées dans les modèles mathématiques. Nous rappelons tout de même les conditions suivies au moment de générer les instances :

- Le nombre de tâches pour les instances de petites tailles sont 3, 4, 5, 7 et le nombre de machines 2, 3, 4.
- Les temps opératoires sont des entiers uniformément distribués dans $[1, \omega]$, et les temps de montage sont des variables aléatoires entre $[30, \omega]$, où ω peut prendre trois valeurs (ω_1, ω_2 et ω_3) : 50, 75 ou 100. Nous avons décidé de créer un intervalle variable avec des valeurs différentes afin de tester des instances incluant un comportement largement diversifié.
- La date de disponibilité d'une tâche est définie dans un intervalle aléatoire de distribution uniforme décrit par $[0, \lambda \sum_{m=1}^{M} p_{im}]$; où p_{im} est le temps opératoire d'une tâche i sur la machine m et λ est le facteur de date de disponibilité qui peut prendre la valeur de (λ_1, λ_2 et λ_3) : 0.1, 0.3, ou 0.5.
- Le nombre de ressources humaines (opérateurs) a été déterminé par le nombre de machines (nous supposons une ressource par machine), le nombre de compétences pour exécuter les activités est 3, 5 ou 7 et les matrices qui définissent les compétences demandées par chaque opération et les compétences qui sont maîtrisées par les opérateurs sont générées d'une manière aléatoire (0 ou 1), où 1 implique que l'opération demande cette compétence ou que l'opérateur domine cette compétence.

Au sujet des instances qui prennent en compte les contraintes d'outillage, nous avons récupéré les environnements générés dans le chapitre précédent. Ci-après, nous présentons un résumé de sa génération :

- Le nombre de tâches pour les instances de petites tailles reste identique (3, 4, 5, 7) ainsi que le nombre de machines (2, 3, 4). Le reste des données est généré comme expliqué précédemment.
- Le nombre de types d'outils et le nombre de copies de chaque type d'outils ont été générés de manière aléatoire. De cette manière, le nombre de types d'outils est une valeur dans l'intervalle $[M, (N \times M)]$ alors que le nombre de copies par type d'outils est une valeur aléatoire dans l'intervalle [2, M]. La matrice qui définit le type d'outils demandé par chaque opération est créée de manière aléatoire (0 ou 1), où 1 implique que l'opération requiert ce type d'outils pour être exécutée.

4.3.1.2 Instances de grandes tailles

Également, des environnements de production pour les instances de grandes tailles ont été désignés et testés avec les différentes méthodes proposées. Ci-dessous, nous expliquons les paramètres employés dans la génération des différents types de problèmes :

- Le nombre de tâches pour ces instances sont 10, 30 ou 50 et le nombre de machines 5, 10 ou 15.
- Les temps opératoires et de montage sont générés de la même façon que pour les instances de petites tailles. Ce sont des entiers générés dans des intervalles de distribution uniforme $[1, \omega]$ et $[30, \omega]$, respectivement. Où ω peut prendre trois valeurs (ω_1, ω_2 et ω_3) : 50, 75 ou 100.
- Les dates de disponibilité suivent l'intervalle aléatoire de distribution uniforme décrit par $[0, \lambda \sum_{m=1}^{M} p_{im}]$.
- Le nombre de ressources humaines (opérateurs) reste inchangé, une ressource par machine. Le nombre de compétences pour exécuter les activités est 3, 5 ou 7 et les matrices qui définissent les habilités demandées par chaque opération et les compétences qui sont maîtrisées par les opérateurs sont générées de manière aléatoire (0 ou 1), où 1 implique que l'opération demande cette compétence ou que l'opérateur domine cette compétence.

En prenant en compte les ressources additionnelles, c'est-à-dire, l'outillage ; nous reprenons les même critères déjà exposés dans la génération des instances de petites tailles expliquées dans la sous-section 3.4.1 du chapitre 3.

4.3.2 Résultats expérimentaux

Dans cette sous-section, nous présentons les tableaux de données des différents tests réalisés. Tout d'abord, nous avons deux types de résultats à exposer. Les premiers sont les résultats obtenus par les différentes méthodes approchées au moment de tester les instances de petites tailles et les deuxièmes correspondent aux tests liés avec les instances de grandes tailles.

Instances avec contraintes de ressources humaines

De la même façon que dans la sous-section précédente, nous avons abordé les résultats en deux étapes. Cette première description montre les résultats des instances qui prennent en compte l'affectation de personnel avec multi-compétences, nous avons testé les trois algo-

rithmes exposés : l'algorithme génétique, l'algorithme à colonie de fourmis et l'algorithme à colonie de fourmis hybridé avec la logique floue. La deuxième partie des essais sera présentée plus tard.

- Instances de petites tailles

De la même manière que dans le chapitre 3, la notation pour chaque scénario suit la structure suivante : $N \times M\_\omega - \lambda - N_W - N_S$. Les nombres de tâches et de machines considérées sont les mêmes, N = 3, 4, 5 et 7, M = 2, 3 et 4. Les nombres de ressources humaines (opérateurs), N_W = 2, 3 ou 4 et le nombre de compétences N_S = 3, 5 ou 7. Les valeurs de ω et λ qui sont deux paramètres utilisés pour définir les intervalles dans la génération de temps opératoires et de montage ainsi que dans les dates de disponibilité de tâches, restent égaux. Neuf types de problèmes ont été générés de façon aléatoire pour chaque scénario, dans lequel chaque problème a été lancé dix fois de manière répétitive. Le nombre de tests et de lancements a été choisi pour fournir un large échantillon de différents environnements mais ils ont été limités en taille à cause de temps de calcul. Ainsi, nous avons désigné 117 scénarii différents qui représentent 1053 instances testées (9 problèmes de chaque type de scénario). Dans chaque scénario nous disposons de 9 combinaisons possibles : ω_1 (λ_1, λ_2, λ_3), ω_2 (λ_1, λ_2, λ_3) et ω_3 (λ_1, λ_2, λ_3). Ensuite, nous présentons les tableaux de résultats qui comparent la performance de chaque méthode approchée en relation avec la méthode exacte. Nous présentons 13 ensembles d'instances (81 problèmes à l'intérieur de chacun) qui regroupent les valeurs des paramètres ω et λ après les différentes combinaisons, en utilisant la notation : $N \times M - N_W - N_S$.

Les paramètres pour lancer l'algorithme génétique ont été déterminés à partir de différents tests : la probabilité de croisement 0.8, la probabilité de mutation 0.2. De la même façon, pour l'ACO, $\tau_0 = 0.5$, $\alpha = 1$; $\beta = 0.8$, $\rho = 0.7$ et $W_f = 50\%$ du nombre de fourmis (la quantité de fourmis considérée pour faire la mise à jour de la phéromone). Au sujet de l'algorithme à colonie de fourmis hybridé avec la logique floue, les paramètres ont été les mêmes que pour l'ACO ($\tau_0 = 0.5$, $\alpha_0 = 1$; $\beta_0 = 0.8$, $\rho = 0.7$ et $W_f = 50\%$), en sachant que les contrôleurs de logique floue sont appliqués aux paramètres α et β. Le critère d'arrêt a été déterminé par le temps d'exécution de l'algorithme à 120 secondes.

Les résultats du tableau 4.11 montrent le comportement des algorithmes au moment de la résolution des différents scénarii proposés. Les $\overline{GAP(\%)}$ représentent l'écart entre la meilleure solution connue (Cplex - modèle MILP) et la meilleure solution obtenue par l'algorithme (AG, ACO et ACO-FLC) en pourcentage de la borne supérieure. Le $\overline{\sigma_{GAP}}$ (déviation standard) montre la quantité de variation ou dispersion des différents GAPs calculée à l'intérieur de chaque scénario. Pour l'ACO et l'ACO-FLC le GAP_{min} est la valeur minimale du GAP

TABLE 4.11 – Comparaison de résultats pour les petites instances

Problème	MILP			AG		ACO			ACO - FLC		
	Solution optimale	CPUs									
	# Inst. résolues	Moyenne	σ_{CPUs}	$\overline{GAP(\%)}$	$\overline{\sigma_{GAP}}$	$\overline{GAP(\%)}$	GAP_{min}	$\overline{\sigma_{GAP}}$	$\overline{GAP(\%)}$	GAP_{min}	$\overline{\sigma_{GAP}}$
3X3-3-3	81/81	0,65	0,28	0,00	0,0000	0,00	0,00	0,0000	0,00	0,00	0,0000
3X3-3-5	81/81	0,56	0,16	0,00	0,0000	0,00	0,00	0,0000	0,00	0,00	0,0000
4X2-2-5	81/81	0,50	0,09	0,00	0,0000	0,00	0,00	0,0000	0,00	0,00	0,0000
4X3-3-3	81/81	2,41	1,23	0,68	0,0027	0,00	0,00	0,0000	0,00	0,00	0,0000
4X3-3-5	81/81	3,49	5,50	0,74	0,0025	0,00	0,00	0,0000	0,00	0,00	0,0000
4X4-4-3	81/81	20,69	76,77	1,84	0,0031	0,07	0,00	0,0006	**0,03**	0,00	0,0004
4X4-4-5	81/81	41,09	102,10	1,84	0,0030	0,06	0,00	0,0007	**0,03**	0,00	0,0004
5X4-4-3	72/81	434,49	524,42	2,81	0,0051	0,23	0,00	0,0011	**0,11**	0,00	0,0006
5X4-4-5	68/81	566,79	650,07	2,78	0,0043	0,24	0,00	0,0013	**0,11**	0,00	0,0006
5X4-4-7	65/81	514,15	653,05	2,83	0,0048	0,26	0,00	0,0013	**0,12**	0,00	0,0007
7X3-3-3	30/81	1399,79	590,09	5,28	0,0101	1,19	0,85	0,0027	**0,88**	**0,45**	0,0034
7X3-3-5	35/81	1290,00	656,01	5,37	0,0113	1,25	0,80	0,0032	**0,96**	**0,46**	0,0040
7X3-3-7	25/81	1463,30	577,48	5,50	0,0128	1,29	0,69	0,0036	**0,98**	**0,43**	0,0044

obtenue dans les problèmes qui appartiennent au scénario.

A partir du tableau 4.11, nous pouvons dire que pour les premiers scénarii (les plus petits - 8 ou 9 opérations) nous avons le même comportement dans les résultats des trois algorithmes. Cependant, en augmentant le nombre des opérations, l'AG commence à perdre en efficacité par rapport aux deux autres méthodes. Pour les problèmes de 20 opérations ou plus l'ACO-FLC montre de meilleurs résultats que l'ACO.

Le nombre de compétences demandées a un impact sur la solution calculée, néanmoins, les configurations où les opérateurs maîtrisent plus de compétences sont généralement plus difficiles à résoudre, puisque la ressource peut être utilisée dans plusieurs opérations, augmentant ainsi la difficulté pour trouver une bonne affectation pour exécuter les différentes tâches.

De cette façon, nous pouvons constater que dans le temps limite de calcul fixé et les différentes configurations proposées comme instances de petite taille, la méthode qui a des meilleurs résultats est l'algorithme à colonie de fourmis hybridé avec la logique floue.

- Instances de grandes tailles

Afin de continuer avec la résolution de ce problème qui prend en compte les contraintes de disponibilité de ressources humaines selon leurs compétences, nous présentons les tests sur les instances de grandes tailles.

Les scénarii suivent la même notation que les instances de petites tailles : $N \times M\_\omega - \lambda - N_W - N_S$. Les tailles des tâches et machines considérées sont $N = 10$, 30 ou 50, $M = 5$, 10 et 15. Le nombre de ressources humaines (opérateurs) pour ces cas est le nombre de machines et le nombre de compétences est généré de façon aléatoire entre $N_S = 3$, 5 ou 7. En utilisant la même procédure, neuf types de problèmes ont été générés de façon aléatoire pour chaque scénario, dans lequel chaque problème a été lancé dix fois de manière répétitive. Les paramètres ω et λ sont sélectionnés aléatoirement entre les valeurs déjà exposées (50, 75 ou 100 et 0.1, 0.3 ou 0.5, respectivement).

Les paramètres de l'algorithme génétique ont été déterminés à partir de plusieurs tests où les meilleurs résultats en termes de qualité de la solution et temps de calcul ont été obtenus avec un nombre de génération de 500, une taille de la population de 50, une probabilité de croisement de 0.8 et une probabilité de mutation de 0.2. Par rapport à l'ACO et l'ACO-FLC le nombre d'itérations a été fixé à 500 et le nombre de fourmis à 95% du nombre de tâches, les paramètres restants ont été récupérés de la description précédente faite pour les instances de petites tailles.

Les tableaux 4.12 et 4.13 montrent les résultats obtenus par les différentes méthodes de résolution. Dans les 18 scénarii proposées l'ACO - FLC trouve les meilleures solutions dans des temps de calcul raisonnables. L'ACO présente des temps de calculs un peu plus courts que l'ACO hybridé mais la qualité des solutions diminue. Enfin, nous avons l'algorithme génétique qui montre des résultats moins performants en termes de solution moyenne.

Instances avec contraintes de ressources humaines et d'outillage

Dans cette deuxième partie, nous montrons les résultats des instances qui prennent en compte l'affectation de personnel avec multi-compétences et d'outillage. Nous avons réalisé les essais avec les trois mêmes algorithmes qui ont été décrits dans l'étape précédente : l'algorithme génétique, l'algorithme à colonie de fourmis et l'algorithme à colonie de fourmis hybridé avec la logique floue. Dans la même logique nous présentons des problèmes de petites et grandes tailles.

- Instances de petites tailles

En ajoutant les contraintes de disponibilité d'outils, la notation pour chaque scénario garde la structure suivante : $N \times M\_\omega - \lambda - N_W - N_S - N_K$. Les tailles de tâches et machines considérées, le nombre de ressources humaines (2, 3 ou 4), le nombre de compétences (3, 5 ou 7) et le nombre de types d'outils (valeur aléatoire dans l'intervalle $[M, (N \times M)]$) n'ont pas changés. Comme dans le cas précédent, neuf types de problèmes ont été générés de façon

TABLE 4.12 – Comparaison de résultats - instances de grandes tailles : RH (I)

Problème	Méthode	Solution moyenne	GAP (%)*	Temps de calcul moyen
10X5_50-0.1-5-3	AG	6586	3,24	*15,022*
	ACO	6422	0,74	19,903
	ACO-FLC	**6373**	0,00	23,112
10X5_75-0.3-5-5	AG	9752	0,68	*17,193*
	ACO	9721	0,36	20,945
	ACO-FLC	**9686**	0,00	22,420
10X5_100-0.5-5-3	AG	11468	4,60	*22,164*
	ACO	10963	0,19	23,589
	ACO-FLC	**10941**	0,00	24,908
10X10_50-0.1-10-5	AG	8891	0,03	*25,050*
	ACO	8900	0,12	33,585
	ACO-FLC	**8889**	0,00	37,186
10X10_75-0.5-10-3	AG	11995	1,66	*40,655*
	ACO	11824	0,24	47,544
	ACO-FLC	**11795**	0,00	51,375
10X10_100-0.1-10-7	AG	16852	5,69	*58,665*
	ACO	15990	0,57	63,540
	ACO-FLC	**15894**	0,00	67,556
30X5_50-0.1-5-3	AG	59956	1,52	*44,321*
	ACO	59421	0,63	67,413
	ACO-FLC	**59044**	0,00	71,791
30X5_75-0.5-5-5	AG	83039	2,47	*63,645*
	ACO	81256	0,33	87,081
	ACO-FLC	**80986**	0,00	92,218
30X5_100-0.1-5-7	AG	108962	1,33	*86,971*
	ACO	108008	0,45	108,590
	ACO-FLC	**107516**	0,00	113,913

*Écart par rapport à la meilleure solution trouvée parmi toutes les méthodes (%)

TABLE 4.13 – Comparaison de résultats - instances de grandes tailles : RH (II)

Problème	Méthode	Solution moyenne	GAP (%)*	Temps de calcul moyen
30X15_50-0.5-15-3	AG	77896	2,81	*269,854*
	ACO	76185	0,61	303,119
	ACO-FLC	**75708**	0,00	309,638
30X15_75-0.1-15-5	AG	113573	4,37	*418,755*
	ACO	109747	1,00	450,156
	ACO-FLC	**108609**	0,00	454,095
30X15_100-0.3-15-3	AG	135980	2,90	*517,338*
	ACO	132652	0,46	565,289
	ACO-FLC	**132032**	0,00	568,875
50X5_50-0.1-5-3	AG	173656	5,82	*109,660*
	ACO	165358	1,05	138,281
	ACO-FLC	**163543**	0,00	142,289
50X5_75-0.5-5-5	AG	231841	3,89	*156,391*
	ACO	224755	0,84	178,572
	ACO-FLC	**222819**	0,00	182,370
50X5_100-0.1-5-7	AG	312844	3,77	*224,319*
	ACO	302961	0,61	251,237
	ACO-FLC	**301055**	0,00	254,751
50X10_50-0.1-10-3	AG	192474	10,88	*474,222*
	ACO	182157	5,52	513,161
	ACO-FLC	**171536**	0,00	520,950
50X10_75-0.5-10-7	AG	283530	7,06	*535,461*
	ACO	272309	3,11	589,057
	ACO-FLC	**263499**	0,00	598,185
50X10_100-0.1-10-5	AG	363540	7,34	*660,807*
	ACO	347248	2,86	724,273
	ACO-FLC	**336866**	0,00	733,364

*Écart par rapport à la meilleure solution trouvée parmi toutes les méthodes (%)

aléatoire pour chaque scénario, dans lequel chaque problème a été lancé dix fois de manière répétitive. Nous avons 117 scénarii différents (1053 instances testées et 9 problèmes de chaque type de scénario). Dans chaque scénario nous disposons de 9 combinaisons possibles : ω_1 (λ_1, λ_2, λ_3), ω_2 (λ_1, λ_2, λ_3) et ω_3 (λ_1, λ_2, λ_3). Le tableau 4.14 expose 13 ensembles des instances qui regroupent les valeurs des paramètres ω et λ après les différentes combinaisons, de telle manière que la notation est la suivante : $N \times M - N_W - N_S - N_K$.

Nous avons gardé les mêmes paramètres pour les algorithmes testés. Pour l'ACO, $\tau_0 = 0.5$, $\alpha = 1$; $\beta = 0.8$, $\rho = 0.7$ et $W_f = 50\%$ du nombre de fourmis (la quantité de fourmis qui est considérée pour faire la mise à jour de la phéromone). Pour l'algorithme à colonie de fourmis hybridé avec la logique floue, les paramètres ont été les mêmes ($\tau_0 = 0.5$, $\alpha_0 = 1$; $\beta_0 = 0.8$, $\rho = 0.7$ et $W_f = 50\%$), et les contrôleurs de logique floue ont été appliqués aux paramètres α et β. Le critère d'arrêt a été limité par le temps d'exécution de l'algorithme ; soit 120 secondes.

TABLE 4.14 – Comparaison de résultats pour les petites instances

	MILP			**ACO**			**ACO - FLC**		
Problème	Solution optimale	CPUs							
	# Inst. résolues	Moyenne	σ_{CPUs}	$\overline{GAP(\%)}$	GAP_{min}	$\overline{\sigma_{GAP}}$	$\overline{GAP(\%)}$	GAP_{min}	$\overline{\sigma_{GAP}}$
3X3-3-3-3	81/81	0,77	0,34	0,00	0,00	0,0000	0,00	0,00	0,0000
3X3-3-5-3	81/81	0,72	0,25	0,00	0,00	0,0000	0,00	0,00	0,0000
4X2-2-5-5	81/81	0,58	0,14	0,00	0,00	0,0000	0,00	0,00	0,0000
4X3-3-3-8	81/81	3,95	6,82	0,00	0,00	0,0000	0,00	0,00	0,0000
4X3-3-5-7	81/81	2,74	1,48	0,00	0,00	0,0000	0,00	0,00	0,0000
4X4-4-3-10	81/81	20,60	34,98	0,15	0,00	0,0006	**0,08**	0,00	0,0004
4X4-4-5-12	80/81	54,08	238,37	0,13	0,00	0,0007	**0,09**	0,00	0,0004
5X4-4-3-18	76/81	409,81	533,05	0,43	0,00	0,0011	**0,30**	0,00	0,0006
5X4-4-5-15	66/81	546,70	698,15	0,45	0,00	0,0013	**0,29**	0,00	0,0006
5X4-4-7-11	60/81	707,76	720,54	0,50	0,00	0,0013	**0,35**	0,00	0,0007
7X3-3-3-9	19/81	1546,55	522,48	1,81	1,52	0,0037	**1,49**	**1,15**	0,0044
7X3-3-5-17	13/81	1644,73	363,40	1,94	1,47	0,0042	**1,57**	**1,09**	0,0050
7X3-3-7-21	11/81	1647,33	359,47	1,98	1,42	0,0046	**1,64**	**1,22**	0,0054

Le tableau 4.14 montre la comparaison des deux algorithmes présentés. Comme on avait expliqué précédemment, les $\overline{GAP(\%)}$ représentent l'écart entre la meilleure solution connue (Cplex - modèle MILP) et la meilleure solution obtenue par l'algorithme (ACO et ACO-FLC) en pourcentage de la borne supérieure. Le $\overline{\sigma_{GAP}}$ représente la dispersion des différents GAPs calculés à l'intérieur de chaque scénario. Enfin, le GAP_{min} est la valeur minimale du

GAP obtenue dans les problèmes qui appartiennent au scénario.

A partir des résultats du tableau 4.14, nous pouvons noter que pour les instances qui ont 12 machines, les deux algorithmes présentent de bons résultats. Cependant, quand on passe à la résolution de problèmes avec 16 tâches, l'ACO-FLC commence à montrer de meilleures solutions. La difficulté pour trouver une bonne solution augmente chaque fois que le nombre de tâches à ordonnancer augmente. Également, l'augmentation du nombre de compétences à prendre en compte pour les opérateurs et le nombre des outils demandés par les machines fait qu'une instance est plus difficile à résoudre qu'une autre.

Dans le temps de calcul fixé, l'algorithme à colonie de fourmis hybridé avec la logique floue montre de meilleurs résultats. Même les GAP_{min} présente des valeurs plus petites que l'ACO.

- Instances de grandes tailles

Les scénarii gardent la même notation que les instances de petite taille : $M\_\omega - \lambda - N_W - N_S - N_K$. Les tailles de tâches sont $N = 10$, 30 ou 50 et de machines $M = 5$, 10 et 15. Le nombre d'opérateurs est le nombre de machines, et le nombre de compétences est généré de façon aléatoire entre 3, 5 ou 7. Également, neuf types de problèmes ont été générés de façon aléatoire pour chaque scénario, dans lequel chaque problème a été lancé dix fois. Les paramètres ω et λ sont choisis de façon aléatoire entre 50, 75 ou 100 et 0.1, 0.3 ou 0.5, respectivement.

Comme il a déjà été mentionné dans les paragraphes précédents, pour les paramètres de l'ACO et l'ACO-FLC, le nombre d'itérations a été établi à 500, le nombre de fourmis à 95% du nombre de tâches et les paramètres restants ont été repris des instances de petites tailles.

Les résultats trouvés dans les tableaux 4.15 et 4.16 montrent que l'ACO a une meilleure performance pour les 4 premiers scénarios (trois de 50 opérations et un de 100 opérations). Cependant, dès lors de l'augmentation du nombre de tâches et machines (nombre d'opérations à ordonnancer) la meilleure méthode est l'ACO - FLC puisqu'elle présente les meilleures solutions pour les 14 scénarios restants. De plus, les temps de calcul sont compétents. L'AG expose des solutions moins performantes, avec des temps d'exécution plus petits puisque la procédure à l'intérieur de l'algorithme est plus simple en termes de processus de calcul (il ne possède pas de processus de recherche locale ou de contrôleurs de logique floue dans sa structure).

TABLE 4.15 – Comparaison de résultats - instances de grandes tailles : RH+outils (I)

Problème	Méthode	Solution moyenne	GAP (%)*	Temps de calcul moyen
10X5_50-0.1-5-3-15	AG	8099	5,89	*18,050*
	ACO	**7622**	0,00	27,903
	ACO-FLC	7653	0,38	23,612
10X5_75-0.3-5-5-35	AG	12102	2,07	*23,152*
	ACO	**11851**	0,00	28,945
	ACO-FLC	11976	1,03	31,420
10X5_100-0.5-5-3-42	AG	15283	2,62	*21,863*
	ACO	**14883**	0,00	31,589
	ACO-FLC	14991	0,71	33,908
10X10_50-0.1-10-5-25	AG	11401	1,51	*36,335*
	ACO	**11230**	0,00	43,585
	ACO-FLC	11289	0,52	43,186
10X10_75-0.5-10-3-70	AG	15495	5,81	*47,921*
	ACO	14724	0,83	56,544
	ACO-FLC	**14595**	0,00	56,375
10X10_100-0.1-10-7-55	AG	21083	7,92	*59,372*
	ACO	19531	0,56	71,540
	ACO-FLC	**19414**	0,00	74,556
30X5_50-0.1-5-3-100	AG	65190	2,48	*59,693*
	ACO	64311	1,13	76,413
	ACO-FLC	**63575**	0,00	81,791
30X5_75-0.5-5-5-123	AG	86204	2,44	*81,327*
	ACO	84456	0,42	97,081
	ACO-FLC	**84098**	0,00	102,218
30X5_100-0.1-5-7-19	AG	116344	3,28	*104,318*
	ACO	114832	1,98	118,590
	ACO-FLC	**112528**	0,00	122,913

*Écart par rapport à la meilleure solution trouvée parmi toutes les méthodes (%)

TABLE 4.16 – Comparaison de résultats - instances de grandes tailles : RH+outils (II)

Problème	Méthode	Solution moyenne	GAP (%)*	Temps de calcul moyen
30X15_50-0.5-15-3-62	AG	86567	4,31	*301,824*
	ACO	84296	1,68	320,119
	ACO-FLC	**82840**	0,00	324,638
30X15_75-0.1-15-5-99	AG	124288	5,25	*448,679*
	ACO	119434	1,35	462,156
	ACO-FLC	**117759**	0,00	466,095
30X15_100-0.3-15-3-160	AG	151362	5,67	*564,128*
	ACO	146687	2,58	579,289
	ACO-FLC	**142783**	0,00	579,875
50X5_50-0.1-5-3-182	AG	191823	7,65	*144,639*
	ACO	182678	2,89	158,281
	ACO-FLC	**177144**	0,00	154,875
50X5_75-0.5-5-5-45	AG	251225	5,48	*182,213*
	ACO	242435	1,98	194,572
	ACO-FLC	**237471**	0,00	201,370
50X5_100-0.1-5-7-138	AG	333201	5,65	*251,987*
	ACO	319318	1,48	265,237
	ACO-FLC	**314376**	0,00	273,751
50X10_50-0.1-10-3-200	AG	207831	11,09	*519,654*
	ACO	194498	4,67	532,161
	ACO-FLC	**184790**	0,00	534,950
50X10_75-0.5-10-7-120	AG	304898	8,07	*594,951*
	ACO	291163	3,57	600,057
	ACO-FLC	**280293**	0,00	615,185
50X10_100-0.1-10-5-317	AG	389224	10,33	*705,271*
	ACO	361607	3,24	738,273
	ACO-FLC	**349013**	0,00	748,364

*Écart par rapport à la meilleure solution trouvée parmi toutes les méthodes (%)

4.4 Conclusion

Les méthodes approchées sont très nombreuses, c'est pourquoi nous avons choisi de développer des méthodes d'optimisation efficaces, basées sur l'algorithme génétique, les colonies de fourmis et l'hybridation avec la logique floue pour résoudre un problème d'ordonnancement de type open shop avec des contraintes de ressources humaines et d'outillage.

Nous avons décidé d'implémenter un algorithme génétique grâce aux résultats obtenus dans les différents problèmes d'ordonnancement et spécialement dans le problème de type open shop qui ont été mentionnés dans le deuxième chapitre (l'état de l'art). L'AG permet d'explorer l'espace de recherche en employant des solutions complètes. Autrement dit, nous utilisons plusieurs opérateurs génétiques qui font la recombinaison de deux solutions trouvées pour en construire de nouvelles. Ces mécanismes sont réalisés selon quelques valeurs de probabilité (croisement et mutation).

Concernant l'algorithme à colonie de fourmis, nous avons aussi trouvé une augmentation sur son application aux problèmes d'ordonnancement. Cette méthode d'optimisation construit une solution étape par étape selon les valeurs des paramètres appelés phéromone, désirabilité et évaporation. Un algorithme à colonie de fourmis construit les solutions nœud par nœud, et la sélection d'un nœud de l'ensemble de nœuds disponibles va conditionner le choix du prochain. Ensuite, nous avons hybridé cette méthode avec la logique floue parce qu'elle génère des affectations de corrections dynamiques sur les valeurs de paramètres (α et β). Nous avons aussi appliqué la recherche locale comme un moyen d'explorer les différents voisinages d'une solution trouvée pendant les processus d'affectation des ressources humaines en ajoutant des mouvements intelligents basés sur les théorèmes proposés par Naderi *et al.* [136] afin d'éviter l'exploration des solutions redondantes, réduire les temps de calcul et obtenir de nouvelles solutions (voisins) avec de meilleures solutions.

De cette façon, avec ces méthodes proposées nous avons étudié différentes façons pour explorer l'espace de solution de notre problème d'ordonnancement de type open shop avec contraintes d'affectation de ressources humaines multi-compétentes et la disponibilité d'outillage.

Les méthodes de résolution proposées ont été comparées entre-elles et avec la meilleure formulation mathématique trouvée dans le chapitre 3 (MILP - lancé sur CPLEX). Cette comparaison a été faite sur des instances générées aléatoirement. Les instances ont été générées à partir de la procédure proposée par différents auteurs qui ont traité des sujets d'ordonnancement, nous avons testé des instances de petites et grandes tailles.

Dans un premier temps, nous avons comparé les instances en prenant en compte que

l'affectation de ressources humaines et leurs compétences pour exécuter une opération. Un AG, un ACO et un ACO-FLC ont été proposés et comparés. Nous avons trouvé que la méthode de l'ACO-FLC a montré les meilleurs résultats pour les instances de petites et grandes tailles.

De la même manière, nous avons ajouté des contraintes d'affectation d'outillage et nous avons comparé les deux algorithmes qui ont donné les meilleurs résultats dans l'étape précédente, l'ACO et l'ACO-FLC. D'après les différents essais, nous avons trouvé que l'ACO-FLC est la méthode qui présente les meilleurs résultats.

Pour encore améliorer ces résultats, nous pourrions essayer des méthodes qui ont fait leur preuve comme des algorithmes à essaims particulaires, ou bien des algorithmes par essaims d'abeilles. Nous pourrions également hybrider ces algorithmes avec une recherche tabou ou un beam search afin de mesurer quelles améliorations nous pouvons apporter.

Comme nous l'avons mentionné au début de ce chapitre, notre intérêt de développer des méthodes approchées efficaces pour résoudre cette problématique d'ordonnancement de type open shop est aussi motivé par l'envie de proposer des mécanismes d'aide à la décision pour apporter des solutions dans le cas industriel qui concerne notre étude de recherche. Ces méthodes ont été proposées en raison de la limite de taille du problème que les modèles mathématiques développés dans le chapitre 3 peuvent résoudre. Les instances sur les données de l'entreprise Norelem seront décrites et traitées dans le chapitre 6 de ce mémoire.

Ce chapitre a fait l'objet de publications dans des conférences internationales [29], [32], [31] et une conférence nationale [30]. De plus, un article dans un journal international a été soumis [33].

Chapitre 5

Ordonnancement d'Open shop multi-objectif

Dans les deux chapitres précédents nous avons étudié le problème d'ordonnancement de type open shop en optimisant un seul objectif à la fois, la minimisation du temps total de séjour. Des formulations mathématiques ont été proposées et testées sur plusieurs instances numériques qui ont été générées selon les critères existants dans la littérature. De la même façon, en connaissant les limitations des méthodes exactes en termes de temps de calcul, nous avons développé des méthodes de résolution approchées afin de proposer des méthodes pour résoudre des problèmes avec des tailles plus proches des cas que nous pouvons trouver dans la réalité.

Dans un premier temps, nous avons abordé ce problème d'ordonnancement de type open shop comme une problématique mono-objectif motivée par les conditions de l'atelier de notre partenaire industriel, l'entreprise Norelem, afin de modéliser et d'identifier le plus rapidement possible un axe d'action pour commencer à résoudre la situation problématique. De cette façon, nous avons décrit un problème open shop avec contraintes d'affectation de ressources humaines et d'outillage qui cherche minimiser un seul objectif. Cependant, d'après cette première approche nous avons trouvé qu'avec les contraintes que présentent l'atelier Norelem et l'idée de minimiser le temps total de séjour des tâches dans le système de production, il y avait aussi des aspects à côté, l'équilibrage de charge de ressources employées, pendant le processus de production qui sont des éléments clés à considérer dans un analyse ultérieure au moment de penser à améliorer la synergie des activités de production.

Dans ce cadre, nous avons décidé de traiter une prochaine étape de notre problématique en ajoutant des objectifs additionnels, afin d'avoir un problème qui prend en compte plusieurs critères au moment de proposer une solution d'ordonnancement. Dans ce chapitre

nous présentons des méthodes de résolution pour notre problème d'ordonnancement de type open shop en considérant 3 fonctions objectifs, c'est-à-dire que, nous optimisons trois critères en même temps. Ci-après, nous décrirons en quoi consiste l'optimisation multi-objectif, quelles mesures de comparaisons peuvent être utilisées, quels sont les objectifs et enfin quelles méthodes multi-objectif nous avons employées.

5.1 Introduction

Les problématiques industrielles actuelles demandent de plus en plus la prise de décisions de courts délais car la concurrence augmente à mesure que l'exigence des clients augmente. De plus, nous sommes dans un environnement économique et social qui évolue très vite grâce aux avancements technologiques et de la connaissance. Ces facteurs font que les entreprises les mieux adaptées et les plus dynamiques ont des avantages compétitifs par rapport aux autres. Dans ce cadre, toute entreprise : petite, moyenne ou grande, doit faire face à plusieurs défis qui généralement cherchent soit la minimisation ou la maximisation d'un objectif, mais en plus de cet objectif principal il y aura toujours des objectifs en parallèle qui peuvent suivre la même ligne d'optimisation ou qui peuvent s'occuper de sujets indirects. Un exemple classique montre un atelier de production où l'objectif est de maximiser le flux de fabrication en cherchant la minimisation des temps de production ainsi que les coûts générés, donc nous pouvons identifier clairement que le but primordial (augmenter les flux de fabrication) a une relation directe ou indirecte avec les deux autres objectifs. De cette façon, nous trouvons que dans le marché actuel il existe un fort intérêt à résoudre des problématiques qui considèrent deux ou plus objectifs en même temps.

L'optimisation multi-objectif a été un sujet d'intérêt pour les chercheurs depuis 1970 dans divers domaines. Dans un problème multi-objectif, il n'existe pas toujours une solution qui est la meilleure en considérant tous les objectifs. Il existe souvent un ensemble de solutions, qui sont non dominées par d'autres solutions dans l'espace de recherche quand tous les objectifs sont pris en compte mais qui sont dominées par d'autres solutions dans un ou plusieurs objectifs et ces solutions sont appelées solutions non-dominées (nous expliquerons ce concept dans une sous-section postérieure).

Dans ce chapitre, nous présentons une introduction sur l'optimisation multi-objectif, quelques notions et concepts sur le sujet. Ensuite, nous exposons les objectifs que nous avons choisis dans notre cas d'étude, après nous détaillons les méthodes employées pour résoudre notre problématique et enfin des comparaisons entre les résultats obtenus seront présentées afin de proposer les conclusions respectives.

5.2 État de l'art

Dans la même logique, l'optimisation multi-objectif occupe une place assez importante dans les champs de recherche actuels en facilitant l'approche aux problèmes réels d'ingénierie rencontrés dans le monde actuel. Ce type d'optimisation permet de prendre en compte plusieurs critères en même temps. De cette manière, comme nous l'avons déjà mentionné, nous ne cherchons pas à trouver une solution optimale mais un ensemble de solutions qui respectent des contraintes pour différents critères.

Un problème d'optimisation multi-objectif consiste à trouver un vecteur $\overrightarrow{x}^* = [x_1^*, x_2^*, ..., x_n^*]^T$ de variables de décision qui satisfait les m contraintes d'inégalité et les p contraintes d'égalité (5.1), afin d'optimiser le vecteur $f(\overrightarrow{x})$ de la fonction (5.2).

$$\begin{aligned} g_i(\overrightarrow{x}) &\geq 0, \quad i = 1, 2, ..., m \\ h_i(\overrightarrow{x}) &= 0, \quad i = 1, 2, ..., p \end{aligned} \tag{5.1}$$

$$\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_k(\overrightarrow{x})]^T \tag{5.2}$$

En d'autres mots, la solution de notre problème d'optimisation est un ensemble de solutions qui présent un compromis pour tous les objectifs satisfaisant les différentes contraintes. Le concept de dominance d'une solution sur une autre est déterminé selon certains critères que nous exposerons dans une section ultérieure. Dans ce cadre, nous obtenons un ensemble de solutions qui appartient au front non dominé mais aucune de ces solutions ne peut être considérée comme la meilleure solution du problème, c'est-à-dire que le choix de la solution à appliquer dépend des besoins du problème étudié et d'un nombre de facteurs additionnels [167].

Dimopoulos [51] a présenté un état de l'art sur les différentes applications de l'optimisation multi-objectif : comme la logistique, l'ordonnancement, et bien d'autres.

5.2.1 Concepts de résolution

D'après la littérature, différents concepts ont été employés pour résoudre les problèmes d'optimisation multi-objectif. Une de ces approches consiste à générer un problème d'optimisation monocritère en ajoutant tous les objectifs dans un même niveau. De cette façon, une somme linéaire pondérée est construite comme cela est exposé par Ishibuchi *et al.* [85] ou Hani *et al.* [79]. Cependant, cette manière de modéliser l'objectif comme une équation

unique n'est pas une tâche facile et peut générer des perturbations dans la conception du modèle, se ramener à une seule fonction objectif peut biaiser la modélisation au moment de pondérer l'importance d'un ou autre objectif [44].

Dans ce cadre, l'approche la plus utilisée pour résoudre ce type de problèmes est la notion de dominance de Pareto, laquelle sera détaillée dans la sous-section suivante.

5.2.1.1 Dominance de Pareto

Le concept de la dominance de Pareto est une des principales notions pour la résolution d'un problème d'optimisation multi-objectif. Cette relation de dominance considère que les solutions optimales d'un problème à plusieurs objectifs sont celles qui sont non dominées et groupées dans un ensemble de solutions (un front), où la solution optimale est généralement appelée solution optimale de Pareto (proposée par Vilfredo Pareto en 1896).

Dans un problème d'optimisation (minimisation), nous disons qu'une solution A domine une autre solution B au sens de Pareto si et seulement si pour n'importe quel objectif i (i est un des k objectifs considérés), la solution présente les mêmes performances ou mieux que la solution B et pour au moins un objectif j, la solution A doit être la meilleure solution.

$$\begin{aligned} &\forall i \in \{1, 2, ..., k\} : \ f_i(A) \leq f_i(B) \\ &\exists j \in \{1, 2, ..., k\} : \ f_j(A) < f_j(B) \end{aligned} \tag{5.3}$$

Un point $x^* \in \Omega = \mathbb{R}^n$ d'un ensemble de solutions réalisables, est dit Pareto optimal par rapport aux autres points de cet ensemble $x \in \Omega$, si et seulement si $\forall x \in \Omega - \{x^*\}$, $f_i(x^*) \leq f_i(x)$ et $\exists j \in \{1, ..., k\} : \ f_j(x^*) < f_j(x)$. Ainsi, nous pouvons définir l'ensemble des solutions Pareto optimales P^* comme :

$$P^* = \{x \in \Omega | \ \nexists \ x' \in \Omega : \ \overrightarrow{f}(x') \prec \overrightarrow{f}(x)\} \tag{5.4}$$

Où $\prec$ représente la dominance de $\overrightarrow{f}(x)$ sur $\overrightarrow{f}(x')$

De cette façon pour un problème multi-objectif et un ensemble d'optimums de Pareto P^*, le front de Pareto $(P(F)^*)$ est défini comme :

$$P(F)^* = \{\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_k(\overrightarrow{x})]^T : \ x \in P^*\} \tag{5.5}$$

5.3 Des approches pour l'optimisation multi-objectif

Concernant les problèmes d'ordonnancement, l'étude de la littérature [175] montre que des méthodes pour résoudre les problèmes multi-objectif ont été généralement classées dans 5 types différents : les méthodes scalaires, interactives, floues, les méthodes qui utilisent les métaheuristiques et les méthodes d'aide à la décision.

L'approche employée dépend du but de l'étude et du contexte de l'application traitée. Dans le problème d'ordonnancement, les métaheuristiques [139], [124] ont été utilisées dans la résolution de critères multiples.

Différents types de recherches ont été menés sur l'application des algorithmes évolutionnaires (MOEA's : multi-objective evolutionary algorithms) aux problèmes d'optimisation multi-objectif [42] puisqu'ils ont été les premières métaheuristiques utilisées dans ce domaine. Les algorithmes évolutionnaires ont démontré être puissants et robustes pour résoudre des problèmes multi-objectif et ils sont de plus en plus utilisés dans les sujets d'ordonnancement. Étant donnée l'attention particulière portée à ces méthodes, plusieurs alternatives de résolution ont été proposées ainsi qu'un classement dans trois catégories : la somme pondérée des fonctions objectifs, des approches basées sur la population et des approches basées sur le concept de Pareto.

En commençant par la première catégorie, la somme pondérée de fonctions objectif combine tous les sous-objectifs dans un seul objectif qui peut être linéaire ou non de tel façon que sa complexité peut varier d'un problème à l'autre ([84],[43]).

Les approches suivantes, basées sur la population, considèrent la recherche par l'évolution de la population mais sans considérer la dominance de Pareto dans le processus de sélection. Une approche représentative de ce type est le VEGA (vector evaluated genetic algorithm - exposé par Schaffer [158]) où la population est divisée en sous-populations qui optimisent son propre objectif.

Enfin, la troisième catégorie fait référence aux méthodes qui utilisent le mécanisme de sélection à partir du critère d'optimalité de Pareto. Dans ce cadre, nous trouvons le Non-dominated Sorting Genetic Algorithm (NSGA) proposé par Srinivas *et al.* [167]. Actuellement, il y a une grande quantité de travaux qui se sont concentrés sur ce type d'approches, algorithmes génétiques multi-objectif (MOGA) [64], des algorithmes génétiques avec niche de Pareto (NPGA) [81], [1]. Deb *et al.* [49] ont développé le NSGA-II qui est la deuxième version du NSGA, qui donne des résultats très bons dans diverses applications. D'autres algorithmes évolutionnaires comme le SPEA et SPEA2 [190], [189] (strength Pareto evolutionary algorithm) ont aussi été proposés comme une autre méthode multi-objectif.

Également, plusieurs autres méthodes ont été développées à partir de différents mécanismes de sélection avec des combinaisons du front de Pareto. Ainsi, l'inclusion des colonies de fourmis a été étudiée par Yagmahan *et al.* [178], Udomsakdigool *et al.* [173], ou même Dridi *et al.* [56] qui ont proposé un algorithme à colonie de fourmis hybride pour résoudre des problèmes d'ordonnancement.

5.4 Critères de comparaison du front de Pareto

Maintenant que nous avons introduit les concepts principaux de l'optimisation multi-objectif, nous devons aborder les diverses métriques qui peuvent être utilisées afin de comparer les fronts de solutions non dominées fournis par les différents algorithmes. Cette comparaison est cruciale pour déterminer la qualité des solutions obtenues et surtout pour établir quel algorithme a l'avantage sur l'autre. Entre les différents critères employés, nous pouvons trouver deux types : les métriques propres à un front et les métriques de comparaison entre différents fronts.

De cette manière, concernant les métriques propres à un seul front, nous pouvons identifier :

- Le nombre de solutions dans le front optimal n_f
- L'hypersurface [44]
- L'hypervolume [176]
- L'espacement [44]
- La métrique HRS [44]

Pour les critères propres à deux fronts, nous trouvons :

- La mesure de Zitzler [190]
- La mesure de progression [44]
- La métrique de Laumanns [108]

Nous présentons ci-dessous plus en détail la métrique de l'hypervolume. En effet, cette mesure permet de déterminer la dominance d'un algorithme sur un autre à partir des différents résultats trouvés par chacun.

L'hypervolume

L'hypervolume (HV) est une métrique qui a été employée par plusieurs auteurs [83], [152]. Elle est aussi connue comme la métrique S [188] ou mesure de Lebesgue [108]. L'hypervolume

d'un ensemble de solutions montre la taille de la portion de l'espace de recherche qui est dominée par ces solutions. D'autre part, l'hypervolume présente un résultat numérique qui montre la proximité des solutions à un front optimal ainsi que la dispersion des solutions dans l'espace objectif.

Cette métrique expose le volume de l'espace objectif qui est couvert par les individus d'un ensemble de solutions non-dominées, c'est-à-dire pour les solutions qui appartiennent à un front. Le volume est délimité par deux points : un point de référence nommé point anti-optimal (A) qui est établi comme la plus mauvaise solution dans l'espace objectif et un deuxième point optimal (ou pseudo-optimal, calculé par la méthode de résolution). Généralement, il faut normaliser les valeurs des fonctions objectif avant de calculer l'hypervolume. De cette façon, l'hypervolume est la surface (volume) qui est compris entre les individus D_l (l = 1,...,Υ) d'un ensemble de solutions non-dominées (ND) et le point A, comme il est présenté dans l'expression (5.6) :

$$HV(ND, A) = \wedge(\{\bigcup h(D_l) | D_l \in ND, l = 1, ..., \Upsilon\}) \tag{5.6}$$

Où $h(D_l) = |D_{l1} - A_1| \times |D_{l2} - A_2| \times ... \times |D_{lC} - A_C|$, Υ est le nombre de solutions qui appartiennent au front de Pareto et C le nombre de fonctions objectif.

Dans la figure 5.1 la région en question est définie par la surface de la figure obtenue avec l'union des différents rectangles formés par chaque solution du front (y_1, y_2 et y_3) et le point de référence (A). D'autre part, la figure 5.2 montre le calcul de l'hypervolume avec trois fonctions objectif où il y a cinq solutions dans le front (y_1, y_2, y_3, y_4 et y_5) et un point de référence.

5.5 Description de la problématique

Comme nous l'avons déjà exposé dans les chapitres 3 et 4, nous traitons un problème d'ordonnancement de type open shop avec contraintes d'affectation de ressources humaines (opérateurs) avec multi-compétences ainsi que l'affectation des outils aux opérations. Dans un premier temps nous avons décidé d'aborder le problème avec l'optimisation mono-objectif : la minimisation du temps total de séjour (temps que reste un produit dès qu'il est disponible pour commencer sa production jusqu'à sa dernière opération).

Dans cette partie, comme nous sommes en train de traiter l'optimisation multi-objectif, nous allons considérer trois objectifs à optimiser afin d'obtenir l'ordonnancement de notre problème. Nous avons gardé notre premier objectif proposé dans les chapitres précédents,

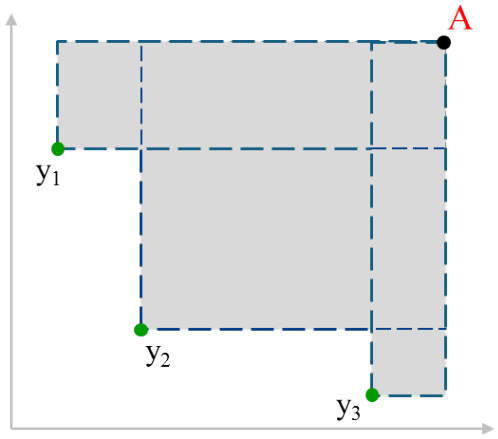


FIGURE 5.1 – Hypervolume pour un problème à deux objectifs

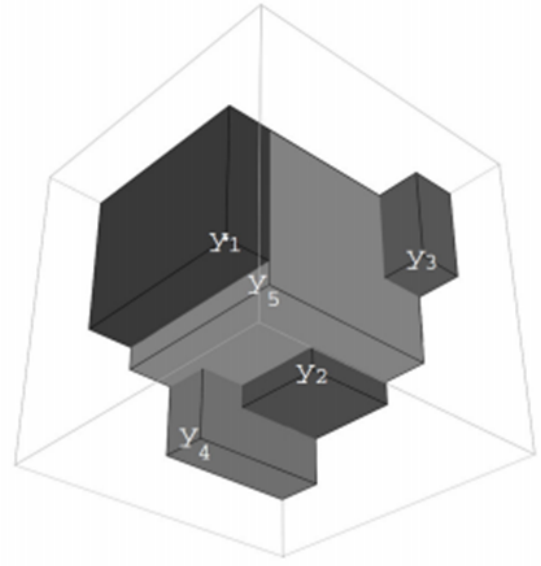


FIGURE 5.2 – Hypervolume pour un problème à trois objectifs

c'est-à-dire, que nous nous sommes toujours intéressés à la minimisation du temps total de séjour ($Min \sum F_i$, figure 5.3 - où nous proposons un exemple graphique de $N = 3$, $M = 3$, $N_w = 4$ et $N_k = 4$)

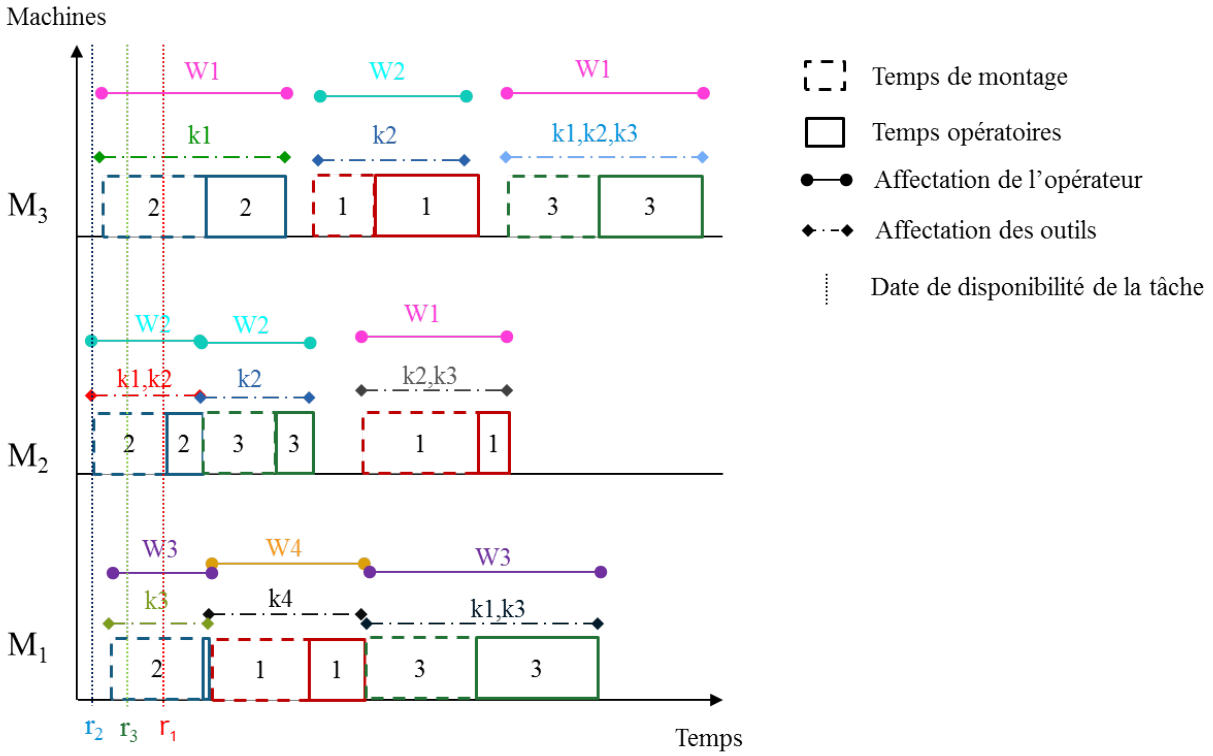


FIGURE 5.3 – F.O. 1 : Minimisation du temps total de séjour

Dans un deuxième temps, il est aussi intéressant d'équilibrer la charge d'utilisation de ressources humaines et matérielles. En sachant que, dans notre problématique, les ressources humaines doivent être affectées pour exécuter une opération selon leurs compétences, nous pouvons avoir le cas où certains opérateurs sont affectés à travailler plus de temps que d'autres (par exemple des opérations plus longues, ou plus polyvalents). De cette manière, nous avons fixé un deuxième objectif à optimiser, l'équilibrage de charge des opérateurs afin de bien distribuer les différentes activités entre l'ensemble du personnel disponible et éviter de surcharger un opérateur.

En prenant comme référence le travail proposé par Ouazene *et al.* [146], nous avons formulé notre deuxième objectif en deux étapes. En premier, il faut déterminer la valeur de la date de fin de la dernière opération exécutée par chaque opérateur w ($E_w = \underset{i\in J, m\in L}{Max}\, C_{wim}$). Une fois que nous avons déterminé ces valeurs, nous minimisons la différence entre la valeur maximale et minimale de dates de fin obtenues (expression 5.7) afin d'équilibrer la charge des opérateurs dans la configuration d'ordonnancement proposée par l'algorithme.

$$Min\left(\frac{1}{N_w}\sum_{w=1}^{N_w} E_w\right) \Leftrightarrow Min\left(\underset{w=1,\ldots,N_w}{Max}\, E_w - \underset{w=1,\ldots,N_w}{Min}\, E_w\right) = Min\left(\Delta E_{RH}\right) \tag{5.7}$$

La figure 5.4 montre l'affectation des opérations sur chaque opérateur w (dans notre exemple nous présentons $N_w = 4$ - même ordonnancement exposé dans l'exemple de notre premier objectif) ainsi que la représentation graphique de ΔE_{RH}. Dans ce cadre notre fonction objectif numéro 2 est la minimisation de ΔE_{RH} ($Min\Delta E_{RH}$).

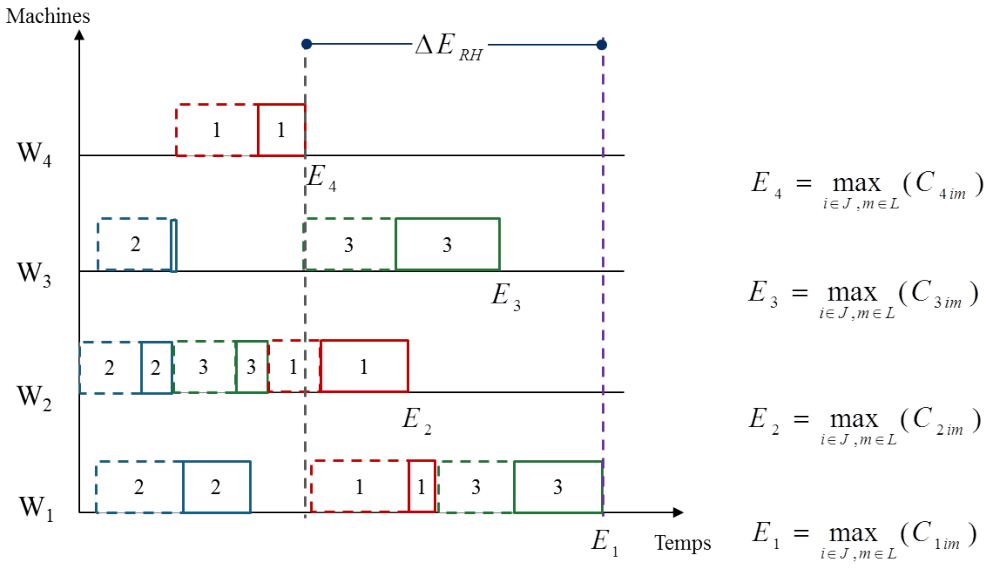


FIGURE 5.4 – F.O. 2 : Équilibrage de charge des opérateurs

De la même façon, nous avons abordé l'équilibrage de charge de machines comme un autre objectif. Nous suivons la même démarche que pour le cas précédent. Tout d'abord, il faut trouver la valeur de la date de fin de la dernière opération réalisée sur la machine m ($E_m = \underset{w\in W, i\in J}{Max} C_{wim}$). Après cela, nous cherchons la minimisation de la différence entre la valeur maximale et minimale des dates de fin calculées par la configuration proposée (équation 5.8)

$$Min\left(\frac{1}{M}\sum_{m=1}^{m} E_m\right) \Leftrightarrow Min\left(\underset{m=1,\ldots,M}{Max} E_m - \underset{m=1,\ldots,M}{Min} E_m\right) = Min\left(\Delta E_{RM}\right) \qquad (5.8)$$

Également, dans la figure 5.5, nous exposons les activités qui ont été affectés aux différentes machines ($M = 3$ dans notre exemple) et la représentation de ΔE_{RM}. Dans cette logique, nous définissons notre troisième et dernier objectif comme la minimisation de ΔE_{RM} ($Min(\Delta E_{RM})$).

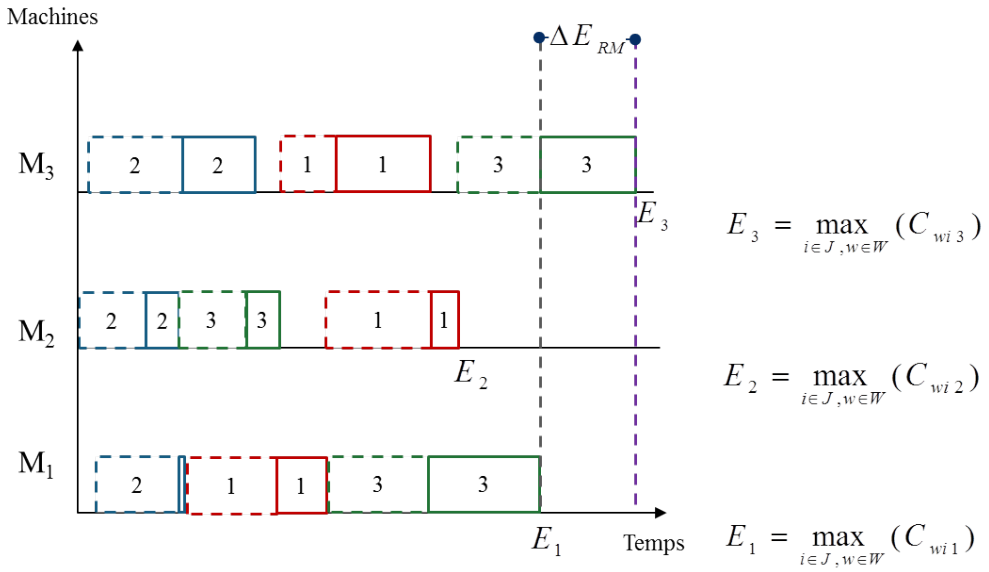


FIGURE 5.5 – F.O. 3 : Équilibrage de charge de machines

En récapitulant brièvement, notre problème est un open shop avec contraintes de ressource où nous cherchons sa résolution en optimisant trois objectifs : la minimisation du temps total de séjour (FO_1), l'équilibrage des charge des opérateurs (FO_2) et l'équilibrage de charge de machines (FO_3).

5.6 Méthodes de résolution

Dans cette section, nous présentons les méthodes que nous avons testées dans la résolution de notre problème d'ordonnancement de type open shop avec contraintes d'affectation de ressources humaines et d'outillage, en prenant en compte trois objectifs.

Ensuite, nous exposons en détail le NSGA-II et le NSGA-III afin d'établir des comparaisons entre les deux méthodes dans la section suivante. Mais tout d'abord, nous commençons par expliquer les concepts communs aux deux méthodes en termes de codage, interprétation et évaluation de la solution.

5.6.1 Représentation d'une solution

Pour coder chaque solution, nous avons repris les vecteurs employés dans les chapitres précédents. Le vecteur d'opérations est crée $(N \times M)$. Chaque opération représente l'affectation d'une tâche à une machine (tableau 4.1 dans la sous-section 5.6.1). D'autre part, le vecteur d'affectation des opérateurs est généré selon les compétences maitrisées par l'opérateur et demandées par l'opération ainsi que le vecteur d'affectation des outils. Nous rappelons les vecteurs dans la figure 5.6.

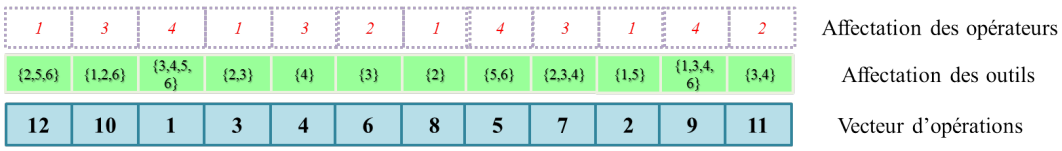


FIGURE 5.6 – Représentation d'une solution d'ordonnancement en considérant des ressources humaines et d'outillage

5.6.2 Évaluation

Pour calculer les fonctions fitness (les fonctions objectif FO_1, FO_2 et FO_3), dans notre cas la minimisation du temps total de séjour $(Min \sum F_i)$, l'équilibrage de charge des opérateurs $(Min\Delta E_{RH})$ et l'équilibrage de charge de machines $(Min\Delta E_{RM})$, il faut créer une séquence d'ordonnancement des opérations qui respecte les contraintes d'affectation des opérateurs et des outils et ensuite réaliser les différents calculs qui considèrent les dates de fin des activités afin d'obtenir les valeurs FO_1, FO_2 et FO_3. Ces valeurs vont déterminer la qualité des solutions obtenues et donneront l'ensemble de solutions à prendre en compte parmi les différents fronts de Pareto.

5.6.3 MOEAs : NSGA - II et NSGA - III

Après avoir décrit les critères à prendre en compte pour nos algorithmes évolutionnaires, nous présentons en détail les algorithmes génétiques multi-objectif utilisés dans notre problème à trois objectifs.

5.6.3.1 NSGA - II

Plusieurs algorithmes évolutionnaires multi-objectif ont été suggérés durant les dernières années. Parmi eux, nous pouvons trouver le NSGA qui pendant quelques années a été critiqué en raison de sa complexité élevée de résolution au moment de trier les solutions (non-dominated sorting), du manque d'élitisme (récupérer les bonnes solutions) et de la spécificité au moment de partager des paramètres.

En réponse aux critiques, Deb *et al.* [49] ont proposé le NSGA-II, lequel a été employé pour résoudre le problème traité dans ce mémoire.

Cet algorithme est constitué de deux parties principales : un classement rapide des solutions non-dominées et la préservation de la diversification des solutions. En prenant en compte ces critères nous allons exposer la procédure utilisée dans la démarche de notre méthode (algorithme 8).

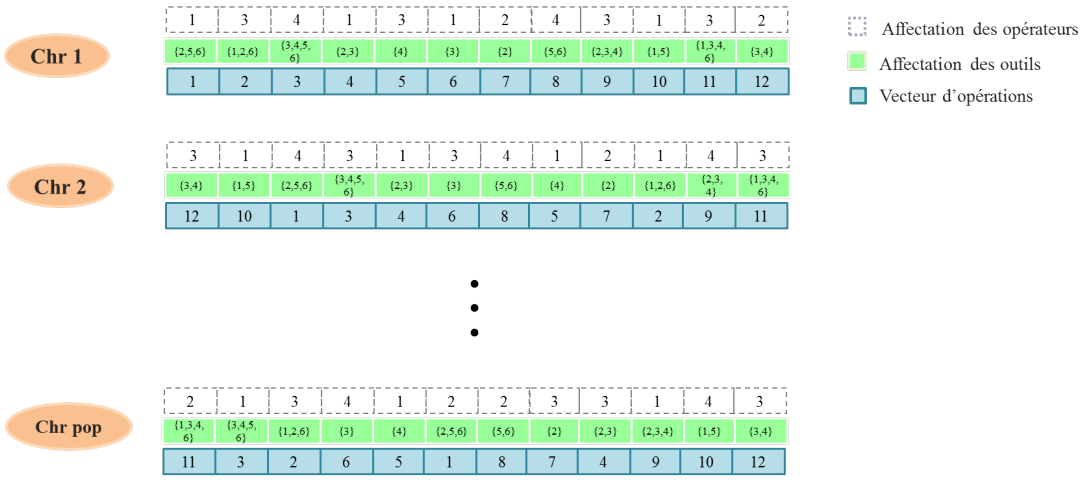


FIGURE 5.7 – Représentation de la population initiale

Population initiale

La population initiale est générée de façon aléatoire selon le nombre de chromosomes définis (POP). Puis, l'affectation des ressources (opérateurs et outils) est faite au hasard

Algorithme 8 NSGA - II

1: Génération de la population initiale
2: Classement des solutions non-dominées
3: **pour** $i = 1$ à *critère d'arrêt* **faire**
4: Sélectionner deux parents $P1$ et $P2$ en suivant la méthode du tournoi
5: Appliquer le croisement entre $P1$ et $P2$ avec une probabilité P_c
6: Appliquer la mutation sur le chromosome sélectionné avec une probabilité P_m
7: Faire le classement des solutions non-dominées
8: Garder les premiers POP chromosomes (POP : taille de la population) pour la prochaine génération
9: **fin pour**

selon les compétences requises pour exécuter l'activité et les compétences maîtrisées par chaque opérateur ainsi que les outils demandés par chaque opération et leur disponibilité. Au moins une activité doit être exécutée par chaque opérateur. A la fin de cette étape nous aurons un chromosome qui sera composé de différentes opérations à réaliser et chacune sera liée avec deux types de ressources (opérateur et outils - figure 5.7).

Classement des individus : solutions non-dominées

Pour établir quelles solutions feront parties de chaque front de Pareto, il faut réaliser un classement des résultats obtenus par chaque séquence de chromosomes générée. Chaque solution doit être comparée avec chaque solution de la population pour déterminer si elle est dominée.

Dans la première étape, nous calculons deux entités : 1) Compteur de domination n_p, qui représente le nombre de solutions qui dominent la solution p et 2) S_p qui est l'ensemble de solutions que la solution p domine.

Toutes les solutions qui font parties du premier front non-dominé doivent avoir leur compteur de domination égal à zéro. Maintenant pour chaque solution p avec $n_p = 0$, nous visitons chaque membre (q) de son ensemble S_p et nous réduisons le compteur de domination. Pendant cette réduction, si un membre q du compteur de domination devient zéro, nous l'ajoutons dans un liste Q. Ces membres appartiennent au deuxième front non-dominé. La procédure décrite est faite avec chaque membre de Q et le troisième front est identifié ainsi de suite. Cette démarche continue jusqu'à l'identification de tous les fronts.

Pour chaque solution p dans le deuxième ou plus haut niveau de non-domination, le compteur de domination n_p peut être au moins POP-1 (POP = taille de la population). Alors

chaque solution p sera visitée au moins POP-1 fois avant que son compteur de domination soit égal à zéro. A ce moment-là, la solution est affectée à un niveau de non-domination et elle ne sera pas de nouveau visitée.

- Préservation de la diversité
  Cette méthode tend, à la fois, à faire progresser les solutions vers le front de Pareto optimal mais aussi à maintenir de la diversité dans la population. De cette façon, nous calculons la densité des solutions qui sont voisines d'une solution particulière de population, nous calculons la distance moyenne de deux points suivant chacun des objectifs (cuboid de la figure 5.8 : ce périmètre est appelé crowding distance [49]). Le calcul de la crowding distance requiert le classement de la population selon chaque valeur de la fonction objectif dans un ordre croissant. Après, pour chaque fonction objectif, les bornes (les solutions avec la plus petite et grande fonction objectif) sont affectées d'une valeur de distance infinie. Toutes les valeurs de distances des solutions intermédiaires sont calculées selon la différence absolue normalisée des valeurs des fonctions de deux solutions adjacentes 5.9 (chaque fonction objectif est normalisée avant de calculer la crowding distance)

$$I[i]_{.distance} = \sum_{c=1}^{C} \frac{I[i+1].c - I[i-1].c}{f_c^{max} - f_c^{min}} \tag{5.9}$$

  Où C est égal au nombre de fonctions objectif et $I[1]_{distance} = I[C]_{distance} = \infty$

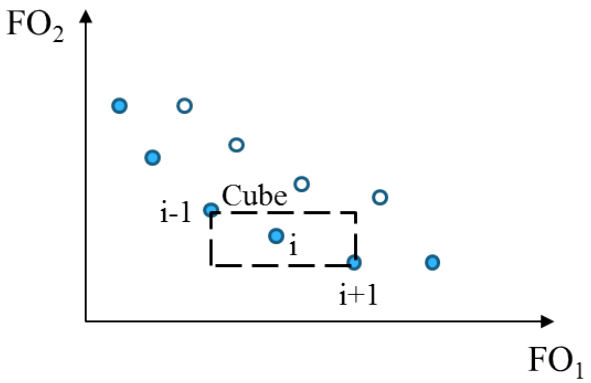


FIGURE 5.8 – Calcul de la crowding distance [49]

A partir de cette distance, nous pouvons comparer deux solutions en termes de proximité avec des autres solutions. De cette manière, entre deux solutions avec différents niveaux de non-domination, nous préférons la solution moins dominée. Par contre, si les deux solutions appartiennent au même front, nous préférons la solution qui est placée

dans la région moins remplie.

Sélection

La sélection détermine quelles solutions de la population actuelle seront préservées pour faire le croisement. Dans cette méthode, la sélection est faite selon la méthode de tournoi, où l'on confronte de deux individus de la population initiale qui sont choisis au hasard afin de stocker les individus qui appartiennent au meilleur front de Pareto. Dans le cas où les deux solutions confrontées seraient membres du même front, la décision sera prise à partir du critère de la crowding distance (figure 5.9).

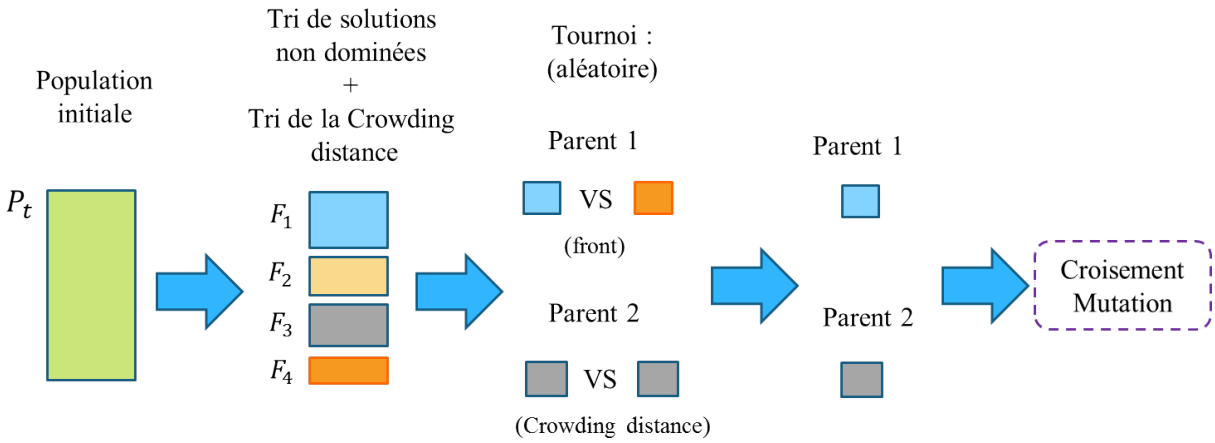


FIGURE 5.9 – Sélection des parents

Croisement

Après l'étape de sélection, le croisement prend les solutions appelées parents et génère de nouvelles solutions en les recombinant. De cette manière la qualité des enfants (fonction de fitness) sera déterminée en grande partie par la qualité des parents. Nous utilisons un croisement en quatre étapes qui est décrit par l'algorithme 9.

Mutation

La mutation génère de nouvelles solutions à partir de la modification des gènes d'un parent. Ce mécanisme aide à préserver la diversité de la population et évite les optimums locaux. Dans notre algorithme, nous adoptons la méthode décrite dans le chapitre 4, laquelle est faite en deux étapes (figure 5.11).

Algorithme 9 Croisement

1: **Étape 1 :** Appliquer la mutation à chaque parent sélectionné selon une probabilité de mutation (même procédure qui est décrite dans les paragraphes suivants), c'est-à-dire, avant de faire la combinaison de parents, des perturbations sont générées dans les vecteurs de solutions qui vont créer les enfants afin de diversifier les individus générés.

2: **Étape 2 :** Sélectionner un point de coupure au hasard et deux sub-divisions sont générées pour chaque parent (qui est composé de trois vecteurs : séquence des opérations, affectation d'outillage et affectation des opérateurs).

3: **Étape 3 :** échanger les sub-divisions de deux parents (le changement de positions des opérations est fait en gardant l'affectation de l'opérateur et d'outillage qui a été déjà faite).

4: **Étape 4 :** Corriger les chromosomes résultants pour éviter la répétition des éléments (opérations) comme il est montré dans la figure 5.10.

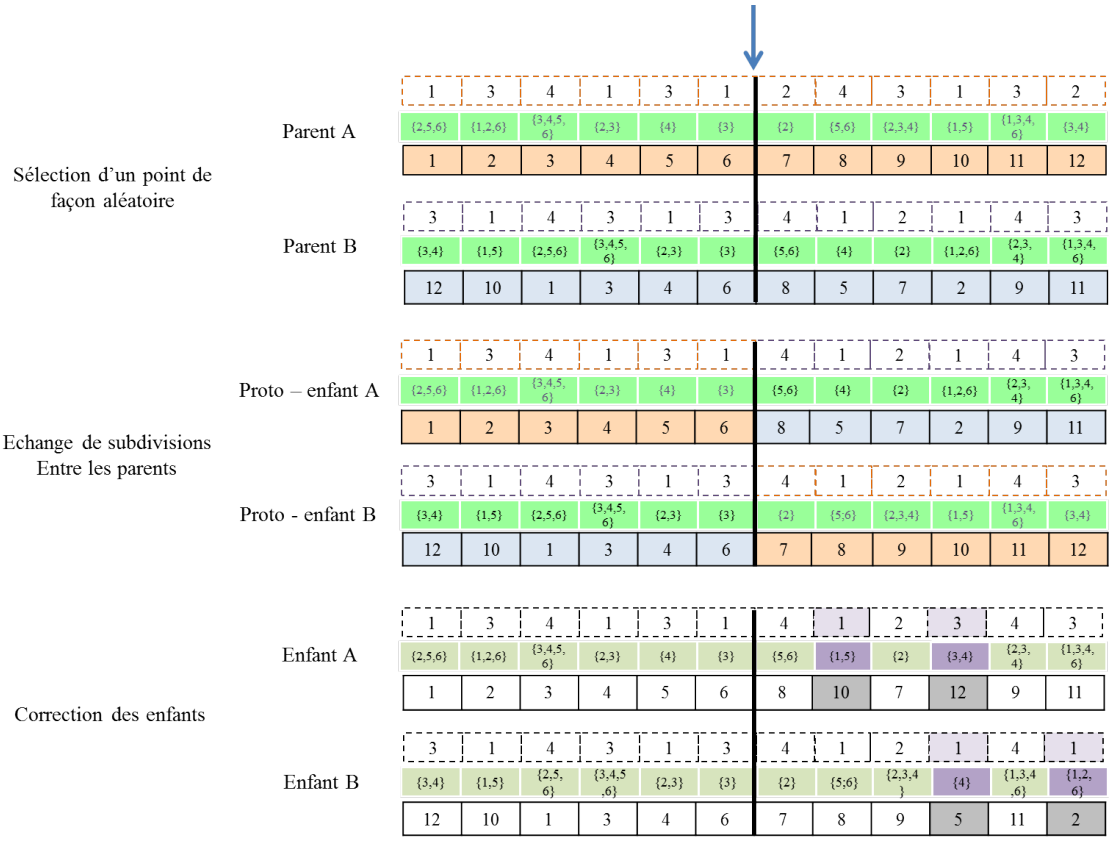


FIGURE 5.10 – Opérateur de croisement

Algorithme 10 Mutation

1: **Étape 1 :** Sélectionner deux points de façon aléatoire, donc le chromosome est divisé en trois sub-vecteurs appelés sv#1, sv#2 et sv#3.

2: **Étape 2 :** Échanger sv#1 et sv#3.

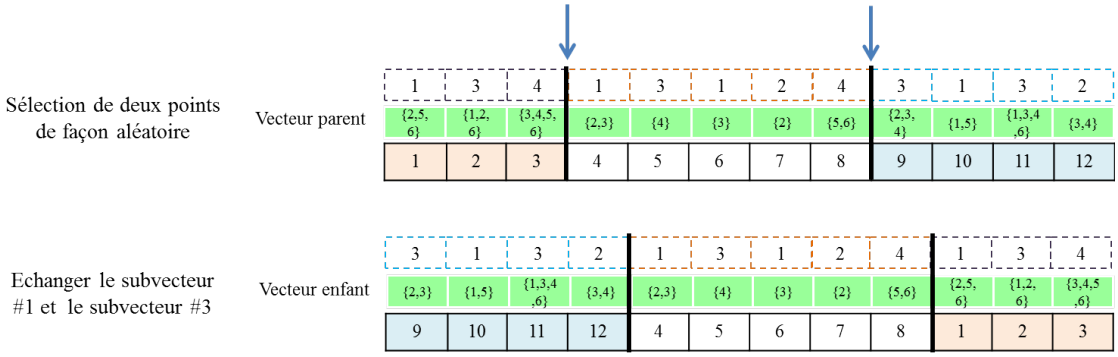


FIGURE 5.11 – Opérateur de mutation

Mise à jour de la population

Après les différents opérateurs génétiques employés et les critères de classement des solutions (fronts et la crowding distance), nous devons déterminer les individus que nous allons garder pour notre prochaine génération. Dans la figure 5.12, nous représentons la façon dont nous sélectionnons des individus d'une génération à une autre. Tout d'abord, nous partons d'une population initiale (P_t) qui a été déjà décrite antérieurement, puis nous générons les enfants (Q_t) en appliquant la sélection, le croisement et la mutation proposés pour avoir une population totale (R_t). Cette dernière sera classée en prenant les différents concepts déjà mentionnés en termes de fronts et de la crowding distance avec pour but de conserver un nombre constant d'individus pour chaque génération (notre taille de la population - POP).

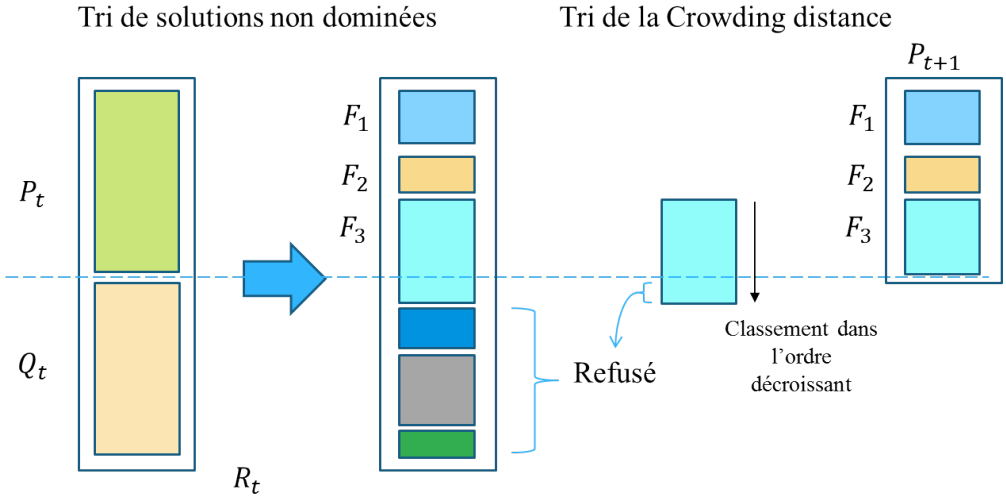


FIGURE 5.12 – Mise à jour de la population [48]

5.6.3.2 NSGA - III

Dans cette section, nous présentons le NSGA - III qui a été désigné comme une réponse aux difficultés qu'ont les algorithmes évolutionnaires pour traiter plusieurs objectifs en même temps (plus de deux). Cet algorithme a été proposé par Deb *et al.* [48] et il s'appuie sur la même structure que son prédécesseur le NSGA - II avec quelques changements importants dans le mécanisme de sélection.

Dans l'algorithme NSGA - II, le classement de la population est fait par différents niveaux de solutions non-dominées (fronts, F_1, F_2, etc.). Ensuite comme nous avons mentionné dans la procédure de la méthode, les différents opérateurs génétiques sont appliqués pour créer la population de la génération suivante. Et selon le nombre de solutions par front et le nombre d'individus à garder, nous prenons en compte la crowding distance comme un critère de sélection. De cette manière les solutions avec les valeurs de crowding distance les plus grandes seront choisies. Le NSGA - III remplace ce critère par les approches qui sont expliquées dans l'algorithme 11 :

Algorithme 11 NSGA - III

1: Calculer le nombre de points de référence (H) à placer sur l'hyperplan
2: Générer la population initiale de façon aléatoire en vérifiant les contraintes d'affectation de ressources (POP chromosomes)
3: Faire le classement des solutions non-dominées
4: **pour** $i = 1$ à *critère d'arrêt* **faire**
5: Sélectionner deux parents $P1$ et $P2$ à partir de la méthode de tournoi
6: Appliquer le croisement entre $P1$ et $P2$ avec une probabilité P_c
7: Faire le classement des solutions non-dominées
8: Normaliser les membres de la population
9: Associer les membres de la population avec les points de référence
10: Appliquer la préservation de la niche (compteur)
11: Garder les solutions obtenues dans la niche pour la prochaine génération
12: **fin pour**

Détermination de points de référence sur un hyperplan

Dans cette partie, il faut prédéfinir un ensemble de points de référence pour assurer la diversité des solutions obtenues. Différents points sont placés sur un hyperplan normalisé qui a la même orientation dans tous les axes.

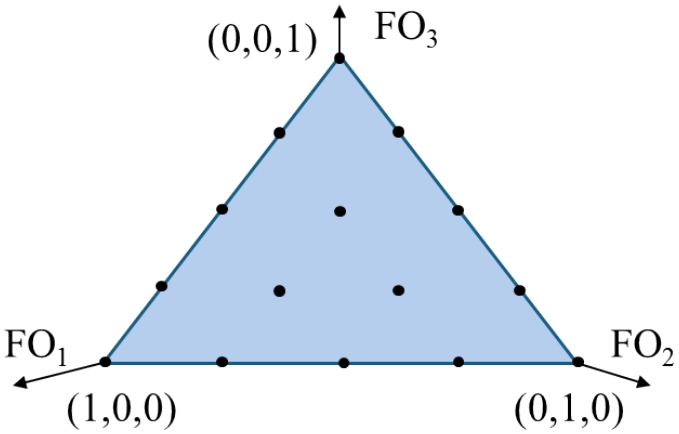


FIGURE 5.13 – Points de référence sur l'hyperplan normalisé [48]

Le nombre de points de référence (H) est défini par l'expression suivante :

$$H = \binom{C+g-1}{g} \tag{5.10}$$

Où C est le nombre de fonctions objectif, g est défini comme le nombre de divisions qui seront considérées sur chaque axe objectif. De cette façon pour 3 objectifs et 4 divisions nous aurons 15 points de référence. Les points de référence sont distribués sur tout l'hyperplan de la même façon que les solutions seront décrites par un front de Pareto. De cette façon les solutions seront plus tard associées avec les points de référence créés.

Normalisation des membres de la population

Afin de déterminer un point idéal de la population en cours, il faut identifier la valeur minimale pour chaque fonction objectif (FO_i^{min}, i = 1,2,...,C), c'est-à-dire que nous aurons un point idéal = ($FO_1^{min}, FO_2^{min}, ..., FO_C^{min}$). Après, chaque fonction objectif doit être déplacée en soustrayant z_i^{min} à l'objectif f_i. De cette façon nous continuons avec la même procédure proposée par [48] afin de générer un hyperplan de la figure 5.14 (Cette démarche permet au NSGA-III de résoudre des problèmes avec des fronts de Pareto où les solutions objectif ont des échelles différentes).

Association entre points de référence et solutions

Après la normalisation de chaque fonction objectif, il est nécessaire d'associer chaque membre de la population avec un point de référence. Pour trouver cette relation, nous définissons une ligne de référence pour chaque point en joignant le point de référence avec

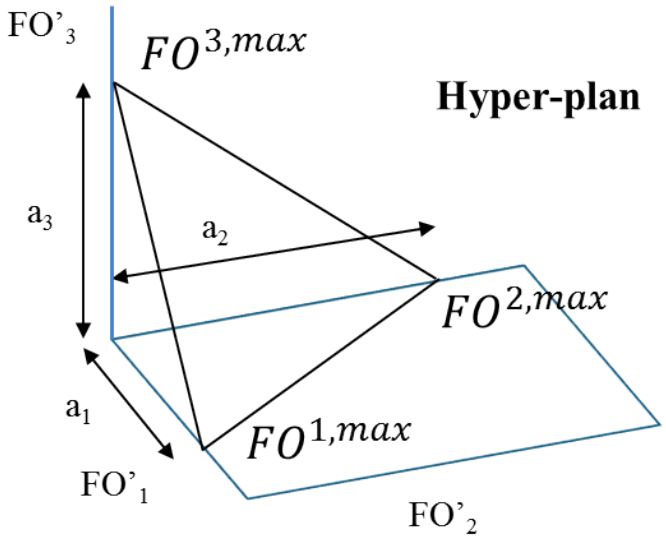


FIGURE 5.14 – Normalisation des membres de la population sur l'hyperplan [48]

l'origine. Ensuite, nous déterminons la distance perpendiculaire entre chaque membre de la population et chacune des lignes de référence. Finalement, le point de référence qui a la ligne de référence la plus proche d'un individu de la population est considéré comme associé au membre de la population (figure 5.15).

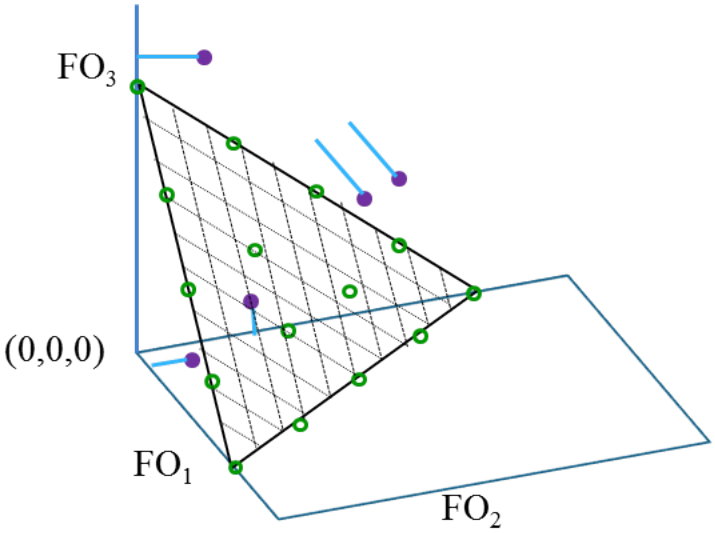


FIGURE 5.15 – Association de la population avec un point de référence [48]

Préservation de la niche

Selon l'étape précédente, un point de référence peut avoir associé un ou plusieurs membres de la solution. Dès lors, il faut compter le nombre de solutions qui sont associées avec chaque

point et ne garder que les solutions qui sont plus proches du point de référence en question (la distance perpendiculaire depuis la ligne de référence - [48])

Opérateurs génétiques

Pour générer les enfants, nous appliquons les opérateurs génériques expliqués dans l'algorithme NSGA - II ainsi que la sélection des individus qui est faite en prenant en compte la diversité de solutions par rapport à la proximité existante entre la solution et le point de référence. De la même façon et à partir des suggestions des auteurs [48] pour donner la même importance à tous les membres de la population, nous fixons la taille de la population (POP) presque égale au nombre de points de référence (H).

5.7 Expérimentations

Cette section présente les résultats de l'application numérique des méthodes multi-objectif décrites dans ce chapitre, sur des instances théoriques du problème d'ordonnancement dans un atelier de production de type open shop. Les méthodes proposées ont été testées, de la même façon que précédemment en employant le logiciel codeblocks 12.11, codées en langage C avec un ordinateur personnel avec CPU CORE i3 2.4 GHz, une mémoire RAM de 4GB et un système d'exploitation windows 7.

Nous présentons d'abord les instances qui ont été sélectionnées pour réaliser les tests et ensuite, les différents tableaux de résultats avec des comparaisons entre les deux méthodes exposées.

5.7.1 Les instances numériques

De la même manière que dans les chapitres précédents, nous avons testé les méthodes de résolution sur des instances théoriques. Différents environnements de production ont été considérés.

Nous avons décidé de reprendre les instances déjà générées au moment de tester le modèle mathématique ainsi que les méthodes approchées qui ont traité le problème d'ordonnancement de type open shop avec les contraintes d'affectation de ressources humaines et d'outillage (section 4.3.1 - petites et moyennes-grandes tailles).

Différents types d'environnements de problèmes ont été générés de façon aléatoire, en prenant en compte plusieurs valeurs pour le nombre de tâches, machines, nombre de com-

pétences, d'outils ainsi que la variation de certains des paramètres (λ et ω) pour générer les valeurs des temps de montage, temps opératoires et dates de disponibilité de tâches.

5.7.2 Résultats expérimentaux

Dans cette sous-section, nous exposons les tableaux de données des différents tests réalisés. Comme nous l'avons mentionné, il y a deux types d'instances : de petites tailles et de grandes tailles. De cette façon, nous allons présenter les divers résultats qui comparent les deux algorithmes employés et les commentaires respectifs par rapport à la qualité de leurs solutions.

5.7.2.1 Instances de petites tailles

En considérant les contraintes d'affectation des opérateurs selon leurs compétences et la disponibilité d'outils, la notation pour chaque scénario suit la même structure que dans le chapitre précédent : $N \times M\_\omega - \lambda - N_W - N_S - N_K$. Les tailles de tâches et machines sont identiques. Le nombre de ressources humaines (2, 3 ou 4), le nombre de compétences (3, 5 ou 7) et le nombre de types d'outils (valeur aléatoire dans l'intervalle $[M, (N \times M)]$) sont restés comme dans le chapitre précédent. D'autre part, chaque scénario résume neuf types de problèmes, dans lesquels chaque problème a été lancé dix fois. Nous présentons 117 scénarii différents (1053 instances testées et 9 problèmes de chaque type de scénario). Dans chaque scénario nous avons 9 combinaisons possibles entre les valeurs de ω et λ comme nous avions déjà mentionné dans les chapitres précédents. Nous exposons 13 ensembles des instances qui regroupent les valeurs des paramètres ω et λ après les différentes combinaisons ($N \times M - N_W - N_S - N_K$).

Par rapport aux paramètres des algorithmes, nous avons fixé la probabilité de croisement à 0.8 et la probabilité de mutation à 0.2. Les deux méthodes ont été lancées par l'évaluation d'un maximum de 23000 fonctions objectif, c'est-à-dire que nous avons fixé le nombre de générations à 250 et une taille de population de 92 (par suggestion de Deb *et al.* [48]).

Concernant le NSGA - III, H n'est pas un paramètre de l'algorithme, puisque la taille de la population (POP) est dépendante de H, donc $POP \simeq H$. De cette façon, selon les suggestions des développeurs des procédures MOEA/D, il est convenable d'utiliser une taille de population de 91 dans un problème de trois objectifs ($C = 3$) [48]. Il faut alors définir $g = 12$ (nombre de divisions considérées sur chaque objectif), pour obtenir $H = 91$ à partir de l'équation 5.10 (de cette façon nous fixons un taille de population $POP \simeq H$).

TABLE 5.1 – La meilleure, la moyenne et la plus mauvaise valeur de l'hypervolume obtenue par le NSGA-II et le NSGA-III - petites tailles

Problème	NSGA - II			NSGA - III		
	H_{max}	$\overline{H}$	H_{min}	H_{max}	$\overline{H}$	H_{min}
$3 \times 3 - 3 - 3 - 3$	0,8715	0,8473	0,8390	**0,8890**	**0,8562**	0,8318
$3 \times 3 - 3 - 5 - 3$	0,8510	**0,8383**	0,8201	**0,8601**	0,8361	0,8294
$4 \times 2 - 2 - 5 - 5$	0,8405	0,8207	0,8060	**0,8526**	**0,8299**	0,8010
$4 \times 3 - 3 - 3 - 8$	**0,8497**	0,8234	0,8001	0,8412	**0,8287**	0,8090
$4 \times 3 - 3 - 5 - 7$	0,8599	0,8364	0,8124	**0,8680**	**0,8402**	0,8099
$4 \times 4 - 4 - 3 - 10$	**0,8419**	0,8161	0,7835	0,8402	**0,8197**	0,7869
$4 \times 4 - 4 - 5 - 12$	0,8314	0,8031	0,7721	**0,8406**	**0,8088**	0,7775
$5 \times 4 - 4 - 3 - 18$	0,8411	0,8125	0,7699	**0,8608**	**0,8275**	0,7768
$5 \times 4 - 4 - 5 - 15$	**0,8407**	0,8064	0,7703	0,8387	**0,8084**	0,7761
$5 \times 4 - 4 - 7 - 11$	0,8496	0,8199	0,7715	**0,8537**	**0,8206**	0,7788
$7 \times 3 - 3 - 3 - 9$	0,8290	0,8015	0,7819	**0,8318**	**0,8119**	0,7836
$7 \times 3 - 3 - 5 - 17$	0,8315	0,8118	0,7807	**0,8402**	**0,8192**	0,7835
$7 \times 3 - 3 - 7 - 21$	0,8297	0,8009	0,7814	**0,8324**	**0,8083**	0,7890

Selon les résultats présentés dans le tableau 5.1 qui montre le meilleur hypervolume (H_{max}), la valeur moyenne ($\overline{H}$) et la valeur de la métrique la plus mauvaise (H_{min}) obtenue dans les différents scénarii. A partir de ces résultats, nous pouvons dire que l'algorithme qui présente de meilleurs résultats dans l'optimisation de nos trois objectifs concernés par le problème, c'est-à-dire que le volume compris entre le point de référence et le meilleur front obtenu par la meilleure solution est plus grand dans le NSGA - III. Cependant, les valeurs moyennes entre une méthode et l'autre sont proches pour les instances de petites tailles testées. De la même façon, nous présentons un exemple d'une solution graphique pour comparer les deux algorithmes (figures 5.16 et 5.17). Dans l'exemple, nous pouvons noter un point de référence qui est la plus mauvaise solution dans l'espace objectif, les points en vert représentent le meilleur front de Pareto trouvé par l'algorithme NSGA - II (16 solutions) et les points en violet le meilleur front de l'algorithme NSGA - III (18 solutions).

5.7.2.2 Instances de moyennes - grandes tailles

Pour ce type d'instances, nous employons la même notation que pour les instances de petite taille : $M\_\omega - \lambda - N_W - N_S - N_K$. Nous avons $N = 10$, 30 ou 50 et $M = 5$, 10 et 15 (tâches et machines respectivement). Le nombre d'opérateurs, le nombre de machines

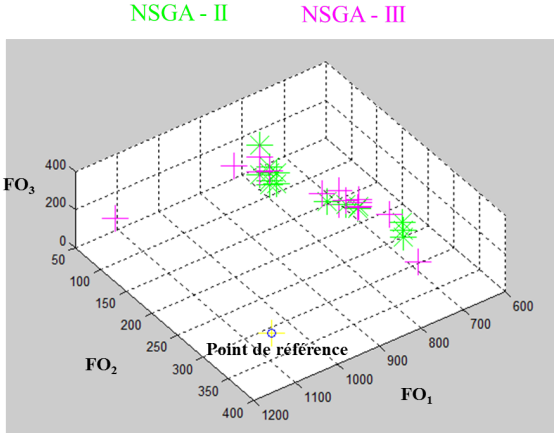


FIGURE 5.16 – Représentation graphique d'un scénario $3 \times 3 - 3 - 3 - 3$

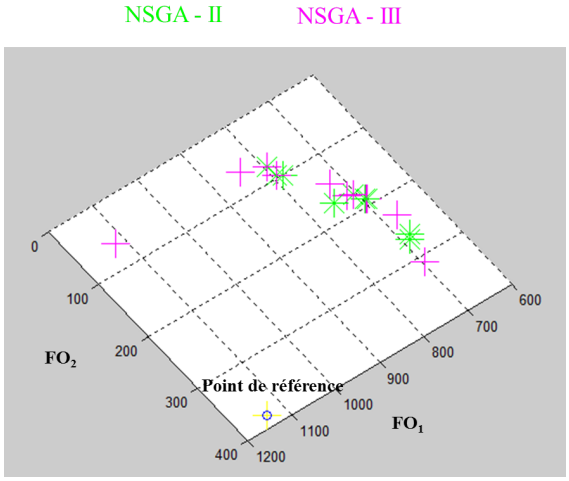


FIGURE 5.17 – Représentation graphique d'un scénario $3 \times 3 - 3 - 3 - 3$

et le nombre de compétences sont choisis de façon aléatoire entre 3, 5 et 7. Neuf types de problèmes sont générés de façon aléatoire pour chaque scénario, où chaque problème a été lancé dix fois. Les paramètres ω et λ sont choisis de façon aléatoire entre 50, 75 ou 100 (ω_1, ω_2, ω_3) et 0.1, 0.3 ou 0.5 ($\lambda 1$, λ_2, λ_3), respectivement.

Comme déjà mentionné dans des paragraphes précédents, nous gardons les mêmes paramètres déjà présentés pour les deux algorithmes dans les instances de petites tailles.

Dans le tableau 5.2, nous exposons les valeurs calculées de l'hypervolume pour les différents environnements générés. Nous exposons la meilleure, la moyenne et la plus mauvaise valeur d'hypervolume dans le scénario testé. Au moment de faire la comparaison entre les deux algorithmes, la méthode du NSGA - III présente les meilleurs résultats, autrement dit, cet algorithme montre une progression importante de la région abordée par les solutions qui appartiennent au premier front Pareto en prenant un point de référence (mauvaise solution initiale). Également, nous pouvons constater que pour les instances qui ont moins de 150 opérations ($N \times M$), la distance entre les valeurs moyennes de l'hypervolume des deux algorithmes est plus petite, mais chaque fois que nous augmentons la taille des opérations le NSGA - III devient plus performant que le NSGA - II pour résoudre notre problème d'ordonnancement de type open shop à trois objectifs.

5.8 Conclusion

Dans ce chapitre nous avons considéré le problème d'ordonnancement de type open shop $O_m|r_i|\sum F_i$ qui a été traité pendant ce mémoire, mais en ajoutant plus de critères ou fonc-

TABLE 5.2 – La meilleure, la moyenne et la plus mauvaise valeur de l'hypervolume obtenue par le NSGA-II et le NSGA-III - grande taille

Problème	NSGA - II			NSGA - III		
	H_{max}	$\overline{H}$	H_{min}	H_{max}	$\overline{H}$	H_{min}
$10 \times 5\_50 - 0.1 - 5 - 3 - 15$	0,7798	0,7602	0,7518	**0,7901**	**0,7790**	0,7598
$10 \times 5\_75 - 0.3 - 5 - 5 - 35$	0,7715	**0,7528**	0,7362	**0,7732**	0,7499	0,7297
$10 \times 5\_100 - 0.5 - 5 - 3 - 42$	**0,7854**	0,7631	0,7401	0,7807	**0,7687**	0,7482
$10 \times 10\_50 - 0.1 - 10 - 5 - 25$	0,7801	0,7523	0,7388	**0,7922**	**0,7618**	0,7435
$10 \times 10\_75 - 0.5 - 10 - 3 - 70$	**0,7810**	**0,7684**	0,7515	0,7790	0,7596	0,7468
$10 \times 10\_100 - 0.1 - 10 - 7 - 55$	0,7935	**0,7732**	0,7586	**0,7986**	0,7681	0,7422
$30 \times 5\_50 - 0.1 - 5 - 3 - 100$	0,7627	0,7438	0,7209	**0,7893**	**0,7691**	0,7438
$30 \times 5\_75 - 0.5 - 5 - 5 - 123$	0,7835	0,7511	0,7291	**0,7950**	**0,7780**	0,7507
$30 \times 5\_100 - 0.1 - 5 - 7 - 19$	0,7854	0,7609	0,7492	**0,8132**	**0,7834**	0,7480
$30 \times 15\_50 - 0.5 - 15 - 3 - 62$	0,7215	0,6967	0,6699	**0,8001**	**0,7810**	0,7690
$30 \times 15\_75 - 0.1 - 15 - 5 - 99$	0,7402	0,7115	0,6898	**0,7865**	**0,7790**	0,7566
$30 \times 15\_100 - 0.3 - 15 - 3 - 160$	0,7317	0,7018	0,6797	**0,7821**	**0,7609**	0,7410
$50 \times 5\_50 - 0.1 - 5 - 3 - 182$	0,7932	0,7568	0,7225	**0,8199**	**0,7810**	0,7583
$50 \times 5\_75 - 0.5 - 5 - 5 - 45$	0,7819	0,7458	0,7167	**0,8102**	**0,7794**	0,7439
$50 \times 5\_100 - 0.1 - 5 - 7 - 138$	0,7928	0,7628	0,7428	**0,8367**	**0,7836**	0,7418
$50 \times 10\_50 - 0.1 - 10 - 3 - 200$	0,7198	0,6874	0,6508	**0,7987**	**0,7225**	0,6815
$50 \times 10\_75 - 0.5 - 10 - 7 - 120$	0,7030	0,6691	0,6131	**0,7624**	**0,7137**	0,6718
$50 \times 10\_100 - 0.1 - 10 - 5 - 317$	0,7152	0,6598	0,6027	**0,8049**	**0,7312**	0,6816

tions objectif à optimiser. Nous avons décidé d'étudier de la même façon que pour les deux chapitres précédents, la minimisation du temps total de séjour et en plus, nous avons travaillé sur l'équilibrage de charge des ressources humaines en parlant des opérateurs qui sont affectés selon leurs compétences et aussi l'utilisation des machines dans l'ordonnancement proposé $O_m|r_i|\sum F_i, \Delta E_{RH}, \Delta E_{RM}$.

Nous avons traité ce problème par deux méthodes étudiées dans la littérature : le NSGA - II et le NSGA - III parce qu'elles sont reconnues par la communauté comme une référence parmi les MOEA. Nous avons réalisé les adaptations respectives à notre situation problématique afin d'obtenir les solutions les plus performantes. A partir de différents tests, nous avons constaté que pour les instances de petites tailles, les deux méthodes montrent une performance similaire où le NSGA - III présente un hypervolume moyen légèrement majeur au NSGA - II. Concernant les autres résultats, l'algorithme NSGA - III est plus performant

que le NSGA - II à mesure que nous augmentons la taille des instances du problème. Nous pouvons donc dire que cette méthode est mieux adaptée pour résoudre des problèmes qui ont plus de 2 objectifs (notre cas d'étude) puisque la procédure de positionnement des solutions dans un espace de référence décrit par un hyperplan et l'existence de points de référence fait une sélection plus pertinente des individus.

Les deux méthodes ont été comparées et analysées grâce au critère de comparaison de front de Pareto, l'hypervolume qui calcule la progression d'un front de Pareto qui est considéré comme le meilleur ensemble de solutions connues par rapport à un point de référence initial.

Nous pourrions proposer une recherche ou bien un réglage dynamique des paramètres avec l'utilisation des contrôleurs à logique floue afin de pouvoir encore améliorer les résultats. Nous pourrions aussi bien essayer d'autres relations de dominance qui contrôlent encore mieux la sélection des solutions et améliorent les algorithmes. De la même façon, nous pourrions essayer d'hybrider les algorithmes NSGA - II et NSGA - III à l'aide de la dominance de Lorenz afin de vérifier quelles nouvelles améliorations nous pouvons obtenir.

Chapitre 6

Application Industrielle

Dans les chapitres précédents, nous avons étudié un problème d'ordonnancement avec des contraintes de ressources liées aux compétences des opérateurs et les disponibilités de l'outillage à utiliser pour exécuter une opération. Nous avons exposé différentes méthodes de résolution pour aborder notre problématique et nous avons testé plusieurs instances théoriques qui ont été générées selon les paramètres décrits dans la littérature.

A partir des résultats obtenus, nous nous intéressons maintenant à la prochaine étape de notre projet de recherche, l'application des méthodes proposées sur un atelier de production réel (l'entreprise Norelem) qui suit les aspects qui ont été présentés dans les différentes parties de ce mémoire.

6.1 Introduction

Le monde industriel porte un intérêt sur le développement d'outils capables de représenter et proposer des solutions pour les différentes problématiques existantes dans divers domaines. Les champs de la recherche et développement deviennent plus demandés afin de trouver de nouvelles technologies et/où méthodes qui contribuent à améliorer les processus au sein de la société (transport, production, etc.) en augmentant son bénéfice.

La recherche opérationnelle se présente comme une alternative de solutions qui s'occupe de l'étude de ces types de problèmes, en proposant les mécanismes pour les résoudre. Les trois niveaux décisionnels à considérer par les entreprises : stratégique, tactique et opérationnel montrent une grande quantité de situations et aspects à prendre en compte au moment d'aborder une situation problématique.

Concernant les entreprises qui ont une relation avec la production, nous pouvons constater

qu'il y a toujours une interconnexion entre les différentes zones de travail qui commencent avec la commande du client, passe par le système de production et finalement inclut les processus de livraison de produits aux clients.

Dans ce cadre, dans les étapes de résolution de notre problème, nous avons fait un analyse du niveau tactique de l'atelier de la société Norelem avant de traiter le sujet d'ordonnancement avec l'objectif de vérifier la configuration physique existante de l'atelier. Ce processus a été fait en prenant une méthode existante dans la littérature proposée par Yalaoui *et al.* [182] qui a montré de très bons résultats dans une application d'un cas industriel également. Cette méthode propose trois étapes, où la première s'occupe de regrouper les machines dans différentes cellules de proximité technologique (maximiser le volume de fabrication à l'intérieur des cellules en utilisant un algorithme génétique). La deuxième partie porte sur l'affectation des machines dans les positions existantes dans les cellules (algorithme à colonie de fourmis). Finalement, la troisième étape évalue la qualité de la solution en considérant des aspects de l'étape une et deux (nombre de cellules à créer, nombre maximum et minimum de machines à allouer par cellule)

Une fois que nous avons appliqué la méthode, nous avons comparé le résultat avec la configuration de l'atelier ainsi qu'avec une configuration proposée par les experts Norelem (chef d'atelier et techniciens) et nous avons trouvé que les changements physiques proposés n'étaient pas assez significatifs pour réaliser un déménagement de l'atelier. Cependant, nous avons constaté que la configuration existante de l'atelier Norelem possède une bonne distribution des machines. De cette manière, nous avons continué avec un problème opérationnel, l'ordonnancement

6.2 Problème opérationnel : ordonnancement

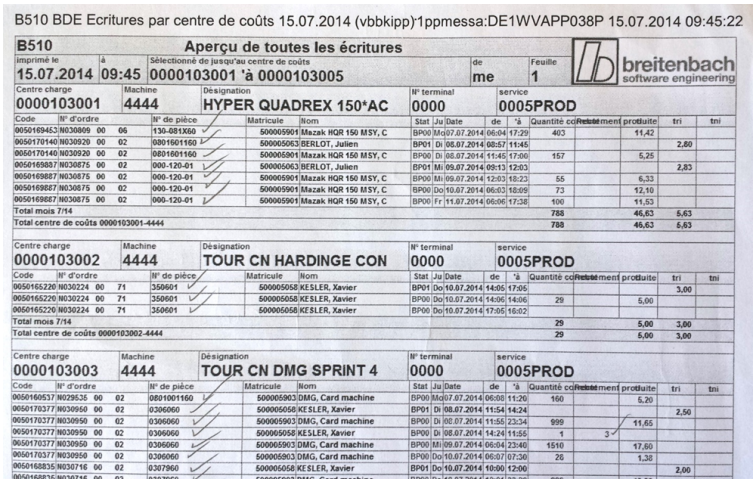
Nous avons abordé l'ordonnancement de la production de l'atelier comme un problème qui demande une réponse très rapide. La prise en compte des différentes contraintes de ressources (humaines et d'outillage) pour générer des solutions faisables et de bonne qualité.

Nous avons déjà exposé les différentes méthodes proposées pour résoudre cette problématique dans les chapitres 3 et 4. Dans cette section, nous allons présenter la démarche pour appliquer les méthodes sur les instances du cas réel, l'interaction avec la base de données de l'entreprise (ERP - logiciel B2), la collecte de données, le classement des données et le développement d'un logiciel qui est en cours d'élaboration.

6.2.1 Collecte de données à Norelem

Comme il a été décrit dans le chapitre 1, l'entreprise Norelem a un ERP appelé B2 qui gère les différentes données concernant le traitement des ordres de fabrication (temps opératoires, temps de montage, dates de disponibilité des tâches, etc.). Dans un premier temps, et afin de tester les méthodes de résolution développées, nous avons fait la collecte des données demandées pour exécuter nos algorithmes, grâce à l'aide du chef d'atelier de la société.

La figure 6.1 montre une extraction des données que l'on peut récupérer, où les différentes tâches sont classées par machines, l'heure et la date de finalisation qui est affichée selon le système de code-barre existant dans l'entreprise. Chaque opérateur fait le scan de la tâche qu'il va exécuter.


B510 BDE Ecritures par centre de coûts 15.07.2014 (vbbkipp)-1ppmessa:DE1WVAPP038P 15.07.2014 09:45:22

B510 — Aperçu de toutes les écritures

imprimé le 15.07.2014 | à 09:45 | Sélectionné de jusqu'au centre de coûts 0000103001 'à 0000103005 | de me | Feuille 1 | breitenbach software engineering

Centre charge	Machine	Désignation	N° terminal	service
0000103001	4444	HYPER QUADREX 150*AC	0000	0005PROD

Code	N° d'ordre		N° de pièce		Matricule	Nom	Stat	Ju	Date	de	'à	Quantité co	Retouche	ment	produite	tri	tni
0050169453	N030809	00	06	130-081X60	500005901	Mazak HQR 150 MSY, C	BP00	Mo	07.07.2014	06:04	17:29	403			11,42		
0050170140	N030920	00	02	0801601160	500005063	BERLOT, Julien	BP01	Di	08.07.2014	08:57	11:45					2,60	
0050170140	N030920	00	02	0801601160	500005901	Mazak HQR 150 MSY, C	BP00	Di	08.07.2014	11:45	17:00	157			5,25		
0050169887	N030875	00	02	000-120-01	500005063	BERLOT, Julien	BP01	Mi	09.07.2014	09:13	12:03					2,83	
0050169887	N030875	00	02	000-120-01	500005901	Mazak HQR 150 MSY, C	BP00	Mi	09.07.2014	12:03	18:23	55			6,33		
0050169887	N030875	00	02	000-120-01	500005901	Mazak HQR 150 MSY, C	BP00	Do	10.07.2014	06:03	18:09	73			12,10		
0050169887	N030875	00	02	000-120-01	500005901	Mazak HQR 150 MSY, C	BP00	Fr	11.07.2014	06:06	17:38	100			11,53		
Total mois 7/14												788			46,63	5,63	
Total centre de coûts 0000103001-4444												788			46,63	5,63	

Centre charge	Machine	Désignation	N° terminal	service
0000103002	4444	TOUR CN HARDINGE CON	0000	0005PROD

Code	N° d'ordre		N° de pièce		Matricule	Nom	Stat	Ju	Date	de	'à	Quantité co	Retouche	ment	produite	tri	tni
0050165220	N030224	00	71	350601	500005058	KESLER, Xavier	BP01	Do	10.07.2014	14:05	17:05					3,00	
0050165220	N030224	00	71	350601	500005058	KESLER, Xavier	BP00	Do	10.07.2014	14:06	14:06	29			5,00		
0050165220	N030224	00	71	350601	500005058	KESLER, Xavier	BP00	Do	10.07.2014	17:05	18:02						
Total mois 7/14												29			5,00	3,00	
Total centre de coûts 0000103002-4444												29			5,00	3,00	

Centre charge	Machine	Désignation	N° terminal	service
0000103003	4444	TOUR CN DMG SPRINT 4	0000	0005PROD

Code	N° d'ordre		N° de pièce		Matricule	Nom	Stat	Ju	Date	de	'à	Quantité co	Retouche	ment	produite	tri	tni
0050160537	N029535	00	02	0801001160	500005903	DMG, Card machine	BP00	Mo	07.07.2014	06:00	11:20	160			5,20		
0050170377	N030950	00	02	0306060	500005058	KESLER, Xavier	BP01	Di	08.07.2014	11:54	14:24					2,50	
0050170377	N030950	00	02	0306060	500005903	DMG, Card machine	BP00	Di	08.07.2014	11:55	23:34	999			11,65		
0050170377	N030950	00	02	0306060	500005058	KESLER, Xavier	BP00	Di	08.07.2014	14:24	11:55	1	3				
0050170377	N030950	00	02	0306060	500005903	DMG, Card machine	BP00	Mi	09.07.2014	06:04	23:40	1510			17,60		
0050170377	N030950	00	02	0306060	500005903	DMG, Card machine	BP00	Do	10.07.2014	06:07	07:30	26			1,38		
0050168835	N030716	00	02	0307960	500005058	KESLER, Xavier	BP01	Do	10.07.2014	10:00	12:00					2,00	

FIGURE 6.1 – Exemple de données ERP - B2 Norelem

FIGURE 6.2 – Centre d'usinage

Nous avons fait une collecte de données sur le terrain (figure 6.3) afin de vérifier les

temps théoriques enregistrés dans l'ERP B2 de l'entreprise. Nous avons suivi les différents ordres de fabrication dans l'atelier. Nous avons constaté que les temps théoriques sont bien définis pour les opérations où les machines sont plus automatisées, c'est-à-dire, les centres d'usinage et les tours à commande numérique. Des opérations comme la peinture, le montage et l'utilisation des rectifieuses, des fraiseuses, des scies à ruban traditionnelles, des perceuses à colonne, entre autres, présentent des temps opératoires et de montage qui sont différents par rapport aux temps théoriques de l'ERP-B2.

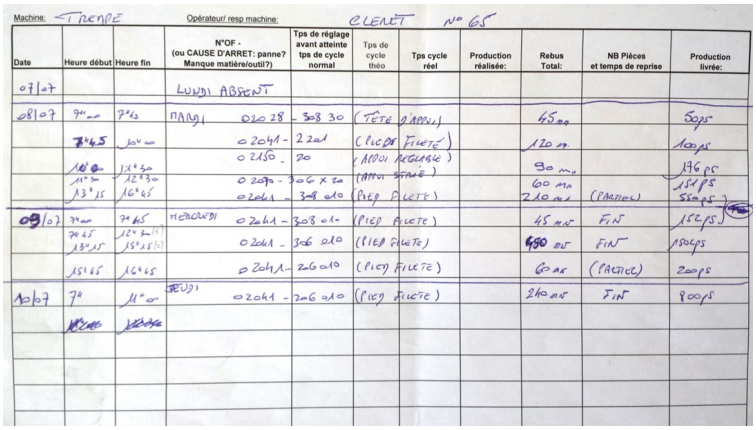


FIGURE 6.3 – Collecte de données sur le terrain

De plus, une matrice avec les différentes compétences maitrisées par opérateur a été mise à jour avec la participation du chef d'atelier (figure 6.4). Cette matrice présente l'ensemble des opérateurs et l'ensemble des machines de l'entreprise, les compétences sont déterminées par la capacité de l'opérateur à exécuter une tâche sur la machine, c'est-à-dire, un opérateur a une compétence s'il peut réaliser l'opération sur la machine demandée, dans le cas contraire il ne possède pas la compétence. Des collectes de données sur les différents types d'outillage ont aussi été faites (figure 6.5).

De cette manière, la mise en place des méthodes d'ordonnancement présentées dans les chapitres précédents dépend de plusieurs facteurs : la qualité des données (collecte de données - ERP de l'entreprise) et de la structure du système d'information (intégration d'outils de décision dans l'activité de l'atelier).

6.2.2 Analyse et test des données

Les données récupérées ont été organisées dans un tableau (figure 6.6) afin de faciliter la lecture des données par les algorithmes de résolution développés.

Pendant la collecte de données sur le terrain nous avons remarqué des temps de montage qui ont été diminués grâce à l'intégration de nouvelles technologies de montage magnétique

	SCIE A RUBAN METORA *ACTUELL MIN* (fonte)	SCIE A RUBAN PETITE KASTO (Fonctional A)	SCIE A RUBAN GRANDE KASTO (TWIN AE7)	SCIE KASTO SEMI AUTOMATIQUE	TOUR CN REALMECA *MINUTES*	TOUR CN HYPERQUADREX MAZAK	TOUR CN HARDINGE CONQUEST 51 *ACTUELL MI	TOUR CN DMG SPRINT 42 LINEAR *ACTUELL MI	TOUR CN MAZAK QTN 250-II	TOUR CN MAZAK QTN 100-II	TOUR CN QT15 MAZAK	TOUR DE REPRISE HAULIN *MINUTES*	TOUR // RAMO *MINUTES*	FRAISEUSE A BRIDES + STRIES	FRAISEUSE VERNIER *ACTUELL MIN*	FRAISEUSE UNIV DUFOUR *ACTUELL MIN*	FRAISEUSE RECKERMANN	FRAISEUSE 244 GROSSE *ACTUELL MIN*	Fraiseuse verticale GRAFFENSTADEN	FRAISEUSE VERTICALE SAJO *ACTUELL MIN*	CU HORIZ. MAKINO A 61*MINUTES*	CU VERTICAL MAZAK 510C *MINUTES*	CU HORIZONT. MAKINO A61NX ET PALETTISEUR	PERCEUSE ALZMETAL *MINUTES*	PERCEUSE RADIALE *MINUTES* (cotè A61NX)	PERCEUSES TARAUDEUSES *MINUTES*	TONNEAU **ACTUELL MIN*	RECTIF. PLANE G.E.R. *MINUTES*	RECTIF. PLANE SHIPMAN *MINUTES*	POINTEUSE HAUSER *MINUTES* (pointage perçage)	RODEUSE CYL. SUNNEN *MINUTES*	EMBALLAGE CONDITIONNEMENT *MINUTES*	GRAVURE (REFERENCES) *MINUTES*	MONTAGE *MINUTES*	POSTE DE SOUDURE TIG *MINUTES*	PRESSE BURGER	PEINTURE VERNISSAGE * ACTUELL MIN*	MACHINE DE TREMPE *ACTUELL MIN*	MAIN D'OEUVRE *ACTUELL MIN*	PERCEUSE CINCINNATI POUR 0714 *MINUTES*	PERCEUSE CINCINNATI *MINUTES*	FOUR *MINUTES*	PERCEUSE ROCHELET *MINUTES*	MONTAGE KIPP *MINUTES*	PRESSE SCHMIDT	CONDITIONNEMENT *MINUTES*	MISE EN STOCK *ACTUELL MIN*	PREPARATION *ACTUELL MIN*	EBAVURAGE/BROSSAGE *ACTUELL MIN*	EMBALLAGE *ACTUELL MIN*	TREMPE AU CHALUMEAU *ACTUELL MIN*
	102002	102004	102005	109021	104010	103001	103002	103003	103005	103004	104006	104003	104005	101001	101007	101009	101012	101014	101017	101016	105001	105002	107001	109001	109007	109015	102006	108004	108005	109008	108006	109004	109006	109009	109017	109022	101003	109012	110001	109002	109003	109013	109016	109020	109024	110002	110003	110004	110005	110006	110007
Opérateur 1																																																			
Opérateur 2																																																		
Opérateur 3																																																		
Opérateur 4																																																		
Opérateur 5																																																		
Opérateur 6																																																		
Opérateur 7																																																		
Opérateur 8																																																		
Opérateur 9																																																		
Opérateur 10																																																		
Opérateur 11																																																		
Opérateur 12																																																		
Opérateur 13																																																		
Opérateur 14																																																		
Opérateur 15																																																		
Opérateur 16																																																		
Opérateur 17																																																		
Opérateur 18																																																		
Opérateur 19																																																		

FIGURE 6.4 – Matrice de compétences maitrisées par opérateur

FIGURE 6.5 – Outils Norelem

Date	Heure début	Heure fin	Référence	Machine	Temps de montage (heure)	Temps usinage (heure)	Quantité	Opérateur	Rebus
Lundi 07/07/14	7h50	16h45	02010-081	108004	0,25	10	1184	Pascal RENARD	
Mardi 08/07/14	7h40	8h40		108004					
Mardi 08/07/14	10h20	10h20		110001		5	1183	Christine LACOUR	
Mardi 08/07/14	10h20	10h20		110003		0,1	1183	Christine LACOUR	
Lundi 07/07/14	12h30	12h30	02010-08	104010		7	508	Christophe FAVRE	6
Mercredi 09/07/14	8h41	8h41		104010		8	936	Christophe FAVRE	32
Jeudi 10/07/14	11h15	11h30		108004	0,25			Pascal RENARD	
Jeudi 10/07/14	11h15	11h15		108004		12	1444	Pascal RENARD	
Vendredi 11/07/14	7h53	7h53		110001		7	1415	Christine LACOUR	
Vendredi 11/07/14	7h54	7h54		110003		0,02	1415	Christine LACOUR	
Mardi 08/07/14	6h30	7h15	024600602002	104010	0,75			Christophe FAVRE	
Mardi 08/07/14	7h15	12h14		104010		4,98	252	Christophe FAVRE	1
Mardi 08/07/14	12h58	13h43		104010	0,75		2	Christophe FAVRE	1
Mercredi 09/07/14	11h58	11h58		104010		6	247	Christophe FAVRE	

FIGURE 6.6 – Tableau des données à tester

qui permettent de gagner beaucoup de temps de bridage des pièces (figures 6.7 et 6.8). Nous avons identifié que pendant les processus de production qui suit un ordre de fabrication, il y

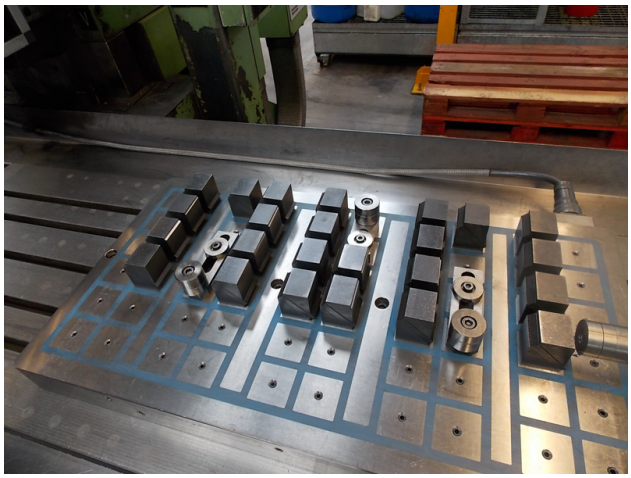


FIGURE 6.7 – Montage - plateau magnétique

FIGURE 6.8 – Pièce sur le plateau magnétique

a des situations qui arrivent, qui augmentent les temps opératoires, le nombre des opérations dans la production ou qui génèrent du retard : l'indisponibilité des machines (panne, occupée dans une autre opération), manque de la matière première, déformation de la pièce après une opération qui a par conséquence d'ajouter des opérations pour la corriger. C'est pour cette raison que le développement d'un outil d'aide à la décision sera un excellent mécanisme pour anticiper ces inconvénients et réagir d'une façon plus rapide sur le chemin.

Dans ce cadre, les deux méthodes développées, les plus performantes (ACO et ACO-FLC) ont été testées avec des données réelles de l'entreprise. Dans un premier temps, nous l'avons pris en compte que les contraintes d'affectation de ressources humaines (multi-compétences des opérateurs).

Nous présentons un test réalisé avec l'algorithme à colonie de fourmis (ACO) pour proposer l'ordonnancement d'une semaine de production réelle. Le cas de l'entreprise a été représenté par 36 machines, 113 références (tâches), 19 opérateurs et 217 opérations.

Les figures 6.9 et 6.10 montrent une représentation partielle (diagramme de Gantt) de l'ordonnancement actuelle et la solution proposée par l'ACO. Nous constatons que l'ACO améliore la solution en plaçant quelques opérations en avance et à partir des échanges sur l'affectation des opérateurs au moment d'exécuter une activité (les opérations entourées en pointillés sont bougées - violet et jaune).

Le tableau 6.1 montre la comparaison entre les deux méthodes de résolution proposées ainsi que la solution Norelem. Nous présentons le temps total de séjour en heures et le pourcentage d'amélioration obtenu par la méthode.

De la même façon, nous avons testé les méthodes proposées en ajoutant les contraintes

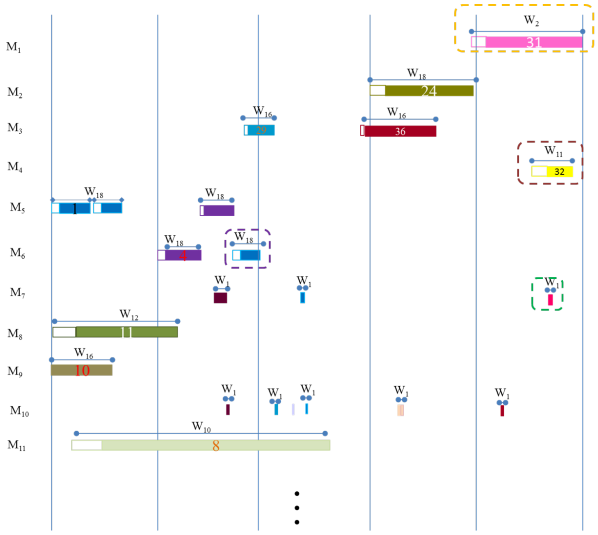


FIGURE 6.9 – Diagramme de Gantt - solution actuelle

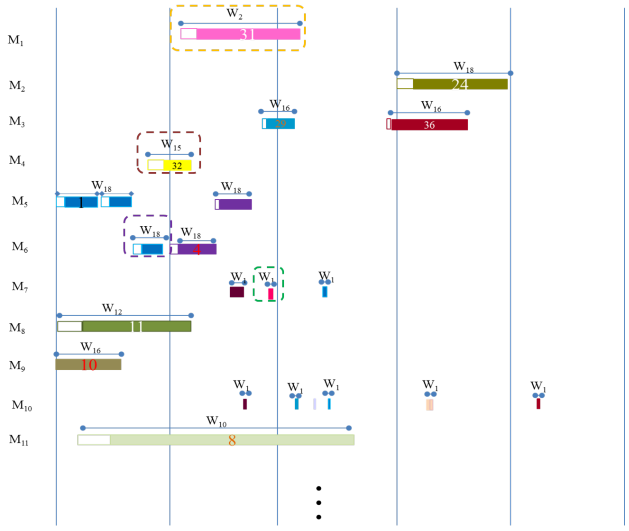


FIGURE 6.10 – Diagramme de Gantt - Solution de l'ACO

TABLE 6.1 – Test sur des instances réelles - ressources humaines

Nombre de machines	36
Nombre de tâches	113
Nombre des opérateurs	19
Nombres des opérations	217

Solution d'ordonnancement		
	Temps total de séjour (heures)	% d'amélioration
Norelem	2341,5	
ACO	2248,7	3,963
ACO-FLC	2226,8	**4,895**

par rapport à la disponibilité d'outillage, le tableau 6.2 montre la même instance des données réelles qu'on avait présentée avec une contrainte en plus : le nombre de types des outils. Nous pouvons vérifier que le pourcentage d'amélioration de la solution par rapport à l'existante diminue un peu (en comparaison avec le tableau 6.1), provoqué par l'ajout de la nouvelle contrainte d'outillage qui conditionne l'exécution des opérations sur les machines (il faut attendre que les outils soient libérés d'une opération à l'autre pour les utiliser).

L'algorithme à colonie de fourmis hybridé avec la logique floue a obtenu les meilleures solutions dans les deux cas du problème (un premier problème : gérer l'affectation de ressources humaines avec multi-compétences et un deuxième : gérer l'affectation de ressources

TABLE 6.2 – Test sur des instances réelles - ressources humaines + outillage

Nombre de machines	36
Nombre de tâches	113
Nombre des opérateurs	19
Nombres des opérations	217
Nombre de types des outils	50

Solution d'ordonnancement		
	Temps de séjour (heures)	% d'amélioration
Norelem	2341,5	
ACO	2291,2	2,148
ACO-FLC	2259,6	**3,498**

humaines et la disponibilité d'outillage en même temps).

A partir de là, nous avons suivi plusieurs semaines de production et nous avons testé les données en prenant en compte l'affectation des ressources humaines et l'outillage avec l'algorithme à colonie de fourmis hybridé avec la logique floue, ci-dessous nous présentons les pourcentages d'amélioration trouvés (tableau 6.3 et figure 6.11) :

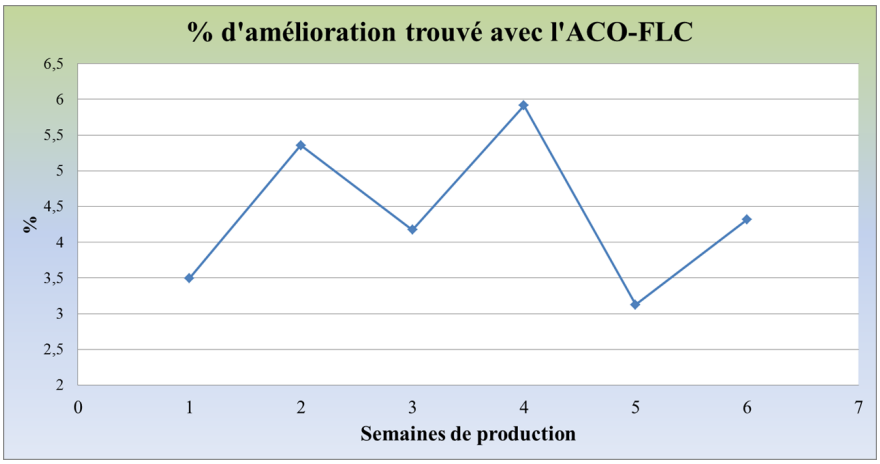


FIGURE 6.11 – Amélioration trouvé par L'ACO-FLC (%)

Les différents tests industriels ont été effectués avec 19 opérateurs, 36 machines, entre 37 et 190 références (tâches), entre 57 et 305 opérations ainsi qu'entre 50 et 125 types d'outils qui ont varié d'une semaine à une autre (la commande client est différente alors le nombre d'ordres de fabrication en cours pendant une semaine est aussi différent)

Nous pouvons constater que selon les résultats obtenus, il y a des améliorations qui sont

TABLE 6.3 – Tableau d'amélioration trouvé par l'ACO-FLC

	Semaines de Production	**% d'amélioration ACO-FLC**
1	01/06/15 - 05/06/15	3,498
2	15/06/15 - 20/06/15	5,358
3	22/06/15 - 27/06/15	4,172
4	06/07/15 - 10/07/15	5,913
5	13/07/15 - 17/07/15	3,124
6	20/07/15 - 24/07/15	4,318

faibles puisque sur des semaines avec une quantité de production plus faible que d'autres, donc il n'est pas possible d'améliorer dans un grand pourcentage le flux de la production.

6.2.3 Logiciel en cours de développement

Comme une étape suivante de ce projet, la mise en œuvre des méthodes de résolution dans l'entreprise demande un processus rapide, facile à comprendre et qui ajoute une réactivité interactive dans toutes les étapes de la proposition d'une solution d'ordonnancement de production. Dans ce cadre, un logiciel est en cours de développement afin d'améliorer l'interface des résultats, sa représentation, des solutions plus graphiques et plus rapides à interpréter.

La figure 6.12 présente un premier dessin fait pour la collecte des données (macro en excel), où on récupère les informations des ordres de fabrication, la référence du produit, la quantité de pièces à produire, les dates de lancement.

Cette idée sera poursuivie par Norelem puisqu'il existe un grand intérêt d'intégrer cet outil d'aide à la décision dans le groupe de travail de l'atelier de production qui est composé du chef d'atelier, des opérateurs et des techniciens. Ce logiciel cherche à offrir un premier mécanisme pour augmenter le flux de la production de l'atelier en considérant les différentes contraintes existantes.

Dans un premier temps, le logiciel sera dédié à résoudre la problématique de l'entreprise, il aura à disposition les différentes méthodes de résolution proposées dans ce mémoire ainsi que des méthodes complémentaires à développer dans un futur proche. Comme une deuxième partie du projet, une boîte noire de méthodes pour résoudre les problèmes d'ordonnancement est envisagé, donc l'idée est d'enrichir le logiciel avec d'autres types de problèmes et à l'avenir d'essayer sa commercialisation comme un outil d'aide à la de décision dans le secteur de production. Il est néanmoins important de voir son positionnement sur le marché de solutions

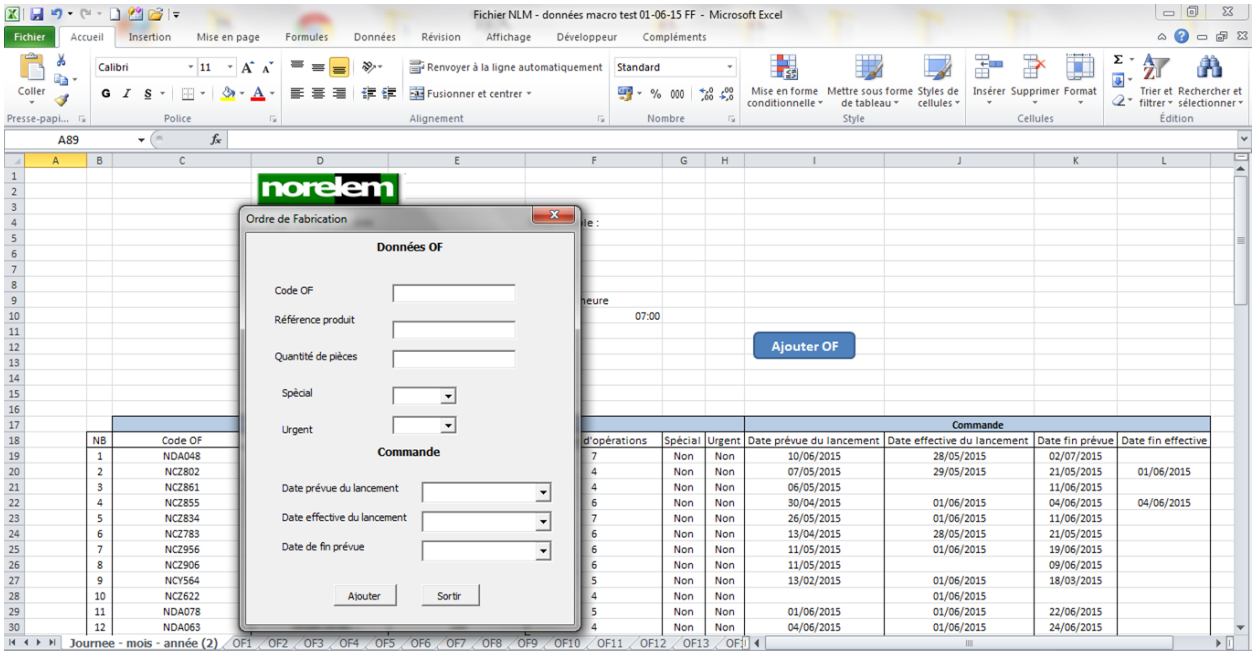


FIGURE 6.12 – Ordres de fabrication à ordonnancer

existantes.

6.3 Conclusion

Dans une première partie, nous nous sommes intéressés à une problématique d'organisation de l'atelier de la société Norelem. Nous avons analysé les différents flux de production et grâce à l'état des lieux trouvé dans l'entreprise, nous avons trouvé le besoin d'améliorer le flux de la production en cherchant la minimisation des retards et la maximisation de la satisfaction des clients. Dans ce processus nous sommes passés par une étape d'analyse de la configuration de l'atelier de Norelem avant de nous occuper du sujet initial de ce mémoire, liée à l'ordonnancement d'un atelier de type open shop.

Pour l'agencement de l'atelier de l'entreprise, nous avons repris la méthode proposée par Yalaoui *et al.* [182] afin de proposer une nouvelle configuration d'atelier. Cette approche a trouvé une amélioration pour l'atelier mais après les analyses, nous avons convenu que ces changements physiques ne seraient pas si significatifs pour l'ordonnancement de la production au regard du coût.

Ensuite, à ce premier sujet de niveau tactique, nous avons traité une problématique de niveau opérationnel avec l'ordonnancement des tâches de production. Des méthodes de résolution approchées ont été développées. Des données réelles ont été collectées et organisées

pour appliquer les méthodes proposées. Les résultats obtenus ont montré une amélioration entre 2% et 5% de la fonction objectif (temps total de séjour) pour les semaines qui sont plus moins chargés en travail, c'est-à-dire que la demande de production n'est pas très haute, il n'y a pas un grand nombre d'ordres de fabrication en cours.

De plus, un logiciel est en cours de développement afin de mieux gérer les méthodes de résolution proposées et faciliter l'interface avec la collecte de données de l'entreprise. Cet outil d'aide à la décision permettra d'ordonnancer les tâches de production de l'atelier d'une façon rapide, facile et interactive. Il sera également prochainement enrichi avec une boîte noire des différentes méthodes pour générer un logiciel qui puisse être commercialisé et mis au service des entreprises industrielles du même domaine de production en particulier.

Conclusions et perspectives

Avec plus de 70 ans d'activités dans la conception et la fabrication des pièces de précision mécanique, Norelem est un élément clef du patrimoine industriel français grâce à un savoir-faire reconnu par l'ensemble de ses clients. La société Norelem a fait appel à l'Université de Technologie de Troyes afin d'optimiser au mieux le flux de production dans son atelier. Cette thèse s'inscrit dans le cadre d'une thèse CIFRE. La direction générale et l'atelier de la société sont basés à Fontaine Les Grès (10). L'objectif principal de cette thèse est de développer des outils efficaces d'aide à la décision afin que la société soit plus compétitive et plus rapide pour satisfaire les exigences de ses clients.

Dans cette thèse, nous avons travaillé sur plusieurs niveaux de décision dans le cas d'une problématique industrielle. Nous nous sommes intéressés plus précisément à résoudre un problème d'ordonnancement. Dans un premier temps, nous avons fait une analyse d'agencement afin de trouver de possibles améliorations physiques dans la configuration de l'atelier avant d'aborder le sujet d'ordonnancement de la production. Cette étape a donné des résultats assez peux significatifs pour motiver un déménagement de l'atelier et nous avait encouragé à trouver des solutions plutôt de niveau opérationnel, l'ordonnancement de l'atelier.

Le premier chapitre a permis de présenter le contexte général de l'étude. Une introduction générale de la société Norelem ainsi qu'une présentation rapide de sa mission et ses services ont été fait. Le problème d'ordonnancement identifié et l'état de lieu de l'entreprise ont été détaillés.

Dans le deuxième chapitre de cette thèse, un rappel des problèmes d'ordonnancement a été exposé. Une description sur le problème d'ordonnancement de type open shop, des méthodes de modélisation et d'évaluation de performances ont aussi été présentées. De plus, une définition de l'optimisation et de ses techniques a été donnée. Ceci a permis de voir l'intérêt grandissant des chercheurs et des industriels à ce domaine et les différents outils disponibles à ce jour pour résoudre ces problématiques. Concernant les méthodes d'optimisation, il existe deux grandes familles d'approches. L'approche exacte qui est souvent utilisée, en sachant qu'il y a une contrainte importante de tailles de problème. Une deuxième famille de mé-

thodes approchées non exactes mais assez efficaces est de plus en plus employée et appréciée par le monde industriel. Celle-ci comprend des heuristiques dédiées à des problématiques spécifiques ainsi que des métaheuristiques qui sont considérées comme des méthodes efficaces et génériques.

Le troisième chapitre présente des modèles mathématiques non-linéaires et linéaires développés pour résoudre un problème d'ordonnancement de type open shop. Celui-ci prend en compte l'affectation de ressources humaines avec multi-compétences et la disponibilité d'outillage. Le modèle non-linéaire a été proposé pour réduire le nombre d'indices utilisés dans les variables de décision (maximum 3) et diminuer le nombre de variables ainsi que le temps de calcul. Des modèles linéaires avec et sans discrétisation du temps ont été exposés. Le modèle le plus performant a été le linéaire sans discrétisation du temps.

Le quatrième chapitre était consacré à l'étude des méthodes approchées pour résoudre notre problématique. Dans un premier temps, nous n'avons considéré que la contrainte d'affectation des ressources humaines avec multi-compétences, puis nous avons ajouté la contrainte de disponibilité d'outillage dans la résolution de nos méthodes. Ce chapitre présente des méthodes basées sur l'algorithme génétique, l'algorithme à colonie de fourmis et l'algorithme à colonie de fourmis hybridé avec la logique floue.

L'étude présentée dans les chapitre 3 et 4 permet de conclure que l'ACO-FLC est la méthode approchée la plus efficace pour résoudre ce type de problème. Ceci est due à son mode de fonctionnement en deux étapes (ordonnancement et puis l'affectation des ressources humaines) et l'utilisation des contrôleurs de floue pour affecter des valeur dynamiques aux paramètres.

Le cinquième chapitre décrit des méthodes multi-objectif pour résoudre un problème d'ordonnancement de type open shop, où trois objectifs sont considérés : la minimisation du temps total de séjour, l'équilibrage de charge des opérateurs et des machines. Des méthodes comme le NSGA - II et le NSGA - III ont été adaptées à la problématique.

Enfin, le chapitre 6 présente l'application industrielle qui a permis de valider en contexte réel la contribution de cette thèse. C'est une étape très importante de l'étude car elle a permis de tester l'ensemble du travail théorique mené en amont.

Suite à la présentation des différents travaux réalisés tout au long de cette thèse, plusieurs pistes intéressantes peuvent être considérées dans le futur.

En ce qui concerne les méthodes de résolution, les points suivants peuvent être approfondis :

- L'amélioration des performances des algorithmes proposés.

- L'application de nouvelles heuristiques ou métaheuristiques.
- L'implémentation des autres méthodes d'optimisation multi-objectif, dominance de Lorenz ou des hybridations avec la logique floue.
- Le fait de considérer les contraintes des ressources humaines peut demander des développements additionnels dans la prévision et la modélisation de comportements humains (fatigue, état émotionnel, etc.) et des situations aléatoires qui arrivent dans un environnement de production.

Plus généralement, l'adaptation ou l'effet d'aborder des problèmes des cas réels avec métaheuristiques promet un champ exploratoire très important pour la recherche à venir. En effet, l'exploitation d'autres méthodes de résolution offre des perspectives intéressantes.

De la même façon, le partenaire industriel, l'entreprise Norelem, a décidé de continuer le projet avec le développement d'un logiciel qui permet d'offrir une interface plus dynamique et interactive au personnel de la société.

Les travaux présentés dans ce mémoire ont donné lieu à plusieurs publications et ont été conclus par la soumission d'un article dans une revue [33] ainsi que des communications à des conférences internationales [29], [32], [31] et une conférence nationale [30].

Bibliographie

[1] M. A. Abido. A niched pareto genetic algorihtm for multiobjective environmental/economic dispatch. *Electrical Power and Energy Systems*, 25 :97 – 105, 2003.

[2] J. O. Achugbue and F. Y. Chin. Scheduling the open shop to minimize mean flow time. *Society for Industrial and Applied Mathematics Journal on Computing*, 11 :709 – 720, 1982.

[3] A. Agnetis, G. Murgia, and S. Sbrilli. A job shop scheduling problem with human operators in handicraft production. *International Journal of Production Research*, 52 :3820 – 3831, 2014.

[4] J.J Aguirre-Solis. Tabu search algorithm for the open shop scheduling problem with sequence dependent setup times. In *Advance Simulation Technologies Conference, Vol. 1, The society for modeling and simulation international, Orlando, 30 march - 3 april*, 2003.

[5] F. Ahmadizar and Mehdi. A novel hybrid genetic algorithm for the open shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 62 :775 – 787, 2012.

[6] D. Alcaide, A. Rodriguez-Gonzalez, and J. Sicilia. A heuristic approach to minimize expected makespan in open shops subject to stochastic processing times and failures. *International Journal of Flexible Manufacturing Systems*, 17 :201 – 226, 2006.

[7] E. Anand and R. Panneerselvam. Literature review of open shop scheduling problems. *Intelligent Information Management*, 7 :33 – 52, 2015.

[8] M. Andresen, H. Bräsel, M. Mörig, J. Tusch, and F. Werner. Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. *Mathematical and Computer Modeling*, 48 :1279 – 1293, 2008.

[9] M. Andresen, H. Bräsel, M. Plauschin, and F. Werner. Using simulated annealing for open shop scheduling with sum criteria. In *In : Tan, C.M., Ed., Simulated Annealing, I-Tech Education and Publishing*, pages 49 – 76, 2008.

[10] C. Artigues and D. Feillet. A branch and bound method for the job-shop problem with sequence-dependent setup times. *Annals od Operations Research*, 159 (1) :135 – 159, 2008.

[11] D. Bai and L. Tang. Open shop scheduling problem to minimize makespan with release dates. *Applied Mathematical Modelling*, 37 :2008 – 2015, 2013.

[12] K. R. Baker. *Introduction to sequencing and scheculing.* Wiley, 1974.

[13] O. Bellenguez and E. Néron. Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In *Practice and Theory of Automated Timetabling V*. Springer Berlin Heidelberg, 2005.

[14] H. Belouadah, M.E. Posner, and C.N. Potts. Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics*, 36 :213 – 231, 1992.

[15] J. Blazewicz, W. Domschke, and E. Pesch. The job shop scheduling problem : conventional and new solution techniques. *European Journal of Operational Research*, 93 :1 – 33, 1996.

[16] J. Blazewicz, E. Pesch, M. Sterna, and F. Werner. Open shop scheduling problems with late work criteria. *Discrete Applied Mathematics*, 134 :1 – 24, 2004.

[17] C. Blum. Beam-aco hybrid ant colony optimization with beam search : an application to open shop scheduling. *Computers & Operations Research*, 32 :1565 – 1591, 2005.

[18] H. Bräsel, A. Herms, M. Mörig, T. Tautenhahn, J. Tusch, and F. Werner. Heuristic constructive algorithms for open shop scheduling to minimize mean flow time. *European Journal Research*, 189 :856 – 870, 2005.

[19] H. Bräsel, D. Kluge, and F. Werner. A polynomial algorithm for an open shop problem wiht unit processing times and tree constraints. *Discrete Applied Mathematics*, 59 :11 – 21, 1995.

[20] H. Bräsel, M. Tautenhahn, and P. Willenius. On the set of solution of open shop problem. *Annals of Operations Research*, 92 :241 – 263, 1999.

[21] J. Breit, G. Schmidt, and V.A. Strusevich. Two-machine open shop scheduling with an availability constraint. *Operation Research Letters*, 29 :65 – 77, 2001.

[22] J. Briet, G. Schmidt, and V.A. Strusevich. Non-preemptive two-machine open shop scheduling with non-availability constraints. *Mathematical Methods of Operations Research*, 57 :217 – 234, 2003.

[23] P. Brucker. *Scheduling Algorithms.* Springer, 2007.

[24] P. Brucker, J. Hurink, B. Jurisch, and B. Wöstmann. A branch & bound algorithm for the open-shop problem. *Discrete Applied Mathematics*, 76 :43 – 59, 1997.

[25] P. Brucker, B. Jurisch, and B. Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49 :107 – 127, 1994.

[26] P. Brucker, S. Knust, T.C. E. Cheng, and N.V. Shakhlevich. Complexity results for flow-shop and open-shop scheduling problems with transportation delays. *Annals of Operation Research*, 129 :81 – 106, 2004.

[27] B. Bullnheimer, R. F. Hartl, and C. Straub. A new rank based version of the ants system - a computational study. *Central European Journal for Operations Research and Economics*, 7 :25 – 38, 1997.

[28] B. Cai, S. Wang, and H. Hu. Genetic algorithm with local search for job shop scheduling problem. *Anvances in information Sciences and Service Sciences*, 3 :42 – 49, 2011.

[29] G. Campos-Ciro, F. Dugardin, F. Yalaoui, and R. Kelly. Open shop scheduling problem with a multi-skills resource constraint : a genetic algorithm approach. In *International Conference on Metaheuristics and Nature Inspired Computing (META), Marrakech, Morocco*, 2014.

[30] G. Campos-Ciro, F. Dugardin, F. Yalaoui, and R. Kelly. Algorithme lonie de fourmis et logique floue pour lordonnancement dopen shop avec ressources. In *ROADEF, Marseille, France*, 2015.

[31] G. Campos-Ciro, F. Dugardin, F. Yalaoui, and R. Kelly. A fuzzy ant colony optimization approach for solving an open shop scheduling problem with multi-skills resource and tool constraints. In *Metaheuristics International Conference (MIC), Agadir, Morocco*, 2015.

[32] G. Campos-Ciro, F. Dugardin, F. Yalaoui, and R. Kelly. A fuzzy ant colony optimization to solve an open shop scheduling problem with multi-skills resource constraints. In *Information Control Problems in Manufacturing (INCOM), IFAC-PapersOnLine, Volume 48, Issue 3, 2015, Pages 715-720*, 2015.

[33] G. Campos-Ciro, F. Dugardin, F. Yalaoui, and R. Kelly. Open shop scheduling problem with a multi-skill resource constraint : a genetic algorithm and an ant colony optimization approach. *International Journal of Production Research*, Published online : 30 december, 2015.

[34] P.-T. Chang, K.-P. Lin, P.-F. Pai, and C.-Z. Zhong. Ant colony optimization system for a multi-quantitative and qualitative objective job-shop parallel-machine-scheduling problem. *International Journal of Production Research*, 46 :5719 – 5759, 2008.

[35] B. Chen and V.A. Strusevich. Approximation algorithms for three-machine open shop scheduling. *ORSA Journal on Computing*, 5 :321 – 326, 1993.

[36] R. Chen, W. Huang, and G. Tang. Dense open shop schedules with relase times. *Theoritical Computer Science*, 407 :389 – 399, 2008.

[37] Y. Chen, A. Zhang, G. Chen, and J. Dong. Approximation algorithms for parallel open shop scheduling. *Information Processing Letters*, 113 (7) :220 – 224, 2013.

[38] T.C. E. Cheng and N. V. Shakhlevich. Two-machine open shop problem with controllable processing times. *Discrete Optimization*, 4 (2) :175 – 184, 2007.

[39] Y. Cho and S. Sahni. Preemptive scheduling of independent jobs with release and due times on open, flow, and job shops. *Operations Research*, 29 :511 – 522, 1981.

[40] H. S. Choi and D. H. Lee. Scheduling algorithms to minimize the number of tardy jobs in two-stage hybrid flow shops. *Computer & Industrial Engineering*, 56 (1) :113 – 120, 2009.

[41] J. Coelho and M. Vanhoucke. Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, 213 :73 – 82, 2011.

[42] C. A. Coello Coello, A. Aguirre, and E. Zitzler. Evolutionary multi-objective optimization. *European Journal of Operational Research*, 181 (16) :1617 – 1619, 2007.

[43] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[44] Y. Collette and P. Siarry. *Optimisation multiobjectif*. EYROLLES, 2002.

[45] S. Dauzère-Pérès, W. Roux, and J. B. Lasserre. Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, 107 :289 – 305, 1998.

[46] D. de Werra and J. Blazewicz. Some preemptive open shop scheduling problems with a renewable or non-renewable resource. *Discrete applied mathematics*, 35 :205 – 219, 1992.

[47] D. de Werra, J. Blazewicz, and W. Kubiak. A preemptive open shop scheduling problem with one resource. *Operations Research*, 10 :9 – 15, 1991.

[48] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I : solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, 18 (4) :577 – 601, 2014.

[49] K. Deb, A. Pratap, S Agarwal, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6 (2) :182 – 197, 2002.

[50] Y. Demir and S. K. Isleyen. An effective genetic algorithm for flexible job-shop scheduling with overoverlapping in operations. *International Journal of Production Research*, 52 :3905 – 3921, 2014.

[51] C. Dimopoulos. A review of evolutionary multiobjective optimization aapplication in the area of production research. *Congress on Evolutionary Computation*, 2 :1487 – 1494, 2004.

[52] M. Dorigo. *Optimization, learning and natural algorithms.* PhD thesis, Dipartimento di electtronica, Politecnico di Milano, Italy, 1992.

[53] Marco Dorigo and Gianni Di Caro. The ant colony optimization meta-heuristic. In *in New Ideas in Optimization*, pages 11 – 32. McGraw-Hill, 1999.

[54] S. H. H. Doulabi. A mixed integer linear formulation for the open shop earliness-tardiness scheduling problem. *Applied Mathematical Sciences*, 4 :1703 – 1710, 2010.

[55] S.H.H. Doulabi, A.A. Jaafari, and M.A. Shirazi. Minimizing weighted mean flow time in open shop scheduling with time-dependent weights and intermediate storage cost. *International Journal on Computer Science and Engineering*, 2 (3) :457 – 460, 2010.

[56] O. Dridi, S. Krichen, and A. Guitouni. A multiobjective hybrid ant colony optimization approach applied to the assignment and scheduling problem. *International Transactions in Operational Research*, 21 (6) :935 – 953, 2014.

[57] I.G. Drobouchevitch and V.A. Strusevich. A polynomial algorithm of the three-machine open shop with a bottleneck machine. *Annals of Operations Research*, 92 :185 – 210, 1999.

[58] M. Dror. Open shop scheduling with machine dependent processing times. *Discrete Applied Mathematics*, 11 :709 – 720, 1992.

[59] F. Dugardin. *Maîtrise des ressources logistiques dans l'optimisation multi-ojectif des lignes de production.* PhD thesis, Université de Technologie de Troyes, 2009.

[60] EcuRed. Image "Brins d'ADN", 2015. Online ; accessed : 2015-04-15.

[61] V. P. Eswaramurthy and A. Tamilarasi. Hybridizing tabu seach with ant colony optimization for solving job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 40 :1004 – 1015, 2009.

[62] H.L. Fang, P. Ross, and D. Corne. A promossing hybrid ga/heuristic approach for open shop scheduling problems. In *11th European Conference on Artificial Intelligence, Amsterdam, The Netherlands, 8 - 12 august, 590 - 594*, 1994.

[63] M. Feldmann and D. Biskup. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers & Industrial Engineering*, 44 :307 – 323, 2003.

[64] C. M. Fonseca and P. J. Fleming. Genetic algorithm for multiobjective optimization : formulation, discussion and generalization. In *Proceedings of the fifth International Conference on Genetic Algorithms. San Mateo : Morgan Kauffman Publishers*, 1993, 416 - 423.

[65] V. Gabrel. Scheduling jobs within time windows on identical parallel machines : New model and algorithms. *European Journal*, 83 :320 – 329, 1995.

[66] M. R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, NY, USA, 1979.

[67] K. Giaro. NP - hardeness of compact scheduling in simplified open and flow shops. *European Journal of Operational Research*, 130 :90 – 98, 2001.

[68] M. X. Goemans, M. Querranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates. *SIAM Journal on discrete mathematics*, 2 :165 – 192, 2002.

[69] M. M. Goharch, B. Karimi, and M. Khademian. A simulation optimization approach for open shop problem with random process times. *International Journal of Advanced Manufacturing Technology*, 70 :821 – 831, 2014.

[70] S. Gonzalez and T. Sahni. Open shop scheduling to minimize finish time. *Journal of the association for the computing machinery*, 23 :665 – 679, 1976.

[71] I. Gonzalez-Rodriguez, J.J. Palacios, C.R. Vela, and J. Puente. Heuristic local search for fuzzy open shop scheduling. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–8, July 2010.

[72] R. L. Graham, E.L. Lawler, J.K. Lentra, and A.H.G. Rinnooy-Kan. Optimization and approximation in deterministic sequencing and scheduling : A survey. *Annals of Discrete Mathematics*, 5 :287 – 326, 1979.

[73] C. Guéret, N. Jussien, and C. Prins. Using intelligent backtraking to improve branch-and-bound methods : an application to open-shop problems. *European Journal of Operational Research*, 127 (1) :344 – 354, 2000.

[74] C. Guéret and C. Prins. A new lower bound for the open shop problems. *Annals of Operations Research*, 92 :165 – 183, 1999.

[75] D. Gupta, R. Jain, and P. Singla. Optimal two stage open shop scheduling, processing time associated with probabilities including transportation time and job block criteria. *International Journal of Mathematical Archive*, 3 (5) :1859 – 1872, 2012.

[76] J.N.D. Gupta and F. Werner. On the solution of 2-machine flow and open shop scheduling problems with secondary criteria. In *15th ISPE/IEE International Conference*

on CAD/CAM, Robotics, and Factories of the Future, Aguas De Lindoia, Sao Paulo, 18 - 20 august, 1999.

[77] N.D. Gupta. Two-stage hybrid flowshop scheduling problem. *The Journal of the Operational Research Society*, 39 (4) :359 – 364, 1988.

[78] O. Guyon, P. Lemaire, E. Pinson, and D. Rivreau. Solving an integrated job-shop problem with human resource constraints. *Annuals of Operations Research*, 2 :147 – 171, 2012.

[79] Y. Hani, H. Chehade, L. Amodeo, and F. Yalaoui. Simulation based optimization of a train maintenance facility model using genetic algorithms. In *Proceedings of the IEEE International Conference on Service Systems and Service Management, pages 513 - 518, Troyes, France*, pages 513 – 518, 2006.

[80] J. Holland. *Adaptation in natural and artifitial systems.* University of Michigan Press, 1975.

[81] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorihtm for multiobjective optimization. In *Proceedings of the first IEEE Conference on Evolutionary Computation*, 1994, 82 - 87.

[82] Y.-M. Huang and J.-C. Lin. A new bee colony optmization algorithm with idle-time based filtering scheme for open shop-scheduling problems. *Experts systems with applications*, 38 :5438 – 5447, 2011.

[83] S. Huband, P. Hingston, L. While, and L. Balone. An evolution strategy with probabilistic mutation for multi-objective optimisation. In *The Congress on Evolutionary Computation, 2003. CEC '03.*, pages 2284–2291 Vol.4, 2003.

[84] D. Indraneel and J. Dennis. A closer look at drawbacks of minimizing weighted sums of oobjective for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14 :63 – 69, 1997.

[85] H. Ishibuchi, K. Narukawa, N. Tsukamoto, and Y. Nojima. An empirical study on similarity-based mating for evolutionary multi-objective combinatorial optimization. *European Journal of Operational Research*, 188 :57 – 75, 2008.

[86] N. Jawahar, P. Aravindan, and S. G. Ponnambalam. A genetic algorithm for scheduling flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 14 :588 – 607, 1998.

[87] Z. Jia, X. Lu, J. Yang, and D. Jia. Research on job-shop scheduling problem based on genetic algorithm. *International Journal of Production Research*, 49 :3585 – 3604, 2011.

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

[89] A. Jougle and D. Savourey. Dominance problem rules for the parallel machine total weighted tardiness scheduling problem with release dates. *Computers & Operations research*, 38 :1259 – 1266, 2011.

[90] B. Jurisch and W. Kubiak. Two-machine open shops with renewable resources. *Operations Research*, 45 :544 – 552, 1997.

[91] H. Kazemipoor, R. Tavakkoli-Moghaddam, and Shahnazari-Shahrezaei. Solving a novel multi-skilled project scheduling model by scatter search. *South African Journal of Industrial Engineering*, 24 :121 – 135, 2013.

[92] S. Khuri and S.R. Miryala. Genetic algorithms for solving open shop scheduling problems. In *EPIA'99 proceedings of 9th Portuguese Conference on Artificail Intelligence, Evora, Protugal, 21 - 24 september, 357 - 368*, 1999.

[93] T. Kis, D. Werra, and W. Kubiak. Two machine open shop scheduling problem with bi-criteria. *Discrete Applied Mathematics*, 38 :129 – 132, 2010.

[94] Z. Kokosinski and L. Studzienny. Hybrid genetic algorihtms for the open shop scheduling problem. *International Journal of Computer Science and Network Security*, 7 :136 – 145, 2007.

[95] O. Koné, C. Artigues, P. López, and M. Mongeau. Event - based milp model for resource - constrained project scheduling problems. *Computers & Operations Research*, 38 :3 – 13, 2011.

[96] O. Koné, C. Artigues, P. López, and M. Mongeau. Comparison of mixed integer linear programming model for the resource-constrained project scheculing problem with consumption and production of resources. *Flexible Services and Manufacturing*, 25 :25 – 47, 2013.

[97] T. Konno and H. Ishii. An open shop scheduling problem with fuzzy allowable time and fuzzy resource constraint. *Fuzzy sets and systems*, 109 :141 – 147, 2000.

[98] A. Kononov, S. Sevastianov, and I. Tchernykh. When difference in machine loads leads to efficient scheduling in open shops. *Annals of Operations Research*, 92 :211 – 239, 1999.

[99] A. Kononov and M. Sviridenko. A linear time approximation scheme for makespan minimization in an open shop with release dates. *Operation Research Letters*, 30 :276 – 280, 2002.

[100] P. Korytkowski, S. Rymaszewski, and T. Wisniewski. Ant colony optimization for job shop scheduling using multi-attribute dispatching rules. *International Journal of Advanced Manufacturing Technology*, 67 :231 – 241, 2013.

[101] C. Koulamas and J. Kyparisis. Single-machine scheduling problems with past-sequence-dependent setup times. *European Journal Of Operational Research*, 187 :1045 – 1049, 2008.

[102] M. Kubale and A. Nadolski. Chromatic scheduling in a cyclic open shop. *European Journal of Operational Research*, 164 (3) :585 – 591, 2005.

[103] W. Kubiak, C. Sriskandarajah, and K Zaras. A note on the complexity of open shop scheduling problems. *Candadian Journal of Information Systems and Operational Research*, 29 :284 – 294, 1991.

[104] M.A. Kubzin and V.A. Strusevich. Planning machine maintenance in two-machine shop scheduling. *Operations Research*, 54 (4) :789 – 800, 2006.

[105] G. J. Kyparisis and C. Koulamas. Open shop scheduling with maximal machines. *Discrete Applied Mathematics*, 78 :175 – 187, 1997.

[106] G.J. Kyparisis and C. Koulamas. Open shop scheduling with makespan and total completion time criteria. *Computers & Operations Research*, 27 (1) :15 – 27, 2000.

[107] H.C.W. Lau, T.M. Chan, W.T. Tsui, and G.T.S. Ho. Cost optimization of the supply chain network using genetic algorihtms. *IEEE Transactions on Knowledge and Data Engineering*, 99 :1–1, 2009.

[108] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Congress on evolutionary Computation Oiscataway, New Jersey, pages 46 - 53*, 2000.

[109] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys. *Sequencing and scheduling : Algorihtms and complexity. Handbook in Operations Research and Management Science*, chapter 9, pages 445 – 521. Elsevier Science, 1993.

[110] E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Recent developments in deterministic sequencing and scheduling : A survey. In M.A.H. Dempster, J.K. Lenstra, and A.H.G. Rinnooy Kan, editors, *Deterministic and Stochastic Scheduling*, volume 84 of *NATO Advanced Study Institutes Series*, pages 35–73. Springer Netherlands, 1982.

[111] J. K. Lenstra and A. H. G. Rinnooy-Kan. New directions in scheculing theory. *Operation Research Letters*, 2 (6) :255 – 259, 1984.

[112] C.-F. Liaw. A tabu search algorithm for the open shop scheduling problem. *Computer & Operations Research*, 26 :109 – 126, 1999.

[113] C.-F. Liaw. A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Operational Research*, 124 :28 – 42, 2000.

[114] C.-F. Liaw. An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. *Computers & Operations Research*, 30 :2081 – 2095, 2003.

[115] C.-F. Liaw. Scheduling two-machine preemptive open shops to minimize total completion time. *Computers & Operations Research*, 31 :1349 – 1363, 2004.

[116] C.-F. Liaw. Scheduling preemptive open shops to minimize total tardiness. *European Journal of Operational Research*, 162 :173 – 183, 2005.

[117] C.-F. Liaw. An improve branch-and-bound algorithm for the preemptive open shop total completion time scheduling problem. *Journal of Industrial and Production Engineering*, 30 :327 – 335, 2013.

[118] C.-F. Liaw, C. Y. Cheng, and M. Chen. The total completion time open shop scheduling problem with a given sequence of jobs on one machine. *Computer & Operations Research*, 29 :1251 – 1266, 2002.

[119] C.F. Liaw. An iterative improvement approach for the non-preemptive open shop scheduling problem. *European Journal of Operational Research*, 111 :509 – 517, 1998.

[120] C.Y. Liu and R.L. Bulfin. Scheduling open shops with unit execution times to minimize functions of due dates. *Operations Research*, 36 :553 – 559, 1988.

[121] S.Q. Liu and H.L. Ong. Metaheuristics for the mixed-shop scheduling problem. *Asia-Pacific Journal of Operational Research*, 21 :97 – 115, 2004.

[122] T. Lorigeon, J.C. Billaut, and J.L. Bouquard. A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint. *Journal of the Operational Research Society*, 53 :1239 – 1246, 2002.

[123] S.J Louis and Z.J. Xu. Genetic algorithms for open shop scheduling and re-scheduling. In *ISCA 11th International Conference on Computers and Their Applications, San Francisco, 7 - 9 March, 99 - 102*, 1996.

[124] T. Loukil, J. Teghem, and D. Tuyttens. Solving multiobjective production scheduling problems using metaheuristics. *European Journal of Operational Research*, 161 :42 – 61, 2005.

[125] A. Lova, P. Tormos, M. Cervantes, and F. Barber. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117 :302 – 316, 2009.

[126] C. Low and Y. Yeh. Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. *Robotics and Computer-integrated Manufacturing*, 25 :314 – 322, 2009.

[127] L. Lu and M.E. Posner. A NP-hard open shop scheduling problem with polynomial average time complexity. *Mathematics of Operations Research*, 18 :12 – 38, 1993.

[128] A. Malapert, H. Cambazard, C. Gueret, N. Jusslen, A. Langevin, and L.M. Rousseau. An optimal constraint programming approach to the open shop problem. *Journal of Artificial Intelligence Research*, 1 :25 – 46, 2009.

[129] T. Masuda and H. Ishii. Two machine open shop scheduling problem with bi-criteria. *Discrete Applied Mathematics*, 52 :253 – 259, 1994.

[130] M.E. Matta. A genetic algorithm for the proportionate multiprocessor open shop. *Computers & Operations Research*, 36 :2601 – 2618, 2009.

[131] K. Mesghouni and S. Hammadi. Evolutionary algorihtms for job shop scheduling. *International Journal of Applied Mathematics and Computer Science*, 14 (1) :91 – 103, 2004.

[132] M. Modarres and M. Ghandehari. Generalized cyclic open shop scheduling and a hybrid algorihtm. *Journal of Industrial and Systems Engineering*, 1 :345 – 359, 2008.

[133] C. Montoya. *New methods for the Multi-Skill Project Scheduling Problem.* PhD thesis, Ecole des mines de Nantes, 2012.

[134] A. Munier-Kordon and D. Rebaine. The two-machine open-shop problem with unit-time operations and time delays to minimize the makespan. *European Journal of Operational Research*, 203 :42 – 49, 2010.

[135] R. Murugesan, S.S. Thamarai, P. A. Rajendran, and V.S. Sampath Kumar. Identification of a rank minimal optimal sequence for open shop scheduling problems. *International Journal of Information and Management Sciences*, 14 :37 – 55, 2003.

[136] B. Naderi, S.M.T. Fatemi Ghomi, and M. Aminnayeri. A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. *Applied Soft Computing*, 10 :703 – 710, 2010.

[137] B. Naderi, S.M.T Fatemi Ghomi, M. Aminnayeri, and M. Zandieh. Scheduling open shops with parallel machines to minimize total completion time. *Journal of Computational and Applied Mathematics*, 235 :1275 – 1287, 2011.

[138] B. Naderi, S.M.T. Fatemi Ghomi, M. Aminnayeri, and M. Zandieh. A study on open shop scheduling to minimise total tardiness. *International Journal of Production Research*, 49 :4657 – 4678, 2011.

[139] V. R. Neppalli, Ch. I. Chen, and J. N. D. Gupta. Genetic algorithms for the two-stage bicriteria flow shop problem. *European Journal Of Operational Research*, 95 :356 – 373, 1996.

[140] R.F. Tavares Neto and M. Godinho Filho. Literature review regarding ant colony optimization applied to scheduling problems : guideline for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, 26 :150 – 161, 2013.

[141] C.T. Ng, T.C.E. Cheng, and J.J. Yuan. Concurrent open shop scheduling to minimize the weighted number of tardy jobs. *Journal of Scheduling*, 6 :405 – 412, 2003.

[142] J. Nonobe and T. Ibaraki. Formulation and tabu research algorithm for the resource constrained project scheduling problem. *Technical Report*, 15 :557 – 588, 2002.

[143] S. Noori-Darvish and R. Tavakkoli-Moghaddam. Solving a bi-objective open shop scheduling problem with fuzzy parameters. *Journal of Applied Operational Research*, 3 :59 – 74, 2011.

[144] S. Noori-Darvish and R. Tavakkoli-Moghaddam. Minimizing the total tardiness and makespan in an open shop scheduling problem with sequence-dependent setup times. *Journal of Industrial Engineering International*, 8 :25, 2012.

[145] Y. Ouazene, F. Hnaien, F. Yalaoui, and L. Amodeo. The joint load balancing and parallel machine scheduling problem. In Bo Hu, Karl Morasch, Stefan Pickl, and Markus Siegle, editors, *Operations Research Proceedings 2010*, pages 497–502. Springer Berlin Heidelberg, 2011.

[146] Y. Ouazene, F. Yalaoui, H. Chehade, and A. Yalaoui. Analysis of different criteria for work balancing on identical parallel machines. In *Procedings of the International Conference on Industrial Engineering and Operations Management, Bali, Indonesia*, 2014.

[147] H. Panahi and R. Tavakkoli-Mogahaddam. Solving a multi-objective open shop scheduling problem by a novel hybrid ant colony optimization. *Expert systems*, 38 :2817 – 2822, 2011.

[148] R. Panneerselvam. Heuristic for moderated job shop scheduling problem to minimize makespan. *Industrial Engineering Journal*, 28 :26 – 29, 1999.

[149] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Dover Publications, 1982.

[150] N. Piersma and W. Van Dijk. A local search heuristic for unrelated parallel machine scheculing with efficient neighborhood search. *Mathematical Computing Modeling*, 24 (9) :11 – 19, 1996.

[151] C. Prins. Competitive genetic algorithms for the open shop scheduling problem. *Mathematical Methods of the Operations Research*, 52 :389 – 411, 2000.

[152] R. Purshouse. *On the evolutionary optimization of many objectives*. PhD thesis, Univ. Sheffield, Sheffield, U.K., 2003.

[153] M. Queyranne and M. Sviridenko. A (2+ϵ) approximation algorithm for the generalized preemptive open shop problem with minsum objective. *Journal of Algorithms*, 45 :202 – 212, 2002.

[154] D. Rebaine. Scheduling the two-machine open shop problem with non-symmetric time delays. In *Congress ASAC, Quebec City, 5 - 8 June, 1 - 10*, 2004.

[155] D. Rebaine and V.A. Strusevich. Two-machine open shop scheduling with special transportation times. *Journal of the Operational Research Society*, 50 :756 – 764, 1999.

[156] T.A. Roemer. A note on the complexity of the concurrent open shop problem. *Journal of Scheduling*, 9 :389 – 396, 2006.

[157] V. Roshnaei, M.M.S. Esfehani, and M. Zandieh. Integrating non-preemptive open shop scheduling with sequence-dependent setup times using advanced metaheristics. *Experts Systems with Applications*, 37 :259 – 266, 2010.

[158] J. D. Schaffer. Multi objective optimization with vector evaluated genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, 1985, 93 - 100.

[159] P. Schuurman and G.J. Woeginger. Approximation algorithms for the multiprocessor open shop scheduling problem. In *Memorandum COSOR 97 - 23, Eindhoven University of Technology, Eindhoven*, 1997.

[160] A. Sedeno-Noda, D. Alcaide, and C. Gonzalez-Martin. Network flow approaches to preemptive open shop scheduling problems with time-windows. *European Journal of Operational Research*, 174 :1501 – 1518, 2006.

[161] P. Senthilkumar and P. Shahabudeen. Ga based heuristic for the open job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 30 :297 – 301, 2006.

[162] S.V. Sevastianov and G.J. Woeginger. Linear time approximation scheme for the multiprocessor an open shop problem. *Discrete Applied Mathematics*, 114 :273 – 288, 2001.

[163] M. A. Shafia, M. P. Aghaee, and A. Jamili. A new mathematical model for the job shop scheduling problem with uncertain processing times. *International Journal of Industrial Engineering Computations*, 2 :295 – 306, 2011.

[164] R. Slowiński. Cost-minimal preemptive scheduling of independent jobs with release and due dates on open shop under resource constrains. *Information processing letters*, 9 :233 – 237, 1979.

[165] F. Sourd. Earliness-tardiness scheduling with setup considerations. *Computers & Operations Research*, 32 :1849 – 1865, 2005.

[166] F. Sourd. Dynasearch neighborhood for the earliness-tardiness scheduling problem with release dates and setup constraints. *Operation Research Letters*, 34 :591 – 598, 2006.

[167] N. Srinivas and K. Deb. Multiobjective function optimization using non dominated sorting genetic algorithms. *Evolutionary Computation*, 2 (3) :221 – 248, 1994.

[168] W. Szwarc and S.K. Mukhopadhyay. Decomposition of the single machine total tardiness problem. *Operations Research Letters*, 19 :243 – 450, 1996.

[169] L. Tang and D. Bai. A new heuristic for open shop total completion time problem. *Applied Mathematical Modelling*, 34 :735 – 743, 2010.

[170] T. Tautenhahn. Scheduling unit-time open shops with deadlines. *Operations Research*, 42 :189 – 192, 1994.

[171] J. T. Tsai, T. K. Liu, W. H. Ho, and J. H. Chou. An improve genetic algorithm for job shop scheduling problems using taguchi-based crossover. *International Journal of Advanced Manufacturing Technology*, 38 :987 – 994, 2008.

[172] A. Udomsakdigool and V. Kachitvichyanukul. Multiple colony ant algorihtm for job-shop scheduling problem. *International Journal of Production Research*, 46 :4155 – 4175, 2008.

[173] A Udomsakdigool and V. Kachitvichyanukul. Solving multi-objective job shop scheduling problem using ant colony optimization. In *APIEMS 2009 Asia Pacific Industrial Engineering & Management Systems Conference*, 2009.

[174] M. v. d. Akker, H. Hoogeven, and G.J. Woeginger. The two-machine open shop problem : to fit or not fit, that is the question. *Operations Research Letters*, 31 :219 – 224, 2003.

[175] X.-J. Wang, C.-Y. Zhang, L. Gao, and P.-G. Li. A survey and future trend of study on multi-objective scheduling. In *Fourth International Conference on Natural Computation, 2008. ICNC '08*, volume 6, pages 382–391, 2008.

[176] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *Evolutionary Computation, IEEE Transactions on*, 10 :29–38, 2006.

[177] Wikipedia. Image "Algorithme à colonie de fourmis", 2015. Online ; accessed : 2015-04-15.

[178] B. Yagmahan and M. M. Yenisey. Aapproach optimization for multi-objective flow shop scheduling problem. *Computers & Industrial Engineering*, 54 (3) :411 – 420, 2008.

[179] F. Yalaoui. Optimisation et gestion de la production : problème de conception et problème d'ordonnancement, 2006. Habilitation à diriger les recherches, Université de Technologie de Compiègne.

[180] F. Yalaoui and C. Chu. A new exact method to solve $pm/r_i/\sum c_i$ the problem. *International Journal of Production Economics*, 100 (1) :168 – 179, 2006.

[181] N. Yalaoui, F. Dugardin, F. Yalaoui, L. Amodeo, and H. Mahdi. Fuzzy project scheduling. In Cengiz Kahraman and Mesut Yavuz, editors, *Production Engineering and Management under Fuzziness*, volume 252 of *Studies in Fuzziness and Soft Computing*, pages 143–170. Springer Berlin Heidelberg, 2010.

[182] N. Yalaoui, H. Mahdi, L. Amodeo, and F. Yalaoui. A new approach for workshop design. *Journal of Intelligent Manufacturing*, 22 :933 – 951, 2011.

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

[184] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8 (3) :338 – 353, 1965.

[185] L. J. Zeballos. A constaint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 26 :725 – 743, 2010.

[186] X.D. Zhang and S. van de Velde. On-line two machine open shop scheduling with time lags. *European Journal Of Operational Research*, 204 :14 – 19, 2010.

[187] R. Zhou, A.Y.C. Nee, and H.P. Lee. Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problem. *International Journal of Production Research*, 47 :2903 – 2920, 2009.

[188] E. Zitzler. *Evolutionary algorithms for multiobjective optimization : methods and applications*. PhD thesis, Swiss Federal Inst. Technology (ETH) Zurich, Switzerland, 1999.

[189] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2 : Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *European Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2001.

[190] E. Zitzler and L. Thiele. Multiobjectif evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3 (4) :257 – 271, 1999.

[191] G.I. Zobolas, C.D. Tarantilis, and G. Ioannou. Solving the open shop scheduling problem via a hybrid genetic-variable neighborhood search algorihtm. *Cybernetics and Systems : An international Journal*, 40 :259 – 285, 2009.

Guillermo CAMPOS CIRO

Doctorat : Optimisation et Sûreté des Systèmes

Année 2015

Développement de méthodes d'ordonnancement efficaces et appliquées dans un système de production mécanique

L'évolution continue des environnements de production et l'augmentation des besoins des clients, demandent un processus de production plus rapide et efficace qui contrôle plusieurs paramètres en même temps. Nous nous sommes intéressés au développement de méthodes d'aide à la décision qui permettent d'améliorer l'ordonnancement de la production. L'entreprise partenaire (Norelem) fabrique des pièces de précision mécanique, il faut donc prendre en compte les différentes contraintes de ressources (humaines et d'outillage) existantes dans l'atelier de production.

Nous avons abordé l'étude d'un atelier d'ordonnancement de type open shop ou chemin ouvert, où une tâche peut avoir de multiples séquences de production puisque l'ordre de fabrication n'est pas fixé et l'objectif à minimiser est le temps total de séjour. Des contraintes d'affectation de ressources humaines (multi-compétences) et de disponibilité d'outillage ont été prises en compte.

Des modèles mathématiques linéaires et non-linéaires ont été développés pour décrire la problématique. Etant donné que les méthodes exactes sont limitées aux instances de petites tailles à cause des temps de calcul, des méthodes de résolution approchées ont été proposées et comparées. De plus, nous avons abordé l'optimisation multi-objectif en considérant trois objectifs, la minimisation du temps total de séjour et l'équilibrage de charge des ressources (humaines et machines).

L'efficacité des méthodes est prouvée grâce à des tests sur des instances théoriques et l'application au cas réel.

Mots clés : recherche opérationnelle - ordonnancement (gestion) - allocation des ressources – algorithmes - optimisation combinatoire - métaheuristiques.

Development of Efficient Scheduling Methods and their Application in a Mechanical Production System

The continuous evolution of manufacturing environments and the growing of customer needings, leads to a faster and more efficient production process that controls an increasing number of parameters. This thesis is focused on the development of decision making methods in order to improve the production scheduling.

The industrial partner (Norelem) produces standardized mechanical elements, so many different resource constraints (humans and tools) are presented in its workshop.

We study an open shop scheduling problem where one job can follow multiple production sequences because there is no fixed production sequence and the objective function is to minimize the total flow time. In addition, multi-skilled personnel assignment and tool's availability constraints are involved.

Mathematical models: linear and non-linear formulations have been developed to describe the problem. Knowing the exact method limitations in terms of instance sizes because of the duration, heuristics methods have been proposed and compared. Besides that, the multi-objective optimization was exposed to deal with three objectives as total flow time minimization and workload balancing concerning both, humans and machines.

The efficiency of these methods was proved by several theoretical instance tests and the application on the real industrial case.

Keywords: operations research - production scheduling - resource allocation - algorithms - combinatorial optimization - metaheuristics.

Thèse réalisée en partenariat entre :

norelem — anrt association nationale recherche technologie — Cifre Conventions — utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"