UNIVERSITÉ DE TOURS

ÉCOLE DOCTORALE MIPTIS

Laboratoire d'Informatique Fondamentale et Appliquée de Tours (EA 6300)

THÈSE présentée par :

Hugo Chevroton

soutenue le 8 décembre 2020
pour obtenir le grade de Docteur de l'Université de Tours

Discipline/Spécialité : Informatique

Résolution par des algorithmes exacts et approchés de problèmes intégrés d'ordonnancement de la production et de tournées de véhicules

Thèse dirigée par :

Billaut Jean-Charles	Professeur, Université de Tours

Rapporteurs :

Absi Nabil	Professeur, École des Mines de Saint-Étienne
Lacomme Philippe	Maître de conférence, HdR, Université de Clermont-Ferrand

Jury :

Absi Nabil	Professeur, Écoles de Mines de Saint-Étienne
Bellenguez Odile	Maître de conférence, HdR, IMT Atlantique
Berghman Lotte	Maître de conférence, Toulouse Business School
Billaut Jean-Charles	Professeur, Université de Tours
Kergosien Yannick	Maître de conférence, Université de Tours
Lacomme Philippe	Maître de conférence, HdR, Université de Clermont-Ferrant
Moukrim Aziz	Professeur, Université de Technologie de Compiègne
Néron Emmanuel	Professeur, Université de Tours

Résumé

Dans cette thèse, nous considérons un ensemble de problèmes de production et de livraison intégrés. Un ensemble de commandes être produit puis livré à des clients. La partie production est modélisée par un *flow-shop* de permutation. Chaque commande doit être préparée sur chaque machine dans un ordre précis, identique pour toutes les commandes et chaque machine doit préparer les commandes dans un ordre à déterminer, identique pour toutes les machines. Des coûts d'inventaire sont pris en compte aux différentes étapes de la production. Pour la partie livraison, les commandes sont regroupées en lot en fin de production et livrées par une flotte de véhicules homogènes. L'ensemble des sites de production et des sites clients sont distincts. Le coût de la livraison correspond à une part fixe par véhicule et une part variable dépendant de la distance parcourue. Enfin, chaque commande est attendue pour une date due de livraison, une livraison en retard engendre des pénalités proportionnelles à la durée de ce retard. Nous considérons deux types d'agents : producteur et prestataire logistique de service (3PL). Ce dernier se charge des livraisons.

Deux variantes de ce problème sont abordées dans cette thèse : une variante à un seul producteur et plusieurs clients et une variante à plusieurs producteurs et un seul client. Dans le premier problème, un producteur doit produire des commandes pour un ensemble de clients qui seront livrées par le 3PL. Deux approches sont considérées. Dans une première approche, nous considérons que le producteur est en mesure d'imposer ses décisions au 3PL. Il fixe la séquence de production des commandes, leurs mises en lot et fait une estimation des dates de livraison. Le 3PL s'adapte et chaque agent cherche à minimiser ses coûts. La résolution du modèle mathématique n'étant efficace que pour des problèmes de petites tailles, un algorithme génétique et un GRASP (*Greedy randomized adaptive search procedure*) sont proposés pour résoudre le problème. Ces deux heuristiques sont évaluées sur un ensemble d'instances générées aléatoirement et les résultats montrent de meilleures performances pour l'algorithme génétique. Dans la seconde approche, producteur et 3PL forment un même agent. Ils prennent leurs décisions et minimisent leurs coûts conjointement. Nous considérons dans cette approche que la composition des lots est déjà connue. Une résolution en deux étapes est proposée. D'abord, pour chaque lot et chaque date de départ possible, un prétraitement détermine une tournée minimisant les coûts de transport et les pénalités de retard. Une procédure par séparation et évaluation ainsi que des heuristiques sont proposées pour réaliser cette étape. Ces premiers résultats sont réutilisés lors de la seconde étape afin de chercher une séquence de production minimisant l'ensemble des coûts du problème. L'ensemble des étapes de cette méthode de résolution sont testées.

Dans le second problème abordé, plusieurs producteurs doivent préparer un ensemble de commandes pour un unique client. Le 3PL réalise des tournées visitant plusieurs producteurs avant d'aller finalement livrer le client. Dans cette approche, nous considérons que les itinéraires de livraison et les dates de passage des véhicules sur les différents sites sont déjà déterminées d'un commun accord entre tous les agents. Les producteurs doivent alors décider de la séquence de production des différents sites et de l'affectation des commandes aux véhicules. Des travaux sont effectués sur la génération d'instances réalisables. Un algorithme génétique est proposé pour résoudre ce problème sur ces instances.

Abstract

We consider in this thesis a set of integrated routing and scheduling problems. A set of orders must be produced and delivered to customers. The production workshop is a permutation flow shop : all the orders are processed in the same order on the machines. We consider inventory costs throughout the entire production process, from the arrival of raw materials until the departure of the final product. Once the orders are completed, they are grouped in batches and delivered to customers by a homogeneous fleet of vehicles. All customers and production locations are different. The delivery cost is composed of a fixed cost per vehicle used and the total traveling costs. Finally, each customer waits for his order at a specific delivery date. If the delivery is late, the agents pay penalty costs. We consider two kinds of agents : the manufacturer and the third-party logistics (3PL) provider. The 3PL provider delivers the orders to the customers.

In this thesis, two versions of the problem are adressed : a first version with one produceur and several customers and a second version with several produceurs and one customer. In the first problem, a manufacturer must produce orders for several customers. Two approaches are proposed. In the first one, the manufacturer dominates the decision-making in the system. The manufacturer is in charge of determining the production scheduling, the number of vehicles and their loads and the departure date of each vehicle. The 3PL provider adjusts its decisions and each agent tries to minimize its own costs. Solving mathematical model is inefficient for large instances, so a genetic algorithm and a GRASP (Greedy randomized adaptive search procedure) are proposed to solve the problem. These two heuristics are tested on randomly generated instances and the results show best performances for the genetic algorithm. In the second approach, both agents are part of the same entity. They take decisions and minimize the whole cost together. Furthermore, we consider the composition of batches is known in advance. A two steps resolution method solves its problem. First, for each batch and each possible departure date, a prepossessing finds the delivery route minimizing the delivery and the penalty costs. A branch and bound procedure and heuristics are developed for this step. The second step reuses these results in order to search a production scheduling minimizing the whole problem cost. Both steps of this method are tested and compared.

In the second problem, several manufacturers must produce orders for a single customer. Each vehicle of 3PL provider deliveries visits fisrt the manufacturers then delivers the customer. In this approach, the different agents have already determined the routes of vehicles and timetable. The manufacturers must determine the scheduling on the different location and the assigment of the orders to vehicles. Feasible instances are generated and a genetic algorithm is proposed to solve these instances.

Table des matières

Table des figures

Liste des tableaux

Introduction

Cette thèse s'intéresse à une problématique d'organisation opérationnelle entre deux types d'agents : un producteur et prestataire logistique de service (3PL). Le producteur représente une entreprise sous contrat avec un ou plusieurs clients auprès desquels il s'engage à produire et acheminer un certain nombre de commandes. Or, depuis quelques années, ce type d'agent préfère se concentrer sur son cœur de métier (la production) et faire sous-traiter les opérations de livraison par un prestataire extérieur. Le 3PL est un agent spécialisé dans les actions de sous-traitance. Il peut réaliser plusieurs prestations, dont la livraison, pour tous types d'agents. Il dispose de ses propres infrastructures et de sa propre flotte de véhicules. Les exemples de collaboration entre producteur et 3PL sont nombreux. Par exemple, le CHU de Tours produit ces propres médicaments à la pharmacie centrale et sous-traite la livraison vers les autres sites de l'hôpital par des véhicules-navettes. De nombreuses grandes surfaces font également appel à des prestataires pour alimenter les stocks de leurs magasins dès qu'ils passent en dessous d'un seuil critique. Le 3PL a parfois accès directement aux informations de stock et planifie lui même les tournées.

Cette thèse donne suite aux travaux réalisés en 2015 par Sonja Rohmer et le professeur Jean-Charles Billaut ([Rohmer, 2015]) sur la modélisation d'un problème intégré entre un producteur et un 3PL. Le producteur doit préparer un ensemble de commandes qui seront livrées par le 3PL. La chaîne de production est un *flow-shop* de permutation prenant en compte des coûts d'inventaire. Les commandes sont regroupées en lot avant d'être livrées par une flotte de véhicule. Un problème de tournée est alors considéré avec des coûts fixes pour les véhicules et des coûts de transport. Des pénalités sont à verser en cas de retard lors de la livraison d'une commande.

Dans un premier temps, nous proposons une présentation générale des champs d'étude de cette thèse. Ce premier chapitre traite de notions de recherche opérationnelle, de méthodes d'optimisation et de classification de problème d'ordonnancement et de livraison. Enfin quelques notions sur la gestion d'une chaîne logistique sont présentées. Un deuxième chapitre propose une revue de l'état de l'art récent.

Le premier problème abordé dans cette thèse est une variante très proche de celle proposée par Sonja Rohmer et Jean-Charles Billaut. Le troisième chapitre de cette thèse présente une description et une modélisation générale de ce problème. Deux variantes sont ensuite abordées. Dans le quatrième chapitre, le producteur a la capacité d'imposer ses décisions au 3PL. Il planifie sa production, la mise en lot des commandes et estime les dates de livraison. Le 3PL doit adapter ses décisions à celles du producteur et minimiser ses propres coûts. Un algorithme génétique et un GRASP (*Greedy randomized adaptive search procedure*) sont proposés pour résoudre le problème lié au producteur. Ces deux algorithmes sont testés et comparés sur la base d'instances générées aléatoirement et les résultats montrent que l'algorithme génétique propose de meilleurs résultats que le GRASP. Dans le cinquième chapitre, les deux agents sont confondus et résolvent le problème de manière globale pour un contexte où les lots sont déjà fixés. Une méthode de résolution en deux phases est proposée. Pour chaque lot et chaque date de départ, une étape de prétraitement définit l'itinéraire de livraison minimisant les coûts de transport et les pénalités de retard des commandes. Les itinéraires trouvés sont utilisés dans différentes méthodes, exactes et approchées, permettant

de résoudre le problème global. Des tests sont réalisés pour évaluer le meilleur paramétrage de la première phase ainsi que celui de la méthode globale.

Le sixième chapitre de cette thèse propose d'aborder un problème différent. Plusieurs producteurs doivent produire des commandes pour un même client. Ils s'associent à un 3PL qui est effectue des tournées de ramassages chez les différents producteurs avant d'aller livrer ce client. Contrairement au problème précédent, la livraison en retard n'est plus permise et les véhicules ont une capacité. La faisabilité des solutions doit donc être prise en compte. Nous nous intéressons à un contexte dans lequel les itinéraires et les dates de passage des tournées sur les différents sites ont déjà été établis d'un commun accord entre les différents agents. Un algorithme génétique est proposé pour résoudre le problème associé. Des travaux sont également effectués pour proposer les instances réalisables utilisées pour tester l'algorithme génétique.

Enfin, nous concluons ce manuscrit par une conclusion générale, dans laquelle nous rappelons les résultats obtenus ainsi que les perspectives de recherche envisageables pour la poursuite de ces travaux.

Chapitre 1

Présentation des champs d'étude

Contents

Ce chapitre a pour objectif de donner un aperçu des domaines abordés dans cette thèse. Sont présentés ici des notions introductives à la recherche opérationnelle, aux problèmes d'ordonnancement et de transport ainsi qu'aux méthodes de résolution. Le sujet de cette thèse est détaillé et mis en perspective avec le domaine de la gestion de chaînes d'approvisionnement. Le plan global de la thèse est présenté en fin de chapitre.

1.1 Introduction à la recherche opérationnelle

La recherche opérationnelle (RO) est une discipline scientifique regroupant un ensemble de méthodes et techniques rationnelles orientées vers la recherche du meilleur choix possible. Elle se base

sur la modélisation et l'analyse du fonctionnement des systèmes de production ou d'organisation, qui aboutit à la génération et la résolution de problèmes combinatoires et d'optimisation. De ce fait, elle se trouve à l'intersection de plusieurs domaines : ingénierie pour la compréhension des systèmes, mathématique pour leurs modélisations et informatique pour automatiser la résolution des problème associés. Elle constitue une aide à la décision dans la mesure où elle permet aux décideurs de comprendre, d'évaluer les enjeux et ainsi d'arbitrer ou de faire les choix les plus efficaces.

1.1.1 Modélisation d'un problème d'optimisation

Chaque problème d'optimisation peut se modéliser à l'aide des quatre types de composantes suivantes. *Les paramètres* sont les données fournies par l'environnement du problème. Ils ne peuvent être modifiés. Un ensemble de paramètres décrit une *instance* du problème. *Les variables* permettent de modéliser les décisions à prendre afin de définir une solution. Les variables peuvent être continues (comprises entre moins l'infini et plus l'infini), entières ou bien binaires. *Les contraintes* imposent des relations entre (et sur) les variables. Ces relations doivent être respectées pour que la solution obtenue soit *réalisable*, c'est-à-dire acceptable comme solution du problème. On note $\mathcal{S}$ *l'ensemble des solutions réalisables* de l'instance d'un problème. *La fonction objectif* définit la qualité d'une solution, que l'on cherche à minimiser (ou maximiser) lors de la résolution d'un problème d'optimisation. En d'autres termes, si l'on considère f une fonction objectif qui associe à une solution s sa *valeur objectif* $f(s)$, résoudre un problème d'optimisation revient à trouver $s^* \in \mathcal{S}$ tel que :

$$f(s^*) = \min_{s \in \mathcal{S}} f(s)$$

Prenons l'exemple du problème de sac à dos [Toth, 1990] dans sa version la plus simple :

Vous êtes Arsène Lupin et vous vous êtes introduit chez un riche négociant. La maison dort, vous venez de forcer le coffre avec succès, les richesses s'étalent devant vous. Cependant, votre sac est encore vide et le temps presse. Vous ne pourrez pas tout emporter car vous charger au-delà d'un poids W risquerait de compromettre votre agilité, votre discrétion et donc votre fuite. Soit J l'ensemble des objets présents dans le coffre, chaque objet $j \in J$ peut se résumer à deux caractéristiques : son poids w_j et son prix de vente au marché noir c_j. W et les ensembles w_j et c_j sont les paramètres de votre problème. Vous avez ces éléments en tête, vous êtes un professionnel. Pour chaque objet j, un choix s'offre à vous, modélisé par une variable x_j. Choisissez de dérober l'objet j, x_j vaudra 1, sinon elle vaudra 0. Vous devez également respecter la contrainte suivante : la somme des poids w_j des objets j dans votre sac ne doit pas dépasser sa capacité W. Votre objectif est de maximiser la valeur de l'ensemble des objets que vous emportez.

Ce problème se modélise par le programme linéaire suivant :

$$\max \sum_{j \in J} x_j c_j \tag{1.1}$$

$$\text{s.t} \sum_{j \in J} x_j w_j \leq W \tag{1.2}$$

La fonction objectif du problème est décrite par l'expression (1.1). La contrainte (1.2) garantit que la somme des poids des objets choisis ne dépasse pas la charge maximale transportable.

Voici une instance réelle du problème : *Votre sac a une capacité W de 10. L'unité utilisée est arbitraire. Le coffre contient 4 objets. L'objet 1 est un magnifique tableau de maître d'une valeur de 25 Millions. Cependant il prend une taille 12 dans votre sac, en d'autres termes, il est trop*

encombrant pour pouvoir être transporté rapidement sans risque d'être abîmé. L'objet 2 est une énorme de liasse de billets. À l'oeil, vous l'estimez à 15 Millions et elle prend 7 de capacité dans votre sac. L'objet 3 est une cassette de pierres précieuses, vous ne pouvez pas l'ouvrir mais, au poids, vous estimez son contenu à 14 Millions. La transporter vous coûtera 5 de capacité. Enfin, l'objet 4 est une figurine en porcelaine. Vous soupçonnez le marchand de la conserver ici plus par sentimentalisme que pour sa valeur marchande estimez à 0.2 Millions pour un place de 3.

Le Tableau 1.1 récapitulatif de l'instance de notre problème.

Objet	Valeur	Poids	
1	25 Millions	12	
2	15 Millions	7	$W = 10$
3	14 Millions	5	
4	0,2 Millions	3	

Tab. 1.1: Récapitulatif de l'instance

Une solution se caractérise par l'état des variables de décision x_j, $j \in \{1, 2, 3, 4\}$ dont chacune désigne le choix de mettre ou non l'objet j dans le sac. L'ensemble des solutions réalisables $\mathcal{S}$ du problème associé à notre instance est le suivant : $S_1 = \{0, 0, 0, 0\}$, $S_2 = \{0, 1, 0, 0\}$, $S_3 = \{0, 0, 1, 0\}$, $S_4 = \{0, 0, 0, 1\}$, $S_5 = \{0, 1, 0, 1\}$ et $S_6 = \{0, 0, 1, 1\}$. La solution S_1 est la solution vide, aucun objet n'est dans le sac. Les solutions S_2, S_3 et S_4 ne sont clairement pas optimales car il est encore possible d'ajouter un autre objet dans le sac. Concernant les deux dernières solutions S_5 et S_6, il faut regarder leurs fonctions objectifs, $f(S_5) = 15, 2$ et $f(S_6) = 14, 2$. Ainsi, la cinquième solution est optimale.

1.1.2 Complexité

1.1.2.1 Introduction à la théories de la complexité des problèmes

La théorie de la complexité propose un ensemble de résultats permettant d'évaluer la difficulté intrinsèque de la résolution d'un problème. Les problèmes peuvent être regroupés en classes de complexité qui indiquent "l'effort minimum" à fournir par les algorithmes pour résoudre ces problèmes. Au sein de la théorie de la complexité, nous nous intéresserons à deux types de problèmes :

– *Les problèmes de décision* sont des problèmes posant une question dont la réponse est "oui" ou "non". Reprenons l'exemple du problème de sac à dos, une variante décisionnelle pour une valeur K donnée est la suivante. Existe-t-il un sous-ensemble d'objets ne dépassant pas la capacité W du sac et ayant une valeur totale supérieure ou égale à K? Le programme linéaire en nombres entiers associé à ce problème est le suivant. Trouver les x_j tels que :

$$K \leq \sum_{j \in J} x_j c_j \tag{1.3}$$

$$W \geq \sum_{j \in J} x_j w_j \tag{1.4}$$

La contrainte (1.4) reste identique à la contrainte (1.2) mais la fonction objectif (1.1) est remplacée par la contrainte (1.3). Il ne s'agit plus de trouver la meilleure solution, mais de trouver une solution telle que sa fonction objectif soit supérieure à K. Si une telle solution existe, la réponse au problème est "oui", sinon la réponse est "non".

– *Les problèmes d'optimisation* tel que présentés Section (1.1.1) possédant une ou plusieurs fonctions objectifs.

1.1.2.2 Complexité des problèmes de décision

La *théorie de la complexité* se base sur le temps (ou l'espace mémoire) nécessaire à un algorithme pour effectuer une opération sur un problème, comme par exemple prouver l'existence d'une solution. De telles valeurs s'évaluent en fonction de la taille de l'instance et de ses données. Très souvent, seule la taille de l'instance, communément notée n, est utilisée (un choix de n objets, n sites à visiter, n tâches à planifier). La complexité est exprimée par une fonction mathématique dépendante de n et considérant le pire des cas. Par exemple, trouver l'index d'un élément dans un tableau à n entiers demande, dans le pire des cas, de parcourir les n éléments du tableau (si l'élément est à la fin ou s'il est absent) soit n opérations. De même, trier un tableau à l'aide d'un algorithme nommé tri à bulle demande dans le pire cas n^2 opérations (le pire des cas étant si le tableau est trié dans l'ordre inverse de celui recherché). Ce nombre d'opérations tombe à $n\ log(n)$ avec l'usage d'un autre algorithme comme le tri par fusion par exemple.

Le temps (ou l'espace mémoire, ou le nombre d'opérations) nécessaire à la résolution d'un problème peut ainsi être exprimé en fonction de la taille de l'instance résolue. La complexité s'évalue d'après le terme de plus haut degré de cette fonction. La Figure 1.1 représente le comportement de certains termes en fonction de petites tailles d'instances.

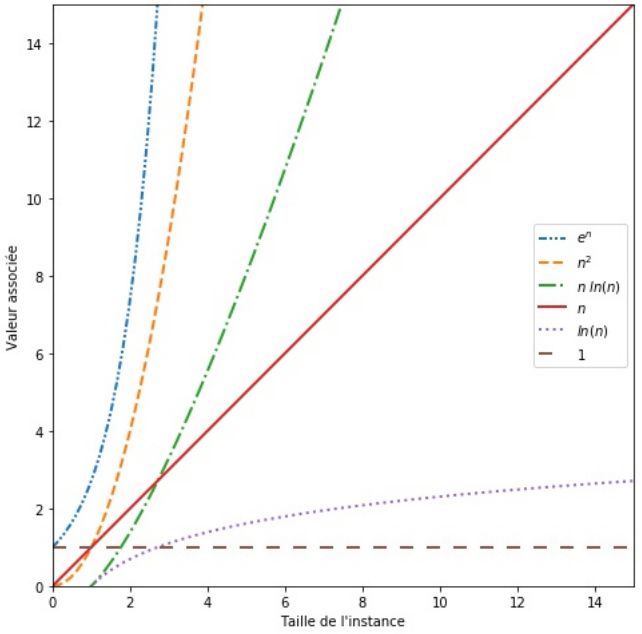

Fig. 1.1: Évolution de la complexité

Lorsque n tend vers l'infini, ces différents termes sont classés dans l'ordre suivant : $1 < log(n) < n < n\ log(n) < n^2 < e^n$. On dit qu'un problème peut être résolu *en temps polynomial* si le nombre d'opérations de l'algorithme utilisé pour sa résolution est borné par une fonction polynomiale de la taille de l'instance de degré k, où k est une constante.

Les problèmes de décision peuvent être associés à différentes *classes de complexité* dont voici les deux principales :

– La classe de complexité notée $\mathcal{P}$ regroupe tous les problèmes de décision résolvables en temps polynomial. C'est-à-dire qu'il existe un algorithme polynomial pour définir si la réponse à la question posée est "oui" ou "non".

– La classe de complexité $\mathcal{NP}$ regroupe tous les problèmes de décision dont une solution peut être vérifiée en temps polynomial ($\mathcal{P}$ est donc inclu dans $\mathcal{NP}$). Dans le cas du problème de sac à dos, dans sa variante décisionnelle, évaluer une solution demande d'évaluer si la somme des poids dans le sac est inférieure à W et si la somme des coûts est supérieure à K. Ces opérations peuvent être réalisées en temps polynomial donc le problème de sac à dos appartient à la classe $\mathcal{NP}$.

Definition 1. ***Réduction entre deux problèmes*** *Un problème de décision P_1 se réduit en temps polynomial à un problème de décision P_2 si et seulement s'il existe un algorithme polynomial f qui peut construire, pour toute instance I_1 de P_1, une instance $I_2 = f(I_1)$ tel que la réponse au problème P_1 pour l'instance I_1 est "oui" si et seulement si la réponse au problème P_2 pour l'instance I_2 est "oui".*

Si un tel algorithme f existe, cela prouve que le problème P_1 peut être résolu par un algorithme du problème P_2. On dit que le problème P_2 est au moins aussi difficile que le problème P_1. Un problème P_1 qui peut se réduire en temps polynomial à un problème P_2 se note $P_1 \propto P_2$.

Les définitions suivantes introduisent différentes sous classes de $\mathcal{NP}$.

Definition 2. *Un problème P est $\mathcal{NP}$-complet si P appartient à $\mathcal{NP}$ et si pour tout problème de $\mathcal{NP}$, il existe une réduction polynomiale vers P.*

Lemme 1.1. *Soit P_1 et P_2 deux problèmes de décision. Si $P_1 \propto P_2$ alors $P_2 \in \mathcal{P}$ implique $P_1 \in \mathcal{P}$ (et de même, $P_1 \notin \mathcal{P}$ implique $P_2 \notin \mathcal{P}$).*

Lemme 1.2. *La relation $\propto$ est transitive. Soit P_1, P_2 et P_3 des problèmes de décisions, $P_1 \propto P_2$ et $P_2 \propto P_3$ implique $P_1 \propto P_3$.*

Lemme 1.3. *Si P_1 et $P_2 \in \mathcal{NP}$, P_1 est $\mathcal{NP}$-complet et $P_1 \propto P_2$, alors P_2 est $\mathcal{NP}$-complet.*

Il existe de nombreux problèmes $\mathcal{NP}$-*complet*. Le premier d'entre eux étant démontré $\mathcal{NP}$--*complet* fut le problème SAT (SATISFIABILITY) [Cook, 1971]. Ce problème consiste à déterminer la satisfiabilité d'une formule de logique propositionnelle. Un autre problème $\mathcal{NP}$-*complet* important est le problème de PARTITION. Une instance du problème de PARTITION se compose d'un ensemble d'objets J où chaque objet j a un poids w_j. Le problème de partition consiste à déterminer si "oui" ou "non" il existe une séparation de J en deux sous ensembles d'objets tel que la somme des poids des objets de chaque sous ensemble soit la même.

La méthode classique pour prouver la complexité d'un problème est de montrer qu'un problème dont on connaît déjà la complexité se réduit à ce problème cible. Reprenons l'exemple du problème de sac-à-dos.

Théorème 1.1. *Le problème de sac à dos, dans sa version décisionnelle, est $\mathcal{NP}$-complet.*

Démonstration. Cette preuve est basée sur la réduction du problème de PARTITION à un problème de sac à dos. Chaque instance I_1 du problème de PARTITION est composée d'un ensemble A d'objets tel que chaque objet a dans cet ensemble a un poids w_a. La somme des poids des objets de A est égale à $2B$. Considérons un algorithme f qui, à toutes instances I_1 associe une instance I_2 du problème de sac à dos composée d'un ensemble d'objets J tel que chaque objet j dans cet ensemble à un poids w_j et un coût c_j et avec une capacité W. À chaque objet $a \in A$ est associé un objet $i \in I$ tel $w_a = w_j = c_j$ et $W = B$. Ainsi, résoudre le problème de PARTITION associé

à l'instance I_1 revient à déterminer s'il existe une solution avec B comme fonction objectif (le sac est complètement rempli). Nous avons donc PARTITION$\propto$ sac à dos.

Ainsi, d'après le Lemme 1.3, le problème de sac à dos est $\mathcal{NP}$-*complet*. □

1.2 Méthode de résolution

Une méthode de résolution est un algorithme visant à apporter une solution à un problème donné pour n'importe laquelle de ses instances. Dans le cadre de cette section nous nous limiterons à la description des méthodes pour les problèmes d'optimisation. On peut distinguer deux grandes familles de méthodes de résolution : les méthodes approchées et les méthodes exactes.

1.2.1 Méthode approchée

Les méthodes approchées sont généralement utilisées pour obtenir des solutions dans des délais raisonnables. Elles ne sont pas assurées de trouver la solution optimale (contrairement aux méthodes exactes), mais fournissent généralement de bonnes solutions, voire des solutions optimales et peuvent parfois donner des garanties sur la qualité des solutions trouvées (solution au pire à 50% de l'optimal par exemple).

Dans la suite de cette section, plusieurs méthodes approchées (appelées aussi heuristiques et métaheuristiques) seront brièvement présentées. Ces méthodes seront d'ailleurs réutilisées dans cette thèse.

1.2.1.1 Heuristique gloutonne

Les heuristiques gloutonnes sont des méthodes de résolution très rapides, souvent des algorithmes polynomiaux, généralement utilisées pour fournir des solutions initiales à des méthodes plus complexes. Ces méthodes construisent une solution par un processus itératif et ne reviennent jamais sur une décision prise.

Reprenons l'exemple du problème de sac à dos décrit en Section 1.1.1, une heuristique de construction gloutonne pourrait être la suivante :

Chaque objet j dans l'ensemble J se voit attribuer une utilité u_j égale au quotient du coût c_j de l'objet sur son poids w_j. Les objets sont ensuite triés par ordre décroisant d'utilité et sont placés dans le sac dans cet ordre. Si un objet ne peut pas être placé dans le sac, on passe à l'objet suivant. L'heuristique prend fin lorsqu'aucun objet ne peut plus être placé dans de sac.

1.2.1.2 Recherche locale

Les recherches locales sont des méthodes plus élaborées visant à améliorer itérativement une solution avec des opérateurs simples. L'algorithme général d'une recherche locale est le suivant.

Une recherche locale démarre avec une solution initiale (généralement aléatoire ou générée par une heuristique gloutonne). Tant que la condition d'arrêt de la recherche locale n'est pas atteinte, elle tente d'améliorer itérativement la solution courante. À chaque itération, un ensemble de nouvelles solutions est généré à partir de la solution courante et d'un opérateur de voisinage. Cet ensemble est d'ailleurs appelé le voisinage de la solution courante. Une nouvelle solution de ce voisinage, généralement la meilleure, remplace la solution courante si elle l'améliore. Si elle ne l'améliore pas, alors on dit qu'un minimum local est atteint. La recherche locale procède alors à une phase de diversification qui modifie fortement la solution courante qui sera de nouveau améliorée jusqu'à atteindre un nouveau minimum local ou le critère d'arrêt de la recherche locale. Une

procédure de diversification peut, par exemple, donner lieu à un certain nombre de modifications aléatoires de la solution courante. Un critère d'arrêt classique se base sur le temps utilisé par la méthode.

Il est à noter qu'un mécanisme de diversification n'est pas présent dans la recherche locale standard. Cependant, la majorité des métaheuristiques basées sur une mécanique de recherche locale dispose d'un mécanisme pour "s'échapper" des minimums locaux.

Ces différentes étapes sont résumées dans l'Algorithme 1.

Algorithme 1 Procédure de la recherche locale avec procédure de diversification

```
1 : s ← Solution initiale()
2 : tant que la condition d'arrêt n'est pas atteinte faire
3 :     répéter
4 :         V ← voisinage(s, opérateur)
5 :         s ← choix d'un voisin(s, V)
6 :     tant que s n'est pas un minimum local
7 :     diversification(s)
8 : fin tant que
```

Ci-dessous se trouvent deux problèmes dont une instance est résolue par recherche locale :

Exemple 1 : Cet exemple vise à illustrer la notion de solution initiale, d'optimum local et de diversification.

On cherche à minimiser la fonction $f(x)$ sur un intervalle. Le profil de $f(x)$ sur cet intervalle est présenté sur la Figure 1.2 (ces informations ne sont pas disponibles pendant la recherche). La recherche locale se comporte de la façon suivante. Une première solution x_s est choisie aléatoirement. Cette solution est ensuite améliorée par une recherche dans son voisinage défini comme l'évaluation de $f(x)$ pour tout x appartenant à $[x_s - \alpha, x_s + \alpha]$ où α est une valeur quelconque. La meilleure solution du voisinage remplace la solution courante tant qu'elle l'améliore. Après une série d'améliorations, la solution courante tombe dans un minimum local, la recherche locale procède alors à une diversification en sélectionnant aléatoirement une nouvelle solution courante dans l'espace de solutions.

Dans notre exemple, une première solution x_{s1} est choisie. Après amélioration de la solution x_{s1} en x_{m1} (Figure 1.3), une nouvelle solution x_{s2} est choisie suite à la diversification et cette solution est améliorée pour donner la solution x_{m2} (Figure 1.4). x_{m1} et x_{m2} sont des minimums locaux. De plus, en tant que plus grand des minimums locaux, x_{m2} est un minimum global (et donc la solution optimale).

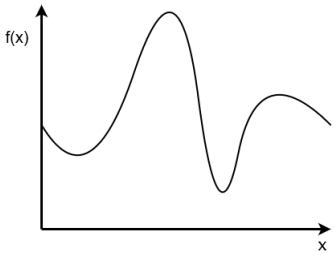

Fig. 1.2: Profil de $f(x)$

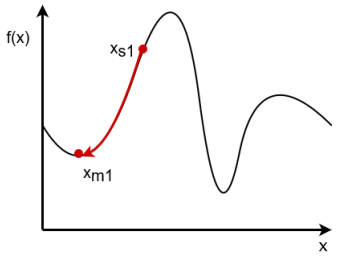

Fig. 1.3: Amélioration de x_{s1} à x_{m1}

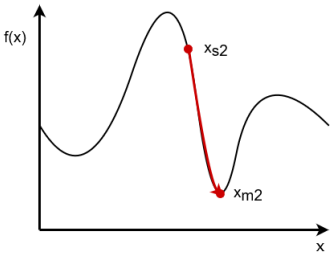

Fig. 1.4: Amélioration de x_{s2} à x_{m2}

Exemple 2 : Cet exemple illustre la notion de solution initiale, d'optimum local et de diversification.

Le problème du voyageur de commerce est un problème classique en recherche opérationnelle. Un voyageur doit visiter un ensemble de sites et revenir à son point de départ en minimisant la distance totale parcourue. Dans cet exemple, nous proposons une instance à 5 villes (j_1, j_2, j_3, j_4, j_5). La matrice des distances entre les différents lieux est symétrique et présentée dans le Tableau 1.2.

	j_1	j_2	j_3	j_4	j_5
j_1	0	3	5	6	7
j_2	3	0	4	6	5
j_3	5	4	0	2	3
j_4	6	6	2	0	3
j_5	7	5	3	3	0

Tab. 1.2: Matrice des distances

Un exemple de recherche locale est présenté pour résoudre ce problème.

La solution initiale est construite grâce à une heuristique gloutonne utilisant la stratégie du plus proche voisin. Le point de départ est choisi aléatoirement (ici le site j_1). À partir du site courant, le prochain site est le site le plus proche n'ayant par encore été visité. On obtient l'itinéraire (j_1, j_2, j_3, j_4, j_5) comme solution initiale avec comme valeur de fonction objectif égale à 18.

Soit un opérateur de voisinage qui consiste à échanger deux villes adjacentes dans l'itinéraire. L'ensemble du voisinage est énuméré en échangeant toutes les paires de villes adjacentes. Ainsi Figure 1.5 représente la solution initiale et les cinq solutions de son voisinage.

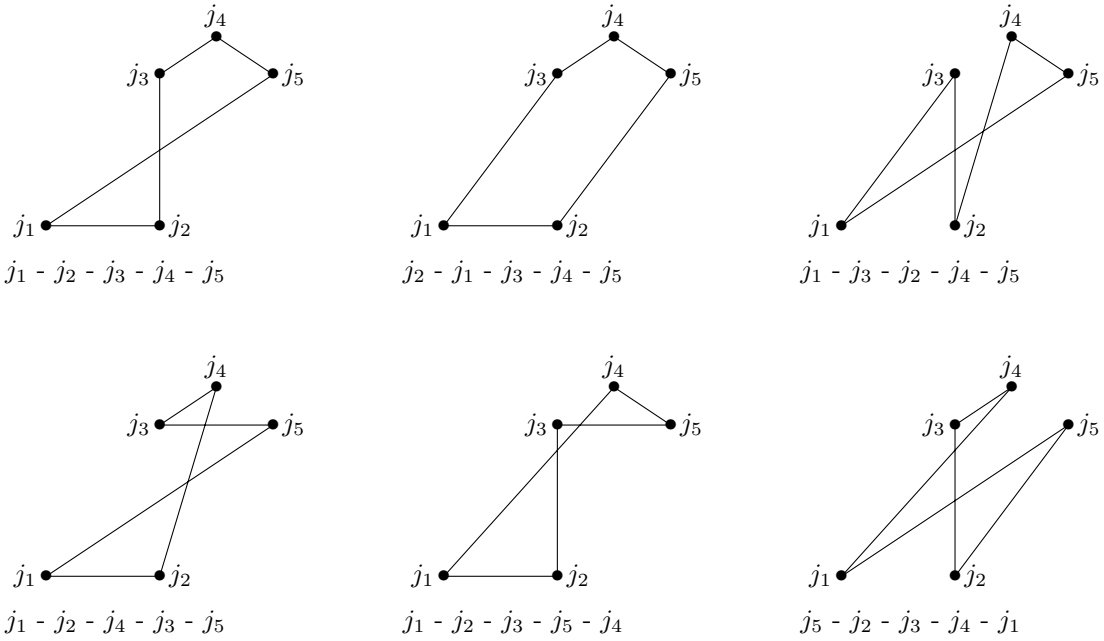

Fig. 1.5: Solution initiale et voisinage

Par exploration du voisinage de la solution initiale, la solution (j_2, j_1, j_3, j_4, j_5) est trouvée avec une valeur de fonction objectif égale à 17. Cette solution est la meilleure du voisinage et c'est aussi la solution optimale de l'instance.

1.2.1.3 Quelques métaheuristiques

Les métaheuristiques sont des méthodes de résolution de haut niveau capables d'être adaptées pour un résoudre une grande variété de problèmes. Le recherche locale est une métaheuristique. Voici quelques nouveaux exemples :

- **La recherche tabou** [Glover, 1997] : Cette méthode se base sur le concept suivant. Pour explorer un grand ensemble de solutions ou échapper à un minimum local, il est parfois nécessaire de dégrader la solution courante. Cependant, à l'étape suivante, il faut veiller à ne pas visiter de nouveau la solution précédente afin d'éviter un cyclage. Ainsi, la recherche tabou possède un mécanisme de mémoire et met à jour la liste des opérations qu'elle effectue sur sa solution courante. Ces opérations deviennent alors interdites pour une certaine durée.
- **Le recuit simulé** [Aarts et al., 1985] : Cette méthode est inspirée du constat en métallurgie qu'un refroidissement lent permet aux métaux d'agencer leurs atomes en une configuration plus solide qu'avec un refroidissement rapide. Ainsi, la méthode de recuit simulé dispose d'une variable température qui décroît lors d'une recherche locale. La méthode explore l'espace de recherche et peut aléatoirement accepter des solutions dégradant la solution courante. Cette probabilité décroît avec la température. Ainsi, la méthode s'autorise de grande dégradation au début de la recherche pour ne plus se concentrer que sur l'amélioration de la solution lorsque la température s'approche de sa valeur de refroidissement finale.

1.2.2 Méthode exacte

Une méthode exacte est une méthode qui trouve toujours la solution optimale d'un problème donné. Elle peut-être rapide comme pour la résolution de problèmes de classe $\mathcal{P}$. Mais elles sont souvent très lentes lors de la résolution de problème $\mathcal{NP}-complet$.

1.2.2.1 Algorithmes polynomiaux

Tout problème de classe $\mathcal{P}$ dispose d'un algorithme polynomial pour le résoudre. Comme dit précédemment, ces algorithmes permettent de trouver la solution optimale en garantissant de le faire en temps polynomial d'après la taille de l'instance. Considérons le problème suivant : minimisation du plus grand des retards pour un problème d'ordonnancement à une machine.

Soit une chaîne de production à une seule machine et un ensemble de tâches J devant être réalisées. Chaque tâche j de J a une durée de production notée p_j et une date de fin de production attendue notée d_j. Chaque tâche est également associée à deux variables, sa date de fin de production C_j et son retard T_j ($T_j = max(0, C_j - d_j)$) qui prennent leurs valeurs une fois la séquence de production fixée. La machine ne peut réaliser qu'une seule tâche à la fois et, une fois lancée, la réalisation d'une tâche ne peut pas être interrompue. La chaîne de production est disponible dès l'instant 0. L'objectif de ce problème est de trouver la séquence de production minimisant le plus grand des retards parmi toutes les tâches (min $max_{j \in J} T_j$).

[Jackson, 1955] propose un algorithme simple pour obtenir la solution optimale de ce problème. Il suffit d'ordonnancer les tâches par date d_j croissantes. Cette règle, très commune en ordonnancement et porte le nom de règle de Jackson ou encore règle EDD (acronyme anglais de Earliest Due Date). La preuve de l'optimalité de cette méthode est la suivante :

Supposons un ordonnancement noté O ne respectant pas la règle de Jackson. Cela signifie qu'il existe au moins deux tâches consécutives j_1 et j_2 tel que $C_{j_1} < C_{j_2}$ et $d_{j_1} > d_{j_2}$. Supposons que O est optimal. Considérons maintenant le même ordonnancement mais en échangeant j_1 et j_2 (ordonnancement O'). Les nouvelles dates de fin des tâches j_1 et j_2 sont notées C'_{j_1} et C'_{j_2} (Figure

1.6). Pour toute tâche $j \in J\backslash\{j_1, j_2\}$, la date de fin ne change pas entre O et O'. T_{j_1}, T_{j_2}, T'_{j_1} et T'_{j_2} sont les retards des tâches j_1 et j_2 pour les ordonnancement O et O'. Notons $\mathcal{T}$ et $\mathcal{T}'$ la valeurs des fonctions objectifs des ordonnancements O et O'.

Pour l'ordonnancement O, $C_{j_1} < C_{j_2}$ et $d_{j_1} > d_{j_2}$ implique que $T_{j_1} \leq T_{j_2}$.

Pour l'ordonnancement O', $C'_{j_2} < C_{j_2}$ implique que $T'_{j_2} \leq T_{j_2}$. De plus, $T'_{j_1} = \max(0, C'_{j_1} - d_{j_1}) = \max(0, C_{j_2} - d_{j_1})$ (car $C'_{j_1} = C_{j_2}$). Donc, $T'_{j_1} \leq T_{j_2}$ (car $d_{j_1} < d_{j_2}$). Donc, la fonction objectif de O' est inférieure ou égale à la fonction objectif de O. Comme O est optimal, O' l'est également.

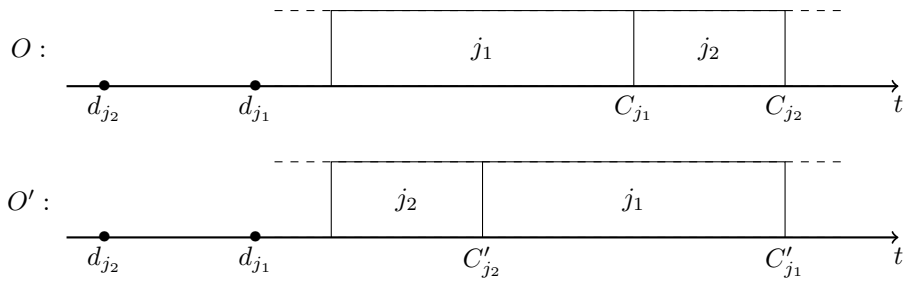

Fig. 1.6: Illustration de la permutation de deux tâches j_1 et j_2

Il est alors possible d'obtenir un ordonnancement optimal, respectant la règle EDD, par une succession d'opérations d'échange de tâches produites sur un ordonnancement qui ne respectent pas la règle de Jackson.

1.2.2.2 Procédure par séparation et évaluation (PSE)

La PSE (ou Branch and Bound en anglais) est une méthode visant à énumérer de manière exhaustive l'ensemble des solutions intéressantes d'un problème. Pour ce faire, la PSE génère un arbre de décision dont chaque embranchement correspond à une décision quant à la solution en cours de construction. Chaque nœud de l'arbre représente une solution partielle. Afin d'accélérer cette exploration, la PSE est dotée d'un mécanisme permettant d'évaluer ces solutions partielles et d'écarter celle ne pouvant aboutir à une solution optimale. Ce mécanisme est basé sur deux bornes : la borne supérieure et la borne inférieure.

Dans cette section, nous allons illustrer la PSE sur le problème de sac à dos décrit Section 1.1.1. Pour ce problème, l'arbre de décisions se construit d'après le choix des éléments mis dans le sac, représenté par la variable x_j (1, l'objet est pris dans le sac, 0 objet n'est pas pris).

Le premier noeud de l'arbre de décision s'appelle le noeud racine et représente une solution partielle vide (aucune décision n'a encore été prise). Chaque noeud a deux noeuds fils, considérant un objet donné j (et sa variable associée x_j), soit cet objet est placé dans la sac, ($x_j = 1$), soit il est exclu du sac ($x_j = 0$). Dans le cas du problème de sac à dos, les variables x_j sont généralement considérées par ordre d'utilité croissantes des objets j. Ce noeud racine et ces deux premiers fils sont présentés Figure 1.7.

Dans le cas d'un problème de maximisation (comme celui du sac à dos), la borne inférieure correspond à la meilleure solution trouvée pour ce problème, elle est mise à jour à chaque fois qu'une solution de meilleure qualité est trouvée. Elle peut-être initialisée grâce à une heuristique qui trouve une première solution réalisable avant de lancer la PSE. La borne supérieure s'applique à une solution partielle et permet d'estimer la valeur de la meilleure solution réalisable que l'on puisse espérer à partir de cette solution. Ainsi, dans le cas où la borne supérieure d'une solution partielle est inférieure à la borne inférieure trouvée pour le problème, cette solution partielle est donc non prometteuse et est exclue de la recherche.

Dans le cas du problème de sac à dos, une borne supérieure peut être calculée en considérant le

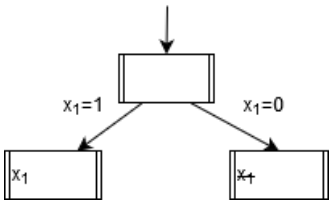

Fig. 1.7: Racine de la PSE

problème de sac à dos en variables continues (des fractions d'objets peuvent alors être placées dans le sac). Le problème de sac à dos en variables continues est de classe $\mathcal{P}$ et peut donc être résolu en temps polynomial de manière exacte.

Nous pouvons distinguer plusieurs types de nœud :

- les nœuds intermédiaires et non dominés, il reste des objets à ajouter dans le sac et la borne supérieure est plus grande que la borne inférieure.
- les nœuds feuilles pour lesquelles toutes les décisions ont été prises et qui peuvent mettre à jour la borne inférieure.
- les nœuds intermédiaires et dominés dans lequel la borne supérieure d'un noeud est inférieure à sa borne inférieure. Plus aucune recherche ne sera effectuée à partir de ce noeud.

Les noeuds non réalisables (avec une solution partielle dépassant la taille du sac) ne sont pas examinés.

Dans cet exemple, nous choisissons d'effectuer une exploration en profondeur, c'est-à-dire que nous explorons le plus profondément en priorité jusqu'à obtenir un noeud feuille, puis les autres nœuds sont explorés en commençant par le nœud le plus bas. Les objets sont placés dans le sac par ordre d'utilité. Pour un noeud donné, nous choisissons également d'explorer en priorité la branche où l'objet est placé dans le sac ($x_j = 1$) par rapport à celle où il ne l'est pas.

Ainsi, en reprenant l'exemple ci-dessus, les noeuds sont numérotés d'après leurs ordre d'exploration. Chaque nouveau noeud intermédiaire est composé d'une solution partielle, d'une borne supérieure (BS), d'une borne inférieure (BI). Les noeuds sont également numérotés en fonction de l'ordre de leur exploration.

Nous allons dérouler la PSE sur l'instance suivante, un sac a une capacité de 10 et 5 objets dont les caractéristiques sont décrites dans le Tableau 1.3.

Indice	1	2	3	4	5
Valeur	21	7	16	7	8
Poids	6	2	5	3	4
Utilité	3.5	3.5	3.2	2.3	2

Tab. 1.3: Instance de sac à dos

Comme première borne inférieure, l'heuristique gloutonne proposée en Section 1.2.1.1 est utilisée et retourne une solution avec une solution avec une valeur totale de 28. Le déroulé de la PSE est présentée en Figure 1.8.

Au départ de la PSE, la borne supérieure du noeud racine correspond à la valeur de l'objet 1 plus celle de l'objet 2 plus $\frac{2}{5}$ de l'objet 3 ($21+7+16\times 2/5$). Après l'ajout des deux premiers objets, le noeud 3 est un noeud feuille dont la valeur de la solution correspond à la borne inférieure déjà

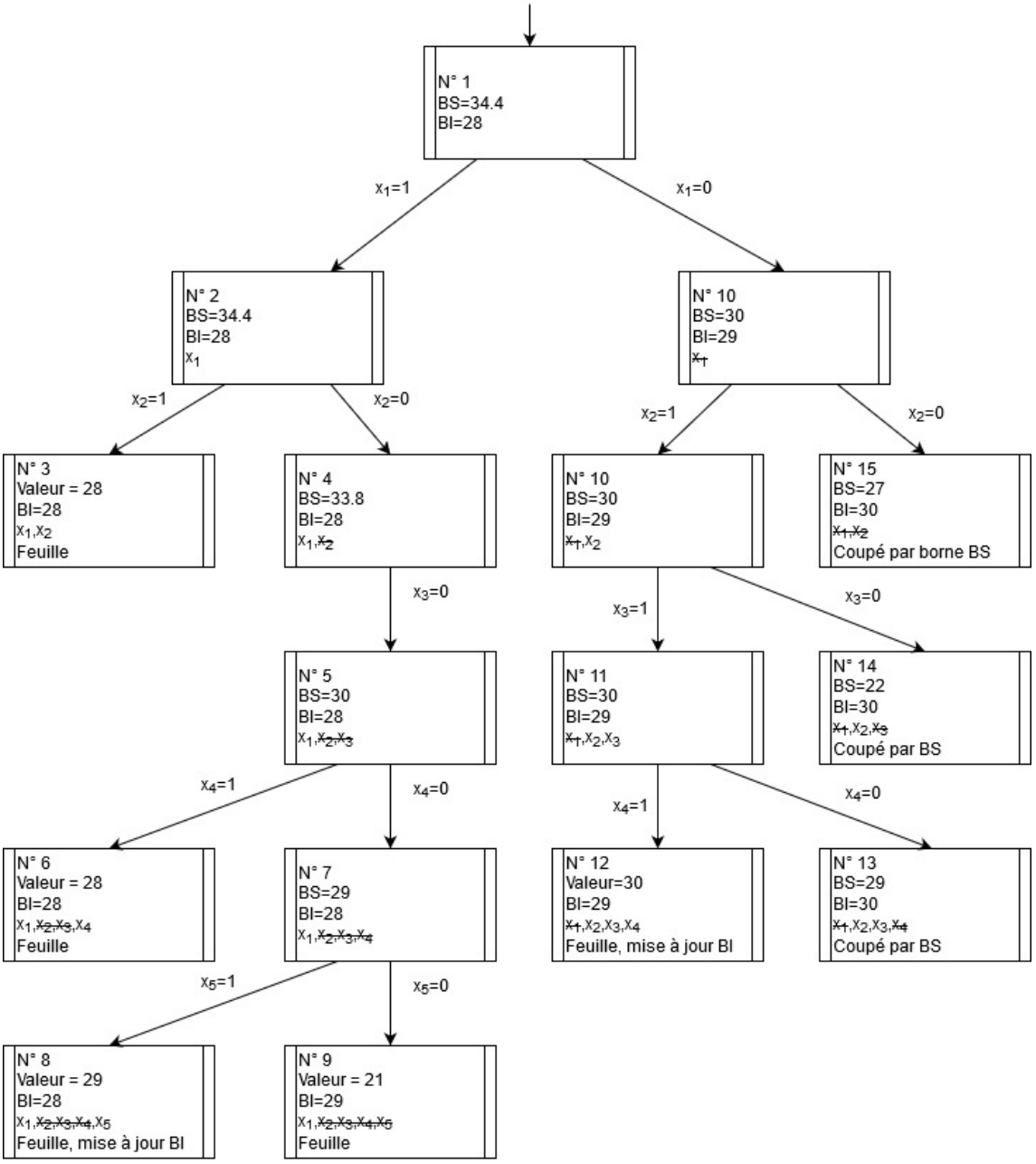

Fig. 1.8: Exemple de PSE

calculée. Il n'y a pas de mise à jour de la borne supérieure et la recherche remonte dans l'arbre (retour au noeud 2) pour continuer l'exploration. À partir du noeud 4, l'option où le premier et troisième objet sont dans le sac n'est pas considérée, car la contrainte de capacité du sac n'est pas respectée. Le noeud 6 est une feuille, mais ne permet toujours pas d'améliorer la borne inférieure. Le noeud 8 permet d'améliorer cette borne qui sera mise à jour pour les noeuds explorés par la suite. Le noeud 9 conclut l'exploration de la branche dans laquelle l'objet 1 est dans le sac. Les noeuds 10, 11, 12 correspondent à l'ajout dans le sac des objets 2, 3, 4 et mènent à l'amélioration de la borne inférieure. Lors de l'exploration du reste de l'arbre, les bornes supérieures des noeuds 13, 14, 15 sont plus faibles que la borne inférieure, ces noeuds sont coupés. La PSE prend fin. La solution optimale a été trouvée au noeud 12, composée des objets 2, 3, 4 avec une valeur de 30.

Il existe plusieurs pistes classiques d'amélioration d'une PSE dont les deux suivantes :

- Pour chaque noeud, calculer automatiquement les bornes supérieures de ses noeuds fils et se servir de cette estimation pour choisir la branche à explorer en priorité.
- Lors du calcul des bornes supérieures dans les nœuds intermédiaire, détecter les bornes réalisables et mettre directement à jour la borne inférieure.

1.3 Problèmes abordés dans cette thèse

Tous les problèmes abordés dans cette thèse partagent la caractéristique suivante : un ensemble de commandes doit être produit et livré à un ou plusieurs clients. Pour illustrer ce problème, considérons un pizzaiolo avec un ensemble de commandes à un instant donné. Chaque commande est une pizza qui doit être préparée en plusieurs étapes (étalage de la pâte, garniture, cuisson), placée dans un lot avec d'autres pizzas et chargée sur un scooter qui la livrera à son client pendant sa tournée. Comme dans cet exemple, nous considérerons des chaînes de production en plusieurs niveaux (appelé *flow-shop*). Nous considérons également des instances où les temps consacrés à la production et à la livraison sont du même ordre de grandeur afin que ces deux parties possèdent un impact similaire sur la date de livraison. Différents types de coûts sont présents, des coûts d'inventaire au sein de la chaîne de production, des coûts de livraison et des pénalités pour toutes commandes livrées en retard.

Ainsi cette thèse est liée à deux familles de problèmes prépondérant dans la recherche opérationnelle. Les problèmes d'ordonnancement (appelés aussi problèmes d'atelier) et les problèmes de livraison. Une rapide introduction à chacun de ces problèmes est fournie dans les Sections 1.4 et 1.5 suivantes.

Chacun de ces domaines possède son vocabulaire spécifique pour désigner une commande : job, travail, produit, bien... Dans cette thèse, nous utiliserons le terme de "commande" aussi bien pour les problèmes d'ordonnancement que de livraison. Par abus de langage, nous parlerons au sein des problèmes d'ordonnancement de production de commandes pour désigner la production du produit associé à la commande. La production d'une commande se fait ainsi par la réalisation d'un ensemble de tâches.

1.4 Les problèmes d'ordonnancement d'atelier

Les modèles *d'ordonnancement d'atelier* modélisent des processus de production que l'on peut rencontrer dans l'industrie. La production d'une commande connaît plusieurs étapes, appelés tâches. Chaque tâche a des caractéristiques différentes (durée de réalisation, consommation de

ressources, relation avec les autres tâches, date due etc...). Une chaîne de production est composée d'un ensemble de machines qui peuvent également avoir des caractéristiques diverses. Ces modèles sont soumis à des contraintes qui portent généralement sur le respect des capacités des machines et sur l'ordre de production des tâches. Les fonctions objectifs sont souvent liées aux dates de fin de production.

Un très célèbre article, [Graham et al., 1979], présente un état de l'art des connaissances de l'époque, ainsi que des résultats de complexité. Il introduit également une notation à trois champs $\alpha|\beta|\gamma$ permettant de décrire et classifier les problèmes d'ordonnancement. Cette notation, appelée notation de Graham, est encore très largement utilisée aujourd'hui. La suite de cette section est d'abord consacrée à la présentation de cette notation et des différents types d'atelier. Ensuite quelques résultats de complexité sont introduits. Notre travail est alors situé par rapport à cet état de l'art.

1.4.1 Notation de Graham

La notation de Graham suppose un ensemble de commandes J qui sont décomposés en un ensemble de tâches devant être réalisées sur un ensemble de machines. Chaque machine ne peut réaliser qu'une seule tâche à la fois.

1.4.1.1 Les commandes

Chaque commande j peut être décrite par les paramètres suivants :

- Un (ou plusieurs) temps de production p_j (ou $p_{i,j}$ où i désigne l'index de la tâche à réaliser).
- une date de disponibilité au plus tôt r_j à partir de laquelle la production de la commande j peut commencer.
- un poids w_j qui indique l'importance relative de la commande j.
- une fonction de coût f_j donnant le coût $f_j(t)$ si la commande j s'achève à la date t.
- une date de livraison d_j pour laquelle la commande est attendue par son client.

1.4.1.2 Environnement de production : champs α

Ce premier champ décrit l'agencement et la composition de l'atelier chargée de la production des commandes. Ce champ se décompose en deux parties : $\alpha = \alpha_1\alpha_2$. Pour les valeurs de $\alpha_1 \in \{\emptyset, P, Q, R\}$ chaque commande n'est composée que d'une seule tâche qui doit être réalisée sur une machine parmi un ensemble de machines. Pour $\alpha_1 \in \{F, J, O\}$, chaque commande à produire correspond à un ensemble de tâches à réaliser sur plusieurs machines. Le champs α_2 décrit le nombre de machines associées à la production. La liste ci-dessous détaille les différents ateliers de production du champs α_1.

- vide : une seul machine, dans ce cas α_2 vaut 1.
- P : machines parallèles identiques, toutes les commandes ont la même durée de production sur chaque machine.
- Q : machines parallèles uniformes, la durée de production des commandes est proportionnel à un facteur dépendant de la machine et s'applique de la même façon pour toutes les commandes.
- R : machines parallèles décorrélées : la durée de production est différente pour chaque commande sur chaque machine.
- F (*flow-shop*) : chaque commande est composée du même nombre de tâches. Chacune de ces tâches doit être réalisée sur une machine différente, l'ordre de passage des tâches sur les machines est le même pour toutes les commandes.

- J (*job-shop*) : Similaire au *flow-shop*, sauf que le nombre de tâches par commande et l'ordre de passage de ces tâches sur les machines sont spécifiques à la commande.
- O (*open shop*) : chaque commande est composée d'un ensemble de tâches qui doivent être réalisées pour que la commande soit achevée. Il n'y a pas d'ordre de passage imposé.

1.4.1.3 Champ des contraintes : champ β

Chacune de ces caractéristiques est optionnelle. Elles sont indiquées dans le champs β et séparées par une virgule. Les plus utilisées sont :

- *pmtn* : préemption autorisée, la réalisation d'une tâche peut être interrompue et reprise plus tard.
- *res* : présence de ressources limitées, la réalisation de chaque tâche mobilise une part de ces ressources. Si une ressource est complètement mobilisée, aucune nouvelle tâche consommant cette ressource ne peut être réalisée.
- *prec* : introduit des relations de précédence entre plusieurs couples de tâches. Lorsque deux tâches sont liées par une relation de précédence, l'une doit être finie avant de pouvoir entamer l'autre.
- r_j : chaque tâche est associée à une date avant laquelle cette tâche ne peut pas commencer.
- d_j : chaque tâche est associée à une date à laquelle cette tâche doit être terminée.
- $p_{i,j} = 1$: les temps d'exécution des différentes tâches valent 1.
- s_{j_1,j_2} : une durée de préparation de la machine est nécessaire entre chaque exécution des tâches j_1 et j_2. Cette durée qui varie selon les tâches qui se succèdent, est aussi appelée temps de *set-up*.

1.4.1.4 Critères d'optimisation : champs γ

Ce champ décrit la fonction objectif. Elle consiste à minimiser ou maximiser un ou plusieurs critères qui dépendent généralement des variables de décisions de chaque commande j comme :

- La date de fin C_j.
- L'écart entre la date de fin et la date souhaitée $L_j = d_j - C_j$.
- Le retard par rapport à la date souhaitée $T_j = \max(0, L_j)$.
- L'information binaire U_j que la commande est en retard ou non.

Chacune de ces variables être pondérée par un poids w_j représentant l'importance de la commande.

Pour définir un critère, on cherchera souvent a minimiser la plus grande valeur d'une de ces variables (C_{max}) ou la somme de ces variables, pondérée ou non ($\sum_{j\in J} C_j$ ou $\sum_{j\in J} w_j C_j$).

1.4.2 Exemples de notations et de résultats de complexité

- $1||T_{max}$: la minimisation du plus grand retard sur une seule machine, présenté en Section 1.2.2.1 est polynomial. Cependant, $1||\sum T_j$ est un problème $\mathcal{NP}$-difficile.
- $1||\sum w_j T_j$: la minimisation de la somme des retards pondérés pour une seul machine est $\mathcal{NP}$-difficile.
- $P2||C_{max}$: la minimisation du temps de production des tâches pour deux machines parallèles est $\mathcal{NP}$-difficile.
- $F2||C_{max}$: la minimisation du temps de production d'un flow-shop à deux machines, est un problème polynomiale, pouvant être résolue par l'algorithme de Johnson [Johnson, 1954]. Cependant, le problème devient $\mathcal{NP}$-difficile pour trois machines et plus.

1.4.3 Positionnement du sujet de thèse en termes d'ordonnancement

Les phases de production abordées dans cette thèse sont modélisées par un flow-shop de permutation ($Fm|perm|-$). Le terme *perm* du champ β décrit un flow-shop avec une contrainte supplémentaire sur l'ordre des commandes sur les machines. Cet ordre doit être le même pour chaque machine. Une solution se résume alors à une seule séquence qui définit dans quelle ordre les commandes doivent être préparé sur toute les machines, plutôt qu'une séquence de production par machine. Cette contrainte est largement répandue dans l'industrie. Elle paraît raisonnable aux agents de terrain et modélise les systèmes de production avec convoyeur (où l'ordre des commandes ne peut pas être modifié). Dans tous les problèmes abordés, un ensemble de lots K est traité. Chaque lot $k \in K$ étant composé d'un ensemble J_k de commandes qui seront livrées dans une même tournées. Des coûts d'inventaire sont pris en compte lors du stockage des matières premières avant le début de la préparation des commandes, pendant le processus de préparation (lorsque que deux tâches d'une commande ne sont pas réalisé strictement à la suite) et après la préparation. Ces coûts d'inventaire augmente généralement entre les différentes étapes de préparation de la commande. Après la production de chacune des commandes du lot, le lot est chargé dans un véhicule qui effectuera la livraison aux clients. La fonction objectif est composée de deux éléments : une somme de coûts d'inventaire pour chaque commande et une somme de coûts liés à la date de fin de production de chaque lot. Trois types de coûts d'inventaire sont pris en compte : un coût due au stockage des matériaux avant la production de la commande, un coût de stockage dès qu'une commande n'est pas traitée sur une machine une fois sa production lancée et un coût de stockage pour chaque commande produite en attente de livraison par un véhicule. L'ensemble des coûts d'inventaire est regroupé sous le sigle IC. Les retards de livraison aux clients sont représentés par le second coût. Ce coût est fonction de la date de fin de production du lot k. La fonction de coût associée à la date de fin de production d'un lot $k \in K$ dépend de la tournée utilisée pour livrer ce lot, nous montrerons que cette fonction est croissante.

Ainsi notre problème peut être décrit comme suit en notation de Graham : $Fm|perm|IC + \sum_{k \in K} f_k(C^k_{max})$, ou C^k_{max} est la date de fin du lot k (et également la date de départ de la tournée chargée de sa livraison) et f_k la fonction de coût associée à cette date de départ.

Un exemple de production et de ces coûts associés est présenté Figure 1.9. La chaîne de production est composée de deux machines m_1 et m_2. Quatre commandes, j_1, j_2, j_3 et j_4 (produites dans cette ordre) sont séparées en deux lots, $B_1 = \{j_1, j_2\}$ et $B_2 = \{j_3, j_4\}$. La partie haute de la Figure 1.9 est un diagramme de Gant représentant les durées opératoires des différentes tâches associées au commande ainsi que leurs dates de début et fin de production sur les machines m_1 et m_2. Les flèches au dessus de la machine m_1 représente les coûts de stockage en début de production, entre m_1 et m_2, les coûts de stockage en cours de production, en dessous de m_2, les coûts de stockage en fin de production.

À ces coûts de stockage s'ajoute un coût supplémentaire pour chaque lot B_1 et B_2. Ces coûts dépendent d'une fonction f_k spécifique à chaque lot k et qui elle-même dépend de la date de fin de production de chaque lot. Sur la figure, ces coûts sont représentés par $f_{B_1}(C^{B_1}_{max})$ et $f_{B_2}(C^{B_2}_{max})$.

1.4.4 Les coûts d'inventaire dans les problèmes d'ordonnancement

Cette section présente les principaux aspects de modélisation des coûts d'inventaire au sein des chaînes de production. Ces coûts peuvent être pris en compte à tout moment du cycle de production d'une commande.

Nous distinguons ici trois types de coûts, les coûts d'inventaires liés au stockage, ceux liés à la péremption des produits et ceux liés à l'immobilisation de capitaux.

Coûts d'inventaire liés au stockage : Ces coûts prennent en compte les dépenses mises en œuvre pour le stockage physique des commandes. Ils peuvent être dus à l'entretien des es-

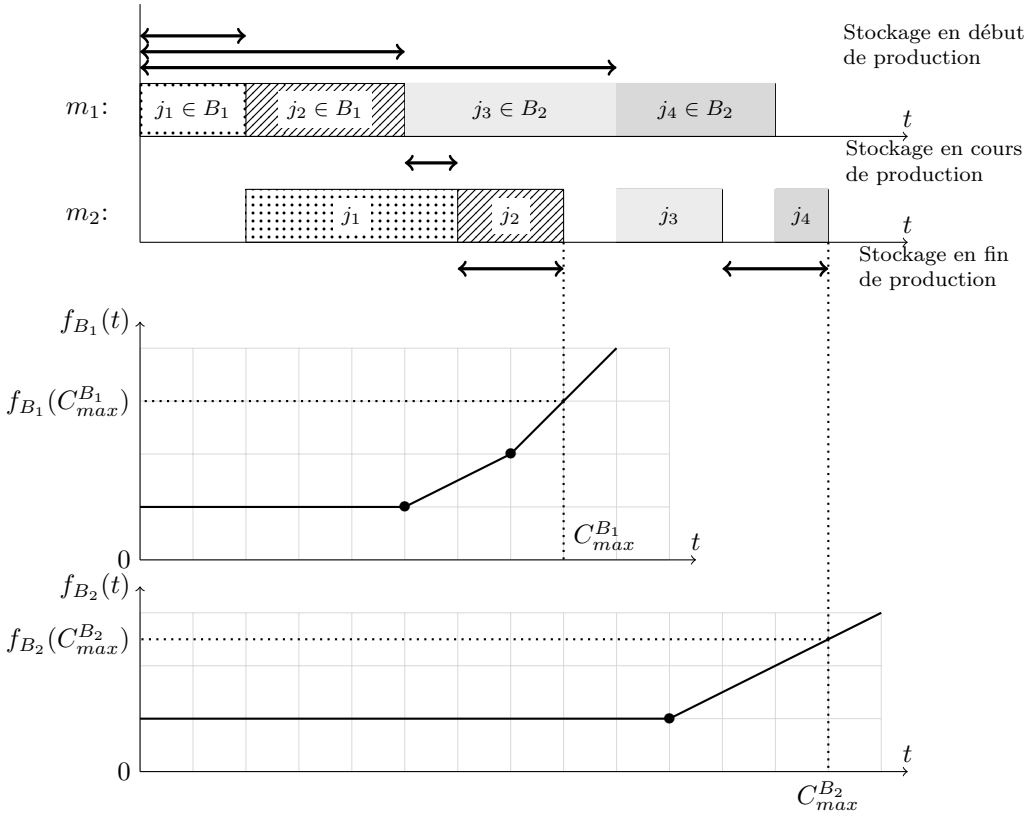

Fig. 1.9: Exemple d'ordonnancement et coûts associés

paces de stockage (entrepôts), ce qui inclut parfois le maintien de conditions particulières (environnement climatisé). Ces coûts sont spécifiques à chaque étape de production et peuvent varier d'une étape à l'autre. Par exemple, un combustible consommé lors d'une étape doit être stocké avant, mais pas après.

Coûts d'inventaire liés à la dégradation de la valeurs des produits : Ces coûts prennent en compte la perte de valeurs des produits sur la chaîne de production. Ces problématiques sont très présentes dans des domaines comme le médical et l'agro-alimentaire (par exemple, des réactifs peuvent perdre leurs efficacités chimiques ou encore des pertes de fraîcheurs alimentaires)

Coûts d'immobilisation des capitaux : Le stockage de produits est une immobilisation de capitaux. Les moyens financiers mobilisés par ce stock ne peuvent donc pas être investis ailleurs (par exemple sur les marchés financiers). C'est donc un coût supplémentaire pour l'entreprise. De plus, ces coûts d'inventaire augmentent au fur et à mesure de la production, lorsque le produit prend de la valeur.

En recherche opérationnelle, les coûts d'inventaire ne sont généralement pas comptabilisés pendant la réalisation d'une tâche sur une machine. Cependant, dans le cas où le temps de réalisation d'une tâche est incompressible, les coûts d'inventaire générés lors de la réalisation de cette tâche sont également incompressibles.

1.5 Les problèmes de tournées

Dans l'état de nos connaissances, il n'existe pas, pour les problèmes de tournées de véhicules, de système de notation aussi répandu et utilisé par la communauté scientifique que la notation de Graham pour les problèmes d'ordonnancement. Cependant des états de l'art sont régulièrement publiés comme par exemple les travaux de Gilbert Laporte ([Laporte, 1992a], [Laporte, 1992b], [Laporte, 2009] et autres).

1.5.1 Description du problème de voyageur de commerce

Le problème de base des problèmes de tournées est le problème du voyageur de commerce (ou Travelling Salesman Problem, ou TSP). Un voyageur doit visiter un ensemble de villes puis revenir à son point de départ. L'objectif pour le voyageur est de minimiser la distance parcourue sur l'ensemble de sa tournée. Le problème se modélise sous forme d'un graphe $G = (V, A)$ avec V l'ensemble des villes clients à visiter ainsi que le site de départ du voyageur et A l'ensemble des arcs reliant ces villes. Généralement, G est un graphe complet, chaque site de V est relié à chacun des autres par un arc dédié. La distance associée à chacune des arcs est répertoriée dans une matrice des distances. Il s'agit alors de minimiser la somme des distances des arcs composant le cycle parcouru par le voyageur.

La première extension de ce problème est le problème de tournées de véhicules (ou *vehicle routing problem*, ou VRP). Sur les mêmes bases que le TSP, il considère qu'une infinité de véhicules est disponible pour livrer ou collecter un ensemble de clients. Il est également assorti d'une contrainte supplémentaire, modélisant la capacité des véhicules en fonction des demandes à livrer ou collecter de chaque client. La somme des demandes collectées ou livrées par un véhicule ne doit pas dépasser sa capacité. On parle alors de *Capacited Vehicule Routing Problem* noté CVRP. À noter que sans une telle contrainte, la solution du TSP serait optimale pour le problème de VRP associé (en support un graphe routier classique qui respecte l'inégalité triangulaire). La fonction objectif est toujours de minimiser la distance totale parcourue par les différents véhicules.

Les paragraphes suivants décrivent quelques éléments importants de variantes du problème de tournées de véhicules.

1.5.1.1 Les matrices

L'utilisation d'une matrice de distances a été évoquée précédemment, mais d'autre types de matrices sont utilisées comme des matrices de durées de trajets ou plus généralement de coûts de trajets. Chacune de ces mesures peut être sujet à minimisation (minimisation du coût d'une tournée) mais également à contrainte. Par exemple, une tournée ne peut faire plus d'une certaine distance, contrainte technique d'autonomie du véhicule. Ou encore, une tournée ne peut faire plus d'une certaine durée due aux contraintes sur les temps de travail du personnel. Si elles sont présentes, certaines propriétés comme l'inégalité triangulaire, peuvent être utilisées dans la résolution des problèmes.

1.5.1.2 Type de flotte

Le nombre de véhicules disponibles rencontré dans les problèmes est soit un nombre fini n soit une infinité. Le nombre de véhicules réellement consacrés à la solution peut varier et chaque utilisation d'un nouveau véhicule peut être soumis à un coût fixe. Une flotte de véhicules peut être homogène (tous les véhicules ont les mêmes caractéristiques) ou hétérogène. La capacité des véhicules peut alors varier selon le véhicule, mais aussi toutes les matrices de coûts, distances, durées associées à un véhicule. Ce cas se rencontre dans le transport urbain de marchandises (camion, voiture, vélo). On parle alors de transport multimodal.

1.5.1.3 Type de livraison

Un problème de type VRP modélise soit un ensemble de biens à livrer depuis le dépôt vers les clients soit de biens à collecter depuis les clients vers le dépôt. Mais il existe également des problèmes de tournées qui considèrent plus généralement des biens à transporter d'un point de collecte vers un point de livraison. On parle alors de problème de *Pick-up and Delivery*. De même, un problème de type *backhaul* désigne le cas où un ensemble de sites doit être visité avant qu'un autre ensemble puisse être livré (souvent dans le cas où une contrainte impose la livraison de la totalité de la marchandise avant d'entamer une nouvelle collecte). Dans certains cas, un même véhicule peut-être utilisé pour effectuer plusieurs tournées, on parle alors de *multi-trip*.

1.5.1.4 Qualité de service

Les dates de livraison ou collecte aux clients sont souvent sujettes à contraintes. Certains sites doivent être livrés ou collectés avant une date donnée ou durant une fenêtre de temps pré-définie. Ce dernier cas est très souvent considéré, on parle alors de VRP avec *time-window* noté VRPTW. Ces fenêtres de temps peuvent être dures ou souples. Dans ce dernier cas, la violation d'une fenêtre de temps génère des pénalités.

1.5.2 Un résultat et quelques algorithmes

1.5.2.1 Le TSP

Le problème de TSP est $\mathcal{NP}$-difficile. Tous les problèmes évoqués précédemment peuvent être réduits à un problème de TSP, ainsi tous ces problèmes sont également $\mathcal{NP}$-difficile.

1.5.2.2 L'algorithme de *giant-tour (découpage de tournée géante)*

Cet algorithme est un exemple d'approche spécifique pour la résolution d'un problème CVRP classique. De plus il sera utilisé et adapté aux problèmes étudiés dans cette thèse.

L'algorithme de *giant-tour* proposé par Prins [Prins et al., 2014] intervient dans la résolution du CVRP (problème de tournées de véhicule avec capacité) à l'aide d'heuristiques. Il est utilisé lors des approches du type *routing first - cluster second*. C'est-à-dire où l'on choisit préalablement un ordre dans lequel les sites seront visités (*routing first*) et l'algorithme permet d'obtenir une décomposition optimale en tournées qui respecte cet ordre (*cluster second*).

Considérons l'instance suivante : une compagnie doit livrer un ensemble de 6 clients, elle dispose d'une flotte de véhicules de capacité 5 et chaque client j_1, j_2, j_3, j_4, j_5, j_6, réclame la livraison d'un colis prenant respectivement 1, 4, 2, 2, 1 et 3 unités de capacité d'un véhicule. Les principales distances entre les différents sites clients ainsi que les tailles des colis demandés sont présentées Figure 1.10.

L'algorithme prend en entrée un ordre dans lequel les sites doivent être livrés, par exemple $\{j_1, j_2, j_3, j_4, j_5, j_6\}$. Un graphe orienté est construit pour cette séquence (Figure 1.11, A). Ce graphe est composé d'un sommet par site plus un pour le retour final au dépôt et chaque arc représente une tournée réalisable par un véhicule respectant l'ordre donnée en entrée. Le point de départ d'un arc représente le premier site visité par la tournée alors que son point d'arrivée représente le site avant lequel la tournée prend fin (ce site n'est donc pas visité). Le poids d'un arc représente le coût de la tournée qui lui est associée. À titre d'exemple, l'arc (j_1, j_2) représente la visite du site j_1 et un retour au dépôt avant la visite du site j_2. Cet aller-retour représente un coût de 8. De même, l'arc reliant j_1 et j_3 représente la livraison des sites j_1 et j_2. Enfin, certaines arcs ne sont pas représentés car ils sont associés à des tournées non réalisables (contrainte de capacité). Par exemple l'arc (j_2, j_4), qui représente la livraison de j_2 et j_3, nécessiterait un véhicule de capacité 6, supérieure à la capacité des véhicules utilisés.

Chaque chemin dans le graphe représente une solution réalisable pour le problème. Un chemin avec 3 arcs représente une solutions avec 3 tournées. La solution optimale de ce sous-problème est donnée par le plus court chemin entre le sommet du premier site de la séquence et le sommet représentant le dépôt. Ce sous-problème est donc polynomial et peut être résolu à l'aide de l'algorithme de Bellman puisque le graphe est acyclique. Le graphe B, Figure 1.11, représente le sous ensemble de tournées optimales respectant la séquence de livraison précédemment énoncée. La fonction objectif vaut 28.

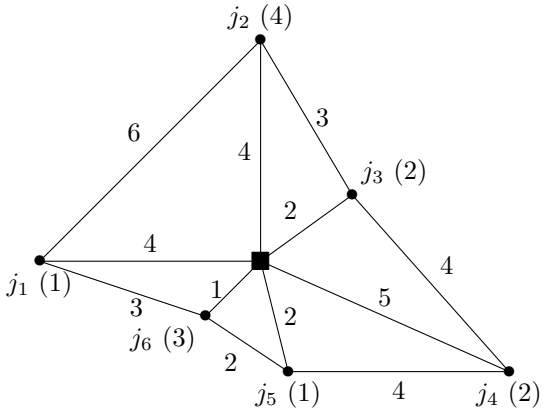

Fig. 1.10: Graphe de distances

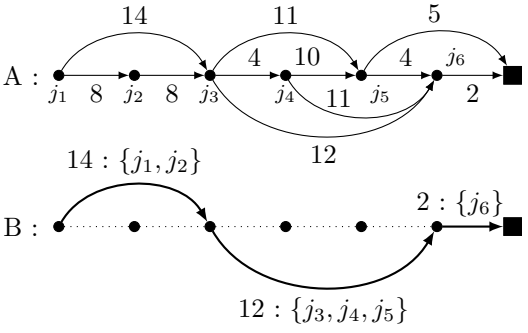

Fig. 1.11: Graphe orienté des tournées réalisables

1.5.3 Positionnement du sujet de thèse en termes de tournées de véhicules

Le problème de livraisons présenté dans cette thèse comporte de nombreux aspects assez classiques. Une flotte homogène de véhicules, sans contrainte de capacité, doit visiter un ensemble de clients, chacun à une date donnée. Une solution est évaluée par plusieurs coûts : un coût fixe pour chaque véhicule utilisé, un coût variable dépendant de la distance parcourue par les véhicules lors des tournées et des pénalités de retard. Dans cette thèse, nous travaillons avec une variante du problème avec fenêtres de temps ([Solomon, 1988]), très étudié dans la littérature. Chaque client a une date à laquelle il souhaite être livré. Mais cette contrainte est souple, une pénalité, proportionnelle à la durée de retard de livraison par rapport à cette date, doit être versée à chaque client livré en retard. Dans l'état de nos connaissances, cet aspect est peu pris en compte dans la littérature des problèmes de tournées.

Dans le cas d'un seul véhicule, le problème est similaire à un problème d'ordonnancement dans la littérature : un ensemble de commandes doit être produit sur une même machine, l'objectif est de minimiser la somme des retards pondérés des tâches en tenant compte de temps de *set-up* [Lee et al., 1997]. Voici sa notation selon Graham : $1|s_{j_1,j_2}|\sum w_j T_j$

Cette version ne tient cependant pas compte d'un cas à plusieurs tournées, du caractère symétrique de la matrice des distances et du coût associé au trajet parcouru.

1.6 Problème de gestion de la chaîne d'approvisionnement dans les grandes surfaces

Les problématiques de production et de livraison s'inscrivent dans le contexte plus large des chaînes de distribution. Cette partie se concentrera sur les chaînes d'approvisionnement des grandes surfaces. En nous appuyant sur [Fulconis et al., 2011], nous décrivons d'abord les différents acteurs de ce processus ainsi que les rapports de dominances qui les unissent. Ensuite, le cas concret d'une plate-forme de *cross-docking* d'une grande enseigne, pivot entre les fournisseurs et les magasins de la firme, est présenté. Pour finir, un parallèle est établit entre cet exemple et les travaux réalisés dans cette thèse.

1.6.1 Trois types d'agents

Les trois principaux agents présentés ici sont les producteurs, les distributeurs et les livreurs. Les producteurs, désignés aussi comme fournisseurs, produisent et écoulent l'essentiel de leurs productions auprès de la grande distribution qui sert d'intermédiaire entre eux et leurs clientèles. Les enseignes de grande distribution centralisent les produits de leurs fournisseurs afin que les particuliers puissent y avoir accès. Par exemple, les grandes surfaces se placent dans le rôle de client des producteurs. Enfin les livreurs (appelés également prestataires logistiques de services ou 3PL) se chargent de transporter les marchandises entre les sites des différents agents.

1.6.1.1 Relation entre fournisseur et distributeur

Le marché de la grande distribution française est globalement dominé par un petit nombre d'enseignes phares implantées sur tout le territoire. La compétition menée entre ces distributeurs pour accroître leurs parts de marché, outre l'importance d'assurer un accès continu au produit pour les particuliers, passe par une guerre des prix. L'argument du prix le plus bas est souvent important pour le particulier. Les distributeurs ont parfois l'ascendant sur leurs fournisseurs, particulièrement dans le cas de fournisseurs régionaux. En effet, le remplacement d'un produit donné par un substitut

d'une autre marque mènera rarement le particulier à se rendre dans une autre enseigne pour acheter ce produit (impact faible pour le distributeur). À l'opposé, un fournisseur peut difficilement se passer d'un distributeur (au vu de leur faible nombre) s'il souhaite toucher la clientèle la plus large possible. Ce rapport de force permet aux distributeurs d'être exigeants quant aux délais, aux quantités, aux fréquences et aux horaires de livraison demandées aux fournisseurs. Les commandes ont des délais courts allant de quelques jours jusqu'à un jour pour certains types de produits.

Par ailleurs, depuis quelques années, la politique du juste à temps (JAT) s'est imposée parmi les distributeurs. Elle permet de gérer la marchandise en flux tendu et ainsi de supprimer les réserves en magasin, ce qui génère des économies substantielles. Le JAT nécessite par contre un approvisionnement très régulier des distributeurs de la part de leurs fournisseurs, mais en plus petite quantité (souvent égal au volume pouvant être contenu en rayon). Ces livraisons ne se font pas directement en magasin, chaque distributeur contrôlant son propre réseau d'entrepôts et de cross dock centralisant les livraisons des fournisseurs avant de les redistribuer en magasin. Cependant le JAT confronte les fournisseurs à une explosion des besoins en transport (chauffeurs et véhicules devant être mobilisés pour livrer des quantités ne permettant pas de remplir les véhicules). Ces derniers ont donc de plus en plus recours à un ou plusieurs 3PL.

1.6.1.2 Les prestataires logistiques de services (3PL)

Depuis leurs apparitions dans les années 70, le rôle des prestataires logistiques de services au sein de la chaîne d'approvisionnement prend une importance grandissante. Profitant de la concentration des producteurs autour de leurs cœurs de métier, c'est-à-dire la conception et la production, les 3PL ont commencé à assumer les aspects transports et stockages de leurs clients puis se sont diversifiés dans des champs d'activités de plus en plus variées (comme le service après-ventes, le conditionnement, la traçabilité des marchandises).

1.6.1.3 Relation entre fournisseur et 3PL

Dans le cas où les deux agents sont liés par un simple contrat de transport, le fournisseur domine le 3PL. Au vue des pressions des grandes surfaces sur leurs fournisseurs, les dates de livraison, quantités et destinations ne sont pas négociables. D'un point de vue du prix, l'opération de transport requiert peu de compétences et par conséquent, de nombreux prestataires peuvent la réaliser. Un fournisseur peut alors choisir le 3PL qui se pliera au mieux à ses conditions.

Ce rapport de dominance s'estompe à mesure que les tâches confiées au 3PL se complexifient. En effet, en s'adaptant au plus juste aux demandes de son client, celui-ci devient indispensable.

1.6.2 Quelques cas concrets

1.6.2.1 Association de plusieurs fournisseurs à un 3PL

[Chenevoy, 2016] fait état de la situation d'un ensemble de fournisseurs du groupe Carrefour, confronté à l'augmentation de la fréquence d'approvisionnement et la diminution des volumes décrites Section 1.6.1.1. Dans un tel contexte, l'article rapporte :

« Notre réflexion a débuté quand une enseigne importante nous a demandé de livrer trois à quatre fois par semaine, non plus à la palette ou à la couche, mais par colis, se souvient Jean-Jacques Hénaff, président du conseil de surveillance de l'entreprise du même nom à l'occasion d'une conférence de GS1 sur la mutualisation logistique, qui s'est tenue le 10 mars à Paris. Et il n'était pas question de délocaliser nos usines. Nous avons alors étudié la possibilité de mutualiser

notre logistique avec d'autres entreprises de la région qui travaillent avec les mêmes clients, et qui sont soumises aux mêmes impératifs. Mieux vaut gagner à plusieurs que de perdre seul !»

Les modalités d'entente entre fournisseurs et 3PL ne sont pas décrites. On se trouve néanmoins dans un nouveau contexte de collaboration entre plusieurs fournisseurs, un 3PL et un distributeur. Dans ces conditions, la dominance des fournisseurs sur le 3PL s'estompe au vu de la qualité de service demandée et de la position centrale de ce dernier.

1.6.2.2 Cas d'un *cross-dock* d'un grand distributeur

Un cross dock est un entrepôt logistique permettant de centraliser les différents produits d'un distributeur avant de les redistribuer à ses différents magasins. Un cross-dock peut être, au choix, propriété de son distributeur ou bien géré par un 3PL. Le cas présenté ici est celui d'un *cross-dock* de produit sec, c'est-à-dire de produits avec une durée de conservation relativement longue et pouvant être stocké. Ce cas s'oppose à celui d'un *cross-dock* de produits frais où les marchandises doivent quasiment être expédiées dans la journée dans un des magasins du distributeur.

Le *cross-dock* tient deux rôles dans la chaîne d'approvisionnement entre les fournisseurs et les distributeurs :

1. le rôle de client lorsqu'il réceptionne les livraisons des fournisseurs pour le distributeur
2. le rôle de fournisseur lorsqu'il redistribue les produits aux différents magasins

***Cross-dock* comme client** Lorsqu'un distributeur passe commande à un fournisseur, les produits sont attendus à une date et un *cross-dock* donné. Cette date de livraison se situe 4 ou 5 jours après l'envoi de la commande au fournisseur. Celui-ci choisit une heure (notée H) pour livrer le jour demandé. Si le camion arrive avant H, il peut éventuellement être reçu en avance, mais sans garantie. S'il arrive entre H et $H+1$ il est considéré comme étant à l'heure. De $H+1$ à $H+3$, il est en retard mais sans pénalité. Passé $H+3$, des pénalités s'appliquent aux fournisseurs. Ce cas arrive pour en moyenne 3% des camions. Si le fournisseur sous-traite ses livraisons à un 3PL, libre à lui de se retourner contre lui. Le *cross-dock* garantit de décharger le camion sous trois heures à compter de H ou de l'heure d'arrivée. Ces chiffres ont été obtenus lors d'un entretien avec un gestionnaire d'un *croos-dock* d'une chaîne française de le grande distribution.

***Cross-dock* comme fournisseur** Chaque jour, le pôle logistique du distributeur passe commande au *cross-dock* de l'ensemble des produits nécessaires aux différents magasins. Dans cette situation, on considère que le livreur domine la situation. Le pôle logistique établit le nombre de camions, leurs chargements, les itinéraires effectués et leurs dates de départ. Le *cross-dock* doit préparer les commandes pour les dates fixées. Des règles d'usages garantissent que l'ensemble des commandes puissent être préparé à l'heure. Les retards sont donc rares et non pris en compte.

1.7 Plan de thèse

Un état de l'art détaillé des problèmes intégrées d'ordonnancement et de tournées de véhicules est présenté Chapitre 2. Un état de l'art sur les problèmes de production d'ordonnancement uniquement et les problèmes de tournées.

Le Chapitre 3 aborde un problème inspiré de la gestion aval d'un *cross-dock*. Un unique producteur doit livrer un ensemble de clients et fait assurer la livraison par un 3PL. Le Chapitre 4 propose une approche où producteur et 3PL sont deux agents distincts, collaborant mais défendant

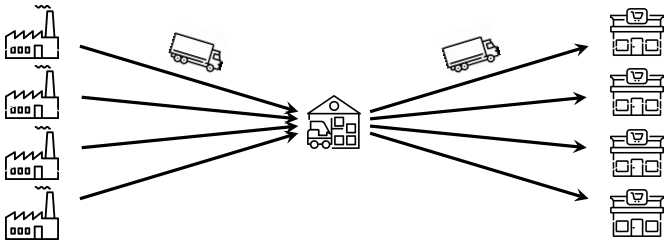

Fig. 1.12: Problème de production en amont et aval du *cross-dock*

leurs propres intérêts. Les conditions de leurs collaboration sont détaillées. Un GRASP et un algorithme génétique sont proposés pour résoudre le problème. Dans le Chapitre 5, la collaboration entre les deux agents est totale, ils cherchent tous deux à minimiser la même fonction objectif. Il est considéré que la composition des lots pour les différentes livraisons est connue. Cette propriété permet de proposer une méthode en deux étapes. Un ensemble de tournées est précalculé pour chaque lot et un itinéraire adapté peut ainsi être proposé quelque soit la date de départ de du lot en livraison. Un méthode à base de recherche locale est alors proposée pour fixer la séquence de production.

Le Chapitre 6 aborde un problème inspiré de la gestion amont d'un *cross-dock*. Un ensemble de producteurs doivent livrer un seul et même client. Afin de minimiser leurs coûts de transport, ils s'associent au même 3PL et mutualisent ainsi leurs livraisons. L'approche proposée considère que les itinéraires et les heures de passages de des tournées ont déjà été fixées d'un commun accord entre le 3PL et les producteurs. Pour chaque ensemble commandes à livrer, les producteurs doivent planifier leurs productions et affecter les commandes aux véhicules.

Enfin, nous concluons ce manuscrit par une conclusion générale, dans laquelle nous rappelons les résultats obtenus ainsi que les perspectives de recherche envisageables pour la poursuite de ces travaux.

Chapitre 2

État de l'art

Contents

Cet état de l'art se compose de trois sections. La Section 2.1 présente un historique de l'évolution des modèles traités dans la littérature des problèmes intégrés puis un état de l'art général sur les problèmes intégrés d'ordonnancement et de tournées de véhicules. La Section 2.2 se focalise sur l'état de l'art de chacun de ces sous-problème en dehors du cadre des problèmes intégrés. Enfin, la Section 2.3 élargit cet état de l'art en présentant un ensemble d'études connexes en lien avec la gestion de chaîne d'approvisionnement et la collaboration entre agents.

2.1 État de l'art sur les problèmes intégrés

2.1.1 Historique des problèmes intégrés

La résolution des problèmes de production et des problèmes de distribution sont deux champs de recherche distincts, même si leurs liens au sein de la chaîne d'approvisionnement sont évidents. Cette situation initiale s'explique simplement. La plupart de ces problèmes pris indépendemment sont déjà difficiles à résoudre (ex : TSP : $\mathcal{NP} - difficile$, $F3||C_{max} : \mathcal{NP} - difficile$).

Cependant les premiers travaux sur les problèmes intégrés montrent qu'une planification conjointe de la production et la distribution mène à une économie significative des dépenses pour les entreprises, comparée à une planification séquentielle où les problèmes associés à la production seraient d'abord résolus, puis les problèmes associés à la livraison ([Federgruen, 1984] : 6 à 7% d'économie, [Golden et al., 1984] : 23% d'économie, [Chandra, 1994] : 3 à 11% d'économie).

La résolution des problèmes intégrés présente donc un intérêt certain. De même, au vu du nombre de problèmes modélisables, son champ d'application est exponentiel, tout en restant limité par la grande complexité de leurs résolutions. Ainsi nous proposons de présenter les étapes de l'évolution des modèles traités pour les problèmes intégrés présentés dans la littérature à travers 3 revues de la littérature espacées dans le temps : [Sarmiento, 1999], [Chen, 2010] et [Moons et al., 2017].

Dans [Sarmiento, 1999], les auteurs présentent un état de l'art dans lequel la plupart des problèmes traités lient la production à la livraison par le biais de stocks. La livraison est alors largement associée à des problèmes de tournées de véhicules mais la production n'est pas abordée au niveau opérationnel. Un exemple de problème classique présenté dans cet état de l'art est l'IRP (Inventory Routing Problem). Un producteur doit assurer l'approvisionnement d'une ressource à un ensemble de clients sur un ensemble de périodes. Les clients et le producteur disposent d'une capacité de stockage tampon. À chaque période, le producteur a la possibilité de lancer la production de cette ressource, pour un coût fixe lié au lancement de la machine et un coût variable lié à la quantité produite. Il peut également réapprovisionner les différents clients en affrétant un ou plusieurs véhicules et en résolvant le problème de tournées associé. L'une des contraintes principales de ce problème est de s'assurer que les clients ont toujours accès à la quantité requise de ressource pour chaque période. L'objectif de l'IRP est de minimiser les coûts de production fixes et variables, les coûts d'inventaire liés au stockage entre les différentes périodes et les coûts de livraison.

Ce problème et ces variantes sont encore largement étudiés aujourd'hui ([Fumero, 1999], [Lei et al., 2006], [Absi et al., 2015], [Absi et al., 2018], [Neves-Moreira et al., 2019]).

Cependant, on peut noter que ce genre de problème ne place pas la production et la livraison sur le même horizon de temps. La production se planifie sur un ensemble de périodes alors que chaque tournée débute et finit à la même période. La livraison est gérée au niveau opérationnel ("au jour le jour") alors que la production est gérée à un niveau plus haut, sur une échelle de temps plus long en se basant sur les stocks comme variables d'ajustement entre la demande et la capacité de production.

Chen ([Chen, 2010]) précise dans son état de l'art de 2010 et que les articles prenant en compte un ordonnancement au niveau opérationnel sont plus récents et que la majorité de ces travaux a été publiée à partir de 2005. Afin de répertorier ces nouveaux modèles, qu'il nomme IPODS (*Integrated Production and Outbound Distribution Scheduling*), il étend la notation de Graham ($\alpha|\beta|\gamma$, Section 1.4) en lui ajoutant deux champs : π et δ, spécifiant respectivement les caractéristiques du processus de livraison utilisé et le nombre de clients livrés. Cet état de l'art classe les modèles en cinq catégories :

1. Modèles avec livraison individuelle des commandes, immédiatement après la fin de leurs préparations.
2. Modèles avec livraison par lot des commandes à un unique client.
3. Modèles avec livraison par lot des commandes et chaque livraison se faisant vers un unique client.
4. Modèles avec livraison par lot des commandes et pour chaque lot, un problème de tournées de véhicules est associé.
5. Modèles avec dates de départ fixes des véhicules.

Comme le remarquent les auteurs de [Moons et al., 2017], les modèles de catégories 1, 2 , 3 et éventuellement 5 de [Chen, 2010] ne considèrent que des livraisons directes aux clients, sans problème de tournées. Ces articles prennent ainsi le contre-pied des travaux présentés dans [Sarmiento, 1999], qui considèrent pour la plupart des problèmes de tournées de véhicules.

Dans [Moons et al., 2017], les auteurs se concentrent sur les problèmes intégrés prenant en compte explicitement un ordonnancement au niveau opérationnel ainsi que des problèmes de tournées de véhicules lors de la livraison des commandes. Il s'agit d'IPODS de catégorie 4 (et éventuellement 5) de la classification de Chen. Il recense en 2017 33 études de cette catégorie avec plus de 60% des travaux publiés après 2010.

Les auteurs de [Moons et al., 2017] concluent que les problèmes abordés dans la littérature doivent encore être adaptés au contexte industriel. En effet, une majorité des études présentées dans leurs états de l'art modélisent la production à l'aide d'une unique machine ou de machines parallèles, alors que les environnement destinés à la production de masse correspondent d'avantage à des *flow-shop* ou des *job-shop*. De même, étant donné que les producteurs font maintenant largement appel à des prestataires de service, la prise en compte de flottes de véhicules hétérogènes dans la modélisation des problèmes serait une plus-value. Enfin les auteurs rappellent que les entreprises sont confrontées à des problèmes de très grandes tailles et qu'il est important de proposer des algorithmes opérant dans un laps de temps court et capable de résoudre des instances de grandes tailles.

On peut remarquer une certaine corrélation entre l'évolution des modèles étudiés sur les problèmes intégrés et les besoins au sein de la chaîne logistique. On peut considérer par exemple l'approvisionnement des grandes surfaces. Jusque dans les années 2000, la part réservée à l'espace de stockage dans les grandes surfaces était importante. Durant cette période, les premiers problèmes intégrés ont été étudiés, avec des modèles de type IRP et en se basant sur les stocks présents dans les usines et dans les magasins. À partir de années 2000, la politique du "juste à temps" se développe et fait diminuer les stocks, apportant de nouvelles problématiques et faisant évoluer les modèles de ces problèmes intégrés.

Aujourd'hui, pour toujours diminuer les stocks et d'augmenter la qualité de service, la politique du "juste à temps" s'est généralisée. La recherche opérationnelle propose maintenant des modèles où la production et la distribution sont étroitement imbriquées, permettant parfois de répondre à une commande (sa production et sa distribution) dans la journée. De plus, les modèles doivent continuer à s'adapter à des situations de plus en plus complexes. On peut remarquer que ces deux évolutions ont été permises par l'augmentation des capacités numériques, telles que le stockage de données (big-data), les capacités réseaux (suivi de stock) et la puissance de calcul.

2.1.2 Littérature des problèmes intégrés

Dans cette section, nous ne considérons que les problèmes intégrés prenant en compte une composante de production au niveau opérationnel (excluent les problèmes de type IRP). Ensuite, nous faisons la distinction entre les articles ne considérant pas de problème de tournées (livraison direct par exemple - Section 2.1.2.2) et les articles considérant les problèmes de tournées (Section 2.1.2.1). Dans chacune de ces sections, les études présentées sont classifiées selon se fait d'après le type de chaîne de production utilisée.

2.1.2.1 Modèles sans prise en compte de problèmes de tournées

Modèles à machine unique : Dans [Condotta et al., 2013], les auteurs proposent un problème à une seule machine où les commandes (liées à une date de début de production au plus tôt et une date de livraison) doivent être regroupées en lot avant d'être livrées à un unique client. La livraison se fait en direct par une flotte de véhicules à capacité limitée. L'objectif est de minimiser le plus grand des retards. Le problème est résolu à l'aide d'une recherche tabou. Dans [Koç, 2017], un ensemble de produits est transporté depuis un entrepôt jusqu'à un site de transformation (à une machine) pour être ensuite ramené à l'entrepôt initial après traitement. Les mêmes véhicules sont

utilisés pour emmener et ramener les produits. Des coûts de stockage sont pris en compte à l'entrée et à la sortie du site de transformation. Une PSE ainsi que plusieurs heuristiques sont mises en place pour résoudre ce problème. Les auteurs de [Cheng et al., 2019] considère un problème similaire, à l'exception de la machine qui traite les commandes par lot. Le temps de traitement d'un lot correspondant à celui de la commande nécessitant le traitement le plus long. Les auteurs proposent des algorithmes polynomiaux avec un ratio de 2 par rapport à l'optimal dans le pire des cas. Dans [Wang et al., 2013], les auteurs présentent également un problème similaire où un unique produit est transporté pour être transformé puis livré à un unique client avant des dates données et dans des quantités précises. La fonction objectif prend en compte les coûts de transport, les coûts d'inventaire et les coûts de lancement de la machine. Un modèle mathématique est également proposé pour résoudre ce problème. Enfin, dans [Fu et al., 2018], les auteurs considèrent un ensemble de commandes avec des dates de disponibilité et des dates de livraison. Ces commandes sont à livrer à un unique client par plusieurs véhicules identiques. L'objectif est de minimiser le nombre de véhicules à utiliser pour livrer les commandes en respectant les délais. Des cas où une commande peut être produite ou livrée en plusieurs fois sont étudiés. Les auteurs fournissent des preuves de complexité, des algorithmes de résolution polynomiaux et des méthodes exactes pour certains cas plus difficiles.

Modèles à machines parallèles : Dans [Cakici et al., 2014], les commandes sont produites sur un ensemble de machines parallèles classique. Les commandes destinées à un même client sont regroupées en lots puis livrées par un unique véhicule à capacité restreinte. Ce modèle minimise la somme des dates de livraison, pondérée en fonction des commandes. Une large gamme d'heuristiques est utilisée pour sa résolution (glouton, sac à dos ...). Dans [Nogueira et al., 2020], le modèle étudié présente un ensemble de machines parallèle préparant les commandes par lot (four de cuisson). Une fois préparées, les commandes sont assignées à une flotte de véhicules puis livrées. Il y a donc deux phases d'affectation : les commandes aux machines et les commandes aux véhicules. Chaque commande a une taille, une durée opératoire, une date de livraison et un profit. L'objectif est de maximiser le profit généré par les commandes livrées à l'heure. Ce modèle est résolu à l'aide d'une modélisation mathématique et d'algorithmes polynomiaux. Enfin, dans [Li et al., 2008], les auteurs présentent un modèle dans lequel le 3PL domine. Un ensemble de commandes doit être ordonnancé sur un ensemble de machines parallèles puis livré. Les dates de départ des véhicules ainsi que leur capacités et leurs itinéraires sont déjà fixés. L'objectif est de minimiser les coûts d'affectation des commandes aux véhicules ainsi que les pénalités d'avance et de retard des commandes. Une méthode de résolution en deux parties est proposée, allouant d'abord les commandes aux véhicules, puis les commandes aux machines. Un recuit simulé est développé pour résoudre ce problème.

Modèles avec d'autres chaînes de production : Dans [Agnetis et al., 2014], les auteurs considèrent deux sites de production distants, composés chacun d'une machine. Les commandes doivent être traitées par le premier puis le second site. Au sortir du premier site, un système de transport est mis en place pour convoyer les commandes semi-finies jusqu'au second en un temps limité. Deux modes de transport sont utilisés : un mode de transport régulier, dont les créneaux sont connus à l'avance, et un mode de transport expresse, qui peut être sollicité à n'importe quel moment mais génère un surcoût. Dans les deux cas, la capacité des véhicules doit être respectée. L'objectif est de minimiser les coûts de transport. Deux scénarios sont considérés : un en faveur du producteur et un autre en faveur du 3PL. Les auteurs présentent des études de complexité et exhibent certains cas polynomiaux et d'autres $\mathcal{NP} - difficiles$. Dans [Dong et al., 2013], les auteurs présentent un *open-shop* à deux machines où les commandes doivent être livrées à un unique client en utilisant un seul véhicule avec capacité. L'objectif est de minimiser la dernière date de livraison au client. Pour résoudre ce problème, les auteurs proposent un algorithme polynomial, avec une garantie que la solution retournée n'est pas 2 fois pire que la solution optimale dans le pire

des cas (algorithme de ratio 2) et montrent que ce ratio peut être amélioré si l'on considère chaque job livré indépendemment. Les auteurs de [Han et al., 2019] propose un modèle où les commandes sont produites en deux étapes sur des sites distincts avant d'être livrées individuellement à leurs client. Les dates de départ pour la livraison finale étant fixées, un coût d'inventaire est prise en compte entre la fin de la deuxième étape de production et le départ de la livraison. Après une étude de complexité les auteurs proposent des algorithmes polynomiaux pour certains types d'instances. Dans [Hassanzadeh et al., 2016], les auteurs considèrent un problème de *flow-shop* bi-objectif où les commandes sont livrées par lot, mais ne considérant qu'un coût fixe pour chaque livraison. Plusieurs heuristiques multi-objectif sont proposées pour résoudre ce problème.

2.1.2.2 Modèles avec prise en compte de problèmes de tournées

Modèles à machine unique : Dans [Armstrong et al., 2008], les auteurs proposent un problème incorporant une chaîne de production à une seule machine produisant des commandes pour un ensemble de clients. Dans ce problème le véhicule effectue une unique tournée une fois la production terminée. La séquence de production des commandes et la séquence de livraison des clients sont identiques et déjà connues. En raison de certaines contraintes supplémentaires (durée de vie des commandes et fenêtres de livraison), l'ensemble des clients ne peux être livré dans des conditions satisfaisantes, il s'agit donc de choisir le sous ensemble de clients à livrer afin de maximiser le nombre total de demandes satisfaites. En exploitant la structure du problème, les auteurs développent une PSE pour ce problème ainsi qu'une borne inférieure permettant d'améliorer la vitesse de résolution. Cependant une erreur dans cette méthode est trouvée et corrigée dans [Viergutz, 2014]. Dans ce nouvel article, les auteurs proposent d'étendre le modèle de [Armstrong et al., 2008] en permettant de modifier conjointement la séquence de production et la séquence de livraison. Une recherche tabou ainsi qu'une recherche locale sont développées pour résoudre ce nouveau problème. Dans [Devapriya et al., 2017], les auteurs prennent également en compte la durée de vie des commandes. Dans ce problème, une machine fabrique un produit périssable qui est livré par une flotte de véhicules avec capacité. L'objectif est de satisfaire l'ensemble des demandes tout en minimisant les coûts fixes et variables liés au transport. Les auteurs proposent une série d'heuristiques pour résoudre ce problème dont la qualité des solutions est évaluée à l'aide d'une borne inférieure. Dans [Cheref et al., 2016], les auteurs proposent également un problème avec une chaîne de production à une machine et un seul véhicule pour réaliser les livraisons. Les commandes ne sont disponibles à la production qu'à partir d'une certaine date et sont munies de dates de disponibilité et de date de livraison. Cet article propose des solutions robustes pour ce problème. Les auteurs considèrent que chaque durée de production et chaque durée de livraisons appartient à un intervalle de temps. Ces durée ne sont pas connues avec exactitude. L'objectif est de trouver une solution satisfaisante pouvant être ajustée pour satisfaire l'ensemble des scénarios et maximiser un critère de robustesse. Une recherche tabou est proposée comme méthode de résolution. Les auteurs de [Geismar et al., 2008] abordent un problème où un unique type de produit est fabriqué. Ce produit est périssable et doit être livré dans des délais courts une fois produit. La livraison est faite par un unique véhicule. La résolution de ce problème se fait en deux phases. D'abord l'ensemble des trajets réalisables sont explorés, (cet ensemble est restreint du fait de la durée de vie des produits et de la capacité des véhicules) puis la production est planifiée. De même, les auteurs de [Lacomme et al., 2018] considèrent un unique produit au sortir d'une chaîne de production et devant être livré à un ensemble de clients en respectant sa durée de vie. Les cas de un et plusieurs véhicules chargés de la livraison sont considérés, l'objectif étant de satisfaire l'ensemble des demandes le plus rapidement possible. Les auteurs développent un GRASP et une recherche locale qui montre de meilleurs performances que la littérature déjà existante pour ce problème. Dans [Li et al., 2005], les auteurs explorent une série de cas, variant autour de la capacité du véhicule (limitée ou illimitée), et du nombre de clients (un ou plusieurs). Pour tous ces cas, l'objectif est de minimiser la somme des dates de livraison. Des algorithmes polynomiaux sont proposés pour résoudre certains modèles de manière exacte.

Modèles à machines parallèles : Dans [Fu et al., 2017], chaque commande peut être réalisée simultanément sur plusieurs machines, cependant un temps et un coût de *set-up* (phase de préparation d'une machine pour traiter un type de tâche différent de celui traité précédemment) sont à prendre en compte avant la production de chaque type de commandes. La livraison aux clients est effectuée par une flotte de véhicules hétérogènes et doit respecter des fenêtres de temps. De plus, deux modes de transport sont utilisés, un mode livrant plusieurs clients en une seule tournée et un mode de livraison en direct. L'objectif de ce problème est de minimiser les coûts de *set-up* liés à la production ainsi qui les coûts de transport. Une modélisation mathématique est proposée pour ce problème ainsi qu'une heuristique de résolution en deux phases. Dans leurs articles, les auteurs de [Tavares-Neto, 2019] considèrent également des temps de *set-up* entre la production de chaque commande. Les commandes sont livrées par un unique véhicule effectuant plusieurs voyages. Les méthodes de résolution proposées sont un GRASP et un algorithme génétique. Dans [Chang et al., 2014], les auteurs considèrent un ensemble de machines parallèles non corrélées avec une livraison effectuée par une flotte homogène de véhicules avec capacités. Un algorithme de colonie de fourmis est proposé pour résoudre le modèle et minimise la somme des coûts de transport et des dates de livraison. Dans un contexte chimiothérapie, comme dans [Kergosien et al., 2017], les médicaments ont une courte durée de vie entre la fin de leurs productions et leurs inoculations aux patients. Les médicaments sont produits sur un ensemble de machines parallèles et livrés aux différents sites par un unique véhicule. L'objectif est de minimiser le plus grand des retards. Une heuristique, basée sur la décomposition de Benders, est proposée ainsi que plusieurs bornes, inférieures et supérieures, pour résoudre ce problème. La préparation de repas industriels nécessite aussi de considérer la production et la livraison des commandes de manière conjointe pour garantir leurs fraîcheurs. Dans [Farahani et al., 2012], juste après sa production, chaque commande génère des coûts supplémentaires jusqu'à sa livraison aux clients. Ces coûts simulent la perte de qualité du produit. La méthode de résolution proposée est composée de 3 étapes successives : les commandes sont d'abord mises en lot, puis affectées aux différentes machines (parallèles) et les itinéraires des tournées de livraisons sont finalement établis. Les auteurs comparent leurs méthodes sur la base d'instances fournies par un entreprise danoise et parviennent à améliorer les précédents résultats fournis par cette entreprise. L'article [Amorim et al., 2013] se place également dans le contexte où de la nourriture périssable doit être produites sur un ensemble de machines parallèles puis être livrée. L'objectif est de respecter les fenêtres de livraisons des clients tout en minimisant les coûts de transport. Cette article propose également une modélisation où une même commande peut être produite sur plusieurs machines. Des problèmes intégrés sont également utilisés pour modéliser l'impression et la diffusion de journaux. Par exemple, [Chiang et al., 2009] présente un cas réel de cette industrie où différentes éditions locales doivent être imprimées à l'aide d'une série de presses (machines parallèles). Une fois produits, les journaux sont envoyés dans plusieurs centres de distribution avant d'être livrés. L'objectif est de minimiser l'ensemble des coûts de transport des journaux. La méthode de résolution proposée est une heuristique en deux étapes, résolvant d'abord une version relâchée du problème, puis l'améliore par recherche tabou pour obtenir une solution réalisable. Les auteurs proposent aussi une évaluation de la robustesse des solutions trouvées. Dans les deux problèmes précédents, le temps consacré à la production et à la distribution sont du même ordre de grandeur. Dans le cas de l'impression de journaux, la production démarre vers minuit, avec l'arrivée des nouvelles de dernières minutes et la livraison chez l'abonné doit se faire à sept heure le lendemain. La synchronisation entre la production et la distribution est alors capitale. Enfin, les auteurs de [Wang et al., 2019a] considèrent une livraison effectuée par une flotte de véhicules avec capacité. Des fenêtres de livraison doivent être respectées malgré des temps de transport variables. L'objectif est de maximiser la robustesse de la solution. Un algorithme génétique est proposé pour résoudre le problème.

Modèles avec d'autres chaînes de production : Le seul article référencé par [Moons et al., 2017] et proposant un système de production de type *flow-shop* est [Scholz-Reiter et al., 2010]. Dans cet

article, plusieurs machines sont disponibles à chaque étape du *flow-shop* pour réaliser les tâches des commandes. Le stockage d'une commande entre chaque étape de production génère des coûts d'inventaires. Une fois produites, elles sont assignées aux véhicules en charge de leurs livraisons. L'objectif global est de minimiser la somme des coûts d'inventaire, des coûts de transport et de retards à la livraison. Ainsi ce modèle est complet, mais aucune méthode de résolution n'est proposée en dehors de la résolution du modèle mathématique. Plus récemment, les auteurs de [Yağmur, 2020] proposent un nouveau modèle dans lequel un ensemble de commandes doit être réalisé par un *flow-shop* de permutation avant d'être livré aux différents clients par un seul véhicule à capacité limitée. L'objectif est de minimiser simultanément le temps de transport et les retards de livraison aux clients. Les auteurs proposent un algorithme mémétique (algorithme à population) pour résoudre ce problème. Les auteurs de [Wang et al., 2019b] considèrent un *flow-shop* à 3 niveaux et un problème de livraison avec plusieurs véhicules. L'objectif est de minimiser la dernière date de livraison. Deux méthodes de résolution sont proposées : une procédure de recherche par voisinage et une heuristique constructiviste. Dans [Mohammadi et al., 2020], une compagnie propose de fabriquer et livrer une série de produits personnalisés à leurs clients. La chaîne de production est composée d'un *job-shop* dans lequel chaque machine ne peut réaliser qu'un sous-ensemble de tâches définies. Les commandes sont livrées par lot grâce à une flotte hétérogène de véhicules de capacité limitée. Des fenêtres de temps doivent être respectées lors de la livraison. Le problème est bi-objectif. Dans un premier temps, la compagnie cherche à minimiser les coûts de production et de distribution, et dans un second temps, les coûts liés à l'avance et au retard des commandes. Certains article considèrent également le cas d'une production multi-site. Dans [Marandi, 2018], les auteurs considèrent un ensemble de commandes devant être produites sur un ensembles de sites distincts, chaque site ne pouvant réaliser qu'un sous-ensemble de tâches. À chaque étape de la production, la commande doit être transportée un site capable de réaliser la tâche courante. Chaque site est composé d'une unique machine. Les commandes finalisées sont acheminées dans un dépôt et des tournées regroupant les commandes de plusieurs clients sont effectuées pour assurer la livraison finale. Les auteurs cherchent à minimiser les pénalités de retard de livraison aux clients ainsi que l'ensemble des coûts de transport.

2.1.2.3 Synthèse

Le Tableau 2.1 est une synthèse des articles présentés dans cette section. Pour chaque article est indiqué le type de chaîne de production utilisé, les type de coûts d'inventaire ("STR" pour les coûts avant la production, "WIP" pour les coûts en cours de production et "FIN" pour les coûts en fin de production, avant le départ de la tournée), le type de problème de tournée (livraison direct ou non, avec ou sans capacité). De plus, nous donnons un aperçu des différents coûts considérés dans la fonction objectif ("CdP" : coûts de production, "IC" : coûts d'inventaire, "RC" : coûts de transport, fixe ou variable, "PC" : pénalités dues aux clients livrés en retard et des méthodes de résolution.

Dans [Nogueira et al., 2020], chaque commande est associée à un bénéfice qui est rapporté si la commande est livrée dans les temps, d'où la notation "Max bénéf.". Dans [Li et al., 2008], les auteurs minimisent les coûts d'affectation des commandes aux véhicules, d'où la notation "Affec.". D'autres types d'objectifs peuvent être minimisés : (D_{max} : minimisation de la dernière de date livraison, C_{max} : minimisation de la dernière date de production, T_{max} : minimisation du plus grand des retards, $\sum D_i$ et $\sum w_i D_i$: minimisation de la somme des dates de livraison, pondérée ou non). Enfin, le terme "Robustesse" décrit la maximisation de la résistance d'une solution au aléas. C'est-à-dire, le fait que qu'une solution reste réalisable, même si les données de l'instance résolue varient.

Concernant les méthodes de résolutions, ce tableau utilise de nombreux acronymes et abréviations : Algo poly : Algorithme polynomial, AG : Algorithme génétique, GRASP : *Greedy randomized adaptive search procedure*, PSE : Procédure par séparation et évaluation, ELS : *Evolutionary lo-*

cal search, PSO : *Particule Swarm Optimisation*, Bender : décomposition de Bender, ACO : *Ant Colony Optimisation*.

2.2 Littérature spécifique aux sous-problèmes de production et de distribution

Les problèmes intégrés résolus dans cette thèse sont tous composés d'un sous-problème de production et d'un sous-problème de livraison. Cette partie vise à situer chacun de ces sous-problèmes par rapport à sa littérature d'origine.

2.2.1 Le problème de *flow shop* avec coût d'inventaire

La partie production du problème intégré étudié dans cette thèse est composée d'un *flow-shop* de permutation prenant en compte des coûts d'inventaire en début, en cours et en fin de production. La modélisation d'une mise en lots pour livrer la commandes dans le problème global impacte le modèle de production de deux façons : 1) les coûts d'inventaire en fin de production de chaque commande dépendent de la mise en lot. 2) Pour chaque lot est associé un coût de livraison. Ce coût de livraison est croissant en fonction de la date de fin de préparation du lot.

Le problème de "flow-shop" est très largement étudié dans la littérature des problème d'ordonnancement et ceux depuis de nombreuses années ([Gupta, 1979], [Minella et al., 2008]). Ce problème autorise de nombreuses variantes parmi lesquelles la variante "de permutation" est très étudiée comme en attestent [Ruiz, 2005] et [Fernandez-Viagas et al., 2017]. En effet, l'hypothèse que la séquence de préparation des commandes soit la même sur toutes les machines de la chaîne de production est couramment employée dans l'industrie.

La prise en compte des coûts d'inventaire au sein d'une chaîne de type *flow-shop* est aussi présente dans la littérature. Dans [Bülbül et al., 2004], les auteurs étudient un "flow-shop" à m machines (pas de permutation) minimisant les coûts d'avance et de retard des commandes produites ainsi que les coûts d'inventaire intermédiaires. Ils proposent une heuristique basée sur une relaxation Lagrangienne combinée à une décomposition de Dantzig-Wolfe, l'ensemble de la méthode de résolution s'appuyant sur une génération de colonne. Cette méthode parvient à trouver des résultats proche de l'optimal dans des temps très courts pour des instances jusqu'à 30 commandes. Dans [Mishra, 2018], les auteurs présentent un problème de "flow-shop" de permutation prenant en compte de coûts d'inventaire en début, en cours et en fin de production ainsi que le retard des commandes, l'ensemble de ces critères devant être minimisé. Les auteurs proposent trois algorithmes de population différents pour résoudre ce problème ainsi qu'un recuit simulé. Une comparaison de ces différentes méthodes est proposée.

La mise en lots des commandes lors de la production est également présente dans la littérature, et sert à modéliser des situations diverses. Par exemple, dans [Potts, 2000], les auteurs présentent un état de l'art des problèmes d'ordonnancement avec mise en lots et détaillent deux grands types de problèmes. Dans un cas, des similarités au sein des commandes justifie un classement en famille. Les commandes d'une même famille doivent être regroupées de préférence au sein du même lot pour diminuer les temps et les coûts de *set-up*. Dans un autre cas, le regroupement en lots est nécessaire pour des machines traitant plusieurs commandes simultanément (opération de cuisson par exemple). Dans [Lin, 2005], les auteurs proposent une problème de *flow-shop* à deux machines avec une mise en lots nécessaire à l'entrée de la seconde machine, l'objectif étant de réguler les temps de *set-up* et de minimiser la date de fin de production générale. Quelques cas particuliers sont résolus de manière polynomiale, le cas général est résolu à l'aide d'une heuristique. Dans [Shen et al., 2014], les auteurs considèrent des familles de commandes à ordonnancer sur une chaîne de type *flow-shop* et prenant en compte des temps de *set-up*. L'objectif est toujours de minimiser

Auteurs	Chaîne de production			Coût d'inventaire	Livraison		Véhicule avec capacité	Qualité de service		Fct. oblectif (Minimisation)	Méthode de résolution
	Une machine	Machines parallèles	Autres		Directe	TSP		Date de livraison	Durée de vie		
[Mohammadi et al., 2020]			•			•	•	•		CdP, PC, VC	Heuristique
[Nogueira et al., 2020]		•			•		•	•		Max bénéf.	Algo poly
[Yağmur, 2020]			•			•	•	•		RC, PC	AG
[Cheng et al., 2019]	•			STR & FIN	•		•			CdP, IC, RC	Algo poly
[Tavares-Neto, 2019]		•		FIN		•	•			C_{max}	Glouton
[Han et al., 2019]			•	FIN	•		•	•		IC, RC, PC	Algo poly
[Wang et al., 2019a]		•					•	•		Robustesse	AG
[Wang et al., 2019b]			•			•	•	•		D_{max}	Heuristique
[Fu et al., 2018]	•				•		•	•		RC	PSE, Algo poly
[Lacomme et al., 2018]	•					•	•		•	C_{max}	GRASP et ELS
[Marandi, 2018]			•			•		•	•	RC, PC	Heuristique, PSO, AG
[Devapriya et al., 2017]	•					•	•		•	RC	AG
[Fu et al., 2017]		•		STR & FIN	•		•	•		CdP, RC	Heuristique
[Koç, 2017]	•					•				IC, RC	Heuristique
[Kergosien et al., 2017]		•				•			•	T_{max}	Bender
[Cheref et al., 2016]	•			WIP & FIN	•			•	•	Robustesse	Tabou
[Hassanzadeh et al., 2016]			•		•			•	•	IC, RC, PC	Glouton, AG
[Cheng et al., 2015]	•				•		•			CdP, RC	ACO
[Agnetis et al., 2014]			•		•		•	•		RC	Algo Poly
[Cakici et al., 2014]		•			•		•			$\sum w_i D_i$	Heuristique
[Chang et al., 2014]		•					•			RC, $\sum w_i D_i$	ACO
[Amorim et al., 2013]		•				•	•		•	RC	Modèle math
[Condotta et al., 2013]	•				•		•	•		T_{max}	Tabou
[Dong et al., 2013]			•		•		•			D_{max}	Algo Poly
[Ullrich, 2013]		•				•	•			PC	AG
[Wang et al., 2013]	•			WIP, FIN	•		•	•		CdP, IC, RC	PSE, Heuristique
[Farahani et al., 2012]		•				•		•		CdP, RC, PC	Heuristique
[Scholz-Reiter et al., 2010]			•	WIP & FIN		•	•	•		IC, RC, PC	-
[Chiang et al., 2009]		•				•		•		RC	Heuristique, Tabou
[Armstrong et al., 2008]	•						•	•	•	PC	PSE
[Geismar et al., 2008]	•				•		•		•	C_{max}	AG
[Li et al., 2008]		•			•		•	•		Affec., PC	Recuit simulé
[Li et al., 2005]	•					•	•			$\sum D_i$	Algo Poly
Positionnement Chapitre 3, 4, 5			•	STR, WIP, FIN		•		•		IC,RC,PC	
Chapitre 6			•	STR, WIP, FIN		•	•	•		IC,RC,PC	

Tab. 2.1: Tableau récapitulatif

la date de fin de production. Les auteurs proposent de résoudre le problème à l'aide d'un ensemble de recherche tabou.

Dans l'état de nos connaissances, il n'existe pas de problème considérant un problème de "flow-shop" (sans partie livraison) et prenant en compte simultanément des contraintes de mise en lots en vue d'une livraison et des coûts d'inventaire.

2.2.2 Le problème de livraison avec pénalité de retard

La partie livraison des problèmes peut être traitée de deux façons. Soit les lots doivent être composés et le problème est alors une variante du VRP ("*Vehicule Routing Problem*"). Soit les lots sont déjà composés et résoudre ce problème revient alors à résoudre une succession de problèmes plus petits, variantes du TSP ("*Travelling Salesman Problem*"). Dans ces deux cas, la plupart des modèles rencontrés sont standards : chaque client doit être visité une et une seule fois, la flotte de véhicules est homogène (avec ou sans capacité), les coûts de transport sont fixes (pour chaque véhicule utilisé) et variables (en fonction de la distance parcourue).

Les problèmes de VRP ont été intensivement étudiés depuis 60 ans et nous vous renvoyons aux différentes revues de littérature de Gilbert Laporte sur le sujet ([Laporte, 1992a], [Laporte, 1992b], [Laporte, 2009]).

La spécificité des problèmes de tournées de cette thèse réside dans les coûts à verser aux clients. Cette spécificité est modélisée dans la littérature comme une variante d'un problème avec des fenêtres de temps "souples" (*soft time windows*). Ce problème se note alors dans littérature VRPSTW (*vehicule routing problem with soft time windows*) par opposition au VRPHTW (*vehicule routing problem with hard time windows*) dans lequel les fenêtres de temps doivent être respectées strictement. Le second cas est le plus étudié et se retrouve souvent sous la notation VRPTW. Le livre [Cordeau, 2000] propose un état de l'art des ces différents problèmes.

Les fenêtres de temps "souples" peuvent être prises en charge de différentes façons. Dans [Mouthuy et al., 2015], les auteurs minimisent trois objectifs dans l'ordre lexicographique : le nombre de véhicule utilisé, le nombre de fenêtre non respectées puis la distance parcourue. Il résolve se problème en utilisant une recherche de voisinage élaborée. Dans [Salani et al., 2016], les auteurs cherchent à minimiser le coût associé au dépassement des fenêtres de temps sous contraintes de ne pas dépasser un certain coût de transport. Dans la mesure où le problème considère un ensemble réduit de tournée, les auteurs présentent des méthodes exactes pour le résoudre. Dans [Taillard et al., 1997], les auteurs considèrent toujours un problème de VRP où un véhicule peut arriver chez un client avant le début de la fenêtre de temps mais doit attendre le début de celle-ci pour pouvoir livrer. Le 3PL paye une pénalité de retard si celui-ci livre au-delà de la fenêtre. Le problème est résolu à l'aide d'une recherche tabou. Les problèmes de tournées abordés dans cette thèse sont similaires à ceux de [Taillard et al., 1997], il suffit de considérer que le début de toutes les fenêtres de temps se trouve en 0.

Plus récemment, les auteurs de [Shelbourne et al., 2017] traitent un problème de livraison considérant explicitement deux dates par commande. Une date avant laquelle la commande ne peut pas partir en livraison et une date due de livraison. L'objectif est de minimiser la somme pondérée du temps de trajet et des pénalités de retard. Le problème est résolu à l'aide d'heuristiques.

2.3 Études connexes

Cette section a pour but d'élargir l'état de l'art et donner des pistes de rapprochement entre la recherche opérationnelle et la gestion de la chaîne d'approvisionnement, en particulier sur les possibilités de collaboration entre les différents agents de cette chaîne.

2.3.1 À propos de la chaîne d'approvisionnement

Pour commencer, dans [Goel, 2018], les auteurs remarquent qu'une grande majorité des articles issus de journaux spécialisés en recherche opérationnelle et en lien avec le transport de marchandises se focalisent principalement sur les contraintes physiques (capacités des véhicules, fenêtre de temps pour les visites des clients) en négligeant les contraintes humaines (limitation du temps de tournée, respect de temps de pause). Les solutions proposés ne sont ainsi pas directement applicables en pratique. Cet article propose une étude de la réglementation légale en Europe et pointe les principaux écarts entre cette réglementation et les modèles abordés en RO. Il propose également des solutions pour réduire cet écart.

Pour avoir un aperçu plus détaillé des relations entre agents dans la chaîne logistique (relation de dominance, service échangé, évolution des relations), nous renvoyons vers [Fulconis et al., 2011]. Cet ouvrage est issu de la littérature de management. À titre d'exemple, le terme de "distribution" ne désigne pas une opération de livraison par un véhicule mais touts les mécanismes politiques et marketing employés par les entreprises et visant à organiser la mise à disposition de produits à leurs clients ("la grande distribution"). Cette ouvrage ne présente cependant pas de cas concret (contexte précis, description de contraintes, données) permettant la définition directe d'un modèle de recherche opérationnelle.

2.3.2 À propos de la collaboration

Concernant la collaboration entre agents au sein de la chaîne d'approvisionnement, il existe deux axes d'étude : le partage de l'information et le partage des coûts.

La plupart des articles incluant plusieurs agents dans leurs modèles font l'hypothèse d'un partage total de l'information. Or, nous savons qu'en pratique, les opérations de production et de livraison sont souvent réalisées par des entreprises distinctes qui ne partagent pas toutes leurs données. Dans [Viet et al., 2018], les auteurs proposent un état de l'art centré sur la *valeur de l'information* (VOI). Une étude sur la VOI cherche à déterminer quels types d'informations doivent être échangés entre agents et de quelle manière cet échange impacte la collaboration au sein de la chaîne d'approvisionnement.

La question du partage des coûts entre les agents peut également composer un sujet d'étude. Dans [Tijs, 1986], les auteurs présentent un court état de l'art sur le problème de répartition des coûts conjoints. Ce problème se rencontre en pratique lorsque des entreprises décident de s'associer pour minimiser les coûts. En d'autres termes, si une première entreprise réalise seule un ensemble de tâches pour un coût A et une second entreprise réalise seule un autre ensemble de tâches similaires pour un coût B et si ces deux entreprises décident de s'associer pour réaliser ensemble leurs deux tâches pour un coût C (C < A+B), comment le coût C doit être réparti entre les deux entreprises ? Cet article propose plusieurs méthodes (issues de la théorie des jeux) pour résoudre ce problème.

De manière plus concrète, dans [Hezarkhani et al., 2016], les auteurs adressent un problème de livraison de type *pick-up delivery* (Section 1.5.1.3). Deux compagnies de livraison réalisent le même type de prestation. Dans un premier temps, l'article décrit un scénario dans lequel chaque compagnie réalise seule ses livraisons puis décrit un scénario dans lequel les compagnies mettent en commun leurs ressources pour réaliser l'ensemble de leurs prestations. L'article propose une solution de répartitions des coûts et prouve que cette solution garantit certaines propriétés d'équité et de compétitivité.

Enfin, la thèse de [Phouratsamay, 2017] propose un panel d'approches très large sur le partage des coûts et des informations au sein de la chaîne logistique. Le cadre de cette thèse est un problème de *lot-sizing* à deux niveaux qui modélise les capacités de production et les besoins d'un producteur et d'un distributeur. En posant des hypothèses sur le modèle, comme les relations de dominance des deux agents et les informations disponibles, l'auteur dérive différents problèmes et propose de

méthode de résolution, souvent polynomiales.

2.4 Conclusion

Les problèmes intégrés suscitent l'intérêt de la communauté de la recherche opérationnelle depuis de nombreuses années. Étant des problèmes difficiles, les modèles abordés par les chercheurs ont évolués avec les capacités de résolutions mais aussi avec les besoins de l'industrie. Aujourd'hui, le champ d'étude des problèmes intégrés est très large.

La première partie de ce chapitre résume, sur la base de différentes revues de la littérature, les différentes évolutions des modèles intégrés étudiés en recherche opérationnelle. On peut ainsi distinguer trois étapes. D'abords sont principalement étudiés des problèmes où seul la livraison est considérée à un niveau opérationnel. Puis, courant des années 2000, apparaissent des modèles où l'ordonnancement est étudié au niveau opérationnel, mais alors le problème de livraison se résume souvent à des livraisons directes aux clients. De nos jours, de plus en plus d'articles présentent des modèles avec l'ordonnancement traité au niveau opérationnelle et des problème de tournées de véhicules à résoudre pour planifier la livraison.

La deuxième partie propose un état de l'art des problèmes intégrés considérant un ordonnancement au niveau opérationnelle. Les problèmes sont classés d'après le mode de production utilisé et le fait qu'ils considèrent ou non un problème de tournée pour la livraison. Dans cette thèse, nous choisissons d'étudier des modèles dont toutes les caractéristiques ont déjà été étudiées au sein de problèmes intégrés, mais jamais ensemble dans un le même modèle. À titre d'exemple, et dans l'état de nos connaissances, il n'existe pas de problème intégré traitant simultanément un *flow-shop* de permutation, des coûts d'inventaire et des livraisons modélisées par des tournées de véhicules.

La troisième partie de ce chapitre propose deux état de l'art spécifiques : un sur les *flow shop* avec coûts d'inventaire et un autre les problèmes de livraison considérant des pénalités de retard.

Enfin, la dernière partie élargit cet état de l'art en présentant un ensemble d'études connexes en lien avec la gestion de chaîne d'approvisionnement et la collaboration entre agent.

Chapitre 3

Problème général : un producteur, un distributeur et plusieurs clients

Contents

3.1 Présentation du problème

Dans cette partie, nous étudions un problème intégré au niveau opérationnel au sein de la chaîne d'approvisionnement et impliquant deux acteurs : un producteur et un prestataire logistique de services (3PL). Une première version de ce problème a d'abord été décrite dans [Rohmer, 2015]. Le problème intégré consiste en deux sous-problèmes : un producteur doit planifier la réalisation des tâches associées à un ensemble de commandes en résolvant un problème d'ordonnancement de type *flow-shop* de permutation prenant en compte des coûts d'inventaire. Une fois la production effectuée, les commandes à livrer sont regroupées en lots et confiées à un 3PL qui devra en planifier la livraison. Chaque tournée a un coût fixe et un coût variable et des pénalités doivent être versées à chaque client livré en retard.

Concernant la chaîne de production, les matières premières nécessaires à la fabrication d'une commande sont initialement stockées jusqu'au début de la production de cette commande. Toutes les commandes à préparer sont divisées en un ensemble de tâches. Chaque tâche doit être réalisée sur une machine dédiée. L'ordre de réalisation des tâches des différentes commandes sur les machines est le même pour toutes les machines, de même que l'ordre de réalisation des tâches d'une commande est le même pour toutes les commandes (*flow-shop* de permutation). Une commande est ordonnancée successivement sur chaque machine de la chaîne de production. Une commande est stockée si, à la fin de sa préparation sur une machine, elle n'est pas directement réalisée par la machine suivante. La finalisation d'une commande sur la dernière machine entraîne également son stockage jusqu'au départ du véhicule chargé de sa livraison. Chaque étape de stockage génère un coût, dépendant de la commande et proportionnel à la durée du stockage. On parle de stockage avant production, en cours de production et en fin production.

Après la production, les commandes sont regroupées en lot pour être livrées par le 3PL. Chaque

lot mobilise un véhicule et génère un coût fixe. Ce véhicule part dès que les commandes de son lot sont terminées. Ainsi, le temps de stockage d'une commande après sa production dépend du lot auquel elle appartient. Enfin le 3PL établit l'ordre de livraison de chaque commande pour chacun des véhicules. Le trajet entre chaque site client génère des coûts de transport. Ces itinéraires permettent de déterminer la date de livraison des commandes. Chaque commande est attendue pour une date de livraison donnée. Si la date de livraison réelle dépasse cette date, une pénalité est versée au client, proportionnelle à la durée retard de sa commande. La tournée débute au dépôt du producteur où se situe la chaîne de production et prend fin au dépôt du 3PL.

On suppose par ailleurs que le 3PL connaît à l'avance la composition des lots à transporter et peut adapter si nécessaire la taille du véhicule avant le départ de la tournée pour y loger toutes les commandes du lot.

La décision charnière entre les deux acteurs est la composition des lots. Elle affecte à la fois la production par le biais des coûts d'inventaire et la livraison. En effet, tous les clients d'un lot doivent être visités par la même tournée.

Pour résumer, ce problème comporte trois prises de décision :

- l'ordonnancement des tâches associées aux commandes,
- le choix du nombre de lots et leurs compositions,
- l'itinéraire de chaque tournée destinée à livrer les commandes d'un lot.

Enfin, la solution de ce problème doit prendre en compte trois types de coûts :

- les différents coûts d'inventaire liés à la production,
- les coûts, fixes et variables, liés à la livraison des lots,
- les pénalités dues aux clients pour le non respect des conditions de livraison.

Ce problème est relativement générique. Il est particulièrement pertinent dans un contexte pharmaceutique ou de restauration à domicile dans lesquels les produits sont périssables. Dans ces situations, un producteur spécialisé (laboratoire ou restaurant) délègue souvent l'opération de livraison à un prestataire extérieur. De plus, un tel contexte implique, soit un laps de temps très court entre l'arrivée de la commande et la date de livraison souhaitée, soit des temps de stabilité des produits également très courts (ce qui est particulièrement vrai dans le cas de production de chimiothérapies). Dans les deux cas, production et livraison représentent une part importante du délais entre le début de production de la commande et sa livraison au client. C'est pourquoi production et livraison doivent être planifiées ensemble afin de minimiser les retards.

La prise en compte de coût d'inventaire le long de la chaîne de production se justifie par plusieurs aspects. Toujours dans le cas de produits périssables, ces coûts représentent la perte de valeur d'une commande si elle doit être stockée. Ces coûts prennent aussi en compte la consommation d'infrastructures (espace, environnement climatisé, ...) pendant le stockage et éventuellement l'immobilisation de capitaux financiers. Ainsi, nous considérons dans ces chapitres que les coûts d'inventaire unitaires associés à chaque commandes augmentent le long de la chaîne de production.

Dans de nombreux contextes comme celui de la pharmaceutique, la taille des lots à distribuer est généralement petite. Il est donc raisonnable de considérer que chaque transporteur a une capacité suffisante dans son véhicule.

Enfin, la prise en compte de pénalité de retard est très présente en pratique et peut se concevoir à la fois comme une perte de qualité du produit, mais également comme une insatisfaction du client. Ces deux éléments augmentent avec la quantité de retard.

3.2 Formulation mathématique

Cette section introduit les notations, paramètres et variables utilisés pour modéliser le problème général. Chaque instance du problème est définie par un ensemble de machines I de taille m, un ensemble de commandes L de taille o et d'un ensemble de véhicules de taille suffisante pour transporter chaque commande individuellement (au moins o véhicules). Pour chaque commande l, la tâche réalisée sur la machine i est associée à un temps de production $p_{i,l}$. Chaque commande l possède également un coût de stockage de sa matière première avant production, noté h_l^{START}, un coût de stockage entre la machine i et la machine $i+1$, noté $h_{i,l}^{WIP}$, un coût de stockage en fin de production, noté h_l^{FIN}, une date de livraison souhaitée par son client, notée d_l et une pénalité de retard π_l en cas de non respect de la date de livraison. Tous les coûts et pénalités présentés ci-dessus sont exprimés par unité de temps. Le coût fixe d'une tournée se note c^V. Le temps et le coût de transport associés aux déplacements d'un véhicule entre le site l_1 et le site l_2 se notent tt_{l_1,l_2} et tc_{l_1,l_2}.

Les variables utilisées pour décrire ce problème sont les suivantes. Les variables y_{l_1,l_2} servent à modéliser la séquence de production choisie. Les variables $C_{i,l}$ correspondent aux dates de fin de préparation des tâches associées aux commandes. Les variables Z_k indiquent si le véhicule k est utilisé et les variables $z_{l,k}$ indiquent si la commande l est transportée par le véhicule k. Les variables x_{k,l_1,l_2} indiquent si deux sites visités lors de la même tournée se succèdent dans cette tournée. Les ensembles F_k et f_l indiquent respectivement les dates de départ du site de production des véhicules et des commandes. D_l et T_l représentent respectivement les dates de livraison et le retard des commandes.

L'ensemble des coûts associés à ce problème est formulé en quatre expressions : IC, les coûts d'inventaire. VC et RC, les coûts fixes et variables de livraison. PC, les pénalités dues aux clients. La Figure 3.1 représente, pour un ordonnancement d'une commande l, les différents types de périodes de stockage impliquant des coûts d'inventaire pour un *flow-shop* à deux machines.

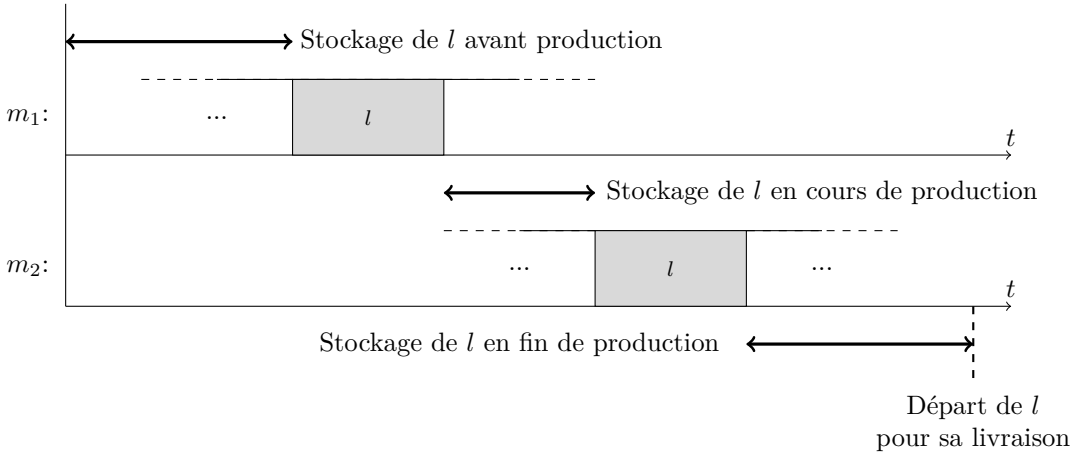

Fig. 3.1: Période de stockage d'une commande l au cours de sa production

L'ensemble des notations du problème est résumé dans le Tableau 3.1 et la modélisation mathématique du problème est présentée par le Modèle 3.1.

Lors de la production, nous considérons que les index 0 et $o+1$ représentent des commandes fictives en début et en fin de production. De même, lors de la livraison, les index 0 et $o+1$ représentent les points de départ et d'arrivée d'une tournée.

Tab. 3.1: Résumé des notations

Ensembles	
m	Nombre de machines
M	Ensemble des machines ($\{1, \ldots, m\}$)
o	Nombre de commandes
L	Ensemble des commandes ($\{1, \ldots, o\}$)
K	Ensemble des véhicules
Paramètres	
$p_{i,l}$	Temps d'exécution de la commande l sur la machine i, $\forall i \in M, \forall l \in L$
h_l^{START}	Coût de stockage par unité de temps en début de production de la commande l, $\forall l \in L$
$h_{i,l}^{WIP}$	Coût de stockage par unité de temps en cours de traitement de la commande l entre la machine i et la machine $i+1$, $\forall l \in L$, $\forall i \in M \backslash \{m\}$
h_l^{FIN}	Coût de stockage par unité de temps en fin de production de la commande l, $\forall l \in L$
c^V	Coût fixe d'une tournée
tt_{l_1,l_2}	Temps de trajet du site l_1 au site l_2, $\forall (l_1, l_2) \in (0, \ldots, o+1)^2$
tc_{l_1,l_2}	Coût du trajet du site l_1 au site l_2, $\forall (l_1, l_2) \in (0, \ldots, o+1)^2$
d_l	Date de livraison souhaitée pour la commande l, $\forall l \in L$
π_l	Pénalité par unité de temps de retard de livraison la commande l, $\forall l \in L$
Variables	
y_{l_1,l_2}	1 si les tâches associées à la commande l_1 sont réalisées juste avant celles de la commande l_2, 0 sinon, $\forall (l_1, l_2) \in \{0, \ldots, o+1\}^2$
Z_k	1 si le véhicule k est utilisé, 0 sinon, $\forall k \in K$
$z_{l,k}$	1 si la commande l est chargée dans le véhicule k, 0 sinon $\forall l \in L, \forall k \in K$
x_{k,l_1,l_2}	1 si les sites clients l_1 et l_2 sont visités par le véhicule k et si l_2 est visité juste après l_1, 0 sinon, $\forall k \in K, \forall (l_1, l_2) \in \{0, \ldots, o+1\}^2$
$C_{i,l}$	Date de fin de préparation de la tâche associé à la commande l sur la machine i, $\forall l \in L$, $\forall i \in M$
F_k	Date de départ du véhicule k $\forall k \in K$
f_l	Date de départ de la commande l, $\forall l \in L$
D_l	Date de livraison de la commande l, $\forall l \in L$
T_l	Retard de la commande l, $\forall l \in L$
Expressions des coûts	
IC	Coût d'inventaire
VC	Coût fixe des tournées
RC	Coût variable des tournées
PC	Coût de pénalité du au client

HV désigne une valeur arbitrairement grande.

Les valeurs des différents coûts sont calculées par les expressions (3.1), (3.2), (3.3) et (3.4)

$$IC = \sum_{l \in L} (C_{1,l} - p_{1,l}) h_l^{START} + \sum_{l \in L} \sum_{i \in M \backslash m} (C_{i+1,l} - p_{i+1,l} - C_{i,l}) h_{i,l}^{WIP} + \sum_{l \in L} (f_l - C_{m,l}) h_l^{FIN} \tag{3.1}$$

$$VC = \sum_{k \in K} Z_k c^V \tag{3.2}$$

$$RC = \sum_{k \in K} \sum_{l_1 \in 0 \cup L} \sum_{l_2 \in L \cup o+1} tc_{l_1,l_2} x_{k,l_1,l_2} \tag{3.3}$$

$$PC = \sum_{l \in L} \pi_l T_l \tag{3.4}$$

Le modèle mathématique de ce problème appelé Modèle 3.1 est le suivant :

$$\text{Minimise } IC + VC + RC + PC \tag{3.5}$$

$$(3.1), (3.2), (3.3), (3.4) \tag{3.6}$$

$$\sum_{l_2=1}^{o+1} y_{l_1,l_2} = 1 \qquad \forall l_1 \in \{0, \ldots, o\} \tag{3.7}$$

$$\sum_{l_1=0}^{o} y_{l_1,l_2} = 1 \qquad \forall l_2 \in \{1, \ldots, o+1\} \tag{3.8}$$

$$p_{1,l} \leq C_{1,l} \qquad \forall i \in M, \forall l \in L \tag{3.9}$$

$$C_{i,l} + p_{i+1,l} \leq C_{i+1,l} \qquad \forall i \in \{1, \ldots, m-1\}, \forall l \in L \tag{3.10}$$

$$C_{i,l_1} + p_{i,l_2} - HV(1 - y_{l_1,l_2}) \leq C_{i,l_2} \qquad \forall l_1, l_2 \in L^2, \forall i \in M \tag{3.11}$$

$$\sum_{k \in K} z_{l,k} = 1 \qquad \forall l \in L \tag{3.12}$$

$$C_{m,l} - HV(1 - z_{l,k}) \leq F_k \qquad \forall l \in L, \forall k \in K \tag{3.13}$$

$$F_k - HV(1 - z_{l,k}) \leq f_l \qquad \forall l \in L, \forall k \in K \tag{3.14}$$

$$\sum_{l \in L} z_{l,k} \leq oZ_k \qquad \forall k \in K \tag{3.15}$$

$$\sum_{l_2 \in L} x_{k,0,l_2} = Z_k \qquad \forall k \in K \tag{3.16}$$

$$\sum_{l_1 \in L} x_{k,l_1,o+1} = Z_k \qquad \forall k \in K \tag{3.17}$$

$$\sum_{l_2=1}^{o+1} x_{k,l_1,l_2} = z_{k,l_1} \qquad \forall k \in K, \forall l_1 \in L \tag{3.18}$$

$$\sum_{l_1=0}^{o} x_{k,l_1,l_2} = z_{k,l_2} \qquad \forall k \in K, \forall l_2 \in L \tag{3.19}$$

$$x_{k,l_1,l_2} = 0 \qquad \forall (l_1, l_2) \in \{0, \ldots, o+1\}^2 \text{ tel que } l_1 \geq l_2 \text{ et } tt_{l_1,l_2} = 0, \forall k \in K \tag{3.20}$$

$$f_l + tt_{0,l} \leq D_l \qquad \forall l \in L \tag{3.21}$$

$$D_{l_2} + tt_{l_1,l_2} - M(1 - x_{k,l_1,l_2}) \leq D_{l_2} \qquad \forall l_1, l_2 \in L^2, \forall k \in K \tag{3.22}$$

$$D_l - d_l \leq T_l \qquad \forall l \in L \tag{3.23}$$

Modèle 3.1: Modèle général

L'expression (3.5) correspond à la fonction objectif du modèle. Les contraintes (3.7) (resp. (3.8)) imposent que les tâches de chaque commande (à l'exception de la commandes fictive d'index $o+1$ (resp. 0)) possèdent un successeur (resp. un prédécesseur) dans la séquence de production. Les contraintes (3.9) fixent la date de fin de la première tâche de chaque commande sur la première machine. Les contraintes (3.10) sont les contraintes de gamme des tâches lors de la production d'une même commande. Les contraintes (3.11) imposent le non chevauchement des tâches de chaque

commande sur une même machine. Les contraintes (3.12) assurent que chaque commande est affectée à exactement un véhicule. Les contraintes (3.13) imposent que le départ d'un véhicule a lieu après que toutes les commandes lui ayant été affectées ne soient terminées. Les contraintes (3.14) lient le départ des commandes à la date de départ du véhicule auquel elles sont affectées. Les contraintes (3.15) assurent qu'un véhicule est bien utilisé si au moins une commande lui est assignée. Les contraintes (3.16) et (3.17) garantissent qu'un véhicule utilisé commence et termine sa tournée au dépôt. Les contraintes (3.18) (resp. (3.19)) imposent que chaque site visité lors d'une tournée k (à l'exception du site d'arrivée d'index $o+1$ (resp. de départ d'index 0) possède un successeur (resp. un prédécesseur) au sein de sa tournée. Les contraintes (3.20) brisent les symétries dans le cas où deux sites auraient les mêmes coordonnées (et donc un temps de transport nul). Les contraintes (3.21) fixent la date de la première livraison d'une tournée. Les contraintes (3.22) calculent pour chaque tournée, la date de livraison des commandes en fonction de l'itinéraire choisi. Ces contraintes permettent également d'éliminer les sous-tours. Les contraintes (3.23) déterminent le retard de chaque commande.

3.3 Résultats préliminaires

La résolution du problème général décrit par le Modèle 3.1 à l'aide d'un solveur commercial est testée.

Les résultats présentés dans cette section sont réalisés sur un ordinateur équipé d'un processeur Intel Core i7-7820HQ et de 16 Go RAM. Les algorithmes sont exécutés sur un seul thread sous un système Windows et sont développés en C++ avec Visual Studio. IBM Ilog Cplex 12.7 est utilisé comme solveur. Le temps de résolution laissé au solveur pour chaque instance vaut 1h.

Sur la base des instances présentées dans le Chapitre 4, les expérimentations montrent que le solveur ne parvient pas à résoudre des instances d'une taille supérieure à 6 commandes.

3.4 Conclusion

La Figure 3.2 rappelle les différentes composantes du problème présenté dans ce chapitre ainsi que les coûts induits par chacune d'elle. Cette figure schématise également les deux approches proposées pour résoudre ce problème dans les chapitres suivants.

La première approche considère que le producteur et le 3PL sont deux agents distincts avec un rapport de dominance du premier sur le second. Le producteur prend une série de décisions avec pour objectif de minimiser les coûts à sa charge (coûts d'inventaire, coûts fixes des véhicules, pénalités de retard). Il fait pour cela des hypothèses sur les décisions que peut prendre le 3PL. Ensuite, le 3PL s'adapte pour minimiser ses propres coûts. Cette approche est détaillée dans le Chapitre 4.

La seconde approche considère que le producteur et le 3PL travaillent ensemble (comme deux services ou département d'une même entreprise par exemple). Ils sont assimilés à un seul et même agent qui minimise l'ensemble des coûts simultanément. Dans cette approche, la mise en lot des commandes dans les véhicules est déjà connue. Elle modélise par exemple un cas où le mode de conditionnement des commandes lors de la livraison varie (environnement réfrigéré) ou un cas où l'affectation des commandes aux véhicules est déjà effectuée pour des raisons logistiques. Cette approche est détaillée dans le Chapitre 5.

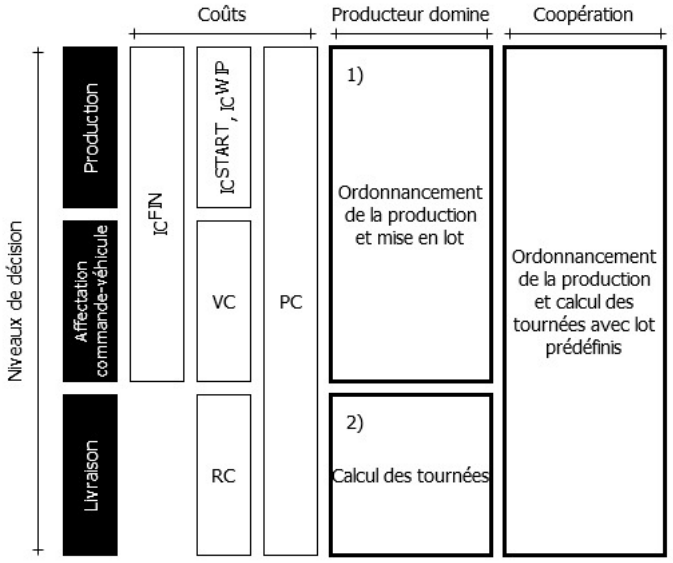

Fig. 3.2: Décomposition du problème et approches

Chapitre 4

Approche où le producteur domine

Contents

Cette partie présente une approche du problème présentée au Chapitre 3 où le producteur et le 3PL sont deux entités distinctes. On considère que le partage d'informations entre les deux agents n'est que partiel et que chacun prends ses décisions indépendemment. Ce chapitre étudie un scénario dans lequel le producteur domine la négociation. Dans ce contexte, le producteur impose au 3PL le nombre et la composition des véhicules ainsi que leurs dates de départ du site de production. Cependant, ne pouvant pas déterminer exactement les temps de livraison, le producteur fait une estimation des dates de livraison des commandes. Tout en s'adaptant aux décisions imposées par le producteur, le 3PL planifie ses livraisons. Les deux agents cherchent à minimiser leurs propres coûts. Le producteur paye ses coûts d'inventaire, un coût fixe pour chaque véhicule (versé au 3PL) et les pénalités dues aux clients. Le 3PL prend en charge les coûts de transport variable.

4.1 Contexte

Comme vu dans les chapitres précédents, la production et la livraison représentent une part importante de la gestion de la chaîne d'approvisionnement. Leur planification efficace permet de réduire les coûts d'une entreprise de façon importante.

Cependant une prise en compte simultanée de ces deux aspects est souvent compliquée car chaque secteur fait appel à des agents différents, aux rôles et compétences diverses. Dans cet environnement, ces agents coopèrent mais veillent à défendre leurs intérêts. Toutes les informations ne peuvent être mises à disposition, ni de tous, ni tout le temps. Ce manque d'information force les agents à anticiper les décisions de leurs partenaires avant de prendre leurs propres décisions. De plus, certains rapports de force peuvent permettre à un agent d'imposer ses décisions aux autres, restreignant leurs possibilités.

Dans ce contexte et dans le cadre du problème présenté au Chapitre 3, nous supposons que le producteur est en position de force et domine la prise de décision. Le producteur a un ensemble de commandes à produire et faire livrer. Pour ce faire, il embauche le 3PL et le rémunère à raison d'un coût fixe par tournée. Dans ce scénario, le producteur impose toutes les décisions liées aux coûts qu'il supporte, à savoir les différents coûts d'inventaire, le coût fixe lié au déploiement d'un véhicule et les pénalités de retard dues aux clients. Ces coûts sont liés à la séquence de production des commandes, à la composition des lots et au temps nécessaire pour livrer chaque client. Ainsi, le producteur, en plus de planifier la séquence de production, décide du nombre de véhicules, de la composition des lots transportés et de leurs dates de départ. Cependant, c'est le 3PL qui décide de l'itinéraire final des tournées. Le producteur ne peut donc pas déterminer les dates de livraisons des commandes dont dépendent les retards. Le producteur estime donc ces dates exactes de livraison en supposant que les commandes d'une même tournée sont livrées par dates de livraison croissantes (EDD). Il en découle une estimation des pénalités de retard à verser au client. L'estimation des dates de livraison (réalisables pour le 3PL) est garantie par contrat. Si, lors de la livraison, le 3PL livre une commandes après la date de livraison estimée par le producteur, il doit reverser des pénalités à ce dernier. À partir de ces informations, le 3PL planifie les itinéraires de ses différentes tournées en cherchant à minimiser les coûts de transport. Une fois les itinéraires des tournées fixés, le producteur paie aux clients les pénalités de retard réelles, qu'elles soient revues à la hausse ou à la baisse par rapport à son estimation initiale. Ainsi le problème à résoudre par le producteur et celui à résoudre par le 3PL sont interconnectés. Chaque agent cherche à minimiser ses propres coûts en s'adaptant au comportement de l'autre et en ne considérant que les informations partagées. La Figure 4.1 illustre les échanges monétaires et flux d'informations entres les agents.

4.2 Adaptation de la modélisation

La modélisation de ce nouveau problème suppose une modification du Modèle 3.1 pour tenir compte des prises de décisions successives du producteur et du 3PL, ainsi que de certains coûts intermédiaires.

Dans ce chapitre, les notations introduites à la Table 3.1 (Chapitre 3) restent valables et seront enrichies. Les Sections 4.2.1 et 4.2.2 décrivent chacunes des sous-problèmes rencontrés par les différents agents. La Section 4.2.3 résume l'approche de résolution du problème du producteur.

4.2.1 Problème de production

Le producteur planifie la production de ses commandes, décide du nombre de tournées et de la composition des véhicules. Enfin, il estime les délais de livraison des commandes en faisant l'hypothèse que le 3PL livre en respectant l'ordre EDD. Les notations déjà introduites au Chapitre

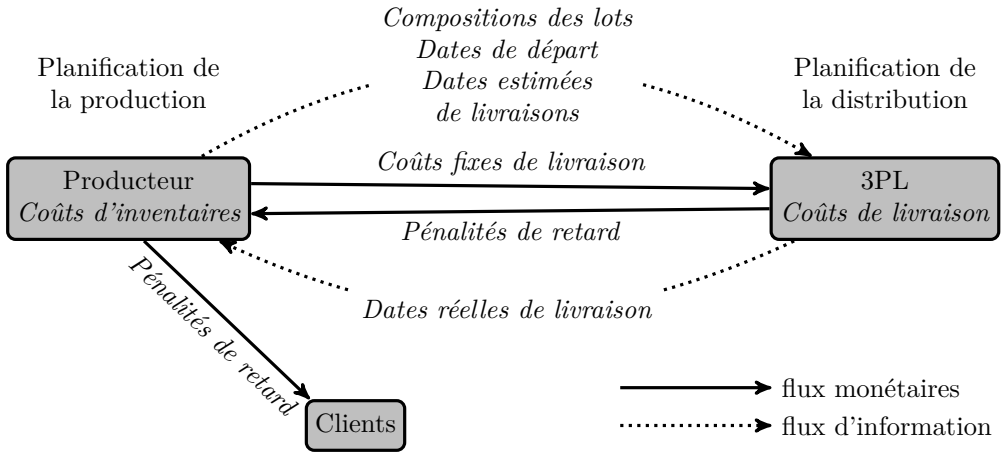

Fig. 4.1: Relation contractuelle entre les agents

3 sont rappelées dans la Table 4.1. Les nouvelles notations nécessaires à la présentation de ce problème sont introduites à la Table 4.2.

Tab. 4.1: Notations introduites au Chapitre 3

Ensembles	
m	Nombre de machines
M	Ensemble des machines ($\{1, \ldots, m\}$)
o	Nombre de commandes
L	Ensemble des commandes ($\{1, \ldots, o\}$)
K	Ensemble des véhicules
Paramètres	
$p_{i,l}$	Temps de préparation de la commande l sur la machine i, $\forall i \in M, \forall l \in L$
h_l^{START}	Coût de stockage par unité de temps en début de production de la commande l, $\forall l \in L$
$h_{i,l}^{WIP}$	Coût de stockage par unité de temps en cours de traitement de la commande l entre la machine i et la machine $i+1$, $\forall l \in L$, $\forall i \in M \backslash m$
h_l^{FIN}	Coût de stockage par unité de temps en fin de production de la commande l, $\forall l \in L$
c^V	coût fixe d'une tournée
tt_{l_1,l_2}	Temps de trajet du site l_1 au site l_2, $\forall (l_1, l_2) \in (0, \ldots, o+1)^2$
tc_{l_1,l_2}	Coût du trajet du site l_1 au site l_2, $\forall (l_1, l_2) \in (0, \ldots, o+1)^2$
d_l	Date de livraison souhaitée pour la commande l, $\forall l \in L$
π_l	Pénalité par unité de temps de retard de livraison la commande l, $\forall l \in L$
Variables	
y_{l_1,l_2}	1 si les tâches associées à la commande l_1 sont réalisées juste avant celles de la commande l_2, 0 sinon, $\forall (l_1, l_2) \in \{0, \ldots, o+1\}^2$
Z_k	1 si le véhicule k est utilisé, 0 sinon, $\forall k \in K$
$z_{k,l}$	1 si la commande l est chargée dans le véhicule k, 0 sinon $\forall l \in L, \forall k \in K$
$C_{i,l}$	Date de fin de préparation de la tâche associée à la commande l sur la machine i, $\forall l \in L$, $\forall i \in M$
F_k	Date de départ du véhicule k $\forall k \in K$
f_l	Date de départ de la commande l, $\forall l \in L$
Expressions de coût	
IC	Coût d'inventaire

VC	Coût fixe des tournées

HV désigne une valeur arbitrairement grande

Tab. 4.2: Nouvelles notations pour le problème de production

Variables	
D_l^M	date de livraison de la commande l, d'après la séquence de livraison EDD estimée par le producteur, $\forall l \in L$
T_l^M	retard de la commande l, d'après la séquence de livraison EDD estimée par le producteur, $\forall l \in L$
x^M_{k,l_1,l_2}	1 si les clients sites l_1 et l_2 sont visités par le véhicule k et que l_2 est visité juste après l_1 d'après la séquence de livraison EDD estimée par le producteur, 0 sinon, $\forall k \in K, \forall (l_1, l_2) \in \{0, \ldots, o+1\}^2$
Expressions de coût	
PPC^M	estimation des coûts de pénalité des clients livrées en retard à verser par le producteur

L'expression des coûts IC et VC restent identiques à celles proposées dans le Chapitre 3, (3.1) et (3.2). L'expression du coût PPC^M est la suivante :

$$PPC^M = \sum_{l \in l} \pi_l T_l^M \tag{4.1}$$

Le modèle mathématique associé au problème de production est présenté dans le Modèle 4.1.

$$\text{Minimise } IC + VC + PPC^M \tag{4.2}$$

(3.7)−(3.15)

$$\sum_{l_2 \in L} x^M_{k,0,l_2} = Z_k \qquad \forall k \in K \tag{4.3}$$

$$\sum_{l_1 \in L} x^M_{k,l_1,o+1} = Z_k \qquad \forall k \in K \tag{4.4}$$

$$\sum_{l_2 \in \{L/d_{l_1} \leq d_{l_2}\} \cup o+1} x^M_{k,l_1,l_2} = z_{k,l_1} \qquad \forall k \in K, \forall l_1 \in L \tag{4.5}$$

$$\sum_{l_1 \in 0 \cup \{L/d_{l_1} \leq d_{l_2}\}} x^M_{k,l_1,l_2} = z_{k,l_2} \qquad \forall k \in K, \forall l_2 \in L \tag{4.6}$$

$$x^M_{k,l_1,l_2} = 0 \qquad \forall l_1, l_2 \in \{0, \ldots, o+1\}^2 \text{ tel que } l_1 \geq l_2 \text{ et } tt_{l_1,l_2} = 0, \forall k \in K \tag{4.7}$$

$$f_l + tt_{0,l} \leq D_l^M \qquad \forall l \in L \tag{4.8}$$

$$D_{l_1}^M + tt_{l_1,l_2} - M(1 - x^M_{k,l_1,l_2}) \leq D_{l_2}^M \qquad \forall l_1, l_2 \in L^2, \forall k \in K \tag{4.9}$$

$$D_l^M - d_l \leq T_l^M \qquad \forall l \in L \tag{4.10}$$

Modèle 4.1: Modèle producteur

L'expression (4.2) correspond à la fonction objectif du problème. Les contraintes de (3.7) jusqu'à (3.15), qui assurent la modélisation du système de production et l'affectation des commandes aux véhicules, proviennent du Modèle 3.1 et restent valables pour ce modèle-ci. (4.3) et (4.4) sont équivalentes aux contraintes (3.16) et (3.17) en s'adaptant aux nouvelles variables x^M_{k,j_1,j_2}. Les contraintes (4.5) et (4.6) sont équivalentes aux contraintes (3.18) et (3.19) tout en garantissant une livraison respectant la séquence EDD. Les contraintes (4.7), (4.8), (4.9) et (4.10) sont équivalentes aux contraintes (3.20), (3.21), (3.22) et (3.23) et s'appliquent aux nouvelles variables spécifiques introduites pour le problème du producteur.

Suite à la résolution du problème du producteur, la séquence et les dates de production des différentes commandes, ainsi que le nombre et la composition des tournées sont fixés (variables y_{l_1,l_2}, $C_{i,l}$, $z_{k,l}$, Z_k, f_l, F_k, $\forall i \in M, \forall k \in K, \forall (l_1, l_2) \in L^2$) lors de la résolution du problème de distribution de 3PL qui détermine les itinéraires et les dates réelles de livraison des commandes.

4.2.2 Problème de distribution

Sur la base des décisions prises par le producteur (nombre de tournées, composition des véhicules, estimation des heures de livraison D^M_l), le 3PL établit l'itinéraire des tournées. Les dates D^M_l fournies par le producteur sont maintenant de nouvelles dates de livraison à respecter par le 3PL. Nous considérons que ces dates sont transmises telles quelles même si $D^M_l < d_l$. En effet, de par sa position dominante, le producteur n'est pas tenu d'être transparent avec le 3PL. En cas de retard, le 3PL devra payer des pénalités au producteur.

Tab. 4.3: Notations introduites au Chapitre 3

Variables	
D_l	date de livraison de la commande l, $\forall l \in L$

De plus, chaque problème de livraison associé à un lot k peut être résolu indépendemment.

Tab. 4.4: Nouvelles notations pour le problème de livraison associé au lot k

Ensembles	
K^M	Ensemble des véhicules prévu par le producteur.
k	Index du lot considéré dans ce problème, $k \in K^M$.
L_k	Ensemble des commandes assignées au véhicule k.
Paramètres	
D^M_l	Les dates de livraison fixées par le producteur remplace dans ce problème les dates d_l initiales, $\forall l \in L_k$.
F_k	Date de départ du véhicule k.
π^{3PL}_l	Montant des pénalités unitaires versées par le 3PL au producteur en cas de retard $\forall l \in L_k$.
Variables	
x_{k,l_1,l_2}	Ces variables correspondent à celles représentant l'itinéraire des tournées dans le Chapitre 3. Il est possible de ne conserver que les variables désignant un couple de commandes l_1 et l_2 transporté dans le même véhicule k, $(l_1, l_2) \in L_k$.
T^{3PL}_l	Retard de livraison du 3PL par rapport aux dates D^M_l fixées par le producteur, $\forall l \in L_k$.
Expressions des coûts	
PC^{3PL}_k	Pénalités payées par le 3PL au producteur pour le retard des commandes du lot k.
RC_k	Coût de transport du lot k payé par le 3PL

Les expressions des coûts PC^{3PL}_k et RC^{3PL}_k sont les suivantes :

$$PC_k^{3PL} = \sum_{l \in L_k} \pi_l^{3PL} T_l^{3PL}$$
$$RC_k = \sum_{l_1 \in 0 \cup L_k} \sum_{l_2 \in L_k \cup o+1} tc_{l_1,l_2} x_{k,l_1,l_2}$$

Ainsi le modèle mathématique associé au problème de livraison du lot k est présenté par le Modèle 4.2.

$$\text{Minimise } RC_k + PC_k^{3PL} \tag{4.11}$$
$$\sum_{l_2 \in L_k \cup o+1} x_{k,l_1,l_2} = 1 \qquad \forall l_1 \in L_k \cup 0 \tag{4.12}$$
$$\sum_{l_1 \in 0 \cup L_k} x_{k,l_1,l_2} = 1 \qquad \forall l_2 \in L_k \cup o+1 \tag{4.13}$$
$$x_{k,l_1,l_2} = 0 \qquad \forall (l_1,l_2) \in (L_k \cup \{0, o+1\})^2 \text{ tel que } l_1 \geq l_2 \text{ et } tt_{l_1,l_2} = 0 \tag{4.14}$$
$$F_k + tt_{0,l} \leq D_l \qquad \forall l \in L_k, \tag{4.15}$$
$$D_{l_2} + tt_{l_1,l_2} - M(1 - x_{k,l_1,l_2}) \leq D_{l_2} \qquad \forall l_1, l_2 \in L_k^2 \tag{4.16}$$
$$D_l - D_l^M \leq T_l^{3PL} \qquad \forall l \in L_k \tag{4.17}$$

Modèle 4.2: Modèle de livraison du lot k

L'expression (4.11) correspond à la fonction objective de ce modèle. Les contraintes (4.12), (4.13), (4.14), (4.15), (4.16) et (4.17) sont équivalent aux contraintes (3.18), (3.19), (3.20), (3.21), (3.22) et (3.23).

Chaque problème de livraison étant indépendant, les coûts de pénalités et de transport de chaque tournée doivent être sommés pour obtenir les coûts totaux. On peut également noter que les variables D_l sont les dates de livraison réelles des commandes aux clients.

4.2.3 Synchronisation et résumé

À la suite de la résolution des problèmes de livraison du 3PL associés aux différents lots, le producteur peut estimer le retard réel de la livraison des commandes chez les clients ($T_l = \max(0, D_l - d_l)$). Il calcule donc les pénalités à verser aux clients ($PC = \sum_{l \in L} \pi_l T_l$) qui remplacent l'ancienne estimation PPC^M. Pour rappel, cette estimation découlait d'une approximation du producteur qui supposait que les commandes d'une même tournée étaient livrées par ordre EDD.

En résumé, l'approche suivante est proposée pour résoudre le Modèle 3.1. Le producteur résout d'abord le problème décrit par le Modèle 4.1. Il détermine la séquence et les dates de production, le nombre et la composition des véhicules ainsi que leurs dates de départ. Par ailleurs, il impose de nouvelles dates de livraison au 3PL, également soumises à des pénalités de retard. La solution du producteur génère un coût d'inventaire IC et un coût de location des véhicules VC qui sera reversé au livreur. Basé sur ces informations, le 3PL résout pour chaque lot le problème décrit par le Modèle 4.2. Il détermine ainsi l'itinéraire des tournées et les dates de livraison des commandes. Ces solutions génèrent les coûts de transport RC_k et les coûts PC^{3PL} qui représente les pénalités

de retard dues au producteur par le livreur. Enfin, à l'aide des itinéraires fournis par le 3PL le producteur peut calculer le montant des pénalités PC qu'il doit aux clients.

L'Algorithme 2 reprend de manière plus formelle cette approche générale.

Algorithme 2 Approche générale

Le producteur optimise sa production :
Résolution de Modèle 4.1 minimisant $IC + PPC^M + VC$
pour k **de** 1 **à** V **faire**
    # Le 3PL optimise l'itinéraire de livraison de chaque lot k :
    Résolution du Modèle 4.2 minimisant $RC_k + PC_k^{3PL}$.
fin pour
Calcul des coûts totaux du 3PL : $\sum_{k=1}^{K^M}(RC_k+PC_k^{3PL})-VC$ (VC payé au 3PL par le producteur)
Calcul des coûts totaux du producteur : $IC + PC^M - \sum_{k=1}^{K^M}(PC_k^{3PL}) + VC$

4.3 Méthodes de résolution

Du fait de sa complexité, une résolution à l'exact du problème du producteur (Modèle 4.1) est très difficile en temps raisonnable. Nous proposons donc des heuristiques pour trouver de bonnes solutions à ce problème. Cependant, cette section ne propose pas de méthode innovante pour résoudre le problème du 3PL. En effet chaque problème de tournée associé à un lot k un problème de type TSP (*Travelling Salesman Problem* ou problème du voyageur de commerce) avec des dates de livraison souples, pouvant être résolues à l'optimalité en utilisant le Modèle 4.2. Cette section se concentre donc exclusivement sur le problème du producteur.

Dans ce contexte, nous décrirons les structures de données utilisées pour représenter une solution ainsi que les méthodes pour évaluer les coûts (coûts d'inventaire, coûts de location des véhicules, estimation des pénalités de retard) d'une solution (Section 4.3.1). Ensuite, nous présenterons deux métaheuristiques pour résoudre ce problème : un algorithme GRASP (Section 4.3.2.4) et un algorithme génétique (Section 4.3.3.1).

4.3.1 Représentation et évaluation d'une solution

4.3.1.1 Représentation d'une solution

Une solution complète du problème est définie par les dates de début d'exécution de chacune des tâches sur chacune des machines, par le nombre de véhicules utilisés et par la composition des lots (un lot est un ensemble de commandes assignées au même véhicule). À partir de ces informations, il est possible de déterminer en temps polynomial les coûts d'inventaire, la date de départ des véhicules, la date de livraison de chaque commande et ainsi de calculer le coût total d'une solution.

Une solution est représentée par la séquence de production des commandes associées à toutes les machines (cas d'un *flow-shop* de permutation). Cette séquence est ensuite subdivisée en lots qui seront chacun affectés à un véhicule.

Exemple : Nous considérons le codage de solution suivant pour un problème de $n = 8$ commandes.

$$\sigma = \{\{l_4, l_2, l_3\}, \{l_1, l_6\}, \{l_8, l_5, l_7\}\}$$

Cette solution correspond à la séquence de production des huit commandes égale à $\{l_4, l_2, l_3, l_1, l_6, l_8, l_5, l_7\}$. Elle est composée de trois lots, le premier composé des commandes l_4, l_2 et l_3, le deuxième des commandes l_1 et l_6 et le troisième des commandes l_8, l_5 et l_7.

Cette représentation d'une solution suppose une hypothèse forte. Les tâches composant les commandes d'un même lot sont réalisées à la suite sur les différentes machines de la chaîne de production. Cette hypothèse réduit l'espace de recherche de solution réalisables et exclu potentiellement la solution optimale. Cependant, ce choix est motivé par deux principaux arguments : cette hypothèse est communément admise dans l'industrie et elle a pour objectif de diminuer les coûts de stockage en fin de production tout en permettant aux véhicules de partir plus tôt.

Ces deux arguments s'illustrent dans l'exemple suivant. Considérons une chaîne de production à 2 machines et 4 commandes, l_1, l_2, l_3 et l_4 avec les temps de production suivant pour les tâches associés, $p_{1,1} = 2$, $p_{2,1} = 2$ pour la commande l_1, $p_{1,2} = 2$, $p_{2,2} = 3$ pour la commande l_2, $p_{1,3} = 2$, $p_{2,3} = 2$ pour la commande l_3 et $p_{1,4} = 1$, $p_{2,4} = 4$ pour la commande l_4. Les commandes l_1 et l_2 sont regroupées dans le lot B_1 et les commandes l_3 et l_4 sont regroupées dans le lot B_2. Les Figures 4.2 et 4.3 représentent un ordonnancement des tâches sur les machines ainsi que les coûts de stockage des commandes en fin de production. La Figure 4.2 représente la séquence de production l_1, l_2, l_3 et l_4 et la Figure 4.3 représente la séquence l_1, l_3, l_2 et l_4.

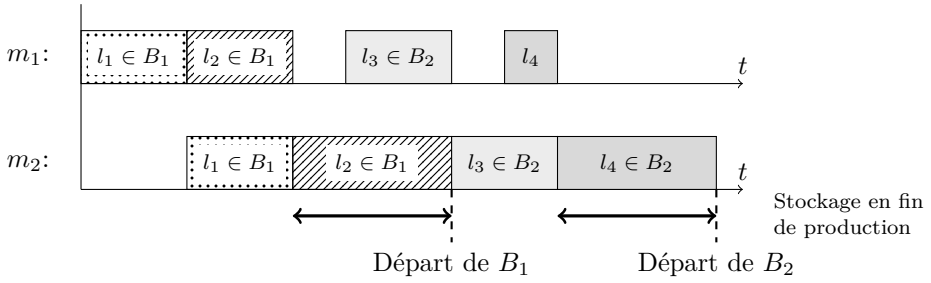

Fig. 4.2: Séquence de production l_1, l_2, l_3 et l_4

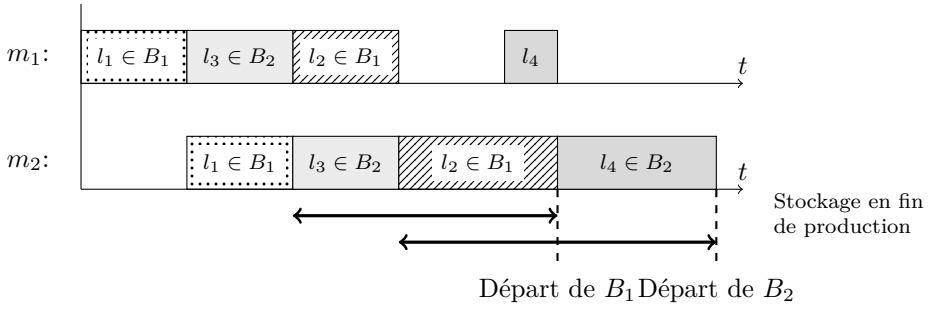

Fig. 4.3: Séquence de production l_1, l_3, l_2 et l_4

On voit que la deuxième séquence (Figure 4.3) augmente les temps de stockage en fin de production et que la date de départ du lot B_1 augmente alors que celle de B_2 reste fixe. Considérons maintenant une instance avec les commandes l'_1, l'_2, l'_3 et l'_4 et les temps de production suivants : $p_{1,1} = 2$, $p_{2,1} = 2$ pour la commande l'_1, $p_{1,2} = 2$, $p_{2,2} = 3$ pour la commande l'_2, $p_{1,3} = 2$, $p_{2,3} = 2$ pour la commande l'_3 et $p_{1,4} = 1$, $p_{2,4} = 4$ pour la commande l'_4. Les commandes l'_1 et l'_2 sont regroupées dans le lot B'_1 et les commandes l'_3 et l'_4 sont regroupées dans le lot B'_2. Nous considérons les Figures 4.4 et 4.5, similaires aux Figures 4.2 et 4.3 précédentes.

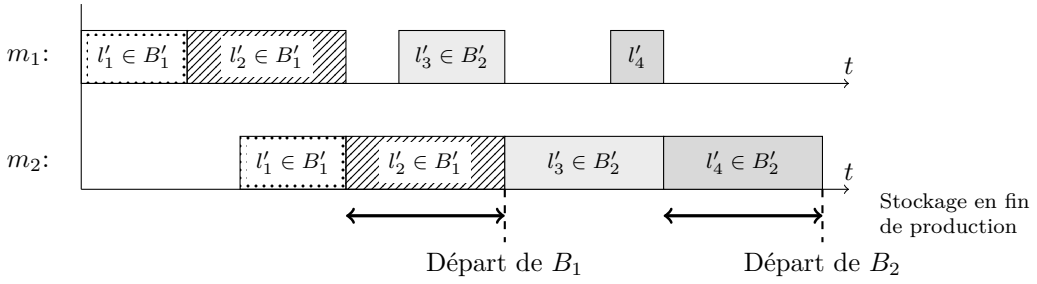

Fig. 4.4: Séquence de production l'_1, l'_2, l'_3 et l'_4

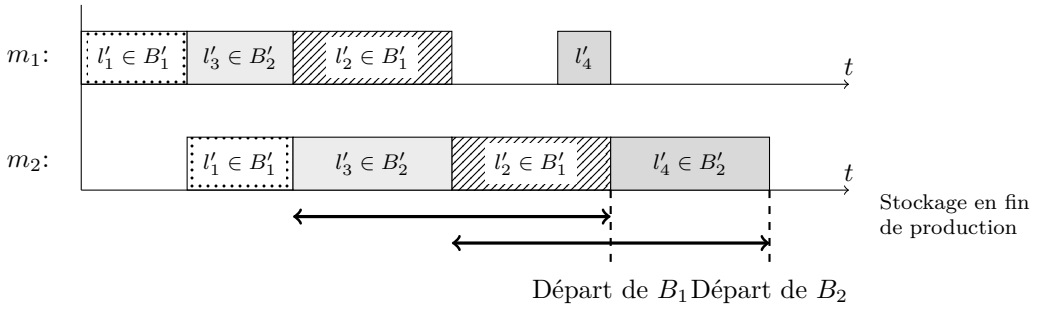

Fig. 4.5: Séquence de production l'_1, l'_3, l'_2 et l'_4

On voit que la deuxième séquence (Figure 4.5) augmente les temps de stockage en fin de production et que la date de départ du lot B'_1 augmente également. Cependant, la date de départ du lot B'_2 est avancée d'une unité de temps, passant de 14 à 13. Dans certains cas, les gains provoqués par l'avancement de cette date de départ peuvent compenser l'augmentation des coûts. Une séquence de production mélangeant des commandes affectées à des lots différents peut également avoir un impact sur les coûts de stockage en cours de production ainsi que sur la date de départ des lots produits par la suite. Nous avons cependant choisi de conserver cette hypothèse de production par lots pour ce chapitre.

4.3.1.2 Méthode d'évaluation exacte d'une solution

Pour une séquence de production et une composition des lots connues, cette méthode résout un modèle mathématique qui minimise les coûts totaux tout en déterminant les dates de productions des commandes sur les différentes machines et fixant les dates de départ des véhicules.

Considérons l'encodage d'une solution σ. Le coût VC est une constante ($VC = c^V|K^M|$) proportionnelle au nombre de véhicules choisis par le producteur. Nous supposons, sans perte de généralité, que les commandes dans σ sont numérotées de 1 à n.

Nous savons quelle est la dernière commande de chaque lot et nous notons E_k l'indice de la dernière commande du lot k, $k \in K^M$. Nous connaissons également le lot assigné à chaque commande et notons K_l le lot assigné à la commande l, $l \in L$.

À partir d'un lot et de la règle EDD, il est possible de déterminer simplement le temps nécessaire pour livrer une commande après le départ de son véhicule. Nous notons TT_l la durée de livraison de la commande l depuis le dépôt.

Un ordonnancement optimal des tâches des différents commandes (incluant leurs dates de production) peut alors être déterminé par le programme linéaire suivant LP^{Fit}.

Les notations utilisées ont déjà été définies à la Section 4.2.1 et les coûts IC et PPC^M sont définis par les expressions (3.1) et (4.1).

$$LP^{Fit}\text{: min } IC + PPC^M \tag{4.18}$$
$$p_{1,1} \leq C_{1,1} \tag{4.19}$$
$$C_{i,l} + p_{i+1,l} \leq C_{i+1,l} \qquad \forall l \in L, \forall i \in M \backslash m \tag{4.20}$$
$$C_{i,l} + p_{i,l+1} \leq C_{i,l+1} \qquad \forall l \in L \backslash o, \forall i \in M \tag{4.21}$$
$$F_k = C_{m,E_k} \qquad \forall k \in K^M \tag{4.22}$$
$$f_l = F_{K_l} \qquad \forall l \in L \tag{4.23}$$
$$D_l^M = f_l + TT_l \qquad \forall l \in L \tag{4.24}$$
$$D_l^M - d_l \leq T_l^M \qquad \forall l \in L \tag{4.25}$$

Modèle 4.3: Modèle de la fonction d'évaluation

Dans ce modèle, VC est pré-déterminé donc la fonction objectif (4.18) est équivalente à la fonction objectif (4.2). L'ensemble des contraintes de ce modèle (de (4.19) à (4.25)) sont adaptées du Modèle 4.2 pour le cas où la séquence de production et la composition des lots sont connues. L'ensemble de contraintes (4.21) permettent le non chevauchement de deux tâches successives sur une machines i car les commandes sont indexées d'après leur ordre dans σ.

4.3.1.3 Méthode d'évaluation approchée d'une solution

Comme précédemment, cette fonction d'évaluation propose des dates de préparation des commandes sur les différentes machines respectant une séquence de production et une composition des lots définie préalablement, réduisant les coûts totaux mais sans garantie d'obtenir les coûts optimaux. Pour l'encodage d'une solution σ connue (la séquence de production et la mise en lot sont connues), cette fonction procède en 2 étapes :

1. En respectant la séquence de production, l'ensemble des dates de production sur chaque machines des commandes est déterminé de manière à ordonnancer chaque tâches au plus tôt. Cette méthode permet d'obtenir les dates de départ des véhicules au plus tôt et ainsi de minimiser les pénalités de retard PC. En effet, l'itinéraire de chaque tournée et la composition des lots étant déjà fixés, PC ne dépend plus que de la date de départ des véhicules.

2. En considérant que ces dates de départ des véhicules calculées en 1 (date de fin de production de la dernière commandes ordonnancer pour chaque lot) sont fixées, l'ensemble des dates de préparations des commandes est déterminé de manière à ordonnancer chaque tâche au plus tard. Cette opération permet de minimiser les coûts d'inventaire IC^{FIN} pour la séquence de production donnée.

À titre d'exemple, les Figures 4.6 et 4.7 illustrent les deux étapes présentées ci-dessus. Dans cet exemple, cinq commandes l_1, l_2, l_3, l_4, et l_5 reparties en deux lots B_1 (commandes à point) et $B2$ (commandes grisées) doivent être ordonnancées. La Figure 4.6 représente le résultat de l'étape 1, la production est planifiée au plus tôt et les dates de départ des lots sont fixées. La Figure 4.7 représente le résultat de l'étape 2, la production sur la première machine des commandes l_3, l_4, et l_5 et la production sur la deuxième machine de la commande l_1 sont avancées d'une unité de temps, réduisant les coûts d'inventaire.

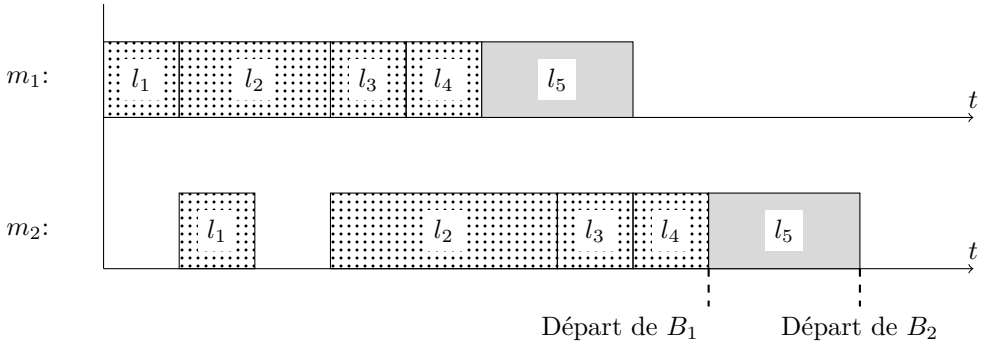

Fig. 4.6: Planification de la production au plus tôt

La Figure 4.7 présente un cas où la méthode d'approximation n'est pas optimale. Par exemple , supposons que la commande l_5 associée au lot B_2 ne soit pas en retard pour sa date de départ. Il est alors possible de décaler le départ de B_2 sans augmenter PC, ce décalage permet de réduire le coût d'inventaire associé aux commandes l_3 et l_4 du lot B_1 (Figure 4.8).

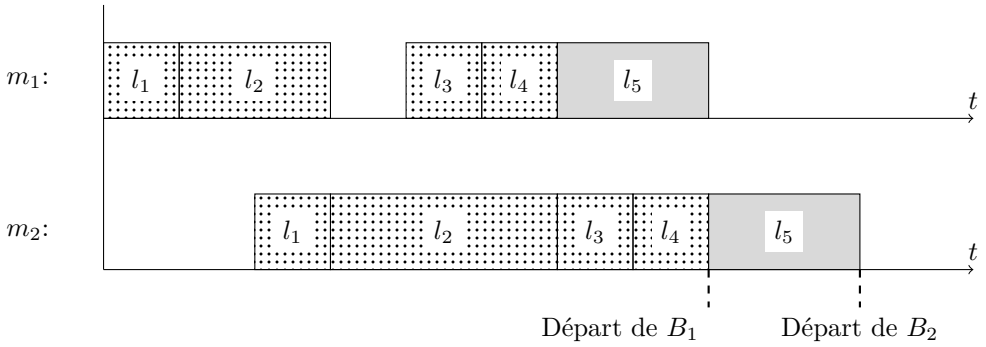

Fig. 4.7: Planification de la production au plus tard avec respect des dates de départ

4.3.2 Algorithme GRASP

Nous proposons d'abord un algorithme GRASP (*Greedy Randomized Adaptive Search Procedure*) pour résoudre ce problème en utilisant ce codage de solution. Cette méthode consiste à générer une série de solutions indépendantes, chacune d'elle construite de manière gloutonne puis améliorée à l'aide d'une recherche locale. Nous allons maintenant présenter la méthode de construction des solutions initiales et la recherche locale utilisées pour améliorer ces solutions. La méthode globale est résumée à la fin de la section.

Le choix de la méthode d'évaluation utilisée par les méthodes de résolution correspond au paramètre τ^{eval} qui vaut "OPT" lors que la méthode exacte (Section 4.3.1.2) est utilisée et "HEU" lorsque la méthode approchée (Section 4.3.1.3) est utilisée.

4.3.2.1 Méthode de génération d'une séquence de production initiale

Pour construire une solution initiale, nous déterminons d'abord la séquence de production des commandes puis la composition des véhicules (ou lots).

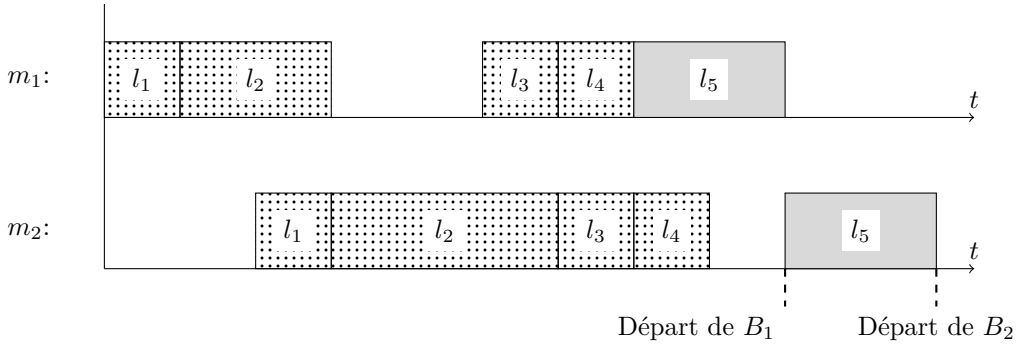

Fig. 4.8: Décalage de la date de départ du dernier lot

La séquence de production est construite itérativement, sur le principe d'une roulette biaisée. Un paramètre λ ($0 < \lambda < 1$) est défini. De ce paramètre découle la taille d'un intervalle basé sur les dates de livraison souhaitées d_l : $[d^{min}, d^{min} + \lambda \times (d^{max} - d^{min})]$, où d^{max} et d^{min} sont respectivement les dates de livraison souhaitées maximum et minimum des commandes restant à planifier. Toute commande ayant sa date de livraison souhaitée dans cet intervalle est incluse dans une *liste restreinte* de candidats et pourra être sélectionnée comme prochaine commande à ordonnancer. Le choix du candidat dans la liste est aléatoire.

Le détail de cet algorithme est donné dans Alg. 3.

Algorithme 3 Heuristique de séquencement

Paramètre : λ
$\sigma = \emptyset$, $\mathcal{L} = L$
tant que $\mathcal{L} \neq \emptyset$ **faire**
    $d^{min} = \min_{l \in \mathcal{L}} d_l$
    $d^{max} = \max_{l \in \mathcal{L}} d_l$
    Liste $= \{l \in \mathcal{L} : d^{min} \leq d_l \leq d^{min} + \lambda \times (d^{max} - d^{min})\}$
    Sélection aléatoire de l dans Liste
    $\sigma \leftarrow \sigma + l$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus l$
fin tant que
renvoyer (σ)

4.3.2.2 Méthode de mise en lot

Après avoir défini la séquence de production des commandes sur les machines, nous proposons une méthode inspirée par l'algorithme Split de Prins [Prins, 2004] pour répartir les commandes au sein des lots. Cet algorithme est présenté pour le *DVRP* (*Distance constraint Vehicule Routing Problem* ou problème de tournées avec contrainte de distance) afin de déterminer un ensemble de tournées optimal respectant un ordre de visite de tous les clients appelé *Giant Tour*. Le principe est de modéliser le problème à l'aide d'un graphe orienté. Chaque tournée réalisable est représentée par une arête du graphe dont le poids correspond aux coût de la tournée. Le plus court chemin dans ce graphe donne l'ensemble des tournées de coût minimum.

Dans notre adaptation de l'algorithme, l'entrée est la séquence σ des commandes calculée par l'heuristique de séquencement (Algorithme 3). Les commandes sont indexées d'après cette séquence. L'ensemble des dates de production des tâches associés aux commandes est prédéterminé en considérant que la production est ordonnancé au plus tôt. Soit un graphe $G = (V, A)$. V contient $n + 1$

sommets chacun associé à une commande, (l_0 étant une commande fictive). A possède un arc pour chaque pair de sommets (l_1, l_2), avec $l_1 < l_2$. Un arc représente le lot contenant la $l_1 + 1^{eme}$ commande jusqu'à la l_2^{eme} commande de la séquence. Le coût associé à cet arc est la somme de deux coûts liés au lot : (1) les pénalités de retard à la livraison estimés par le producteur, (2) le coût fixe d'un véhicule. Pour calculer les pénalités et donc les dates de livraison des commandes, on considère que la date de départ du véhicule est égale à la date de fin de production de la dernière commande du lot.

Nous utilisons ensuite l'algorithme de Bellman pour trouver le plus court chemin au sein de ce graphe. Chaque arête composant ce chemin représente un lot présent dans la solution. Cette évaluation est rapide car le graphe est acyclique (de l'ordre de $O(n^2)$ arêtes).

Cette méthode ne prend cependant pas en compte les coûts d'inventaire, mais minimise de manière optimale, pour une séquence de production σ donnée, la fonction objectif $VC + PPC^M$.

Exemple : Considérons une instance à deux commandes l_1, l_2 et deux machines avec $p_{1,1} = p_{1,2} = p_{2,2} = 1, p_{2,1} = 2$ et $c^V = 5$. Les coûts d'inventaire ne sont pas considérés ici. La séquence de production σ associée est (l_1, l_2). Le graphe G pour cette instance contient 3 sommets et 3 arêtes correspondant aux 3 lots possibles, $\mathcal{B}_1 = \{l_1\}$, $\mathcal{B}_2 = \{l_2\}$ et $\mathcal{B}_3 = \{1, 2\}$. La Figure 4.9 présente les dates de production des commandes pour les différents lots.

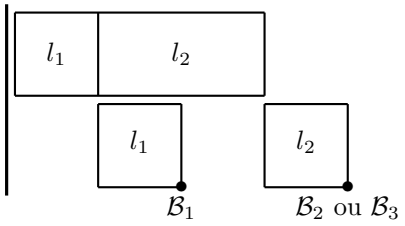

Fig. 4.9: Production et dates de livraison des lots $\mathcal{B}_1$, $\mathcal{B}_2$ et $\mathcal{B}_3$

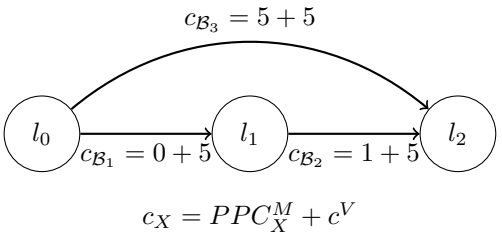

Fig. 4.10: Graphe des coûts de livraison

La livraison du lot $\mathcal{B}_1$ débute à la fin de la production de l_1 (à la date 2). La livraison des lots $\mathcal{B}_2$ et $\mathcal{B}_3$ démarre après la production de l_2 (à la date 4). Dans le cadre de cet exemple, nous supposons que le calcul des pénalités de retard mène aux résultats suivants : $PPC^M_{\mathcal{B}_1} = 0, PPC^M_{\mathcal{B}_2} = 1, PPC^M_{\mathcal{B}_3} = 5$.

En notant c_X le coût associé au lot $\mathcal{B}_X$, nous avons $c_X = PPC^M_X + c^V$. Les différents coûts sont placés sur les arêtes associées du graphe représenté par la Figure 4.10.

En appliquant l'algorithme de Bellman sur ce graphe, nous voyons que le chemin le plus court est égal à $\{(0, l_2)\}$ avec un coût associé de 10. Cela signifie que les commandes l_1 et l_2 sont livrées ensemble. Si le coût fixe des véhicules c^V diminue de 5 à 3, le plus court chemin devient $\{(0, l_1), (l_1, l_2)\}$ pour un coût total associé de 7.

4.3.2.3 Algorithme de recherche locale

Un algorithme de recherche locale est utilisé pour améliorer une solution rapidement. La recherche est soumise à trois paramètres. Le premier, noté τ^{eval} indique le mode d'évaluation de chaque nouvelle solution explorée. Pour $\tau^{eval} = OPT$, la solution est évaluée en utilisant la fonction d'évaluation LP^{Fit}. Pour $\tau^{eval} = HEU$, la solution est évaluée en utilisant la méthode d'évaluation approchée proposée Section 4.3.1.3. Le deuxième paramètre, notée τ^{eff}, décrit le caractère intensif ou non de la recherche locale et le dernier paramètre Δ définit la portée des opérateurs de voisinages utilisés.

L'algorithme utilise quatre types de voisinage (Inv, MOVE, Fusion, Div) et la stratégie "*first improvement*", toute solution évaluée et améliorant la solution courantes devient la nouvelle solution courante.

L'opérateur Inv échange deux commandes dans la séquence de la solution associée aux positions i et j. Pour éviter des évaluations redondantes, (échanger i avec j est équivalent à échanger j avec i) deux commandes ne sont échangées que pour des positions $i < j$. Par exemple :

$$\text{Inv de 2 et 5 :}\{\{1,\underline{2},3,4\}\{\underline{5},6,7\}\} \rightarrow \{\{1,\underline{5},3,4\}\{\underline{2},6,7\}\}$$

L'opérateur Move extrait une commande de sa position i et la réinsère dans la séquence d'une solution à une position j.

$$\text{Move de la position 2 à la position 5 : } \{\{1,\underline{2},3,4\}\{5,6,7\}\} \rightarrow \{\{1,3,4\}\{5,\underline{2},6,7\}\}$$
$$\text{Move de la position 5 à la position 2 : } \{\{1,2,3,4\}\{\underline{5},6,7\}\} \rightarrow \{\{1,\underline{5},2,3,4\}\{6,7\}\}$$

La commande réinsérée appartient au lot de la commande dont elle a pris la place. Cet opérateur peut supprimer un lot si la commande déplacée est la dernière de son lot.

Ces deux voisinages sont quadratiques. Pour éviter une explosion du temps de calcul associé à la recherche locale, ces opérateurs sont limités par le paramètre Δ. Δ réduit la distance entre les deux positions des commandes considérées par Inv et Swap. Une opération associée aux positions i et j n'est acceptée que si $|j - i| \leq \Delta$. Les opérateurs sont testés pour toutes valeurs de i et par valeurs croissantes de i. Pour chaque valeur de i, toutes les valeurs de j respectant le paramètre Δ sont testées par valeurs croissantes.

L'opérateur Fusion fusionne deux lots consécutifs et l'opérateur Div sépare un lot en deux juste après la position ciblée (cet opérateur est le seul à pouvoir augmenter le nombre de lots).

$$\text{Fusion de } \mathcal{B}_1 : \{1,2,3\} \text{ et } \mathcal{B}_2 : \{4,5\}: \quad \{\underline{\{1,2,3\}\{4,5\}}\{6,7\}\} \rightarrow \{\underline{\{1,2,3,4,5\}}\{6,7\}\}$$
$$\text{Div de } \mathcal{B}_1 : \{1,2,3,4,5\} \text{ après la position 3:} \quad \{\{1,2,\underline{3},4,5\}\{6,7\}\} \rightarrow \{\{1,2,3\}\{4,5\}\{6,7\}\}$$

L'opérateur Fusion est appliqué à n'importe quel couple de lots produit à la suite. L'opérateur Div est appliqué à la position de chaque commande n'étant pas la dernière produite de son lot.

Chaque solution générée par un opérateur est évaluée par la fonction d'évaluation sur l'ensemble de la séquence. Ces différents opérateurs sont toujours évalués dans l'ordre suivant : Inv - Move - Fusion - Div. Si un opérateur n'améliore pas la solution courante, l'opérateur suivant est utilisé.

Lors que tous les opérateurs de voisinages ont été utilisés une première fois, la suite de l'algorithme dépend du paramétrage τ^{eff} qui peut prendre l'une des deux valeurs suivante :

- Paramètre "O" : La recherche locale prend fin.
- Paramètre "S" : La recherche locale teste de nouveau l'ensemble des opérateurs et prend prend fin lorsqu'aucun opérateur ne peut améliorer la valeur de la solution courante. Cette solution est alors un minimum local.

Cette méthode est notée LS.

4.3.2.4 GRASP complet

Le principe de cet algorithme est de rapidement générer un grand nombre de solutions de bonne qualité et de renvoyer la meilleure d'entre elles après un délai limite $CPUmax$. Les solutions sont d'abord générées en utilisant les procédures de séquencement et de mise en lot (Alg. 3 avec $\lambda \in [0,1]$.

Elles sont ensuite améliorées par LS jusqu'à ce qu'un minimum local (τ^{eff} = "S") soit atteint (Alg. 4). Le résultat de cet algorithme est la meilleure solution trouvée après $CPUmax$. Cette solution est réévaluée par la fonction LP^{Fit} si τ^{eval} vaut HEU.

Algorithme 4 GRASP

```
1 : Paramètres λ, CPUmax
2 : f(S*) = ∞
3 : tant que CPU ≤ CPUmax faire
4 :     // Création d'une solution initiale
5 :     σ = Heuristique_de_squencement(λ) (Alg. 3)
6 :     S = Heuristique_de_mise_en_lot(σ)
7 :     // Application de la recherche locale
8 :     S = LS(S)
9 :     // Amélioration de la meilleure solution
10 :    si f(S) ≤ f(S*) alors
11 :        S* = S
12 :    fin si
13 :    Mise à jour du temps CPU
14 : fin tant que
15 : renvoyer S*
```

4.3.3 Algorithme génétique (AG)

L'AG [Hollande, 1975] s'inspire de la théorie Darwinienne pour faire évoluer une population de solutions en utilisant des opérateurs tels que la sélection, la reproduction et la mutation. Au fil des générations, une population de solutions évolue et s'améliore pour proposer des solutions de qualité croissante au problème que l'on souhaite résoudre.

Dans l'AG proposé, la population est composée de δ_{pop} solutions. L'encodage d'une solution correspond à la séquence de production des commandes. Pour évaluer cet encodage, l'algorithme de mise en lot (Section 4.3.2.2) et une fonction d'évaluation (Section 4.3.1.3 et 4.3.1.2) sont utilisés.

4.3.3.1 Population initiale

Les solutions de la population initiale sont générées à l'aide de la méthode de génération de séquences de production (Section 4.3.2.1) et de la méthode de mise en lots (Section 4.3.2.2) utilisées par le GRASP. Pour rappel, la première méthode est soumise à un paramètre λ. Des solutions sont générées jusqu'à ce que les δ_{pop} solutions de la population soient uniques (plus de δ_{pop} générations peuvent être nécessaires). Deux solutions sont considérés comme identiques si elles ont la même fonction objectif. En fonction du paramètre τ^{eval}, une solution peut être évaluée de manière exacte ou approchée.

4.3.3.2 Sélection et croisement

Une nouvelle solution (un enfant) est une combinaison de deux solutions (deux parents). Chaque parent est sélectionné au cours d'un tournoi binaire : deux solutions sont choisies aléatoirement dans la population et celle avec la meilleure fonction objectif est conservée. Un couple de parents est composé de deux solutions distinctes et est utilisé pour générer deux enfants.

Pour le premier enfant, les parents se voient attribuer un rôle, Parent 1 et Parent 2, puis un opérateur de croisement est appliqué pour définir l'enfant à partir de ses parents. Nous utilisons

l'opérateur LOX (*Linear Order crossover*, [Lacomme et al., 2001]). Le principe est de conserver la partie centrale de la séquence de production du premier parent (délimitée par deux indices choisis aléatoirement). Les commandes restantes sont décalées en début et fin de production en accord avec la séquence du second parent. Le but est de préserver les parties supposées bonnes chez chacun des parents ainsi que l'ordre relatif des commandes. Un exemple présenté dans le Tableau 4.5 illustre l'opération de croisement.

Parent 1 :	1	2	3	4	5	6	7
Parent 2 :	5	6	4	1	2	3	7
Étape 1 :							
Indices aléatoires : 2 et 5							
Étape 2 :							
Parent 1 :	1	2	|3	4	5|	6	7
Parent 2 :	5	6	4	1	2	3	7
Enfant :	-	-	|3	4	5|	-	-
Étape 3 :							
Parent 2 :	X	6	X	1	2	X	7
Enfant :	6	1	|3	4	5|	2	7

Tab. 4.5: Croisement LOX

Dans cet exemple, chaque solution est une séquence de 7 commandes. Lors de la première étape du croisement, les indices 2 et 5 sont tirés. Ainsi, lors de la second étape, la portion de la séquence solution comprise entre les indices 2 et 5 du Parent 1 est copiée dans la séquence solution de l'Enfant. Cette séquence est complété en étape trois en se basant sur la séquence solution du Parent 2.

Pour définir le second enfant, les rôles Parent 1 et Parent 2 sont échangés et les deux indices choisis aléatoirement lors de la procédure de croisement du premier enfant sont réutilisés. Une fois la séquence de production générée, l'algorithme de mise en lot (Section 4.3.2) est appliqué.

Après le croisement, chaque enfant mute avec une probabilité p_{mut}. Cette mutation consiste à appliquer la recherche locale LS (soumise aux paramètres τ^{eff} et Δ).

4.3.4 Gestion de la population

À chaque génération, $\delta_{pop}/2$ couples de parents sont formés lors de tournois binaires. Le tirage des couples se déroule avec remise, c'est à dire qu'un parent peut appartenir à plusieurs couples. δ_{pop} enfants sont ainsi générés.

Chaque enfant est ajouté à la population seulement si une solution identique n'est pas déjà présente. Pour accélérer les calculs, on considère que deux solutions sont identiques si elles ont la même fonction objectif. Lorsqu'un enfant est ajouté, il prend la place d'un individu parmi la seconde moitié de la population en terme de fonction objectif. Ce mécanisme permet de s'assurer qu'une "bonne" solution ne soit pas remplacée et perdue. Afin de procéder à cette opération facilement, la population est triée.

De nouvelles générations sont générées jusqu'à atteindre la limite de temps. À la suite de quoi, l'algorithme génétique renvoie la meilleure solution de la population.

4.3.5 Paramétrage de l'algorithme génétique

L'AG est soumis a différents paramètres. D'abord les paramètres qui lui sont propres : δ_{pop}, la taille de la population et $p_{mut} \in [0,1]$, la probabilité de mutation d'un enfant. Ensuite le paramètre $\tau^{eval} \in \{HEU, OPT\}$ définit la méthode servant à évaluer une solution. La génération d'une solution initiale dépend du paramètre $\lambda \in [0,1]$ qui intervient lors de la création d'une séquence de production. Enfin, la mutation dépend du paramétrage de la recherche locale LS et donc des paramètres τ^{eff} et Δ.

4.4 Expérimentations et résultats

Les résultats présentés dans cette section sont réalisés sur un ordinateur équipé d'un processeur Intel Core i7-7820HQ et de 16 Go RAM. Le code a été réalisé en C++ sous Visual Studio et compilé pour Windows 10. Le solveur commercial utilisé est Cplex 12.7.1.

4.4.1 Génération des instances

Les résultats présentés dans cette section ont été obtenus sur des instances générées selon les spécifications suivantes. Nous considérons un *flow-shop* de permutation à 5 machines. Nous introduisons un nouveau paramètre $\alpha = 100$. Pour chaque commande l et chaque machine i, le temps de production $p_{i,l}$ est tiré aléatoirement entre 1 et α et la date de livraison d_j aléatoirement entre 1 et $\alpha \times n$. Les différents sites clients sont répartis dans un carré de taille $3\alpha \times 3\alpha$ avec une distance tt_{l_1,l_2} entre les sites l_1 et l_2 correspondant à la distance euclidienne. De plus, nous considérons que les coûts de transport sont égaux à la distance parcourue ($tt_{l_1,l_2} = tc_{l_1,l_2}$).

Le paramétrage des coûts a été étudié pour que la solution optimale d'une instance utilise un nombre de véhicules diffèrent de 1 (toutes les commandes livrées en une fois) ou n (chaque commande est livrée par un véhicule dédié). Nous considérons dans cette partie que le coût d'inventaire h_l^{START} est nul. Les autres coûts d'inventaire d'une commande l sont générés de telle sorte qu'ils augmentent de 1 ou 2 unités d'une machine à l'autre : le coût d'inventaire $h_{1,l}^{WIP}$ est tiré aléatoirement dans $[1,2]$, $h_{i+1,l}^{WIP}$ est tiré aléatoirement dans $[h_{i,l}^{WIP}+1, h_{i,l}^{WIP}+2]$ et le coût d'inventaire final h_l^{FIN} est tiré aléatoirement dans $[h_{m,l}^{WIP}+1, h_{m,l}^{WIP}+2]$. Les pénalités unitaires π_l versées par le producteur aux clients pour chaque unité de retard sont tirées aléatoirement dans $[5,10]$. Le coût fixe de location d'un véhicule c^V vaut 4000.

Des ensembles de 10 instances sont générés pour chaque valeur de n dans l'ensemble $\{6,7,8,9,10,20,30,40,50,60,70,80,90,100\}$. Les instances avec $n \in \{10,20,30,40,50,60,70,80,90,100\}$ sont utilisées pour tester les heuristiques et les instances avec $n \in \{6,7,8,9,10\}$ sont utilisées pour tester la résolution des modèles mathématiques.

4.4.2 Évaluation des heuristiques

Les heuristiques proposées résolvent uniquement la première étape du problème (comme définie en Section 4.2.1) où le producteur planifie la production des commandes et la composition des lots utilisés lors des livraisons. Pour rappel, la fonction objectif est égale à la somme des coûts d'inventaire (IC), des coûts fixes de location des véhicules (VC) et de l'estimation des pénalités de retard (PPC^M). La limite du temps de calcul est fixée à $\lceil \frac{n}{10} \rceil$ minutes par instance.

4.4.2.1 Évaluation du GRASP

Le GRASP présenté Section 4.3.2 est soumis à une série de paramètres : la valeur de λ utilisée pour générer la séquence de production initiale, la taille du voisinage Δ utilisée pendant la recherche locale et la méthode d'évaluation d'une solution utilisée pendant la recherche locale ($\tau^{eval} \in \{OPT, HEU\}$). Le degré d'intensité de la recherche locale τ^{eff} est toujours fixé à "S" dans le GRASP. Toute solution retournée est donc un minimum local.

Des études préliminaires ont montré que la méthode d'évaluation de la solution a un impact déterminant sur les résultats du GRASP. En effet, cette méthode est appelée un très grand nombre de fois pendant la recherche locale et la version approchée ($\tau^{eval} = HEU$) de cette méthode fournie un résultat bien plus rapidement que la version exact ($\tau^{eval} = OPT$).

Ainsi, deux paramétrages de GRASP ont été testés, chacun basé sur une méthode différente d'évaluation des solutions. La méthode de GRASP avec $\tau^{eval} = OPT$ a le paramétrage suivant : $\lambda = 0.1$, $\tau^{eff} = S$ et $\Delta = 5$. Cette méthode se note $GRASP^{OPT}$ (ou G^{OPT} en abrégé). Ce choix de paramétrage limite l'exploration du voisinage de l'ensemble des solutions mais est adapté à une fonction d'évaluation coûteuse en temps. La méthode GRASP avec $\tau^{eval} = HEU$ a les valeurs suivantes de paramètres : $\lambda = 0.2$, $\tau^{eff} = S$ et $\Delta = 20$. Cette méthode se note $GRASP^{HEU}$ (ou G^{HEU} en abrégé). Ce choix de paramétrage permet une plus large exploration du voisinage.

Le Tableau 4.6 indique le nombre de solutions générées par chacune des deux méthodes dans le même temps impartie.

n	Nb. d'itérations	
	G^{HEU}	G^{OPT}
10	12096	168
20	6670	61
30	3572	32
40	3585	18
50	2432	12
60	1757	8
70	1300	6
80	1034	5
90	860	4
100	691	3

Tab. 4.6: Nombre d'itérations des méthodes G^{HEU} et G^{OPT}

Des études supplémentaires ont montré que la durée d'évaluation de la méthode approchée ne dépasse pas 10 microsecondes pour toutes les tailles d'instances alors que la durée d'évaluation de la méthode exacte varie entre 1 et 20 millisecondes pour les différentes tailles d'instances, soit un rapport 1000 entre les deux durées.

Le nombre d'itérations (nombre de solutions générées par chacune des méthodes) diminuent dramatiquement avec la taille des instances. Il est divisé par 18 pour $GRASP^{HEU}$ (de 12096 pour des instances de 10 commandes à 691 pour des instances de 100 commandes) et 56 pour $GRASP^{OPT}$(de 168 pour des instances de 10 commandes à 3 pour des instances de 100 commandes). Ces résultats s'expliquent en partie par l'augmentation du temps d'évaluation d'une solution. De plus, le nombre de solutions à évaluer pour trouver un minimum local augmente également avec la taille des instances. Enfin, le nombre de solutions localement optimales retournées par $GRASP^{OPT}$ pour des instances de taille 60 ne dépasse pas 10. On peut penser que ce nombre de solutions (construites sur une base aléatoire), n'est pas assez élevé pour une méthode de type GRASP.

Le Tableau 4.7 compare les performances des deux méthodes sur la qualités des solutions trouvées. La colonne n indique le nombre de commandes des instances résolues. Les colonnes "Nb. Itération" indiquent le nombre de fois qu'un optimal locales est atteinte pour chaque paramétrage de GRASP. Les colonnes "Amélioration" présentent les pourcentages d'amélioration moyens aux différentes étapes de chaque méthode. Pour $GRASP^{HEU}$, la colonne "A1" représente l'amélioration moyenne entre la solution initiale et celle obtenue grâce à la recherche locale. La colonne "A2" représente l'amélioration moyenne obtenue en appliquant la méthode d'évaluation exacte sur la solution retournée par la recherche locale. Pour $GRASP^{OPT}$, chaque solution étant toujours évaluée de manière exacte, seule l'amélioration due à la recherche locale est présentée. Les colonnes "Nb. Meilleures" indiquent le nombre de fois où la méthode trouve la meilleure solution sur chaque instance (sur une base de 10 instances par ensemble) et les colonnes "Écart" fournissent l'écart moyen entre les solutions trouvées par les méthodes et les meilleurs solutions.

n	Nb.itérations		Amélioration (%)			Nb. meilleures		Écart (%)	
	G^{HEU}	G^{OPT}	G^{HEU} A1	G^{HEU} A2	G^{OPT} A1	$\#G^{HEU}$	$\#G^{OPT}$	G^{HEU}	G^{OPT}
10	12095	168	27.8 %	0.0 %	26.2 %	9	6	0.1 %	0.3 %
20	6670	62	36.0 %	0.1 %	33.0 %	10	4	0.0 %	1.2 %
30	3571	33	40.4 %	0.3 %	34.8 %	9	1	0.1 %	2.1 %
40	3585	18	45.7 %	0.4 %	37.8 %	9	1	0.0 %	2.8 %
50	2433	12	62.3 %	0.5 %	52.5 %	10	0	0.0 %	3.2 %
60	1756	8	78.0 %	0.4 %	64.9 %	8	2	0.1 %	2.3 %
70	1299	6	94.9 %	0.5 %	79.6 %	10	0	0.0 %	2.7 %
80	1033	5	112.3 %	0.5 %	90.3 %	10	0	0.0 %	2.5 %
90	859	4	132.0 %	0.5 %	111.1 %	10	0	0.0 %	3.4 %
100	689	3	158.2 %	0.5 %	136.9 %	10	0	0.0 %	2.9 %

Tab. 4.7: Performance moyenne des méthodes G^{HEU} et G^{OPT}

Comme le montre les colonnes "Amélioration", l'amélioration apportée par la recherche locale augmente avec la taille des solutions pour les méthodes (de 27.6% et 36.2% pour des instances de taille 10 jusqu'à 158.2% et 136.9% pour des instances de taille 100). Cet élément explique en partie la diminution du nombre d'itérations des méthodes. En effet, le potentiel d'amélioration d'une solution initiale pour des instances de grande taille est plus important et nécessite une phase de recherche plus longue. Pour la méthode $GRASP^{HEU}$, la colonne "A2"montre que cette amélioration augmente légèrement avec la taille des instances mais reste marginale (pas plus de 0.5%). Les dernières colonnes ("Nb. meilleures" et "Écart") indique clairement que la solution $GRASP^{HEU}$ propose de meilleures performances pour toutes les tailles d'instance.

Ces résultats montrent qu'utiliser une méthode d'évaluation approchée à plusieurs avantages. D'abord, cette évaluation permet de trouver une fonction objectif pour une solution proche de la fonction objectif optimale, ce que montre la colonne "Amélioration - A2". Ensuite, l'usage de ce mode d'évaluation est beaucoup plus rapide, ce qui permet d'effectuer des recherches avec des voisinages plus large et donc plus performantes (comparaison des colonnes "Amélioration $GRASP^{HEU}$ - A1" et "Amélioration $GRASP^{OPT}$") avec plus d'itérations. En conclusion, les résultats de $GRASP^{HEU}$ sont meilleurs.

4.4.2.2 Évaluation de l'algorithme génétique (AG)

L'AG présenté Section 4.3.3 est soumis à une série de paramètre : pop_{size} est la taille de la population, λ est la valeur utilisée pour générer la séquence initiale de production des premières solutions de la population, p_{mut} est la probabilité de mutation de chaque nouvel individu généré, Δ définit la taille du voisinage utilisé pendant la recherche locale, $\tau^{eff} \in \{O, S\}$ définit si la recherche

locale atteint (“S”) ou non (“S”) un minimum local et le paramètre $\tau^{eval} \in \{HEU, OPT\}$ désigne si chaque solution explorée est évaluée de manière exacte ou approchée.

Des études préliminaires ont montré qu'une taille de population pop_{size} faible (égale à 20) fournissait de meilleurs résultats dans la plupart des configurations. Pour permettre une diversité au sein de la population initiale, le paramètre λ est fixé à 0.2. De même, la probabilité de mutation p_{mut} est fixée à 0.3.

Sur cette base, les différentes combinaisons des paramétrages restant ont été testés, à l'exception de l'association $\tau^{eval} = OPT$ et $\Delta = 20$ qui ne permet pas de réaliser assez d'itération de l'AG dans le temps imparti. Dans la suite de cette section, chaque méthode sera nommée d'après son paramétrage d'après le code suivant : $AG^{\tau^{eval},\tau^{eff},\Delta}$.

Le Tableau 4.8 indique le nombre moyen de générations réalisées pour chaque type de méthodes.

n	$AG^{OPT,S,5}$	$AG^{OPT,O,5}$	$AG^{HEU,S,20}$	$AG^{HEU,S,5}$	$AG^{HEU,O,20}$	$AG^{HEU,O,5}$
10	22	44	920	958	917	965
20	9	21	859	991	924	1056
30	4	12	565	863	746	925
40	2	8	402	657	603	752
50	2	6	224	452	486	615
60	2	5	142	345	413	544
70	1	4	107	280	349	375
80	1	4	80	235	299	325
90	1	3	68	202	264	385
100	1	2	55	166	228	343

Tab. 4.8: Nombre moyen de génération par méthode

De la même manière que pour l'algorithme $GRASP$, ce tableau montre que le paramétrage τ^{eval} est la paramétrage avec l'impact le plus grand sur le nombre d'itérations de l'algorithme. Pour $\tau^{eval} = OPT$, les méthodes associées, quelque soit les autres paramètres, ne génèrent pas plus de 10 itérations pour des instances de taille supérieure à 30. Le second paramètre avec le plus d'impact est τ^{eff} ($\tau^{eff} = 20$ implique moins de génération), suivi du paramètre Δ.

Le Tableau 4.9 se focalise sur l'évaluation du paramétrage τ^{eff}. Pour chaque combinaison de paramétrages $(\tau^{eval}, \Delta) \in \{(OPT, 5), (HEU, 5), (HEU, 20)\}$, chacune des deux méthodes associées (pour $\tau^{eff} = O$ et $\tau^{eff} = S$) est évaluée sur la base de la meilleure solution trouvée par les deux méthodes exécutées à la suite. L'écart moyen d'une méthode à la meilleure solution trouvée par son couple est reporté dans le Tableau 4.9 (en pourcentage). Le paramétrage commun aux deux méthodes est indiqué sur la première ligne du tableau, la valeur de τ^{eff} sur la seconde.

On constate, pour toutes valeurs de (τ^{eval}, Δ) que le paramétrage $\tau^{eff} = O$ propose des solutions de moins bonne qualité que le paramétrage $\tau^{eff} = S$. Toutes les méthodes testées avec le paramétrage $\tau^{eff} = O$ ne trouve pas plus d'une meilleure solution sur 10 instances de taille 50 et plus. Enfin, les méthodes avec le paramétrage $\tau^{eff} = O$ ont été testées avec un délais de résolution plus long (30 minutes pour des instances de 30 commandes). Les résultats montrent que, pour certaines instances, ce délais supplémentaire ne permet toujours pas à ces méthodes de trouver des solutions équivalentes à celles trouvées par les méthodes avec $\tau^{eff} = S$ sans délai supplémentaire. En effet pour chaque nouvel individu généré dans la population, l'algorithme de mise en lot (Section 4.3.2) est appliqué sur la séquence de production et peut introduire une perte d'optimalité (cet algorithme est une heuristique). Corriger cette perte peut demander une modification importante de la solution. Cette modification peut intervenir lors de la mutation, cependant le paramétrage $\tau^{eff} = O$ appliqué à la recherche locale ne permet pas une amélioration suffisante de la solution.

n	$\tau^{eval} = OPT, \Delta = 5$		$\tau^{eval} = HEU, \Delta = 5$		$\tau^{eval} = HEU, \Delta = 20$	
	$\tau^{eff} = O$	$\tau^{eff} = S$	$\tau^{eff} = O$	$\tau^{eff} = S$	$\tau^{eff} = O$	$\tau^{eff} = S$
10	0.0 %	0.0 %	0.4 %	0.0 %	0.1 %	0.1 %
20	0.7 %	0.2 %	0.6 %	0.3 %	0.1 %	0.0 %
30	0.4 %	0.7 %	1.3 %	0.1 %	1.4 %	0.0 %
40	1.5 %	0.3 %	2.3 %	0.1 %	1.0 %	0.1 %
50	4.7 %	0.0 %	3.8 %	0.0 %	2.7 %	0.0 %
60	8.3 %	0.0 %	7.1 %	0.0 %	4.8 %	0.0 %
70	11.2 %	0.2 %	9.6 %	0.0 %	4.8 %	0.0 %
80	18.4 %	0.0 %	11.6 %	0.0 %	6.9 %	0.0 %
90	27.4 %	0.0 %	13.5 %	0.0 %	9.7 %	0.0 %
100	32.3 %	0.0 %	15.6 %	0.0 %	9.2 %	0.0 %

Tab. 4.9: Écart moyen (%) à la meilleure solution trouvée lorsque τ^{eff} varie.

Le Tableau 4.10 compare l'impact de Δ lorsque $\tau^{eval} = HEU$ et $\tau^{eff} = S$. La colonne "Écart" (resp. "Nb. meilleures") indique l'écart à la meilleure solution trouvée (resp. le nombre de meilleures solutions trouvées) pour chacune des deux valeurs de Δ.

	Écart (%)		Nb. meilleures	
n	$\Delta = 20$	$\Delta = 5$	$\Delta = 20$	$\Delta = 5$
10	0.1 %	0.0 %	9	10
20	0.2 %	0.2 %	5	8
30	0.0 %	0.9 %	9	1
40	0.0 %	1.1 %	10	0
50	0.0 %	1.7 %	10	0
60	0.0 %	1.7 %	10	0
70	0.0 %	2.1 %	10	0
80	0.0 %	2.5 %	10	0
90	0.0 %	3.0 %	10	0
100	0.0 %	3.5 %	10	0

Tab. 4.10: Évaluation du paramètre Δ pour $\tau^{eval} = HEU$ et $\tau^{eff} = S$

On remarque que le paramétrage $\Delta = 5$ ne trouve aucune meilleure solution pour des instances de taille supérieure ou égal à 40. Cependant ce paramétrage trouve plus de meilleures solutions pour des instances de petites tailles (10 et 20 commandes). Du fait de la taille de ces instances, un Δ petit permet une exploration exhaustive du voisinage de la solution. De plus, d'après le Tableau 4.8, on sait que $\Delta = 5$ permet de générer 4% et 15% de solutions en plus par rapport à $\Delta = 20$. On remarque cependant que, même pour ces petites instances, l'écart moyen des solutions trouvées par la méthode avec $\Delta = 20$ reste faible (inférieur ou égal à 0.2%).

Les résultats précédents montrent que les valeurs de τ^{eff} et Δ fournissant les meilleurs résultats pour une méthode avec $\tau^{eval} = OPT$ sont $\tau^{eff} = S$ et $\Delta = 5$ et $\tau^{eff} = S$ et $\Delta = 20$ pour une méthode avec $\tau^{eval} = HEU$.

Ainsi, le Tableau 4.11 compare la méthode $AG^{OPT,S,5}$ et la méthode $AG^{HEU,S,20}$. La colonne "Écart" (resp. "Nb. meilleures") indique l'écart à la meilleure solution trouvée (resp. le nombre de meilleurs solution trouvées) pour chacune des deux méthodes.

Ce tableau montre que $AG^{HEU,S,20}$ présente de meilleurs résultats que $AG^{OPT,S,5}$ (ou du moins équivalent pour les petites instances). Ces résultats s'expliquent principalement par le manque

n	Écart (%) $AG^{HEU,S,20}$	Écart (%) $AG^{OPT,S,5}$	Nb. meilleures $AG^{HEU,S,20}$	Nb. meilleures $AG^{OPT,S,5}$
10	0.1 %	0.2 %	9.0	9.0
20	0.3 %	0.3 %	6.0	6.0
30	0.1 %	2.1 %	8.0	2.0
40	0.0 %	2.9 %	10.0	0.0
50	0.0 %	4.6 %	10.0	0.0
60	0.0 %	5.4 %	10.0	0.0
70	0.0 %	7.2 %	10.0	0.0
80	0.0 %	5.6 %	10.0	0.0
90	0.0 %	6.2 %	10.0	0.0
100	0.0 %	4.9 %	10.0	0.0

Tab. 4.11: Comparaison des méthodes $AG^{HEU,S,20}$ et $AG^{OPT,S,5}$

d'itérations effectuées par $AG^{OPT,S,5}$ (pas plus de 10 en moyenne pour des instances de taille 20 et plus). De ce fait, les mécanismes propres aux algorithmes génétiques n'ont pas le temps d'être efficace et de converger vers de bonnes solutions

4.4.3 Comparaison du GRASP et de l'algorithme génétique

Dans cette section, nous comparons le GRASP et l'AG pour leur meilleurs paramétrage. Pour le GRASP, il s'agit de la méthode $GRASP^{HEU}$ ($\tau^{eval} = HEU$, $\lambda = 0.2$, $\tau^{eff} = S$ et $\Delta = 20$). Pour l'AG, il s'agit de la méthode ($pop_{size} = 20$, $p_{mut} = 0.3$, $\lambda = 0.2$, $\tau^{eval} = HEU$, $\tau^{eff} = S$ et $\Delta = 20$).

Le Tableau 4.12 compare les performances des deux méthodes. La colonne "Écart" (resp. "Nb. meilleures") indique l'écart à la meilleure solution trouvée (resp. le nombre de fois où la méthode a trouvé la de meilleure solution) pour chacune des deux méthodes.

n	Écart (%) G^{HEU}	Écart (%) $AG^{HEU,S,20}$	Nb. meilleures G^{HEU}	Nb. meilleures $AG^{HEU,S,20}$
10	0.06 %	0.00 %	9	10
20	0.03 %	0.46 %	9	4
30	0.16 %	0.28 %	6	4
40	0.28 %	0.17 %	1	9
50	0.58 %	0.00 %	0	10
60	1.02 %	0.00 %	0	10
70	0.82 %	0.02 %	2	8
80	1.24 %	0.01 %	1	9
90	0.96 %	0.16 %	2	8
100	0.79 %	0.02 %	1	9

Tab. 4.12: Comparaison des méthodes G^{HEU} et $AG^{HEU,S,20}$

Les deux méthodes sont équivalentes pour des instances de taille 10, les deux instances trouvent les mêmes résultats pour 9 instances sur 10 et l'écart à l'optimal de la solution trouvée est faible (autour de 0.6%). Pour les instances de taille 20 et 30, la méthode G^{HEU} trouve plus de meilleures solutions (9 et 6 contre 4 et 4) avec des écarts moyens plus faible (0.03% et 0.16% contre 0.46% et 0.28%). Cependant, pour les instances de plus grandes tailles, la méthode $AG^{HEU,S,20}$ montre de meilleures performances avec au moins 8 meilleures solutions trouvées sur 10 et des écarts moyens

inférieurs à 0.17%. Pour ces instances, $GRASP^{HEU}$ ne trouve pas plus de 2 meilleures solutions avec un écart moyen entre 0.28% et 1.24%.

D'après les écarts affichés, on remarque que les deux méthodes fournissent des solutions assez proches en termes de fonction objectif pour le même temps de résolution. On peut également remarquer que les paramètres $\tau^{eval} = HEU$, $\tau^{eff} = S$ et $\Delta = 20$ sont identiques pour les deux méthodes, ce qui implique une recherche locale identique. Cependant, on constate que les mécanismes évolutionnaires de $AG^{HEU,S,20}$ lui permettent d'obtenir des solutions de meilleure qualité pour des instances de grande taille. En effet, alors que la méthode $GRASP^{HEU}$ applique une recherche locale sur une séquence de production générée aléatoirement, $AG^{HEU,S,20}$ applique une recherche locale sur une séquence de production issue d'une population de solution améliorée au fil des itérations. De cette différence vient l'avantage à la méthode $AG^{HEU,S,20}$.

4.4.4 Répartition des coûts de la fonction objectif

Cette section discute de la répartition moyenne des coûts de la fonction objectif pour les différentes méthodes sur les différents type d'instances (10 instances par type).

Le Tableau 4.13 indique la répartition moyenne des coûts des différentes méthodes pour des instances de taille 100. La première colonne indique la méthode concernée par les résultats de la ligne. La colonne "Coût total" indique la valeur moyenne des solutions obtenues par la méthode. Les méthodes sont d'ailleurs triées dans ce tableau par ordre décroissant de cette valeur. Les deux ensembles de colonnes suivants, "Répartition en pourcentage" et "Coût par commande", représente pour chaque type de coût sa répartition moyenne en pourcentage et son coût moyen par commande de l'instance. La toute dernière colonne représente le nombre moyen de commandes par véhicule pour chaque instance résolue. Le coût d'un véhicule étant fixe d'une instance à l'autre (4000), les valeurs de cette colonne sont directement liées à la répartition des coûts VC par commande, qui n'est donc par représentée.

Méthode	Coût total	Répartition en pourcentage				Coût par commande			Cmd. par véhicule
		IC^{WIP}	IC^{FIN}	PC	VC	IC^{WIP}	IC^{FIN}	PC	
$AG^{OPT,O,5}$	238381.5	8.2%	16.1%	17.4%	58.4%	193.1	380.4	418.3	2.9
$AG^{HEU,O,5}$	205553.9	9.7%	21.9%	16.2%	52.2%	199.4	446.6	337.6	3.8
$AG^{HEU,O,20}$	187745.3	10.5%	29.2%	15.3%	44.9%	196.6	545.9	291.0	4.8
$AG^{OPT,S,5}$	180456.7	6.0%	21.9%	17.4%	54.8%	106.6	392.6	321.3	4.1
G^{OPT}	178231.0	5.9%	22.3%	16.2%	55.6%	104.7	397.7	291.9	4.1
$AG^{HEU,S,5}$	177806.8	8.5%	25.0%	16.1%	50.5%	150.0	443.0	289.1	4.5
G^{HEU}	173224.6	7.0%	24.6%	15.2%	53.2%	120.6	424.0	267.6	4.4
$AG^{HEU,S,20}$	171894.6	7.4%	24.6%	15.3%	52.7%	126.1	421.6	267.2	4.4

Tab. 4.13: Répartition moyenne des coûts des différentes méthodes pour des instances de taille 100

En fonction de la répartition de leurs coûts, les méthodes avec des comportements similaires ont été regroupées et chaque groupe est séparé par un espace. Les deux méthodes avec les pires coûts ($AG^{OPT,O,5}$ et $AG^{HEU,O,5}$) se caractérisent par des véhicules faiblement remplis (moins de 4 commandes en moyenne) et paradoxalement avec des pénalités de retard de livraison plus élevé que les autres méthodes. Cela indique une séquence de production de mauvaise qualité. En effet, même avec des temps de livraison courts, les commandes ont un retard plus important pour ces méthodes. Soit la production prend du retard du fait d'une séquence de production qui génère des périodes d'inactivité sur les machines, soit les commandes les plus urgentes ne sont

pas ordonnancées assez tôt. Les coûts IC^{WIP} importants par commandes laissent penser que la séquence de production génère de l'inactivité et donc du stockage. Le second groupe de méthodes propose différents compromis entre le nombre de véhicules utilisés et les pénalités par commandes. En se concentrant sur les méthodes $AG^{HEU,O,20}$ et G^{OPT}, on constate qu'elles ont des pénalités de retard identiques alors que $AG^{HEU,O,20}$ utilise plus de véhicules. Ce résultat s'explique sans doute par une planification de la production plus efficace pour la méthode G^{HEU}. Cette hypothèse est appuyée par des coûts IC^{WIP} par commande plus faible pour G^{HEU} que pour $AG^{HEU,O,20}$. Les méthodes du dernier groupe ont une répartition des coûts très similaires. Ces deux méthodes semblent avoir trouver le meilleur compromis entre le nombre de véhicules utilisé et les pénalités de retard.

Pour finir, le coût IC^{FIN} est directement corrélé au nombre de commandes par véhicules. Concernant la répartition des coûts en pourcentage, ces données sont plus difficiles à exploiter car les coûts varient les uns par rapport aux autres. On peut retenir que la part consacrée au coût IC^{WIP} varie entre 5.9% et 10.5%, celle consacrée au coût IC^{FIN} varie entre 16.1% et 29.2%, celle consacrée au coût PC varie entre 15.2% et 17.4% et celle consacrée au coût VC varie entre 44.9% et 58.4%.

Le Tableau 4.13 indique la répartition moyenne des coûts de la méthode $AG^{HEU,S,20}$. pour les différents instances. La première colonne indique le nombre de commandes des instances concernées par les résultats de la ligne. La colonne "Coût total" indique la valeur moyenne des solutions obtenues. Les deux ensembles de colonnes suivants, "Répartition en pourcentage" et "Coût par commande", représentent pour chaque type de coût sa répartition moyenne en pourcentage et son coût moyen par commande de l'instance. La toute dernière colonne représente le nombre moyen de commande spar véhicule pour chaque instance résolue. Le coût d'un véhicule étant fixe d'une instance à l'autre, les valeurs de cette colonne sont directement liées à la répartition des coûts VC par commande.

n	Coût total	Répartition en pourcentage				Coût par commande			Cmd. par véhicule
		IC^{WIP}	IC^{FIN}	PC	VC	IC^{WIP}	IC^{FIN}	PC	
10	36632.2	2.5%	5.9%	44.0%	47.6%	90.4	208.7	1684.1	2.4
20	54540.2	5.3%	10.2%	36.3%	48.2%	142.0	275.0	1030.0	3.2
30	75371.1	5.6%	15.3%	31.2%	48.0%	135.9	358.3	844.8	3.5
40	92475.4	6.4%	20.2%	27.2%	46.2%	143.6	432.4	695.8	3.9
50	100924.5	6.1%	22.1%	23.9%	47.9%	120.2	429.3	525.0	4.3
60	114867.4	6.6%	22.7%	19.6%	51.1%	124.9	423.1	399.8	4.2
70	127879.1	6.9%	24.6%	17.7%	50.7%	126.0	449.4	325.7	4.3
80	141225.6	7.0%	25.5%	15.1%	52.5%	122.6	448.3	269.4	4.3
90	151679.8	7.6%	27.9%	12.3%	52.2%	128.7	464.8	216.3	4.6
100	171894.6	7.4%	24.6%	15.3%	52.7%	126.1	421.6	267.2	4.4

Tab. 4.14: Répartition moyenne des coûts de la méthodes $AG^{HEU,S,20}$ pour pour toutes les tailles d'instance

L'étude des coût par commandes peut s'axer exclusivement sur le fait que les pénalités dues aux clients diminuent lorsque la taille des instances augmentent. Cette diminution se fait en deux temps. Pour les instances de 10 à 40 commandes, PC diminue alors que le nombre de commandes par véhicule augmente et pour les instances de 50 à 100 commandes PC diminue alors que le nombre de commandes par véhicule reste constant. Les deux mouvements observés lors de la première phase sont normalement opposées, augmenter le nombre de commandes par véhicule devrait augmenter les pénalités de retard. La valeur PC continue à diminuer parce que les dates dues de livraison des commandes pour des instances de plus grandes tailles permettent de livrer les commandes

avec moins de retard. Lors de la génération des instances, les dates dues de livraison sont tirées dans une fenêtre de temps qui évolue dynamiquement avec la taille des instances. Cette fenêtre de temps fournit des dates de livraison très strictes pour les instances de petites tailles (d'où le retard important malgré des livraisons rapides pour ces instances), qui deviennent plus souples quand le nombre de commandes augmentent.Il n'est cependant pas possible de savoir pour les instances de grandes tailles si le coût PC est équitablement réparti sur toutes les commandes, ou si les commandes urgentes concentrent l'ensemble des pénalités de l'instance. Les coûts IC^{FIN} dépendent toujours du nombre de véhicules utilisés et les IC^{WIP} reste relativement constant sur l'ensemble des instances.

Concernant la répartition des coûts en pourcentage, on constate que, malgré ces variations réelles, la part du coût VC reste constante (entre 46.2% et 52.7%). La part consacrée au coût PC chute de 44% pour des instances à 10 commandes à 15% pour des instances à 100 commandes. Le comportement de la répartition des coûts IC^{FIN} et IC^{WIP} est principalement dû aux variations importantes des coûts PC et VC.

4.4.5 Comparaison des méthodes sur des instances de petites tailles

Cette section compare les résultats de la résolution du Modèle 4.2 à l'aide du solveur CPLEX et les différentes variantes de GRASP et de l'algorithme génétique sur des instances de petites tailles $n \in \{6, 7, 8, 9, 10\}$. La limite de temps donnée au solveur est une heure, celle donnée au instance est une minutes. Concernant le temps de résolution du solveur, les instances de 5 commandes sont résolues en 1.7 secondes en moyenne et 48 secondes pour les instances de 6 commandes. Seulement 7 instances sur 10 sont résolues avant la limite de temps pour les instances de taille 7 et toutes les instances de taille 8 ne peuvent pas être résolues dans la limite de temps.

Pour les instances de taille 6, 7, 8, 9 et 10, le Tableau 4.15 indique le nombre de meilleures solutions trouvées pour chacune des méthodes.

Méthode	$n = 6$	$n = 7$	n=8	n=9	n=10
Exact	10	10	9	0	2
G^{HEU}	10	10	9	9	8
G^{OPT}	10	10	9	10	8
$AG^{HEU,S,20}$	10	10	10	10	9
$AG^{HEU,S,5}$	10	10	10	10	10
$AG^{HEU,O,20}$	10	10	9	10	9
$AG^{HEU,O,5}$	10	10	9	10	7
$AG^{OPT,S}$	10	10	10	10	10
$AG^{OPT,O}$	10	10	10	10	10

Tab. 4.15: Nombre de solutions optimales trouvées par méthodes

L'écart moyen à l'optimal des différentes heuristiques est en moyen très faible (moins de 0.3%). Pour la méthode exacte, cet écart est nul pour les instances à 8 commandes et moins, égal à 3.3% pour les instances à 9 commandes et 2.1% pour les instances à 10 commandes.

Outre l'efficacité des heuristiques, cette section valide également la représentation d'une solution avec une production par lot (toutes les commandes d'un même lot sont produites à la suite). En effet, toutes les solutions optimales trouvées pour de petites instances valide cette hypothèse de modélisation.

4.5 Conclusion

Ce chapitre propose une approche du problème intégré de production et distribution décrite dans le Chapitre 3. Cette approche considère deux agents, un producteur et un livreur (3PL) qui

se repartissent les décisions et les coûts du problème initial (coûts d'inventaire, de livraison, de retard). Dans le scénario proposé, le producteur domine la négociation avec le 3PL en lui imposant le nombre et la composition des tournées, ainsi que des dates de départ et d'arrivée pour les commandes.

Un modèle mathématique est proposé pour résoudre le problème de décision du producteur et celui du 3PL. Le modèle associé au problème du producteur ne pouvant être résolu en temps raisonnable pour des instance de plus 8 commandes, un GRASP (*Greedy randomized adaptive search procedure*) et un algorithme génétique sont proposés pour résoudre ce modèle. Comme composant de ces méthodes, deux méthodes d'évaluation (exacte et approchée) sont proposées, ainsi qu'une procédure de mise en lot et une procédure de recherche locale.

Les résultats expérimentaux ont montré que l'utilisation d'une méthode d'évaluation approchée pour ce problème permet d'explorer un ensemble de solution plus large tout en guidant la recherche de manière efficace. L'utilisation d'une méthode approchée dans les algorithmes fournit de meilleurs résultats que la méthode d'évaluation exacte. De plus, les résultats montrent que l'algorithme génétique fournit de meilleurs résultats que le GRASP pour des mécanismes de recherche locale similaires.

Chapitre 5

Approche collaborative avec lots fixés

Contents

Ce chapitre présente une seconde approche du problème présenté dans le Chapitre 3. Cette fois-ci, le producteur et le 3PL sont membres de la même entité. Le partage d'informations est complet et toutes les prises de décisions se font conjointement dans le but de minimiser l'ensemble des coûts. Nous sommes dans le cadre d'un problème intégré classique.

Cependant, nous considérons que le nombre et la composition des lots ont déjà été déterminés au préalable par les deux agents. Ce cas de figure arrive lorsque le mode de conditionnement des commandes lors de la livraison varie ou dans un cas où l'affectation des commandes est déjà effectuée pour des raisons logistiques.

5.1 Définition du modèle de collaboration

Ce modèle se base sur celui proposé Modèle 3.1 à l'exception du nombre et de la composition des lots qui sont prédéfinis (variables Z_k et $z_{k,l}$ fixées). Le Tableau 5.1 rappelle les notations présentées dans le Chapitre 3. Le Tableau 5.2 introduit les nouvelles notations nécessaires à la présentation de ce nouveau problème.

Tab. 5.1: Rappel des notations

Ensembles	
m	Nombre de machines
M	Ensemble des machines ($\{1, \ldots, m\}$)
o	Nombre de commandes
L	Ensemble des commandes ($\{1, \ldots, o\}$)
Paramètres	
$p_{i,l}$	Temps de préparation de la commande l sur la machine i, $\forall i \in M, \forall l \in L$
h_l^{START}	Coût de stockage par unité de temps au début de la préparation de la commande l, $\forall l \in L$
$h_{i,l}^{WIP}$	Coût de stockage par unité de temps en cours de préparation de la commande l entre la machine i et la machine $i+1$, $\forall l \in k$, $\forall i \in M \backslash \{m\}$
h_l^{FIN}	Coût de stockage par unité de temps après la préparation de la commande l, $\forall l \in L$
c^V	coût fixe d'une tournée
tt_{l_1,l_2}	Temps de trajet du site l_1 au site l_2, $\forall (l_1, l_2) \in (0, \ldots, o+1)^2$
tc_{l_1,l_2}	Coût du trajet du site l_1 au site l_2, $\forall (l_1, l_2) \in (0, \ldots, o+1)^2$
d_l	Date de livraison souhaitée pour la commande l, $\forall l \in L$
π_l	Pénalité de retard par unité de temps de livraison de la commande l, $\forall l \in L_k$
Variables	
y_{l_1,l_2}	1 si les tâches associées à la commande l_1 sont réalisées juste avant celles de la commande l_2, 0 sinon, $\forall (l_1, l_2) \in \{0, \ldots, o+1\}^2$
$C_{i,l}$	Date de fin de réalisation de la tâche associée à la commande l sur la machine i, $\forall l \in L$, $\forall i \in \{1, \ldots, m-1\}$
F_k	Date de départ du véhicule k $\forall k \in K$
D_l	Date de livraison de la commande l, $\forall l \in L$
T_l	Retard de la commande l, $\forall l \in L$
Expressions de coût	
IC	Coût d'inventaire
RC	Coût variable des tournées
PC	Coût de pénalité du au client

HV désigne une valeur arbitrairement grande.

Tab. 5.2: Nouvelles notations

Ensembles	
K	Ensemble des véhicules requis
L_k	Ensemble des commandes assignées au véhicule k, $\forall k \in K$
Variables	
x_{k,l_1,l_2}	1 si les sites clients l_1 et l_2 sont visités par le véhicule k et si l_2 est visité juste après l_1, 0 sinon, $\forall k \in K, \forall (l_1, l_2) \in \{L_k \cup \{0, o+1\}\}^2$

Par ailleurs, la connaissance de la composition des lots permet d'associer directement la date de départ d'une commande à la date de départ de son lot. Cela permet de ne pas utiliser les variables f_l. L'expression du coût d'inventaire IC est alors réécrite en modifiant le terme concernant les coûts en fin de production.

$$IC = \sum_{l \in L} C_{1,l} h_l^{START} + \sum_{l \in L} \sum_{i \in M/m} (C_{i+1,l} - p_{i+1,l} - C_{i,j}) h_l^{WIP} + \sum_{k \in K} \sum_{l \in L_k} (F_k - C_{m,l}) h_l^{FIN} \quad (5.1)$$

Le modèle mathématique associé à ce problème est présenté par le Modèle 5.1.

$$\text{Minimise } IC + RC + PC \ (5.1), (3.4), (3.3) \tag{5.2}$$

$$\sum_{l_2=1}^{o+1} y_{l_1,l_2} = 1 \qquad \forall l_1 \in \{0,\dots,o\} \tag{5.3}$$

$$\sum_{l_1=0}^{o} y_{l_1,l_2} = 1 \qquad \forall l_2 \in \{1,\dots,o+1\} \tag{5.4}$$

$$p_{1,l} \leq C_{1,l} \qquad \forall i \in M, \forall l \in L \tag{5.5}$$

$$C_{i,l} + p_{i+1,l} \leq C_{i+1,l} \qquad \forall l \in L, \forall i \in \{1,\dots,m-1\} \tag{5.6}$$

$$C_{i,l_1} + p_{i,l_2} - HV(1 - y_{l_1,l_2}) \leq C_{i,l_2} \qquad \forall l_1, l_2 \in L^2, \forall i \in M \tag{5.7}$$

$$C_{m,l} \leq F_k \qquad \forall k \in K, \forall l \in L_k \tag{5.8}$$

$$\sum_{l_2 \in L_k} x_{k,0,l_2} = 1 \qquad \forall k \in K \tag{5.9}$$

$$\sum_{l_1 \in L_k} x_{k,l_1,o+1} = 1 \qquad \forall k \in K \tag{5.10}$$

$$\sum_{l_2 \in \{L_k \cup o+1\}} x_{k,l_1,l_2} = 1 \qquad \forall k \in K, \forall l_1 \in L_k \tag{5.11}$$

$$\sum_{l_1 \in \{0 \cup L_k\}} x_{k,l_1,l_2} = 1 \qquad \forall k \in K, \forall l_2 \in L_k \tag{5.12}$$

$$x_{k,l_1,l_2} = 0 \qquad \forall k \in K, \forall l_1, l_2 \in \{L_k \cup \{0, o+1\}\}^2 \text{ tel que } l_1 \geq l_2 \text{ et } tt_{l_1,l_2} = 0, \tag{5.13}$$

$$F_k + tt_{0,l} \leq D_l \qquad \forall k \in K, \forall l \in L_k, \tag{5.14}$$

$$D_{l_1} + tt_{l_1,l_2} - M(1 - x_{k,l_1,l_2}) \leq D_{l_2} \qquad \forall k \in K, \forall l_1, l_2 \in L_k^2 \tag{5.15}$$

$$D_l - d_l \leq T_l \qquad \forall l \in L \tag{5.16}$$

Modèle 5.1: Modèle de collaboration

La fonction objectif (5.2) est la nouvelle fonction objectif de notre modèle. Le coût VC est maintenant une constante et n'apparaît plus. Les contraintes (5.3), (5.4), (5.5), (5.6), (5.7) et (5.8) correspondent aux contraintes de production et restent inchangées par rapport au Modèle 3.1. Les contraintes (5.9), (5.10), (5.11), (5.12) et (5.13) correspondent aux contraintes de livraison des lots dont la composition est connue. Les contraintes (5.14), (5.15) et (5.16) restent similaires à celles du Modèle 3.1.

5.2 Algorithme glouton

Afin de présenter la valeur ajoutée des méthodes proposées dans cette thèse, un algorithme glouton (GL) a été développé. Cet algorithme construit une solution proche de celle qu'un opérateur humain pourrait obtenir par une résolution "à la main". Cet algorithme considère que les différents

lots sont produits par bloc, i.e. les différentes commandes d'un lot sont produites consécutivement et sans qu'aucune autre commande ne viennent s'intercaler. L'ordre de production des lots suit l'ordre croissant des dates moyennes de livraison souhaitées. La préparation des commandes au sein de ces lots est ordonnancée en utilisant l'heuristique NEH [Nawaz, 1984] et l'ordonnancement est "calé à gauche" afin que chaque tâche soit réalisée le plus tôt possible. Pour finir, les différentes tournées de livraison sont planifiées en suivant la règle du plus proche voisin.

La méthode NEH est résumée dans l'Algorithme 5.

Algorithme 5 Résumé de l'algorithme NEH

1 : Pour chaque commande l, calculer $\sum_{i=1}^{m} p_{i,l}$, la somme des temps de production sur toutes les machines
2 : Créer la liste L_{NEH} avec les commandes triées par $\sum_{l}^{p} p_{i,l}$ décroissant
3 : $S \longleftarrow \{\emptyset\}$
4 : **pour** $l \in [1, \ldots, n]$ **faire**
5 : **pour** $k \in [0, \ldots, |S|]$ **faire**
6 : Évaluer le coût d'insertion de la l^{eme} commande de L_{NEH} à la k^{eme} position dans S
7 : **fin pour**
8 : Insérer la l^{eme} commande de L_{NEH} dans S à la meilleure position
9 : **fin pour**
10 : **renvoyer** S

Considérons un exemple de 5 commandes réparties en 2 lots, produites sur un *flow-shop* de permutation à 2 machines. Nommons chaque commande l_1, l_2, l_3, l_4 et l_5 et chaque lot B_1 composé de l_1 et l_2 et B_2 composé de l_3, l_4 et l_5. Chaque commande est associée à des temps de préparation (un pour chaque machine) et une date de livraison comme définie ci-après suivante : $l_1 : p_{1,1} = 3, p_{2,1} = 1, d_1 = 12$, $l_2 : p_{1,2} = 1, p_{2,2} = 2, d_2 = 10$, $l_3 : p_{1,3} = 1, p_{2,3} = 2, d_3 = 15$, $l_4 : p_{1,4} = 3, p_{2,4} = 2, d_4 = 9$ et $l_5 : p_{1,5} = 3, p_{2,5} = 3, d_5 = 14$. Les coûts d'inventaire n'interviennent pas dans la construction d'une solution avec cette heuristique. Ainsi, nous considérons pour toutes les commandes que $h^{START} = 1$, $h^{WIP} = 2$ et $h^{FIN} = 3$. De même nous considérons que les pénalités unitaires de retard ont le même coût, $\pi = 3$. Ainsi, la moyenne des dates de livraison du lot vaut 11 pour le lot B_1 et 12 pour le lot B_2. Ils seront produits dans cet ordre. L'heuristique NEH fournit la séquence de production $\{l_2, l_1\}$ pour le lot B_1 et $\{l_3, l_5, l_4\}$ pour le lot B_2. Ainsi le lot B_1 est produit à la date 5 et le lot B_2 est produit à la date 14. La Figure 5.1 représente la séquence de production "calée à gauche".

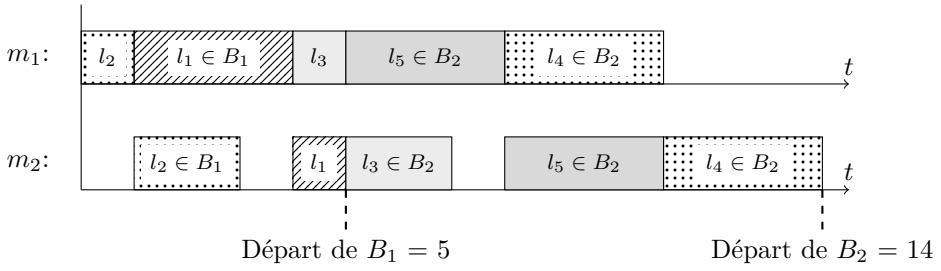

Fig. 5.1: Ordonnancement de la production

Ensuite, les tournées sont planifiées suivant la stratégie du plus proche voisin et les dates de livraison sont calculées. La Figure 5.2 présente les itinéraires empruntés (les coûts de transport sont égaux à la durée du trajet, lesquels sont indiqués sur les arêtes) et les dates de livraison.

Nous pouvons maintenant calculer les coûts associés à cette solution. L'ensemble des coûts

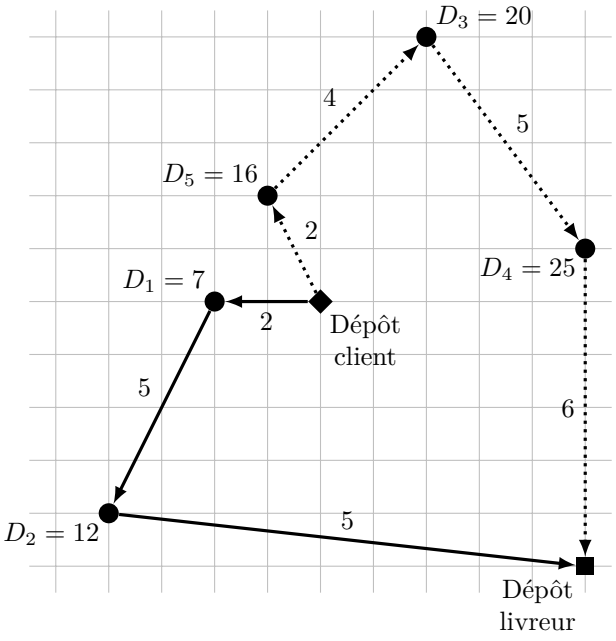

Fig. 5.2: Itinéraire de tournées et dates de livraison

IC^{START} vaut 18, l'ensemble des coûts IC^{WIP} est nul et l'ensemble des coûts IC^{FIN} vaut 36. Le transport coûte 29. Les commandes l_1 et l_5 sont livrées à l'heure, les commandes l_2, l_3 et l_4 ont respectivement 2, 6 et 16 unités de retard pour un PC total de 72. Ainsi le coût total de la solution est 155.

5.3 Méthodes de résolution

Rappelons que la composition de chaque lot est déjà établie et les deux parties du problème doivent être résolues simultanément : l'ordonnancement d'un *flow-shop* de permutation pour la partie production et la planification d'une livraison pour chaque lot pour la partie distribution. Ces deux parties sont liées par les dates de départ des véhicules. Modifier une date de départ peut modifier à la fois les coûts d'inventaire et les coûts de pénalités.

Nous proposons une méthode de résolution en deux phases : une phase de prétraitement et une phase d'exploration des solutions. Dans un premier temps, une tournée minimisant simultanément les coûts de transport et les pénalités de retard des clients est trouvée pour chaque date de départ de la livraison de chaque lot. L'ensemble des coûts minimums des tournées pour toutes les dates de départ possibles permettent d'obtenir un ensemble de fonctions linéaires par morceaux, une pour chaque lot. Cette première étape est réalisée lors de la phase de prétraitement. Cette fonction de coût de livraison (également appelée "fonction linéaire par morceaux de coût de transport et de pénalité" ou fonction DC) est détaillée en Section 5.3.1.1 et différentes méthodes pour la déterminer (exactes et heuristiques) sont proposées Section 5.3.1.2.

La Section 5.3.2.1 présente la seconde étape de la méthode de résolution. Tout d'abord une nouvelle modélisation mathématique du problème incluant les fonctions DC est proposée. Cette inclusion réduit la taille et la complexité du modèle original (Modèle 5.1). Ainsi deux nouveaux modèles sont proposés pour résoudre de manière exacte le problème général. Un modèle "classique" de programmation linéraire (Modèle 5.3) et un modèle de programmation par contraintes (Modèle

5.4).

En supposant que la séquence de production soit fournie, une fonction d'évaluation est obtenue par la résolution d'un programme linéaire dont le modèle est présenté Section 5.3.2.3. Cette fonction d'évaluation permet d'ajuster l'ensemble des dates de réalisation des tâches associées aux commandes, et par conséquent de fixer les dates de départ des véhicules, afin de minimiser conjointement les différents coûts d'inventaire, de transport et pénalités de retard. Cette fonction d'évaluation est utilisée dans une méthode de recherche par voisinage proposée pour déterminer le meilleur ordonnancement possible de la production et donc de résoudre le problème général. Cette dernière méthode est présentée dans la Section 5.3.2.4.

5.3.1 Prétraitement : Fonction linéaire par morceaux de coût de transport et de pénalité (DC)

5.3.1.1 Définition

Dans la suite du document et par soucis de concision, nous nommons l'ensemble des coûts de transport et de pénalités par l'expression "coûts de livraison". Pour chaque lot, nous devons trouver un itinéraire minimisant les coûts de livraison pour chaque date de départ possible afin de réutiliser ces itinéraires lors de la résolution du problème global. Les dates de départ possibles d'un lot sont comprises entre les dates 0 et T^{prod}, avec T^{prod} la plus grande date de départ possible d'une tournée, qui correspond à la plus grande date de fin de production des commandes de tous les ordonnancements possibles. À noter que lorsque la date de départ augmente pour une tournée donnée, les coûts de transport de cette tournée restent identiques alors que les pénalités de retard augmentent linéairement avec la date de départ. Le coefficient de cette augmentation linéaire correspond à la somme des coûts de pénalités unitaires de chaque commande livrée en retard par la tournée. Les coefficients augmentent lorsqu'un nouveau client est livré en retard, la fonction associée aux coûts de livraison d'une tournée est donc croissante et linéaire par morceaux. Le minimum entre deux fonctions des coûts de livraison provenant de deux tournées différentes mène également à une fonction linéaire par morceaux. La fonction DC_k représente la valeur minimale des coûts de livraison pour chaque date de départ $t \in [0, T^{prod}]$ du lot k. Un point d'inflexion d'une fonction DC_k (à une date de départ t donnée) a lieu dans deux cas : lorsque une nouvelle commande devient en retard ou lorsqu'une nouvelle tournée remplace la tournée courante à la date t car son coût de livraison à cette date est plus petit. La Figure 5.3 présente un exemple de profil de fonction de coût de livraison.

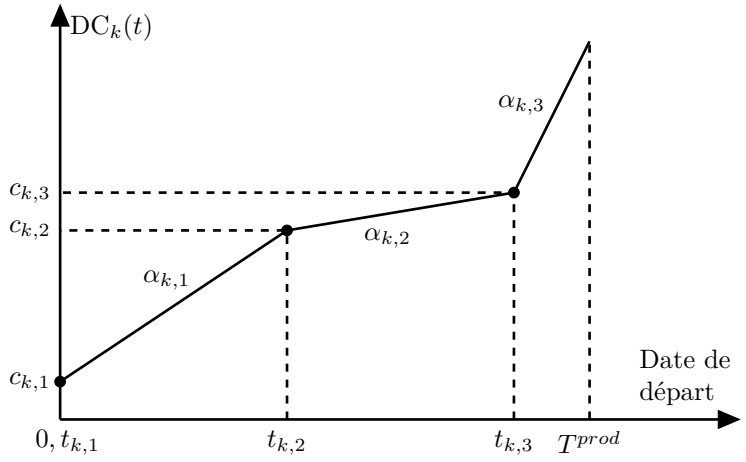

Fig. 5.3: Coûts de livraison du lot k en fonction de sa date de départ

Nous notons S_k le nombre de segments de la fonction DC_k associée au lot k. La première date de départ associée à chacun de ces segments s ($1 \leq s \leq S_k$) est notée $t_{k,s}$ et le coût de livraison pour cette date est noté $c_{k,s}$. Le coefficient directeur de ce segment est noté $\alpha_{k,s}$ et est égal à la somme des pénalités unitaires associées aux commandes en retard.

La date $t_{k,s}$ (origine du nouveau segment s) correspond à un point d'inflexion provoqué par deux causes. Soit la tournée à l'origine du segment précédent $s-1$ a une nouvelle commande en retard, ce qui provoque une augmentation de $\alpha_{k,s-1}$ et donc la création du segment s, soit la tournée à l'origine du segment s a un coût de livraison plus petit en $t_{k,s}$ que la tournée à l'origine du segment $s-1$. Dans ce cas, le coefficient $\alpha_{k,s}$ décroît par rapport au coefficient $\alpha_{k,s-1}$ (sauf rares exceptions où deux courbes sont confondues).

Nous notons $\mathrm{DC}_{k,r}$ la fonction de coût de livraison associée à une unique tournée r qui livre toutes les commandes du lot k. La fonction $\mathrm{DC}_{k,r}$ peut être obtenue en calculant le coût de livraison de r pour chaque date $t \in [0, T]$. Cependant, cette méthode peut être améliorée en identifiant les points d'inflexion de la fonction (dont chacun correspond à une ou plusieurs commandes nouvellement en retard). Dans ce cas $\mathrm{DC}_{k,r}$ est calculé en $O(n \log n)$ avec $n = |L_k|$.

Déterminer DC_k revient à déterminer l'ensemble de tournées noté $\mathcal{R}_k$ tel que :

- pour chaque date t il existe une tournée $r \in \mathcal{R}_k$ tel que le coût $\mathrm{DC}_{k,r}(t)$ soit minimal. ($\mathrm{DC}_k(t) = \min_{r \in \mathcal{R}_k} \mathrm{DC}_{k,r}(t)$)

- pour toute tournée r dans $\mathcal{R}_k$, il existe une date t tel que $\mathrm{DC}_{k,r}(t)$ soit la plus petite valeur parmi toutes les valeurs des autres tournées à cette date. ($\forall r \in \mathcal{R}_k, \exists t \in [0, T^{prod}]$: $\mathrm{DC}_{k,r}(t) = \mathrm{DC}_k(t)$)

Exemple : Soit un lot composé de trois commandes a, b et c. Chaque commande a une date de livraison souhaitée ($d_a = 16,\ d_b = 15, d_c = 22$) et une pénalité unitaire de retard ($\pi_a = 1,\ \pi_b = 3, \pi_c = 33$). Le site de production est noté PS. Le temps de transport est présenté par la Figure 5.4. Nous supposons que le temps et le coût de transport sont égaux. Deux tournées r et r' sont considérées avec $r = (b, a, c)$ et $r' = (a, c, b)$. La Figure 5.5 donne les dates de livraison pour chaque commande sur chaque route en considérant la date de départ $t = 0$.

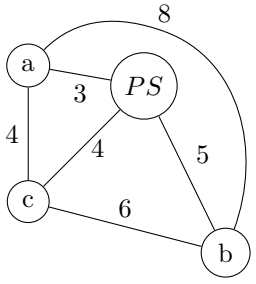

Fig. 5.4: Matrice de distance

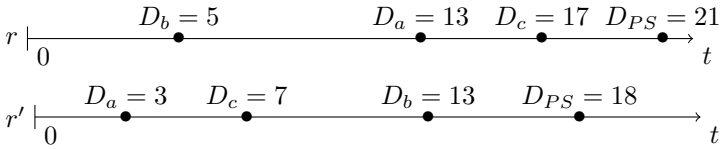

Fig. 5.5: Temps et coût de transport

Les Figures 5.6 et 5.7 représentent les fonctions $\mathrm{DC}_{k,r}$ et $\mathrm{DC}_{k,r'}$ pour $t \in [0, 16]$. Par exemple, pour $t = 8$, les coûts de livraison de la tournée r correspondent à $\mathrm{DC}_{k,r}(8) = 35$ (pénalités de retard pour a et c plus le coût de transport). La Figure 5.8 représente $f(t) = \min(\mathrm{DC}_{k,r}(t); \mathrm{DC}_{k,r'}(t))$. Sur l'intervalle $[0, 3]$ et $[9, 16]$, la tournée r' domine la tournée r et, sur l'intervalle $[3, 9]$, la tournée r domine la tournée r'.

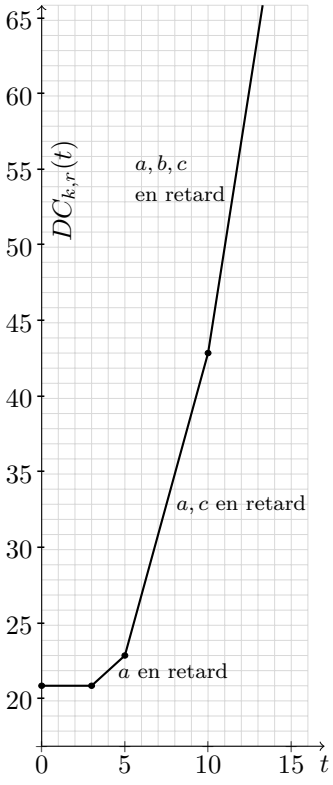

Fig. 5.6: $\mathrm{DC}_{k,r}(t)$

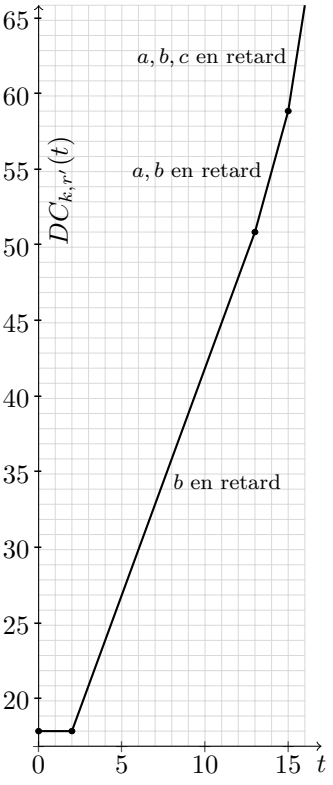

Fig. 5.7: $\mathrm{DC}_{k,r'}(t)$

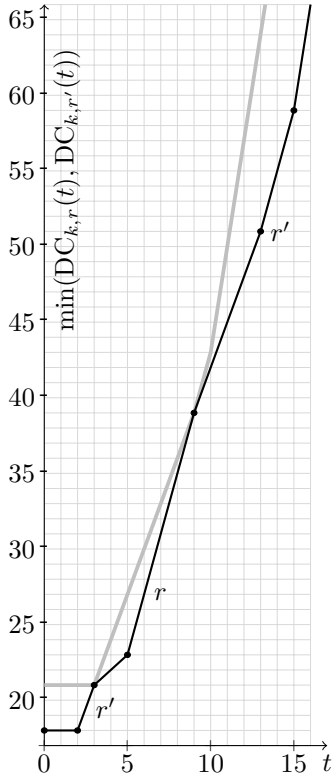

Fig. 5.8: $\min(\mathrm{DC}_{k,r}(t), \mathrm{DC}_{k,r'}(t))$

5.3.1.2 Méthodes de calcul des fonctions DC

Plusieurs méthodes peuvent être utilisées pour construire la fonction DC_k d'un lot. Nous présentons deux méthodes exactes et une heuristique. La première méthode présentée revient à résoudre un modèle mathématique pour chaque lot et chaque date de départ. La second méthode est une heuristique qui sera utilisée comme borne supérieure pour la troisième méthode. Cette dernière méthode, présentée en fin de section, est inspirée de la procédure par séparation et évaluation (PSE ou *Branch-and-Bound*).

Modèle mathématique pour déterminer les fonctions DC_k Afin de déterminer le plus petit coût de livraison pour toutes les dates de départ possibles $t \in [0, T^{prod}]$ du lot k, nous introduisons les coûts de transport RC_k et les pénalités PC_k liés à la livraison du lot k avec :

$$RC_k = \sum_{l_1 \in 0 \cup L_k} \sum_{l_2 \in L_k \cup o+1} tc_{l_1,l_2} x_{k,l_1,l_2}$$
$$PC_k = \sum_{l \in L_k} \pi_l T_l$$

Nous résolvons ensuite le Modèle 5.2 en fixant la date de départ du lot k à t.

$$\text{Minimise } RC_k + PC_k \tag{5.17}$$
$$\sum_{l_2 \in L_k} x_{k,0,l_2} = 1 \tag{5.18}$$
$$\sum_{l_1 \in L_k} x_{k,l_1,o+1} = 1 \tag{5.19}$$
$$\sum_{l_2 \in L_k \cup o+1} x_{k,l_1,l_2} = 1 \qquad \forall l_1 \in L_k \tag{5.20}$$
$$\sum_{l_1 \in 0 \cup L_k} x_{k,l_1,l_2} = 1 \qquad \forall l_2 \in L_k \tag{5.21}$$
$$x_{k,l_1,l_2} = 0 \qquad \forall l_1, l_2 \in \{L_k \cup \{0, o+1\}\}^2 : l_1 \geq l_2, tt_{l_1,l_2} = 0, \tag{5.22}$$
$$t + tt_{0,l} \leq D_l \qquad \forall l \in L_k, \tag{5.23}$$
$$D_{l_1} + tt_{l_1,l_2} - M(1 - x_{k,l_1,l_2}) \leq D_{l_2} \qquad \forall l_1, l_2 \in L_k^2 \tag{5.24}$$
$$D_l - d_l \leq T_l \qquad \forall l \in L \tag{5.25}$$

Modèle 5.2: Modèle de calcul de DC_k pour une date de départ fixée

L'expression (5.17) est la fonction objectif de notre problème. Les contraintes (5.18), (5.19), (5.20), (5.21), (5.22), (5.23), (5.24) et (5.25) sont équivalentes aux contraintes (5.9), (5.10), (5.11), (5.12), (5.13), (5.14), (5.15) et (5.16) du Modèle 5.1 représenté Section 5.1.

Afin de générer la fonction de coût DC_k en entier, ce modèle est résolu pour chaque date de départ $t \in [0, T^{prod}]$.

Heuristique d'évaluation des fonctions DC_k Une heuristique est proposée pour obtenir une bonne approximation des fonctions de coûts avec un temps de calcul court. Cette méthode

peut être utilisée lorsque le nombre de lots ou leurs tailles rendent une résolution à l'exact trop longue. L'Algorithme 6 présente les principales étapes de cette heuristique. Nous notons DC_k^{app} l'approximation de DC_k et $\mathcal{R}_k^{app}$ l'ensemble des tournées impliquées dans le calcul de DC_k^{app}.

Algorithme 6

1 : $\mathcal{R}_k^{app} \leftarrow$ Calcul tournée initiale()
2 : $Tabu \leftarrow \emptyset$
3 : **tant que** $(\mathcal{R}_k^{app} \backslash Tabu) \neq \emptyset$ **faire**
4 : $\quad (t, r) \leftarrow$ Séléction d'une tournée$(\mathcal{R}_k^{app} \backslash Tabu)$
5 : $\quad \{r_{best}, \mathcal{R}_k^{app}\} \leftarrow$ Recherche locale$(t, r, \mathcal{R}_k^{app})$
6 : $\quad Tabu \leftarrow Tabu \cup r_{best}$
7 : **fin tant que**
8 : **renvoyer** $\mathcal{R}_k^{app}$

La première étape de cet algorithme initialise $\mathcal{R}_k^{app}$ avec une seule tournée. Différentes stratégies sont présentées en fin de paragraphe pour la générer.

Ensuite, avant d'améliorer $\mathcal{R}_k^{app}$ en utilisant une recherche locale, une liste tabou est crée. Cette liste tabou contient l'ensemble des tournées ne pouvant plus être améliorées par la recherche locale. À chaque itération de la boucle principale (lignes 3 à 7), une tournée r (n'étant pas dans la liste tabou) est sélectionnée aléatoirement dans $\mathcal{R}_k^{app}$. On considère la plus petite date t telle que $\mathrm{DC}_{k,r}(t) = \mathrm{DC}_k^{app}(t)$ (c'est-à-dire r est la tournée la moins coûteuse dans $\mathcal{R}_k^{app}$ pour livrer le lot k pour un départ du véhicule à la date t). Une recherche locale est utilisée ligne 5 pour améliorer la tournée r afin de minimiser son coût de livraison pour un départ du véhicule à la date t. Cette recherche locale utilise un opérateur de voisinage de type "réinsertion". Toute tournée obtenue en réinsérant une commande de la tournée courante à un autre endroit fait partie du voisinage de la tournée courante. Lors de l'exploration de ce voisinage, si une tournée améliore les coûts de livraison de la tournée courante pour une date de départ en t, cette tournée devient la nouvelle tournée courante (stratégie de "first-improvement"). La recherche locale prend fin lorsque tout le voisinage de la tournée courante est exploré sans qu'aucune de ces tournées n'améliore la solution courante. Un minimum local est atteint. La tournée finale r_{best} est alors ajoutée à la liste tabou (ligne 6).

Lors de l'exploration des voisinages de la recherche locale, n'importe quelle route r' du voisinage de la tournée courante et améliorant $\mathcal{R}_k^{app}$ (i.e. $\exists t \in [0, T^{prod}] : \mathrm{DC}_{k,r'}(t) < \mathrm{DC}_k^{app}(t)$) est ajoutée à $\mathcal{R}_k^{app}$. Après l'ajout d'une tournée dans $\mathcal{R}_k^{app}$, il est possible qu'une autre tournée r'' soit supprimée si $\forall t \in [0, T^{prod}]$ $\mathrm{DC}_k^{app}(t) < \mathrm{DC}_{k,r''}(t)$.

La procédure de la ligne 3 à 7 s'arrête dès que toutes les tournées dans $\mathcal{R}_k^{app}$ sont incluses dans la liste tabou (i.e. toutes ces tournées ont déjà été améliorées par la recherche locale).

La tournée initiale de l'Algorithme 6 est définie par l'une des huit stratégies suivantes :

- NN : La tournée initiale est définie en utilisant la règle du plus proche voisin. [Cover, 1967]
- NN^{Inv} : La tournée initiale est définie en utilisant la séquence inverse de NN.
- $NN_{2,1}$: La tournée initiale est définie en concaténant la seconde moitié de la séquence NN avec sa première moitié.
- $NN_{2,1}^{Inv}$: La tournée initiale est définie en utilisant la séquence inverse $NN_{2,1}$.
- EDD : La tournée initiale est définie en utilisant la règle EDD.
- EDD^{Inv} : La tournée initiale est définie en utilisant la séquence inverse de EDD.

– $EDD_{2,1}$: La tournée initiale est définie en concaténant la seconde moitié de la séquence EDD avec sa première moitié.

– $EDD_{2,1}^{Inv}$: La tournée initiale est définie en utilisant la séquence inverse de $EDD_{2,1}$.

À titre d'exemple, supposons un lot composé des six commandes l_1, l_2, l_3, l_4, l_5 et l_6 et dont la tournée NN associée est $\{l_1, l_2, l_3, l_4, l_5, l_6\}$. La tournée NN^{Inv} associée à ce lot est $\{l_6, l_5, l_4, l_3, l_2, l_1\}$. La tournée $NN_{2,1}$ associée à ce lot est $\{l_4, l_5, l_6, l_1, l_2, l_3\}$. La tournée $NN_{2,1}^{Inv}$ associée à ce lot est $\{l_3, l_2, l_1, l_6, l_5, l_4\}$.

Ainsi, pour générer la fonction de coûts DC_k^{app}, plusieurs méthodes sont considérées. Chaque méthode consiste à exécuter l'Algorithme 6 plusieurs fois avec des tournées différentes à chaque fois. Les différents ensembles de tournées $\mathcal{R}_k^{app}$ obtenus sont ensuite fusionnées pour obtenir l'ensemble final de tournées de la méthode.

La méthode "1-appel" utilise une seule fois Algorithme 6 et l'initialise avec la tournée obtenue par la stratégie NN. La méthode "2-appels" utilise les tournées issues des stratégies NN et EDD. La méthode "8-appels" utilise les tournées issues de l'ensemble des stratégies proposées ci-dessus.

Méthode exacte : procédure par séparation et évaluation (PSE) Dans cette approche par PSE (Section 1.2.2.2), l'objectif est de trouver, pour un lot k, un ensemble de tournées $\mathcal{R}_k$ contenant une tournée avec le coût de livraison minimal pour chaque date de départ t comprise dans l'intervalle $[0, T^{prod}]$. L'ensemble de ces tournées permet de former la fonction $DC_k(t)$ du lot k. L'approche se base sur une recherche arborescente qui explore l'ensemble des tournées possibles et intéressantes en s'appuyant sur un système de coupe pour éviter d'explorer les tournées de mauvaise qualité. Cette exploration consiste à construire au fur et à mesure les tournées (d'abord par le choix de la première commande livrée, puis de la seconde etc...).

Une tournée ne livrant pas toutes les commandes est nommée tournée partielle et est désignée par le symbole σ. L'ensemble des commandes restant à planifier est noté A_σ. Une tournée partielle σ est également associée à un ensemble de dates de départ τ pour lesquelles la tournée σ peut potentiellement améliorée le meilleur coût de livraison connu pour un départ à chacune de ces dates.

À chaque tournée partielle σ et pour chaque date de départ t, une borne inférieure $lb_\sigma(t)$ peut être calculée et correspond à une estimation des coûts minimums de livraison pour une tournée commençant par la séquence σ avec un départ en t. De même une borne supérieure $UB(t)$, correspondant au meilleur coût de livraison trouvé pour la date $t \in [0, T^{prod}]$, est mise à jour lors de la recherche. La borne supérieure UB est initialisée par une fonction DC_k^{app} donnée par l'heuristique présentée précédemment.

Le premier nœud de la recherche (ou nœud racine) est composé de $\sigma = [\emptyset]$ (tournée vide, $A_\sigma = L_k$) et de $\tau = [0, T^{prod}]$. (Aucune décision n'est encore prise au niveau du nœud racine et la borne supérieure $UB(t)$ peut être potentiellement améliorer en tout point). Le principe de construction de l'arbre et le suivant : à partir d'un nœud composé d'une tournée σ et d'un ensemble de dates τ, $|A_\sigma|$ nœuds enfants sont crées et sont composés d'une tournée σ' tel que $\sigma' = [\sigma, l], l \in A_\sigma$. Chacun de ces nœuds correspond à un choix différent quant à la prochaine commande livrée après σ. Pour déterminer τ', l'ensemble de dates de départ associé aux nœuds enfants, la borne inférieure du coût de livraison $lb_{\sigma'}(t)$ est calculée pour σ' et pour chaque $t \in \tau$. τ' vaut alors $\{t \in \tau$ tel que $lb_{\sigma'}(t) < UB(t)\}$. Si τ' est vide, la tournée σ' n'est plus susceptible d'améliorer la meilleure solution trouvée (la borne supérieure UB) pour aucune date de départ. Le nœud est donc coupé (aucune exploration sera mené à partir de ce nœud).

Lorsqu'un nœud contient une tournée σ complète ($A_\sigma = \{\emptyset\}$, toutes les commandes sont livrées), on parle de "nœud feuille". Si l'ensemble de dates de départ τ associé est non vide, alors σ améliore la borne inférieure $UB(t)$ pour tout $t \in \tau$ tel que $DC_{k,t}(t) < UB(t)$. La procédure s'arrête lorsque

tous les nœuds sont soit coupés, soit explorés. La fonction de coûts $DC_k(t)$ recherchée correspond alors à la borne supérieure $UB(t)$.

Calcul de la borne inférieure La borne inférieure $lb_\sigma(t)$ calculée à une date t est composée de trois temps : $lb_\sigma(t) = RPC_\sigma(t) + lb^R_\sigma + lb^P_\sigma(t)$ avec :

- $RPC_\sigma(t)$: le coût de la livraison de la tournée partielle σ.
- lb^R_σ : la borne inférieure des coûts de transport pour livrer toutes les commandes de A_σ.
- $lb^P_\sigma(t)$: la borne inférieure des pénalités de retard dues à la livraison des commandes de A_σ.

lb^R_σ est calculée à partir de l'arbre couvrant de poids minimum d'un graphe noté G^R. Ce graphe complet contient un sommet pour chaque commande dans A_σ, plus un sommet pour la dernière commande livrée par σ et un sommet pour le dépôt. La valeur des arêtes correspond au coût de transport entre deux sites. La valeur de l'arbre de poids minimum de G^R est une borne inférieure des coûts de transport pour livrer toutes les commandes de A_σ. On remarque que lb^R_σ ne dépend pas de la date de départ t.

$lb^P_\sigma(t)$ est calculée à partir d'un graphe noté G^P. Ce graphe complet contient un sommet pour chaque commande dans A_σ, plus un sommet pour la dernière commande livrée par σ. La valeur des arêtes correspond au temps de transport entre deux sites. D'abord, les plus petites dates de livraison possibles sont calculées à partir de l'arbre couvrant de poids minimum de G^P et en s'appuyant sur le Théorème 5.1. Ensuite, le coût minimum des pénalités des retards est calculé en assignant chaque commande à une de ces dates (Théorème 5.2).

Théorème 5.1. *Pour chaque chemin de taille s dans un graphe G, la longueur de ce chemin est supérieure ou égale à la somme des s plus petites arêtes de l'arbre couvrant de poids minimum de G.*

Preuve. *On note T^* l'arbre couvrant de poids minimum de G, $l(T^*)$ sa valeur et $l_k(T^*)$ la somme des k plus petites arêtes de T^*. On note μ un chemin de taille s dans G et A l'ensemble des sommets de μ ($|A| = s+1$). On note T^*_A l'arbre couvrant de poids minimum du sous graphe de G induit par A. La longueur de μ ne peut être plus petite que la somme totale des arêtes de T^*_A, car le chemin μ est un arbre couvrant de A (μ est connecté et sans cycle) : $l(\mu) \geq l(T^*_A)$. De plus, la valeur de T^*_A ne peut être plus petite que $l_s(T^*)$ (T^*_A est un arbre du sous graphe de G induit par A possédant s arêtes) : $l(T^*_A) \geq l_s(T^*)$.* ■

D'après le Théorème 5.1, la i^{eme} plus petite date de livraison possible est toujours plus grande ou égale à la date de la dernière livraison de la tournée σ plus la somme des i plus petites arêtes de l'arbre couvrant de G^P (noté $D^{min}_i(t)$).

Pour calculer la borne inférieure $lb^P_\sigma(t)$, il faut résoudre un problème d'affectation des commandes de A_σ à livrer aux dates de livraisons $D^{min}_i(t)$. La matrice d'affectation $U(t)$ associée à ce problème est de taille $|A_\sigma| \times |A_\sigma|$ avec :

$$u_{l,i}(t) = \max(0, \pi_l(D^{min}_i(t) - d_l)) \quad \forall (l,i)^2 \in \{1, \ldots, |A_\sigma|\}^2$$

Théorème 5.2. *$lb^P_\sigma(t)$ est donné par la solution optimale de ce problème d'affectation.*

Preuve. Dans l'Annexe A

L'algorithme de Kuhn-Munkres [Kuhn, 1955], [Munkres, 1957] peut être utilisé pour résoudre ce problème d'affectation en temps polynomial.

Exemple de calcul de la borne inférieure $lb^P_\sigma(t)$ Dans cet exemple A_σ est composé de 3 commandes l_1, l_2 et l_3 avec : $d_1 = 5$, $\pi_1 = 1$, $d_2 = 9$, $\pi_2 = 2$, $d_3 = 11$, $\pi_3 = 3$. Les temps et les coûts de transport sont supposés égaux. Un arbre couvrant de poids minimum du graphe G^R est présenté Figure 5.9. Le coût de transport minimum lb^R_σ est égal à 7. La date de livraison t de la dernière commande de σ est égale à 8.

On peut alors calculer $D_1^{min} = 8 + 1 = 9$, $D_2^{min} = 8 + 1 + 2 = 11$, $D_3^{min} = 8 + 1 + 2 + 2 = 13$ ainsi que toutes les valeurs $u_{l,i}(t)$ de la matrice $U(t)$ (pour $t = 8$). Par exemple, pour la commande 1 et la second date de départ ($i = 2$), $u_{1,2}(8) = \max\{0, \pi_1(D_2^{min}(8) - d_1)\} = \max\{0, 1(11 - 5)\}$. La matrice $U(8)$ est représentée Figure 5.10. L'algorithme Kuhn-Munfres assigne la commande 1 à la troisième date de livraison, la commande 2 à la première date et la commande 3 à la troisième. Ainsi, la borne inférieure des coûts de livraison $lb^P_\sigma(8)$ vaut $8 + 0 + 0 = 8$.

On suppose que le coût de la livraison de la tournée partielle σ pour $t = 8$ ($RPC_\sigma(8)$) est connu et vaut 10, on peut alors calculer la borne inférieure complète : $lb_\sigma(8) = RPC_\sigma(8) + lb^R_\sigma + lb^P_\sigma(8) = 10 + 7 + 8 = 25$.

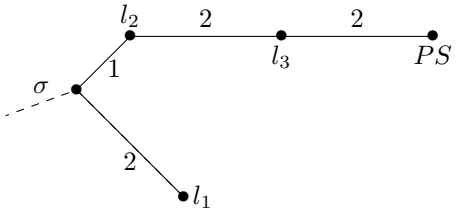

Fig. 5.9: Arbre couvrant de poids minimum du graphe G^R

$u_{j,i}(8)$	$i = 1$	$i = 2$	$i = 3$
Commande l_1	4	6	8
Commande l_2	0	4	8
Commande l_3	0	0	6

Fig. 5.10: Matrice d'affectation $U(8)$

5.3.2 Résolution du problème d'ordonnancement

Une fois ce prétraitement réalisé, les fonctions de coût $\mathrm{DC}_k(t)$ sont connues pour chaque lot k. Ces fonctions permettent d'obtenir les coûts de livraison optimaux associés au lot k et à la date de départ t du véhicule. Pour résoudre la partie ordonnancement du problème général, il faut déterminer la séquence de production des commandes qui peut être évaluée en s'appuyant sur les fonctions de coût de chaque lot.

Dans cette section, nous allons présenter un nouveau modèle mathématique, incluant les fonctions de coût DC afin de résoudre le problème général. Ensuite, une procédure d'évaluation d'une séquence de production sera présentée pour déterminer les dates optimales de début et de fin de préparation des commandes sur chaque machine (et par conséquent, les dates de départ des véhicules) minimisant à la fois les coûts d'inventaire et les coûts de livraison. Enfin, une recherche par voisinage sera proposée pour trouver la meilleure séquence de production possible en utilisant cette procédure d'évaluation.

5.3.2.1 Modèle mathématique incluant les fonctions DC

Le Modèle 5.3, noté DC_{MILP}, est un nouveau modèle mathématique incluant les fonctions DC. Le Tableau 5.3 introduit de nouvelles notations nécessaires à la description de ce problème. Le modèle reprend en plus les notations introduites Section 5.3.1.1.

Tab. 5.3: Nouvelles notations

Ensembles	
Δ_k	Ensemble des segments définissant la fonction de coût DC_k, $\forall k \in K$
S_k	Nombres de segments de Δ_k, $\forall k \in K$
Paramètres	
$t_{k,s}$	Première date associée au segment s du lot k, $\forall k \in K$, $\forall s \in \Delta_k$
$c_{k,s}$	Coût de livraison associé au segment s du lot k pour un départ en $t_{k,s}$ (équivalent à $\mathrm{DC}_k(t_{k,s})$), $\forall k \in K$, $\forall (s, s') \in \Delta_k^2$
$\alpha_{k,s}$	Coefficient directeur du segment s, $\forall k \in K$, $\forall s \in \Delta_k$
Note :	Les segments associés à un lot k sont indexés par dates $t_{k,s}$ croissantes. $(s < s' \iff t_{k,s} < t_{k,s'})$, $\forall k \in K$, $\forall s \in \Delta_k$
T^{prod}	Borne supérieure du temps total de production des commandes
Variables	
$z_{k,s}$	1 si la date de départ F_k du lot k appartient au s^{eme} segment de la fonction DC_k, $\forall k \in K$, $\forall s \in \Delta_k$, 0 sinon
$u_{k,s}$	variables continues positives tel que, $t_{k,s} + u_{k,s} = F_k$ si $z_{k,s} = 1$, sinon $u_{k,s} = 0$, $\forall k \in K$, $\forall s \in \Delta_k$
Expression	
LC	$RC + PC$, représente les coûts de livraison, soit la somme des coûts de transport et des pénalités de retard

L'expression de LC est la suivante :

$$LC = \sum_{k \in K} \sum_{s \in \Delta_k} (z_{k,s} c_{k,s} + u_{k,s} \alpha_{k,s})$$

$$DC_{MILP} : \min\ IC + LC \tag{5.26}$$

$$\sum_{l_2=1}^{o+1} y_{l_1,l_2} = 1 \qquad \forall l_1 \in \{0, \ldots, o\} \tag{5.27}$$

$$\sum_{l_1=0}^{o} y_{l_1,l_2} = 1 \qquad \forall l_2 \in \{1, \ldots, o+1\} \tag{5.28}$$

$$p_{1,l} \leq C_{1,l} \qquad \forall i \in M, \forall l \in L \tag{5.29}$$

$$C_{i,l} + p_{i+1,l} \leq C_{i+1,l} \qquad \forall l \in L, \forall i \in \{1, \ldots, m-1\} \tag{5.30}$$

$$C_{i,l_1} + p_{i,l_2} - HV(1 - y_{l_1,l_2}) \leq C_{i,l_2} \qquad \forall l_1, l_2 \in L^2, \forall i \in M \tag{5.31}$$

$$C_{m,l} \leq F_k \qquad \forall k \in K, \forall l \in L_k \tag{5.32}$$

$$F_k = \sum_{s \in \Delta_k} (z_{k,s} t_{k,s} + u_{k,s}) \qquad \forall k \in K \tag{5.33}$$

$$\sum_{s \in \Delta_k} z_{k,s} = 1 \qquad \forall k \in K \tag{5.34}$$

$$u_{k,s} \leq z_{k,s}(t_{k,s+1} - t_{k,s} - 1) \qquad \forall k \in K, \forall s \in \{1, \ldots, S_k - 1\} \tag{5.35}$$

$$u_{k,S_k} \leq z_{k,S_k}(T^{prod} - t_{k,S_k}) \qquad \forall k \in K \tag{5.36}$$

Modèle 5.3: Modèle incluant les fonctions de coût DC

L'expression (5.26) est la fonction objectif de notre modèle. Les contraintes (5.27), (5.28),

(5.29), (5.30), (5.31) et (5.32) sont similaires aux contraintes (5.3), (5.4), (5.5), (5.6), (5.7) et (5.8) présentées dans le Modèle 5.1. Les contraintes (5.33) font le lien entre les dates de départ des lots dans la partie production et les dates de départ des lots dans la partie livraison. Les contraintes (5.34) jusqu'à (5.36) modélisent les fonctions de coût DC afin d'obtenir les coûts de livraison en fonction de la date de départ.

5.3.2.2 Modèle de programmation par contraintes

Le modèle DC_{CP} est une variante du Modèle 5.3 qui utilise la programmation par contraintes. Le modèle proposé utilise des variables de type intervalle, de type séquence d'intervalles ainsi que des variables réelles classiques. Les intervalles sont couramment utilisés en programmation par contraintes et sont associés à plusieurs attributs comme la "taille", le "début" et la "fin" représentant respectivement la durée, la date de début et de fin d'un intervalle de préparation d'une commande sur une machine. Une variable de type séquence d'intervalles est composée d'un ensemble d'intervalles et modélise un ordre total au sein de cet ensemble. Des contraintes comme "noOverlap" et "mêmeSequence" permettent d'obtenir, respectivement, une chaîne sans chevauchement des intervalles et d'avoir la même séquence entre deux variables. Nous renvoyons le lecteur vers [Laborie et al., 2018] pour plus de détails.

Les variables de décision sont les suivantes :

- l'intervalle $\mathcal{I}_{i,l}$ modélise la préparation de la tâche associée à la commande l sur la machine i $(\forall l \in L,\ \forall i \in M)$.
- la séquence Ψ_i modélise la séquence de tous les intervalles $\mathcal{I}_{i,l}$ sur la machine i $(\forall i \in M\ : \Psi_i = Sequence(\bigcup_{l \in L} \mathcal{I}_{i,l})$.

Dans le cadre de la programmation par contraintes, nous pouvons définir directement les fonctions linéraires par morceaux DC_k grâce aux ensembles de dates $t_{k,s}$ et aux ensembles de coefficients directeurs $\alpha_{k,s}$ $(\forall k \in K,\ \forall s \in S_k)$. La fonction objectif devient :

$$\begin{aligned} DC_{CP} : \min RC + PC + IC = \min & \sum_{k \in K} \mathrm{DC}_k(F_k) + \sum_{l \in L} début(\mathcal{I}_{1,l}) h_l^{START} \\ & + \sum_{l \in L} \sum_{i=1}^{m-1} (début(\mathcal{I}_{i+1,l}) - p_{i,l} - début(\mathcal{I}_{i,l})) h_{i,l}^{WIP} \\ & + \sum_{k \in K} \sum_{l \in L_k} (F_k - p_{m,l} - début(\mathcal{I}_{m,l})) h_l^{FIN} \end{aligned}$$

Les contraintes du modèle sont les suivantes :

L'ensemble de contraintes (5.37) fait correspondre la taille des intervalles aux durées des tâches. Les contraintes (5.38) assurent la production dans l'ordre des tâches sur les différentes machines. Les contraintes (5.39) assurent que la production démarre après la date 0. Les contraintes (5.40) garantissent le non chevauchement des tâches sur les machines. Les contraintes 5.41 garantissent que les commandes soient produites dans le même ordre sur chacune des machines. Enfin, les contraintes (5.42) associent la date de départ des véhicules à la fin de la préparation des tâches.

5.3.2.3 Procédure d'évaluation

Lorsque la séquence de préparation des commandes est connue (i.e. les variables $y_{l1,l2}$ du Modèle 5.3 sont fixées), les dates optimales de fin de production des commandes sur les machines doivent

$$taille(\mathcal{I}_{i,l}) = p_{i,l} \qquad \forall i \in I, \forall l \in L \tag{5.37}$$
$$fin(\mathcal{I}_{i,l}) \leq début(\mathcal{I}_{i+1,l}) \qquad \forall i \in \{1,\dots,I-1\}, \forall l \in L \tag{5.38}$$
$$0 \leq début(\mathcal{I}_{1,l}) \qquad \forall l \in L \tag{5.39}$$
$$noOverlap(\Psi_i) \qquad \forall i \in I \tag{5.40}$$
$$sameSequence(\Psi_1, \Psi_i) \qquad \forall i \in I \backslash \{1\} \tag{5.41}$$
$$fin(\mathcal{I}_{m,l}) \leq F_k \qquad \forall v \in K, \forall l \in L_k \tag{5.42}$$

Modèle 5.4: Modèle de programmation par contraintes

encore être déterminées. On suppose sans perte de généralité, que l'ensemble des commandes L et des ensembles L_k $(k \in K)$ sont réindexés en fonction de la séquence de production. Les dates de production peuvent être obtenues par la résolution du Modèle 5.5. La séquence de production étant fixée, la variables $C_{i,l}$ représente maintenant la date de fin de la tâche associée à la l^{eme} commande de l'ordonnancement sur la machine i.

$$\min IC + LC \tag{5.43}$$
$$s.t.\ C_{1,1} = p_{1,1} \tag{5.44}$$
$$C_{i,l+1} \geq C_{i,l} + p_{i,l+1} \qquad \forall i \in M, \forall l \in \{1,\dots,o-1\} \tag{5.45}$$
$$C_{i+1,l} \geq C_{i,l} + p_{i+1,l} \qquad \forall i \in \{1,\dots,m-1\}, \forall l \in L \tag{5.46}$$
$$C_{m,l} \leq F_k \qquad \forall k \in K, \forall l \in L_k \tag{5.47}$$
$$(5.33), (5.34), (5.35), (5.36)$$

Modèle 5.5: Modèle de la procédure d'évaluation

L'expression (5.43) est la fonction objectif du modèle. Les contraintes (5.44), (5.45), (5.46) et (5.47) sont équivalentes aux contraintes (5.5), (5.6), (5.7) et (5.8) du Modèle 5.1 pour lequel la séquence de production est fixée. Les contraintes (5.33), (5.34), (5.35) et (5.36) proviennent du Modèle 5.1.

Un modèle composé uniquement de variables continues (comme le Modèle 5.5) est un PLS (Programme Linéaire Simple). Il existe des algorithmes polynomiaux pour résoudre ce genre de modèle. Dans ce chapitre, nous choisissons de le résoudre à l'aide d'un solveur commercial, ce qui est rapide également.

Pour cette raison, ce modèle peut être utilisé pour évaluer un grand nombre de séquences de production dans le cadre d'une recherche locale ou autres types d'heuristiques.

5.3.2.4 Recherche locale pour la séquence de production

Pour trouver un bon ordonnancement, nous appliquons successivement deux types de recherches locales. La première agit "au niveau des lots" et explore des séquences de production des lots. On considère alors que les commandes de chaque lot sont produites à la suite sans qu'une commande d'un autre lot ne vienne s'intercaler dans cette sous séquence. Ensuite, la seconde recherche locale agit "au niveau des commandes" et améliore la solution en modifiant la séquence de production des commandes. L'ensemble de ces deux recherches locales se note méthode NS. La méthode NS utilise la procédure d'évaluation présentée Section 5.3.2.3 pour évaluer les solutions explorées. En

effet, la distribution est prise en charge par la procédure d'évaluation, qui intègre les fonctions de coûts de transport DC. Un itinéraire de tournée est automatiquement associé à un véhicule une fois sa date de départ fixée.

Première recherche locale Lors de la première recherche locale, une solution est représentée comme une séquence de lots. Les commandes au sein de chacun de ces lots sont ordonnancées en utilisant l'algorithme NEH

Après le séquencement des commandes dans chaque lot et pour créer la solution initiale, les lots sont ordonnancés par valeur moyenne de date de livraison souhaitée pour leurs commandes ("EDD moyen") et la séquence de production globale est évaluée par la procédure d'évaluation. L'opérateur de voisinage utilisé consiste à déplacer un lot k de sa position a à une position b dans la séquence. Afin de réduire la taille du voisinage, un paramètre λ_{lot} est défini et représente l'écart maximum entre la position initiale d'un lot et la position où ce lot peut être réinséré. L'ensemble des solutions voisines est généré en déplaçant chaque lot de la solution en position a vers toutes les positions b comprises entre $a - \lambda_{lot}$ et $a + \lambda_{lot}$.

Dès qu'une meilleure solution que la solution courante est trouvée en déplaçant un lot, elle devient la nouvelle solution courante. L'exploration reprend ensuite en adoptant une des trois stratégies suivantes :

- stratégie "1" : après chaque amélioration, l'exploration continue depuis la position a et la position d'insertion b et s'arrête dès que $a = n$ et $b = n - 1$.
- stratégie "P" : après chaque amélioration, l'exploration reprend depuis la position $a = \min(a, b)$ et la position d'insertion $b = \min(a, b) - \lambda_{lot}$. Le but est d'intensifier l'exploration autour du point d'amélioration de la nouvelle solution courante. L'exploration s'arrête dès que $a = n$ et $b = n - 1$.
- stratégie "S" : après chaque amélioration, l'exploration continue depuis la position a et la position d'insertion b. Lorsque $a = n$ et $b = n - 1$, l'exploration repart à $a = 1$ et $b = 2$. La recherche locale s'arrête dès qu'un optimum local est trouvé, i.e. quand aucune solution dans le voisinage ne peut améliorer la solution courante.

Deuxième recherche locale Lors de la seconde recherche locale, une solution est représentée comme une séquence de commandes. La solution trouvée par la première recherche locale, évaluée par la procédure d'évaluation, donne la solution initiale de la deuxième recherche locale. L'opérateur de voisinage consiste à déplacer une commande depuis une position a jusqu'à une position b de la séquence. Pour réduire le nombre de solutions dans le voisinage, un paramètre $\lambda_{commande}$ est défini et représente l'écart maximum entre la position initiale d'une commande et la position où cette commande peut être réinsérée. L'ensemble des solutions voisines est généré en déplaçant chaque commande depuis sa position a jusqu'à chaque position b comprise entre $a - \lambda_{commande}$ et $a + \lambda_{commande}$. Dès qu'une solution meilleure que la solution courante est trouvée en déplaçant une commande, elle devient la nouvelle solution courante. L'exploration reprend en adoptant l'une de trois stratégies utilisées pour la première recherche locale.

5.4 Expérimentations et résultats

Avant de présenter les expérimentations évaluant les performances des méthodes proposées, nous décrivons en Section 5.4.1 comment les instances sont générées. Un critère pour comparer les fonctions de coûts DC est présenté et justifié en Section 5.4.2. Enfin les résultats des expériences

sont présentés. Dans un premier temps, nous évaluons l'efficacité des méthodes utilisées pour déterminer les fonctions DC en Section 5.4.3. Ensuite, les résultats de la méthode NS pour résoudre le problème global sont détaillés et discutés en Section 5.4.4.

Les résultats présentés dans cette section sont réalisés sur un ordinateur équipé d'un processeur Intel Core i7-7820HQ et de 16 Go RAM. Les algorithmes sont exécutés sur un seul thread sous un système Windows et sont développés en C++ avec Visual Studio. IBM Ilog Cplex 12.7 est utilisé comme solveur.

5.4.1 Génération des instances

La difficulté de résoudre une instance est principalement liée à un équilibre entre la partie production et la partie livraison du problème. Si une partie domine largement l'autre en termes de coût, se concentrer sur la résolution de cette partie peut être suffisant pour obtenir des solutions de très bonne qualité. Nous considérons ces cas comme "faciles" dans le sens que la résolution de l'une des ces deux parties seule est plus importante que l'autre. Afin de manipuler des instances "difficiles", nous générons des instances aléatoires tel que les deux parties soient équilibrées.

Trois types d'instances ont été créées : *instances à 1 lot (1I)*, *petites instances (SI)* et *grandes instances (LI)*. Les instances 1I sont utilisées pour tester les méthodes de génération des fonctions de coût DC et ne contiennent pas d'informations liées à la production. Les instances SI et LI sont utilisées pour tester les recherches locales utilisées dans la résolution du problème global. Les ensembles 1I, SI et LI sont composés respectivement de 16, 12 et 6 ensembles de 20 instances. Le nombre de commandes et la méthode utilisée pour composer les lots sont différents pour chaque ensemble. L'ensemble 1I est généré pour un nombre de commandes n allant de 5 jusqu'à 20. L'ensemble SI est généré pour un nombre de commandes n allant de 5 jusqu'à 10 et l'ensemble LI pour un nombre de commandes n valant 20, 50 ou 100. Nous considérons un *flow-shop* de permutation à 2 machines pour l'ensemble SI et 5 machines pour l'ensemble LI. Pour chaque commande l, la pénalité de retard unitaire π_l est générée aléatoirement d'après une loi normale de distribution d'espérance 5 et d'écart type 2. Les coûts d'inventaire d'une commande l sont générés de telle sorte qu'ils augmentent de 1 ou 2 unités d'une machine à l'autre : le coût d'inventaire initial h_l^{START} est tiré aléatoirement dans $[1, 2]$, le coût d'inventaire $h_{1,l}^{WIP}$ est tiré aléatoirement dans $[h_l^{START} + 1, h_l^{START} + 2]$, $h_{i+1,l}^{WIP}$ est tiré aléatoirement dans $[h_{i,l}^{WIP} + 1, h_{i,l}^{WIP} + 2]$ et le coût d'inventaire final h_l^{FIN} est tiré aléatoirement dans $[h_{m,l}^{WIP} + 1, h_{m,l}^{WIP} + 2]$. La durée du temps de production $p_{i,l}$ est choisie aléatoirement entre 1 et 10. Les différents sites clients et dépôts sont placés aléatoirement sur un carré de taille 10×10 avec une distance tt_{l_1,l_2} séparant les sites l_1 et l_2 et déterminée en utilisant la distance Euclidienne. Nous considérons que tc_{l_1,l_2} est égal à tt_{l_1,l_2}.

Les dates souhaitées de livraison des commandes sont aléatoirement choisies entre 0 et une estimation de la plus grande date de livraison possible. Cette estimation est calculée comme la somme de T^{prod} (dernière date de production possible) et d'une borne supérieure de la durée de tournée, notée T^{tour}. T^{prod} vaut $(n+m-1) \times 10$, 10 étant le temps de production maximale d'une tâche. T^{tour} vaut $(n/|K|) \times 10\sqrt{2}$, $n/|K|$ étant le nombre moyen de commandes par lot et $10\sqrt{2}$ la longueur maximale d'une arête inscrite dans un carré de 10×10.

Le nombre de lots et leurs compositions sont générés en fonction d'un nombre de commandes moyen par lot, noté γ. Les instances SI sont composées de 2 lots, γ correspond donc à n divisé par 2. Pour les instances LI, γ vaut 5 et le nombre de commandes par lot est tiré aléatoirement dans $[0.6\gamma, 1.4\gamma]$.

Une fois la taille de chaque lot fixée, la composition des lots est déterminée d'après une des deux règles suivantes :

– Lot *Non trié* : les commandes sont assignées aléatoirement aux lots.

– Lot *Trié* : les commandes sont préalablement triées par ordre de dates de livraison souhaitées croissante (EDD) puis assignées aux lots dans cet ordre (i.e. le premier lot est composé des commandes avec les dates dues les plus petites, le second lot est composé des commandes avec les dates suivantes ...)

Les instances 1I ne sont composées que d'un lot. De ce fait, les paramètres *Non trié* et *Trié* n'ont pas d'impact.

5.4.2 Indicateurs de comparaison

Avant de présenter les résultats expérimentaux des méthodes déterminant les fonctions de coût DC, nous introduisons trois indicateurs noté $MI(X), AI(X)$ et $BI(X)$. L'objectif de ces indicateurs est de comparer équitablement les résultats des méthodes exactes et approchées. Chaque méthode retourne un ensemble $\mathcal{R}_k^{app}$ de routes qui décrit la fonction DC_k associée au lot k. Nous évaluons l'ensemble $\mathcal{R}_k^{app}$ de routes trouvé par chaque méthode sur l'intervalle $\{T^{min}, \ldots, T^{max}\}$ où T^{min} représente la date de départ pour laquelle toutes les commandes sont livrées à l'heure quelque soit l'itinéraire choisi et T^{max} correspond à la date de départ telle que toutes les commandes sont livrées en retard quelque soit l'itinéraire choisi. Notons que, contrairement à l'intervalle $[0, T^{prod}]$, cet intervalle est indépendant des données de production. Lors des expérimentations, nous fixons $T^{min} = min_{l \in L}\{d_l - n \times 10\sqrt{2}\}$ et $T^{max} = max_{l \in L}\{d_l\}$.

Les indicateurs visent à évaluer une méthode X en comparant la fonction DC^X trouvée par cette méthode et la fonction DC^* donnée par une méthode exacte (PSE). Plus précisément, les indicateurs reflètent l'écart des deux fonctions sur trois sous intervalles de $\{T^{min}, \ldots, T^{max}\}$: $\{T^{min}, \ldots, A_X\}, \{A_X, B_X\}$ et $\{B_X, \ldots, T^{max}\}$ avec A_X la dernière date pour laquelle l'écart entre DC^X et DC^* sur l'intervalle $\{T^{\min}, A_X\}$ est constant et B_X la première date pour laquelle l'écart entre DC^X et DC^* sur l'intervalle $\{B_X, T^{\max}\}$ est constant. Nous définissons également :

– a_X la date du premier changement de pente de la fonction DC^X,

– a^* la date du premier changement de pente de la fonction DC^*,

– b_X la date du dernier changement de pente de la fonction DC^X,

– b^* la date du dernier changement de pente de la fonction DC^*,

La date de départ a_X (resp. a^*) correspond à la date pour laquelle la tournée avec le coûts de transport minimum de l'ensemble DC^X (resp. DC^*) commence à livrer son premier client en retard. De même, la date de départ b_X (resp. b^*) correspond à la date pour laquelle la tournée avec les coûts de livraison les plus faibles de l'ensemble DC^X (resp. DC^*) pour un départ en T_{max} commence à livrer l'ensemble de ses clients en retard.

Nous avons alors $A_x = min\{a_X, a^*\}$ et $B_X = max\{b_x, b^*\}$. Une illustration de la fonction DC^* et d'une DC^X sur l'intervalle $\{T^{min}, \ldots, T^{max}\}$ est présentée à la Figure 5.11.

L'indicateur principal $MI(X)$, présenté par l'expression 5.48 mesure la différence d'aire sous la fonction DC^X obtenue par la méthode X et l'aire sous la fonction DC^* obtenu par la PSE, exprimé en pourcentage. $MI(X)$ est défini uniquement sur l'intervalle $\{A_X, B_X\}$ et peut être vu comme l'écart relatif entre ces deux courbes : un ratio de 0% indique que les courbes sont identiques et la différence entre les courbes augmente avec le ratio.

$$MI(X) = \frac{\int_{A_X}^{B_X} [\mathrm{DC}^X(t) - \mathrm{DC}^*(t)]\mathrm{d}t}{\int_{A_X}^{B_X} [\mathrm{DC}^*(t)]\mathrm{d}t} \tag{5.48}$$

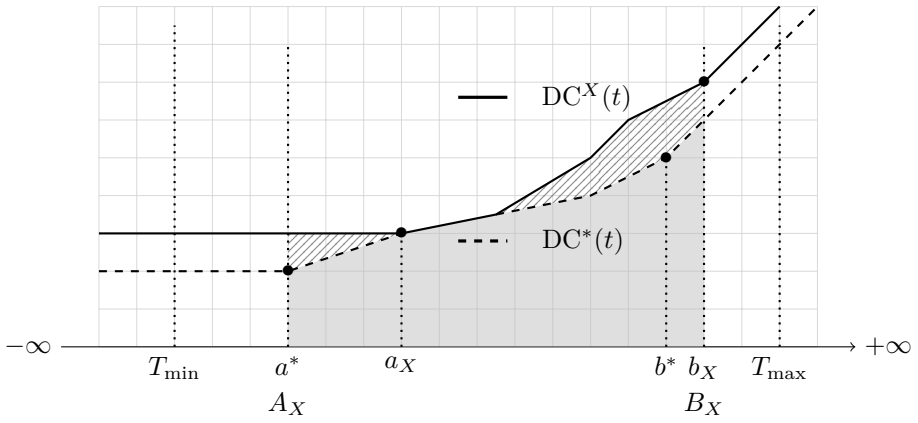

Fig. 5.11: Illustration des indicateurs de comparaison

Les deux autres indicateurs, $AI(X)$ et $BI(X)$ (expressions 5.49 et 5.50) calculent le ratio entre la fonction DC^X et la fonction DC^* pour les dates de départ A_X et B_X respectivement. Notons que cet écart est constant sur l'intervalle $\{-\infty, A_X\}$ et $\{B_X, +\infty\}$.

$$AI(X) = \frac{\mathrm{DC}^X(A_X)}{\mathrm{DC}^*(A_X)} \tag{5.49}$$

$$BI(X) = \frac{\mathrm{DC}^X(B_X)}{\mathrm{DC}^*(B_X)} \tag{5.50}$$

5.4.3 Efficacité des méthodes de prétraitement pour déterminer les fonctions DC

Nous comparons d'abord les résultats obtenus par le modèle mathématique présenté en Section 5.3.1.2 et la procédure par séparation et évaluation (PSE) détaillée en Section 5.3.1.2 sur l'ensemble d'instances 1I. La fonction DC est calculée sur l'intervalle de temps $\{T^{min}, T^{max}\}$. La borne supérieure de PSE est calculée en utilisant la stratégie "8-appels".

La Figure 5.12 montre le temps de calcul moyen des deux méthodes exactes en fonction du nombre de commandes par lot. Nous imposons une limite de temps de 1 heure pour chaque méthode. La résolution du modèle mathématique à l'aide d'un solveur ne résout en moins d'une heure que des instances de taille n inférieure ou égale à 9, et seulement 85% des instances de taille 10. Alors que la PSE trouve la solution optimale de toutes les instances jusqu'à 17 commandes en moins d'une heure. Les instances jusqu'à 13 commandes par lot sont résolues en moins d'une minute.

Les trois versions de l'heuristique (les méthodes "1-appel", "2-appels" et "8-appels") présentées Section 5.3.1.2 sont maintenant évaluées. Figure 5.13 représente le temps moyen de résolution de ces trois versions pour chaque type d'instance. Les Figures 5.14, 5.15 et 5.16 montrent la valeur moyenne des différents indicateurs ($MI(X)$, $AI(X)$ et $BI(X)$) pour chaque version en fonction de la taille des instances résolues.

Dans tous les cas, la qualité des fonctions de coût trouvées par la méthode "8-appels" est meilleure que par la méthode 2-appels, elle même meilleure que par la méthode "1-appel". Cette stratégie d'appels multiples permet d'obtenir une meilleure approximation de la fonction DC au détriment d'une augmentation du temps de résolution. Ce dernier est directement proportionnel au nombre d'appels global. L'heuristique est clairement plus rapide que la PSE pour de grandes tailles d'instance et permet de trouver de bonnes approximations des fonctions DC.

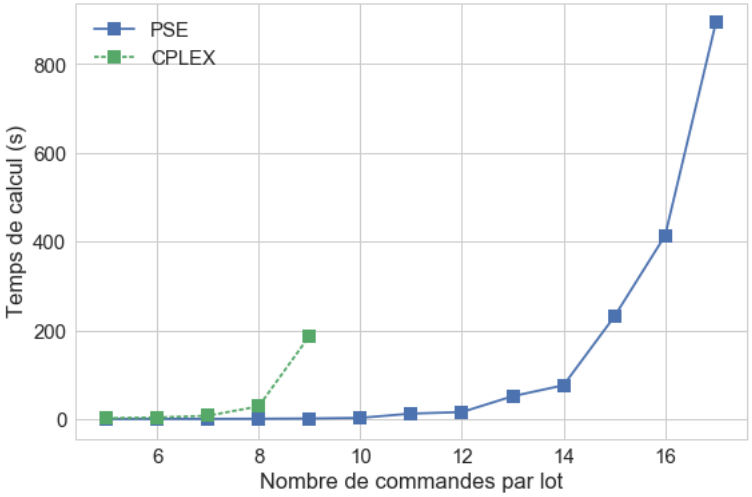

Fig. 5.12: Temps de calcul moyen des méthodes exactes

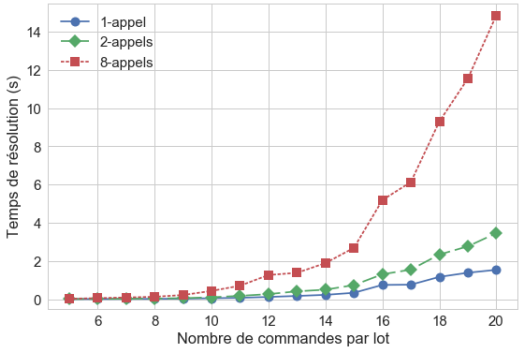

Fig. 5.13: Temps de calcul moyen des méthodes approchées

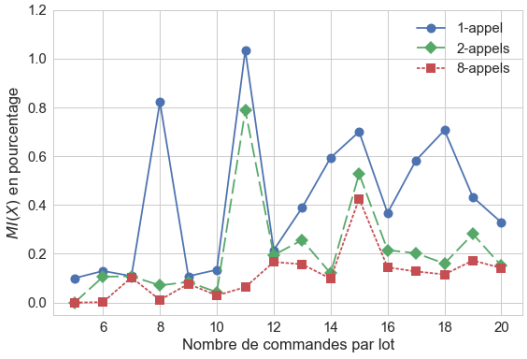

Fig. 5.14: Indicateur de comparaison $MI(X)$

Concernant les trois indicateurs $MI(X)$, $AI(X)$ et $BI(X)$, pour la méthode "8-appels", la valeur moyenne de $AI(X)$ augmente jusqu'à une valeur 3% d'écart moyen alors que les indicateurs $MI(X)$ et $BI(X)$ fluctuent en dessous de 0.45%. Pour expliquer cette différence, nous pouvons noter que la valeur $\mathrm{DC}^*(A_X)$ est bien plus faible (et donc plus sensible à de petites variations) que la valeur de $\mathrm{DC}^*(B_X)$. En effet, la tournée associée à $\mathrm{DC}^*(A_X)$ minimise les coûts de transports entre les clients seulement car les pénalités de retard pour un départ en A_X sont nulles. Alors que $\mathrm{DC}^*(B_X)$ tient compte d'importantes pénalités de retard (devant lesquelles les coûts de transport sont négligeables). Ainsi, le ratio entre $\mathrm{DC}^*(A_X)$ et $\mathrm{DC}^*(B_X)$ est égal à 20 pour les instances de petites tailles et à plus de 100 pour les instances de grandes tailles. Par conséquent, un même écart à l'optimal aura un impact bien plus important sur $AI(X)$ que sur $BI(X)$, ce qui explique l'écart entre les deux indicateurs.

Pour conclure, la PSE est efficace pour générer des fonctions de coût jusqu'à 12 commandes par lot. Pour des instances de plus grandes tailles, la méthode "8-appels" semble la plus adaptée. Rappelons que la PSE utilise la méthode "8-appels" pour générer sa borne supérieure initiale.

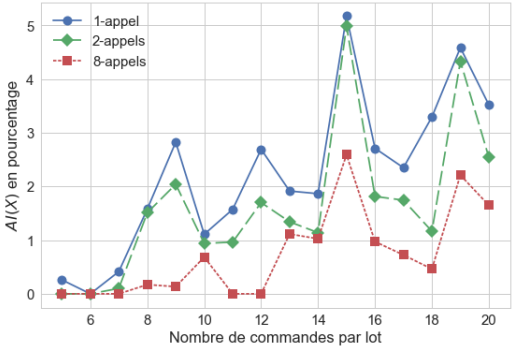

Fig. 5.15: Indicateur de comparaison $AI(X)$

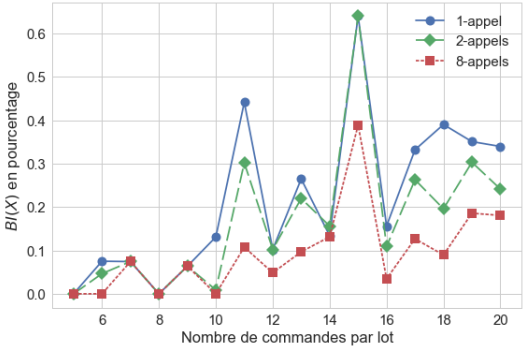

Fig. 5.16: Indicateur de comparaison $BI(X)$

5.4.4 Efficacité des méthodes de résolution du problème global

Cette section présente l'efficacité des méthodes proposées pour résoudre le problème intégré : l'algorithme glouton (GL, Section 5.2), le programme linéaire en nombres entiers (Section 5.3.2.1), le modèle de programmation par contraintes (Section 5.3.2.2) et la méthode de recherche par voisinage (NS, Section 5.3.2.4). La méthode par voisinage est soumise à trois paramètres : la taille du voisinage lors de la recherche locale permutant les lots (notée λ_{lot}), celle lors de la recherche locale permutant les commandes (notée $\lambda_{commande}$) et la méthode d'exploration utilisée (notée "1", "P" ou "S"). Les premiers résultats, donnés en Section 5.4.4.1, comparent les solutions trouvées par les méthodes exactes sur les instances de petite taille (SI). Les seconds résultats présentés en Section 5.4.4.2 rapportent les tests de la méthode sur des instances de grandes tailles (LI) pour les différents ensembles de paramètres présentés plus haut. Du fait des petites tailles de lot de ces instances (jusqu'à 10 commandes par lot pour les instance SI et LI), la PSE présentée en Section 5.3.1.2 calcule les fonctions de coût DC utilisées au sein de la procédure d'évaluation des solutions.

5.4.4.1 Résultat sur les instances SI

Pour des instances de petites tailles, les paramètres λ_{lot} et $\lambda_{commande}$ de la méthode NS (Section 5.3.2.4) n'ont quasiment pas d'impact. En effets, pour les plus grandes de ces instances, le nombre de commandes reste inférieur à 17 et le nombre de lots est égal à 2. Des valeurs faibles de λ_{lot} et $\lambda_{commande}$ sont donc suffisantes pour explorer efficacement l'ensemble des solutions. Ainsi nous fixons le paramètre λ_{lot} à 2 et le paramètre $\lambda_{commande}$ à 10. Les trois stratégies (notées "1", "P" ou "S") sont testées. Le Tableau 5.4 présente l'ensemble des résultats. Les deux premières colonnes indiquent l'ensemble d'instances testé (20 instances par ensemble). L'efficacité de la méthode NS sur de petites instances est évaluée en comparant la solution retournée par cette méthode aux solutions trouvées par la méthode GL (5.2) d'une part, et aux solutions retournées par les modèles DC_{MILP} (5.3.2.1) et DC_{CP} (5.3.2.2) d'autre part.

La colonne "#OPT NS" indique pour chaque stratégie de recherche le nombre d'instances (sur 20) pour lesquelles la méthode NS a trouvé la solution optimale. La colonne "Écart (%)" indique pour chaque stratégie l'écart moyen entre la solution trouvée par la méthode NS et la solution optimale : $\acute{E}cart = \frac{Z(NS)-Z^*}{Z^*}$. La colonne "Écart GL (%)" indique pour chaque stratégie l'écart moyen entre la solution trouvée par la méthode GL avec la solution optimale : $\acute{E}cart = \frac{Z(GL)-Z^*}{Z^*}$. Les colonnes "Temps CPU NS(sec)", "CPU CPLEX (sec)" et "CPU CP (sec)" indiquent le temps de calcul moyen de chacune des trois méthodes. Le temps de résolution de la méthode GL, n'excédant jamais une seconde, n'est pas présenté dans le tableau.

Pour toutes les instances présentées, le modèle CP trouve la solution optimale en moins d'une

	Instances	#OPT NS			Écart NS (%)			Temps CPU NS (sec)			Écart GL	CPU CPLEX	CPU CP
n	Composition	1	P	S	1	P	S	1	P	S	(%)	(sec)	(sec)
5	Non trié	**20**	**20**	**20**	0.00	**0.00**	**0.00**	**0.37**	0.44	0.55	13.38	0.16	**0.21**
	Trié	**18**	**18**	**18**	0.34	**0.34**	**0.34**	**0.36**	0.44	0.48	13.58	0.08	**0.24**
6	Non trié	15	**18**	**18**	0.21	**0.04**	**0.04**	**0.50**	0.70	0.81	13.22	0.31	**0.41**
	Trié	15	**19**	18	0.41	0.02	**0.17**	**0.50**	0.68	0.82	14.54	0.12	**0.28**
7	Non trié	13	**17**	**17**	0.39	**0.26**	**0.26**	**0.71**	0.96	1.32	16.90	1.28	**0.88**
	Trié	17	19	**20**	0.15	0.04	**0.00**	**0.76**	0.88	1.32	12.00	0.65	**0.86**
8	Non trié	11	17	**19**	0.52	0.27	**0.22**	**1.12**	1.33	1.99	18.99	3.28	**0.80**
	Trié	10	**17**	**17**	0.33	0.07	**0.06**	**1.03**	1.35	1.89	15.40	3.97	**0.78**
9	Non trié	8	**17**	**17**	0.79	**0.30**	**0.30**	**1.26**	1.78	2.63	14.92	18.46	**1.17**
	Trié	7	18	**19**	0.56	0.01	**0.00**	**1.21**	1.90	2.82	14.82	33.01	**1.20**
10	Non trié	4	12	**17**	0.80	0.32	**0.11**	**1.62**	2.63	3.79	17.09	241.37	**2.05**
	Trié	5	16	**17**	1.10	0.06	**0.03**	**1.60**	2.73	3.75	14.19	430.61	**1.86**
11	Non trié	6	15	**17**	0.58	**0.03**	0.04	**2.12**	3.46	4.66	19.64	-	**4.68**
	Trié	2	14	**15**	1.87	0.94	**0.90**	**2.02**	3.95	5.41	16.63	-	**5.17**
12	Non trié	2	13	**16**	0.82	0.18	**0.14**	**2.44**	5.09	6.31	18.51	-	**8.42**
	Trié	5	**16**	**16**	1.38	**0.09**	0.17	**2.43**	4.81	6.18	16.21	-	**5.71**
13	Non trié	0	12	**15**	1.26	0.25	**0.11**	**2.92**	6.38	8.89	16.24	-	**14.53**
	Trié	2	**14**	13	1.03	**0.15**	0.19	**2.82**	5.66	7.95	14.60	-	**12.13**
14	Non trié	2	11	**14**	0.85	0.17	**0.11**	**3.30**	6.74	9.52	14.08	-	**31.19**
	Trié	0	12	**17**	1.19	0.25	**0.14**	**3.25**	7.17	9.96	13.80	-	**34.81**
15	Non trié	1	12	**13**	1.06	0.18	**0.13**	**4.23**	9.85	11.73	20.03	-	**112.17**
	Trié	0	8	**12**	1.48	0.19	**0.14**	**4.17**	10.49	12.85	18.58	-	**165.94**
16	Non trié	0	9	**13**	1.16	0.16	**0.14**	**4.80**	11.99	15.79	13.70	-	**348.41**
	Trié	0	6	**10**	1.04	0.27	**0.19**	**4.67**	11.87	13.24	13.17	-	**274.67**
17	Non trié	1	9	**10**	1.21	0.43	**0.32**	**6.07**	13.86	16.95	15.61	-	**1656.20**
	Trié	0	8	**13**	1.69	0.21	**0.09**	**5.84**	14.55	18.18	13.91	-	**1084.14**

Tab. 5.4: Comparaison à l'optimal

heure. Cependant la résolution du modèle CP_{MILP} montre ses limites pour des instances de plus de 10 commandes. La procédure gloutonne GL propose une solution pour toutes les instances de manière instantanée, mais avec un écart à l'optimal entre 12 et 20%.

Quelle que soit la stratégie utilisée, la méthode de voisinage parvient à trouver la plupart des solutions optimales pour les instances de petites tailles (jusqu'à 12 commandes) et de bonnes solutions pour les instances plus grandes (avec un écart moyen autour de 1%), et ce, dans un délais très court. La stratégie "S" est plus efficace comparée à la stratégie "P" mais au prix d'une augmentation du temps de calcul. Notons que cette amélioration n'est pas aussi importante que l'amélioration de la stratégie "P" comparée à la stratégie "1". En effet "S" et "P" restent très similaires du fait du petit nombre de commandes par instance.

5.4.4.2 Résultats sur les instances de grandes taille LI

Cette section présente les résultats de la procédure gloutonne GL et des méthodes de recherche par voisinage NS pour les instances de grandes tailles. Les paramètres $\lambda_{lot} \in \{3, 5\}$ et $\lambda_{commande} \in \{5, 10\}$ sont testés pour les trois stratégies d'exploration ("1", "P" et "S"). Ces 12 combinaisons de paramètres sont testées sur les 6 types d'instances de l'ensemble LI ($n \in \{20, 50, 100\}$ avec instances triées et non triées). Les Figures de 5.17 à 5.22 présentent les performances moyennes de chaque combinaison de paramètres pour NS uniquement, une figure par type d'instance. Chaque figure représente un repère de coordonnées Cartésiennes où chaque point représente une combinaison de paramètres. Les coordonnées d'un point sont données par le temps de calcul moyen de la méthode et son écart moyen entre la solution trouvée par la méthode et la meilleure solution fournie par l'ensemble des 12 méthodes. La forme d'un point dépend des paramètres λ_{lot} et $\lambda_{commande}$ utilisés. L'étiquette d'un point indique la stratégie de recherche locale utilisée.

Pour les instances de 20 commandes, des expériences préliminaires ont montré que le paramètre λ_{lot} avait un impact mineur du fait du faible nombre de lots (égal à 4). Dans les résultats suivants, seul le paramètre $\lambda_{lot} = 3$ est présenté pour les instances de 20 commandes.

Le temps de calcul augmente clairement avec le nombre de commandes des instances. Il est entre 2 et 14 secondes pour les instances de 20 commandes alors qu'il passe entre 100 et 1200 secondes pour les instances de 100 commandes.

L'écart à la meilleure solution connue augmente également avec le nombre de commandes. Par exemple, l'écart pour le paramétrage $\lambda_{lot} = 3, \lambda_{commande} = 5$ et la stratégie de recherche locales "1" vaut 1.4% pour des instances triées de 20 commandes jusqu'à 8% pour des instances de 100 commandes. L'écart et le temps de résolution dépend fortement de la stratégie des recherches locales. La stratégie "1" permet d'obtenir un temps de calcul court mais donne la plupart du temps les pires résultats. La stratégie "S" fournit les meilleurs résultats mais avec un temps de calcul pouvant être jusqu'à 4 fois supérieur à celui des autres méthodes. Ainsi la stratégie "P" constitue la plupart du temps le meilleur compris entre performance et temps de calcul. Pour les instances de 20 commandes (triées ou non triées), seule la stratégie de recherche locale (1, P ou S) a un impact important sur la qualité de la solution trouvée. La raison principale est la structure des instances (date de livraison uniforme pour chaque lot et peu de commandes) qui conduit à des minimums locaux attractifs et implique qu'une recherche local spécifique mènera toujours aux mêmes résultats. Ainsi, les paramètres λ_{lot} et $\lambda_{commande}$ ont un impact sur le temps de résolution de la méthode, mais négligeable sur la qualité des solutions trouvées. Pour les instances de 20 commandes, $\lambda_{lot} = 3$ et $\lambda_{commande} = 5$ sont des paramètres suffisants pour trouver un minimum local mais des paramètres avec de plus grandes valeurs ne permettent pas de sortir de cet optimum local, et ce même pour un temps de recherche plus long. Pour tout type de stratégie, le temps de résolution de la méthode NS pour les instances non triées est toujours plus élevé que pour les instances triées. L'écart à la meilleure solution trouvée est également plus grand. Pour chaque stratégie de recherche locale, nous pouvons observer qu'une augmentation de λ_{lot} de 3 à 5 permet

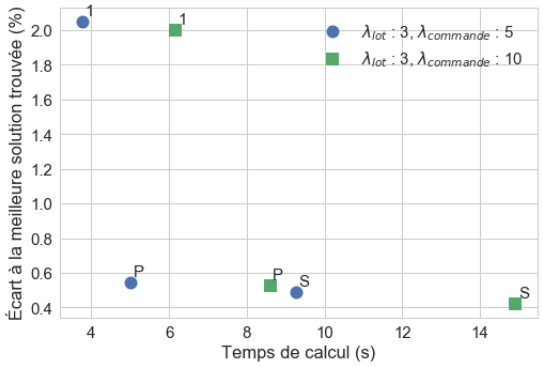

Fig. 5.17: 20 commandes et instances non triées

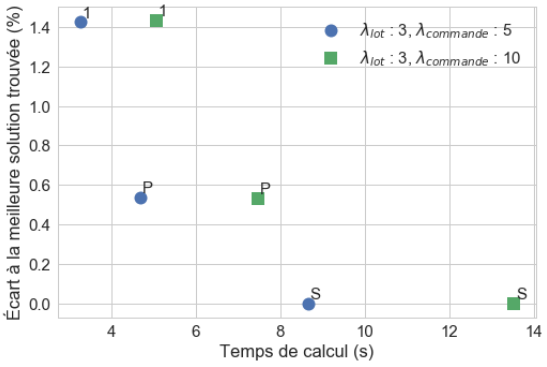

Fig. 5.18: 20 commandes et instances triées

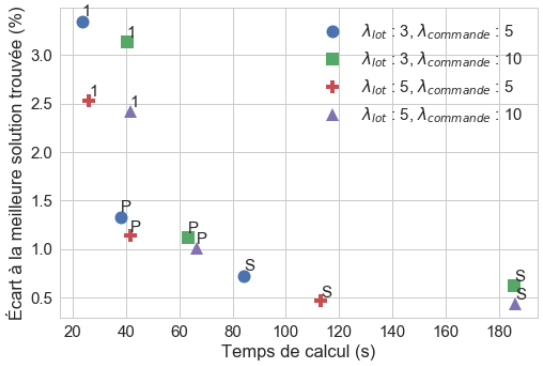

Fig. 5.19: 50 commandes et instances non triées

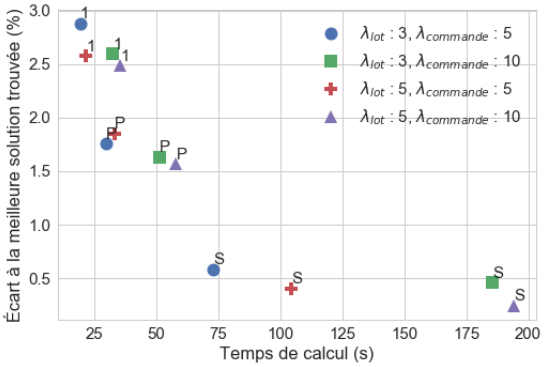

Fig. 5.20: 50 commandes et instances triées

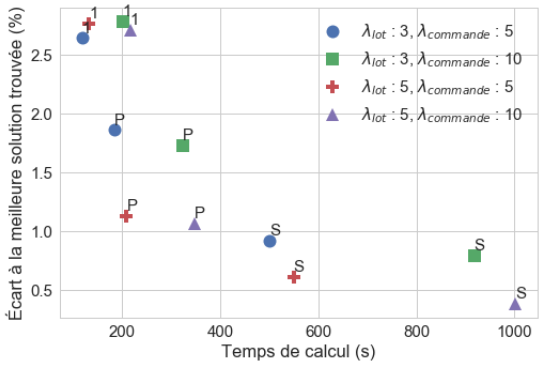

Fig. 5.21: 100 commandes et instances non triées

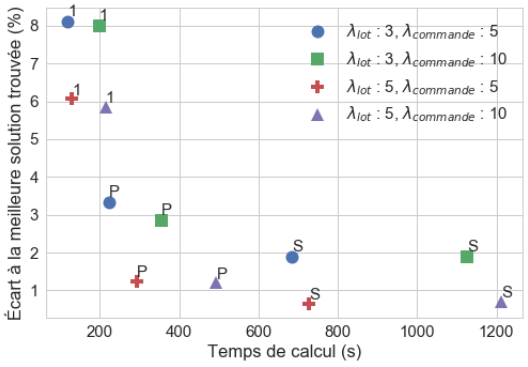

Fig. 5.22: 100 commandes et instances triées

d'obtenir au meilleure amélioration en termes de ratio entre l'écart à la meilleure solution et le temps de calcul comparée à une augmentation de $\lambda_{commande}$ de 5 à 10. De ce fait, le paramétrage $\lambda_{lot} = 3$ et $\lambda_{commande} = 10$ permet d'obtenir les écarts les plus faibles. Finalement, au vu des performance des différentes heuristiques, nous pouvons conclure que si le paramétrage "stratégie S, $\lambda_{lot} = 5$ et $\lambda_{commande} = 10$" donne les solutions de meilleures qualités, ce dernier est très coûteux en temps de calcul et ainsi le paramétrage "stratégie P, $\lambda_{lot} = 5$ et $\lambda_{commande} = 5$" propose un meilleur compromis.

Le Tableau 5.5 montre l'écart entre la solution trouvée par la procédure GL et la meilleure solution trouvée avec les deux stratégies citées précédemment : (S ;$\lambda_{lot} = 5$;$\lambda_{commande} = 10$) et (P ;$\lambda_{lot} = 5$;$\lambda_{commande} = 5$). Nous pouvons observer que la méthode NS apporte une réelle amélioration vis à vis d'une procédure gloutonne avec un écart de 23% à 30%.

Nombre de commandes	20		50		100	
Type	Non trié	Trié	Non trié	Triée	Non trié	Trié
GL vs NS avec (S ;$\lambda_{lot} = 5$;$\lambda_{commande} = 10$)	-30.1%	-24.5%	-27.2%	-27.9%	-30.5%	-27.9%
GL vs NS avec (P ;$\lambda_{lot} = 5$;$\lambda_{commande} = 5$)	-30.0%	-23.8%	-26.4%	-25.8%	-29.8%	-26.8%

Tab. 5.5: Écart entre la méthode gloutonne et celle de voisinage sur les instances LI

5.4.5 Répartition des coûts

Cette section porte sur la répartition des coûts (IC, RC et PC) au sein des meilleures solutions trouvées pour les instances de grandes tailles.

Le Tableau 5.6 correspond à la répartition moyenne (sur 20 instances) des différents coûts divisée par le nombre de commandes de l'instance. Le Tableau 5.7 indique pour chaque type d'instance la valeur moyenne de la meilleure solution trouvée, ainsi que la répartition moyenne (en pourcentage du total) des différents coûts.

La répartition des différents coûts d'inventaire (IC^{START},IC^{WIP} et IC^{FIN}) est présentée ainsi que leur total cumulé (IC). Le coût $RC + PC$ est exprimé comme étant le coût associé aux dates de départ de l'ensemble des véhicules. À noter que RC, le coût de transport, ne représente pas plus de 10 unités de coûts en moyenne par commande sur l'ensemble des instances (soit moins de 3% du coûts pour les plus petites instances).

Instances		Total	IC^{START}	IC^{WIP}	IC^{FIN}	IC	$RC + PC$
n	Composition						
20	Trié	286.9	85.8	31.5	87.3	204.7	82.3
	Non trié	314.2	82.4	32.0	85.9	200.3	113.9
50	Trié	458.1	215.1	44.3	90.5	349.9	108.2
	Non trié	546.9	208.9	45.9	88.2	342.9	204.0
100	Trié	676.3	427.6	54.5	88.5	570.6	105.6
	Non trié	874.4	408.0	55.4	88.6	552.0	322.4

Tab. 5.6: Répartitions des coûts par commande

Concernant la répartition des coûts par commande (Tableau 5.6), on peut classer les coûts en deux catégories, ceux dont le coût moyen par commande reste constant pour les différentes tailles d'instances et ceux dont le coût moyen augmente avec la taille des instances. Concernant les coûts d'inventaire, les coûts IC^{WIP} et IC^{FIN} appartiennent à la première catégorie, (nous considérons que l'augmentation de IC^{WIP} est faible au vu de l'augmentation des autres coûts). Le coût IC^{START} appartient à la second catégorie. En effet, plus un grand nombre de commande est réalisé, plus le temps de stockage amont des dernières commandes à être préparées est grand.

n	Instances Composition	Total	IC^{START} (%)	IC^{WIP} (%)	IC^{FIN} (%)	IC (%)	$RC + PC$ (%)
20	Trié	5738.3	30.5	11.2	31.0	72.6	27.4
	Non trié	6284.9	26.5	10.4	27.7	64.6	35.4
50	Trié	22907.4	47.8	9.8	20.0	77.7	22.3
	Non trié	27347.2	38.5	8.5	16.3	63.3	36.7
100	Trié	67626.4	63.6	8.1	13.2	84.9	15.1
	Non trié	87443.4	46.9	6.4	10.2	63.4	36.6

Tab. 5.7: Répartitions des coûts en pourcentage

Enfin le coûts $RC + PC$ n'augmente pas avec la taille des instances pour les instances triées, mais augmente avec la taille des instances pour les instances non triées. La valeurs stable des coûts type PC (rappelons que RC est faible) sur les instances triées confirme que la répartition des dates de livraisons souhaitées de commandes est bien homogène pour les différentes tailles d'instances (sinon des variations de PC serait remarquer) . L'augmentation des coûts type PC sur les instances non triées montre que le nombre de commandes livrées avec un retard important, voire très important, augmente avec la taille de ce type d'instance.

Concernant la valeur des fonctions objectifs des solutions, on remarque d'abord l'augmentation de la fonction objectif entre les instances triées et les instances non triées (9%, 16% et 29% pour les instances 20, 50 et 100 commandes respectivement). Cette augmentation des coûts est due exclusivement à l'augmentation de pénalités de retard PC. En effet, la part de coût PC est plus importante pour les instances non triées que pour les instances triées. On note également l'évolution de la répartition des coûts d'inventaire avec une augmentation importante des coûts IC^{START}. Comme nous avons vu précédemment que les valeurs absolues des coûts IC^{WIP} et IC^{FIN} restent constantes, cette augmentation est donc due à la seule augmentation des coûts IC^{START}.

Enfin concernant l'équilibre global des instances en terme de répartition des coûts, on remarque que le coût d'inventaire IC^{START} représente une part importante de ces coûts (même si le coût unitaire associé à chaque commande est faible, entre 1 et 2). On peut cependant noter que la majeure partie de ce coût est incompressible. Si l'on ignore le coût IC^{START}, on peut remarquer que la balance entre les coûts d'inventaire (IC^{WIP}, IC^{FIN}) et de livraison ($RC + PC$) est plus équilibrée : 42%-27% et 38%-35% pour les instances de 20 commandes, triés et non triés, 30%-22% et 25%-37% pour les instances de 50 commandes, triés et non triés et 21%-15% et 16%-36% pour les instances de 100 commandes, triés et non triés.

Pour finir, même si les résultats précédents portent sur les meilleures solutions trouvées, des analyses complémentaires ont montré que la répartition des coûts était similaire pour toutes les solutions, quelque soit le paramétrage utilisé pour les générer.

5.5 Conclusion

Ce chapitre propose des méthodes de résolution du problème décrit dans le Chapitre 3, sous hypothèse que la composition des lots pour la livraison est déjà connu. Le problème est résolu en deux étapes. La première étape consiste à résoudre la partie livraison. Pour chaque lot et chaque date de départ, un itinéraire de livraison minimisant le coût de transport et de retard est déterminé. Ces résultats sont agrégés sous forme de fonction linéaire par morceau, une fonction par lot. Pour générer ces fonctions, nous proposons une méthode par séparation et évaluation (PSE), ainsi que des heuristiques. La seconde étape est dédiée à la résolution de la partie ordonnancement. Deux méthodes exactes et une recherche locale avec différentes stratégies sont proposées pour réaliser cette étape. La première méthode exacte est basée sur la résolution d'un programme

linéaire en variables mixtes et la seconde est basée sur la résolution d'un modèle de programmation par contraintes. L'heuristique proposée est basée sur une recherche de voisinage à deux niveaux (niveau des lots et niveau des commandes). Elle utilise une procédure d'évaluation pour trouver les meilleures dates de réalisation des tâches associées aux commandes ainsi que les meilleures dates de départ des véhicules à partir d'une séquence de préparation des commandes et des fonctions linéaires par morceaux de coûts de livraison. Les expérimentations montrent que la résolution du modèle de programmation par contraintes apporte un amélioration significative du temps de résolution par rapport à la résolution du modèle en variables mixtes. Cependant, une fois la taille des instances atteignant 17 commandes, les méthodes exactes nécessitent un temps de calcul trop important et les heuristiques sont alors préférées.

La PSE est capable de trouver rapidement la fonction de coût associée à des lots jusqu'à 12 commandes. Ensuite, des méthodes heuristiques peuvent lui être préférées. Ces méthodes sont capables de générer rapidement (au plus une quinzaine de secondes) des fonctions de coûts à moins de 1% de l'optimal. Pour résoudre le problème général sur des instances de grande taille, plusieurs intensités de recherche locale sont testées pour aboutir à un bon compromis entre temps de résolution et performance. Le paramétrage retenu résout des instances de 100 commandes en 5 minutes avec un écart moyen par rapport à la meilleure solution trouvé en dessus de 2%.

Plusieurs perspectives de recherche peuvent être considérées. La seconde étape de résolution de ce problème (basé sur une recherche locale) pourrait être améliorée en se basant sur les travaux de [Bülbül et al., 2004]. Ces travaux cherchent à combiner des méthodes de génération de colonne et de relaxation Langrangienne afin de résoudre à l'exact un problème de *flowshop* avec coût d'inventaire et pénalisation de la fin de production en avance ou en retard. Une seconde extension consisterait à essayer de résoudre le problème plus général où le nombre et la composition des lots ne sont pas connus. Une génération de colonne pourrait être mise en place dans laquelle chaque colonne serait un lot dont la fonction de coût de livraison serait pré-calculée.

Chapitre 6

Plusieurs producteurs et un client avec livraison fixée

Contents

6.1 Présentation du problème

6.1.1 Description du problème et justification

Ce chapitre porte sur l'étude d'un problème intégré au sein de la chaîne d'approvisionnement. Les deux mêmes types d'acteurs sont présents : producteur et prestataire logistique de services (3PL).

Dans ce chapitre, un ensemble de producteurs approvisionne un même et unique client. Chaque commande produite doit être livrée au client. La livraison est assurée par un unique 3PL associé à l'ensemble des producteurs et disposant d'une flotte homogène de véhicules. Tous les véhicules ont la même capacité. À chaque tournée, un véhicule du 3PL visite un sous-ensemble de producteurs et charge tout ou partie des commandes déjà produites. Il termine sa tournée en allant livrer le client et en respectant les dates dues de livraison des commandes.

La modélisation des systèmes de production de chaque producteur est soumise aux mêmes coûts et aux mêmes contraintes que dans les chapitres précédents (Section 3.1), c'est-à-dire un *flow-shop* de permutation avec coûts d'inventaire. Un ordonnancement se résume à une seule séquence de production commune à toutes les machines du *flow-shop*. Cette contrainte est largement répandue dans l'industrie. Elle paraît raisonnable aux agents de terrain et modélise par exemple les systèmes de production avec convoyeur (ou l'ordre des commandes ne peut pas être modifié). Les matériaux nécessaires à la fabrication d'une commande sont initialement stockés jusqu'au début de sa production. La production d'une commande est divisée en plusieurs tâches réalisées dans un ordre spécifique, chacune sur une machine spécialisée. Chaque tâche est ordonnancée sur chacune de ces machines et chaque commande doit être stockée temporairement si la tâche associée n'est pas directement réalisée sur la machine suivante. La finalisation d'une commande entraîne son stockage jusqu'à son ramassage par un véhicule du 3PL. Chaque étape de stockage génère un coût, dépendant de la commande et proportionnel au temps de stockage. On parle de stockage avant production, en cours de production et en fin production.

La modélisation de la partie livraison de ce problème est inspirée du problème décrit dans [Chenevoy, 2016]. Dans cet article issu d'une revue spécialisée pour la grande distribution, un ensemble de fournisseurs du groupe Carrefour s'associe à un 3PL pour mutualiser les livraisons à leur client. Étant donné que chaque véhicule transporte un ensemble de biens produits par plusieurs producteurs, une contrainte de capacité des véhicules paraît non négligeable. Cette contrainte est notamment attestée dans [Chenevoy, 2016] : "Actuellement, grâce à ce travail collaboratif, le taux de remplissage des camions atteint 80%". Concernant le mise en place de dates dues pour la livraison, nous reconnaissons qu'il existe des mécanismes dans la grande distribution pour pénaliser les retards lors de la livraison. Cependant, nous considérons que ces retards sont principalement le fait d'aléas non pris en compte par la planification. Nous faisons donc l'hypothèse que les commandes sont soumises à des dates dues de livraison impératives, interdisant les retards.

L'approche proposée dans ce chapitre fait l'hypothèse que l'ensemble des livraisons a été fixé lors d'un accord préalable entre les producteurs et le 3PL. Ainsi, les itinéraires des tournées et leurs dates de passages chez les producteurs et le client sont déjà connus. Ce planning de livraison doit être assez robuste pour acheminer dans les temps toutes les commandes que peut passer le client. Ce type d'entente entre acteurs pour assurer la livraison est déjà en place pour livrer des médicaments du CHU Bretonneau de Tours vers différents hôpitaux de la ville. Plusieurs véhicules spécialisés jouent le rôle de navette journalière en suivant un emploi du temps et des itinéraires pré-définis pour livrer des médicaments.

La flotte des véhicules du 3PL est composée de véhicules homogènes avec la même capacité. Lors de son passage chez un producteur, un véhicule n'attend pas, toutes les commandes à charger doivent être prêtes et sont chargées instantanément. En effet, l'attente d'un véhicule est susceptible de générer du retard pour toutes les commandes à livrer par ce véhicule. Chaque commande a un poids et la somme des poids des commandes chargées ne doit pas dépasser la capacité du véhicule. L'ensemble des véhicules pouvant livrer une commande donnée est restreint aux véhicules arrivant chez le client avant la date due de livraison de la commande.

Les producteurs doivent maintenant planifier la séquence de production sur les différents sites ainsi que l'affectation des commandes aux véhicules. Ils prennent ces décisions conjointement et cherchent à minimiser l'ensemble des coûts d'inventaire. On notera que, contrairement aux chapitres

précédents, il faut s'assurer de la faisabilité des solutions lors de la résolution du problème en plus de minimiser les coûts. En comparaison avec les chapitres précédant, cette contrainte de faisabilité remplace les pénalités dues aux clients, qui sont maintenant nulles. La fonction objectif de ce problème ne prend plus en compte que les différents coûts d'inventaire liés à la production.

Exemple : Le paragraphe et les figures suivantes visent à présenter un exemple de solution pour une instance donnée.

Dans l'instance considérée, deux producteurs j_1 et j_2 ont chacun trois commandes à livrer au client. Le système de production de chaque producteur est un *flow-shop* de permutation à deux machines. Les commandes sont livrées par deux véhicules k_1 et k_2 arrivant respectivement chez le client aux dates 10 et 15. Chaque véhicule a une capacité de 3. Seules les caractéristiques principales des commandes sont détaillées ci dessus. Les durées d'exécution des tâches associées aux différentes commandes sont représentées graphiquement sur les Figures 6.1 et 6.2. Les caractéristiques des commandes $l_{1,1}$, $l_{1,2}$ et $l_{1,3}$ du producteur j_1 et $l_{2,1}$, $l_{2,2}$ et $l_{2,3}$ du producteur j_2 sont les suivantes : les commandes $l_{1,1}$ et $l_{2,1}$ ont des dates de livraison inférieures strictement à 15 et ne peuvent être chargées que dans le premier véhicule. Les autres commandes peuvent être chargées dans tous les véhicules. Toutes les commandes prennent un espace de 1 dans les véhicules. Les dates de passage des véhicules sur les différents sites sont présentées Tableau 6.1.

Site	k_1	k_2
3PL	1	5
j_1	4	12
j_2	7	9
Client	10	15

Tab. 6.1: Date de visite $D_{k,j}$ des véhicules

Nous considérons que les deux producteurs s'accordent sur l'affectation de leurs commandes aux véhicules et sur les différentes dates de production des commandes. La solution proposée pour cet exemple est réalisable mais n'est pas forcément optimale. Cette solution est la suivante. Les commandes $l_{1,1}$, $l_{2,2}$ et $l_{2,3}$ sont affectées aux véhicules k_1 et les commandes $l_{2,1}$, $l_{1,2}$ et $l_{1,3}$ sont affectées aux véhicules k_2. La séquence de production du producteur j_1 est $\{l_{1,1}, l_{1,2}, l_{1,3}\}$, la séquence de production du producteur j_2 est $\{l_{2,2}, l_{2,1}, l_{2,3}\}$. Les Figures 6.1 et 6.2 représentent un ordonnancement réalisable pour les commandes de chaque producteur. Sur ces figures, la date $D_{j,k}$ représente la date de passage véhicule k chez le producteur j. La Figure 6.3 présente le trajet des véhicules et l'évolution de leurs chargements. Dans cette figure, le trajet du véhicule k_1 est représenté par les flèches continues et le trajet du véhicule k_2 est représenté par les flèches composées de tirets.

La fonction objectif est composée des coûts de stockage des différentes commandes : coût de stockage initial pour toutes les commandes sauf $l_{1,2}$, coût de stockage en cours de production pour la commandes $l_{1,2}$ et coûts de stockage en fin de livraison pour les commandes $l_{1,2}$, $l_{2,1}$ et $l_{2,2}$.

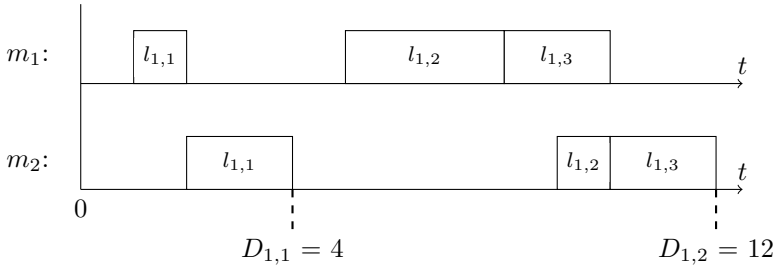

Fig. 6.1: Ordonnancement du producteur j_1

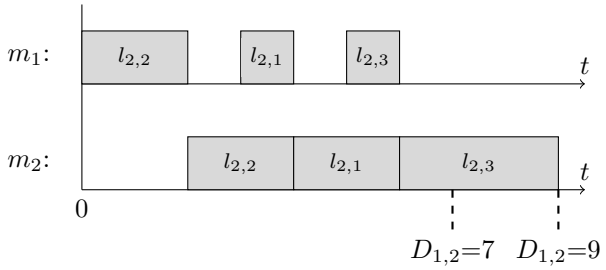

Fig. 6.2: Ordonnancement du producteur j_2

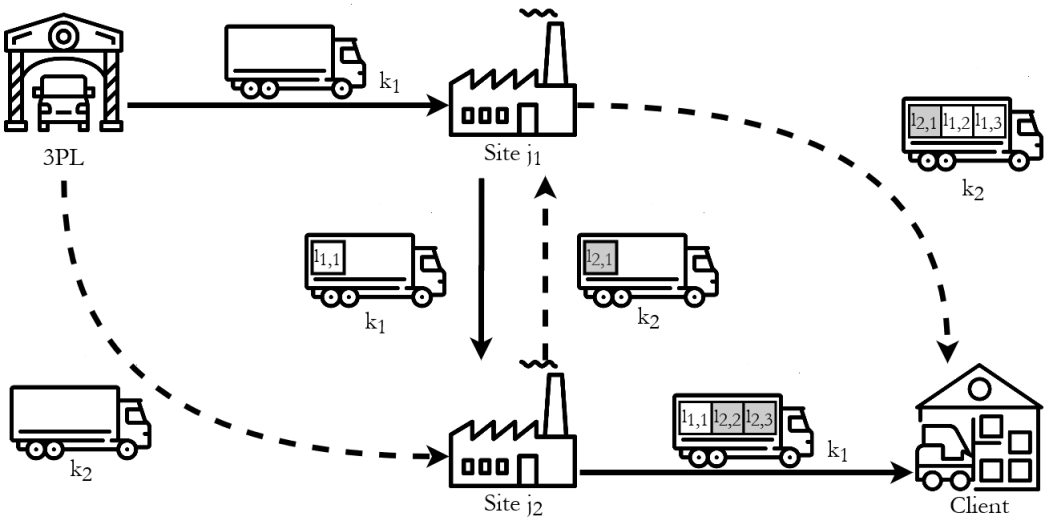

Fig. 6.3: Trajet et chargement des véhicules

6.1.2 Modélisations mathématique

Cette section introduit les notations, paramètres et variables utilisés. Elle présente la formulation linéaire en variables mixtes entières du problème décrit en Section 6.1.

Lors de la production, nous considérons que les index 0 et $o_j + 1$ représentent des commandes fictives en début et en fin de production du producteur j. L'index du client final se note $n + 1$.

Tab. 6.2: Résumé des notations

Ensemble	
n	Nombre de producteurs
J	Ensemble des producteurs ($J = \{1, \ldots, n\}$)
m	Nombre de machines pour chaque producteur
M	Ensemble des machines ($M = \{1, \ldots, m\}$)
o_j	Nombre de commandes du producteur j
K	Ensemble des véhicules
L_j	Ensemble des commandes du producteur j ($L_j = \{1, \ldots, o_j\}$)
K_j	Ensemble des véhicules visitant le producteur j, $\forall j \in J$
$K_{j,l}$	Ensemble des véhicules auquel la commande l du producteur j peut être affectée, $\forall j \in J,\ \forall l \in L_j$
	$d_{j,l}$ la date due de livraison de la commande l du producteur j, $\forall j \in J, \forall l \in L_j$
$L_{j,k}$	Ensemble des commandes du producteur j que le véhicule k est susceptible de livrer, $\forall j \in J,\ \forall k \in K_j$
Paramètres	
$p_{j,i,l}$	Temps d'exécution de la commande l du producteur j sur la machine i, $\forall j \in J, \forall i \in M, \forall l \in L_j$
$h_{j,l}^{START}$	Coût de stockage en début de production de la commande l du producteur j, $\forall j \in J, \forall l \in L_j$
$h_{j,i,l}^{WIP}$	Coût de stockage en cours de traitement de la commande l du producteur j entre la machine i et la machine $i+1$, $\forall j \in J, \forall l \in L_j,\ \forall i \in M \backslash m$
$h_{j,l}^{FIN}$	Coût de stockage en fin de production de la commande l du producteur j, $\forall j \in J, \forall l \in L_j$
$d_{j,l}$	La date dues de livraison de la commande l du producteur j, $\forall j \in J, \forall l \in L_j$
$w_{j,l}$	Poids de la commande l du producteur j, $\forall j \in J, \forall l \in L_j$
W	Capacité d'un véhicule
$D_{j,k}$	Date de visite du véhicule k au producteur j (fixée), $\forall j \in J,\ \forall k \in K_j$
$D_{n+1,k}$	Date de livraison du véhicule k au client $\forall k \in K_j$
Variables	
y_{j,l_1,l_2}	1 si la commande l_1 est produite juste avant la commande l_2 chez le producteur j, 0 sinon, $\forall (l_1, l_2) \in \{0, \ldots, l_j + 1\}^2$
$C_{j,i,l}$	Date de fin de la commande l du producteur j sur la machine i, $\forall j \in J, \forall i \in M$, $\forall l \in L_j$
$f_{j,l}$	Date de départ de la commande l du producteur j, $\forall j \in J, \forall l \in L_j$
$z_{j,l,k}$	1 si la commande l du producteur j est affectée au véhicule k, $\forall j \in J,\ \forall l \in L_j,\ \forall k \in K_{j,l}$
Expressions de coût	
IC	Coût d'inventaire

HV désigne une valeur arbitrairement grande

L'ensemble $K_{j,l}$ est limité pour chaque commande l du producteur j aux véhicules k dont la

date de livraison $D_{n+1,k}$ au client est inférieure ou égale à la date de livraison $d_{j,l}$ de la commande.

L'expression des coûts d'inventaire est identique à celle des chapitres précédents et rappelé dans l'expression (6.1).

$$IC = \sum_{j\in J}\sum_{l\in L_j}\left(C_{j,1,l}h_{j,l}^{START} + \sum_{i\in M/m}(C_{j,i+1,l} - p_{j,i+1,l} - C_{j,i,l})h_{j,l}^{WIP} + (f_{j,l} - C_{j,m,l})h_{j,l}^{FIN}\right) \tag{6.1}$$

Le modèle mathématique de ce problème, appelé Modèle 6.1 est le suivant :

$$\text{Minimise } IC \tag{6.2}$$

$$\sum_{l_2=1}^{o_j+1} y_{j,l_1,l_2} = 1 \qquad \forall j \in J, \forall l_1 \in \{0,\dots,o_j\} \tag{6.3}$$

$$\sum_{l_1=0}^{o_j} y_{j,l_1,l_2} = 1 \qquad \forall j \in J, \forall l_2 \in \{1,\dots,o_{j+1}\} \tag{6.4}$$

$$p_{j,1,l} \leq C_{j,1,l} \qquad \forall j \in J, \forall i \in M, \forall l \in L_j \tag{6.5}$$

$$C_{j,i,l} + p_{j,i+1,l} \leq C_{j,i+1,l} \qquad \forall j \in J, \forall i \in \{1,\dots,m-1\}, \forall l \in L_j \tag{6.6}$$

$$C_{j,i,l_1} + p_{i,l_2} - HV(1 - y_{j,l_1,l_2}) \leq C_{j,i,l_2} \qquad \forall j \in J, \forall l_1, l_2 \in L_j^2, \forall i \in M \tag{6.7}$$

$$\sum_{k\in K_{j,l}} z_{j,l,k} = 1 \qquad \forall j \in J, \forall l \in L_j \tag{6.8}$$

$$C_{j,m,l} \leq f_{j,l} \qquad \forall j \in J, \forall l \in L_j, \forall k \in K_{j,l} \tag{6.9}$$

$$f_{j,l} = \sum_{k\in K_{j,l}} z_{j,l,k}D_{j,k} \qquad \forall j \in J, \forall l \in L_j, \forall k \in K_{j,l} \tag{6.10}$$

$$\sum_{j\in J}\sum_{l\in L_{j,k}} z_{j,l,k}w_{j,l} \leq W \qquad \forall k \in K \tag{6.11}$$

Modèle 6.1: Modèle général

L'expression (6.2) correspond à la fonction objectif de notre modèle, la minimisation des coûts d'inventaire. Les contraintes (6.3) (resp. (6.4)) imposent que chaque commande (à l'exception de la commande fictive d'index $o_j + 1$ (resp. 0)) possède un successeur (resp. un prédécesseur) dans la séquence de production du producteur j. Les contraintes (6.5) bornent la date de fin minimum de la première tâche de chaque commande. Les contraintes (6.6) garantissent la bonne succession des tâches lors de la production d'une commande. Les contraintes (6.7) garantissent la date de fin des tâches en fonction de la séquence de production choisie par le producteur j. Les contraintes (6.8) assurent que chaque commande est affectée à exactement un véhicule (parmi ceux auxquels elle peut être affectée). Les contraintes (6.9) imposent que la date de fin de production de chaque commande est avant sa date de départ. Les contraintes (6.10) imposent que la date de départ de chaque commande correspond à la date de passage du véhicule auquel elle est affectée. Les contraintes (6.11) assurent que la capacité maximale des véhicules est respectée.

6.2 Faisabilité et minimisation des coûts d'un ensemble de séquences de production

Dans cette section, les méthodes de résolution proposées combinent deux phases. La première phase génère un ensemble de séquences de production pour les différents producteurs. À ce niveau, les dates de production ne sont pas encore déterminées. La seconde phase évalue la faisabilité de cet ensemble de séquences en proposant une affectation des commandes aux véhicules.

Cette section propose une méthode vérifiant exclusivement la faisabilité d'une solution et deux méthodes proposant, si une affectation réalisable existe, des dates de production des commandes minimisant les coûts d'inventaire de manière exacte ou de manière approchée. Ces méthodes sont basées principalement sur la résolution de différents modèles mathématiques.

6.2.1 Faisabilité d'un ensemble de séquences de production

La méthode décrite dans cette section vise uniquement à tester la faisabilité d'un ensemble de séquences de production fixées pour les différents producteurs. À partir d'une séquence de production donnée pour un producteur, les dates d'exécution des tâches des commandes sur chaque machine sont déterminées de manière à ordonnancer au plus tôt chaque tâche. Les variables $C_{j,m,l}$ indiquant les dates de fin de production des commandes sont alors fixées.

Le Modèle 6.2 trouve une affectation (si elle existe) des commandes aux véhicules respectant les capacités, les dates de départ des véhicules et les dates dues de livraison des commandes. Sinon, l'ensemble des séquences de production est irréalisable. Ce modèle est composé des contraintes d'affectation (impliquant les variables $z_{j,l,k}$) du Modèle 6.1.

(6.8), (6.9), (6.10),(6.11)

Modèle 6.2: Modèle de contrôle de faisabilité

6.2.2 Minimisation des coûts d'un ensemble de séquences de production

Si une solution réalisable existe, les deux méthodes suivantes retournent une solution proposant une affectation des commandes aux véhicules ainsi que des dates de production des commandes minimisant les coûts d'inventaire.

Cette première méthode retourne la solution réalisable minimisant les coûts d'inventaire de l'ensemble des producteurs. Cette méthode consiste à résoudre le Modèle 6.3. Sans perte de généralité, les commandes de ce modèle sont ré-indexées pour correspondre à la séquence de production de chaque producteur. Ainsi la variable $C_{j,i,l}$ correspond à la date de fin de production de la l^{eme} commande produite par le producteur j sur la machine i.

La plupart de contraintes de ce modèle sont reprises du Modèle 6.1. Les contraintes (6.12) sont l'adaptation des contraintes (6.7) pour une séquence de production fixée chez chaque producteur.

La seconde méthode est proposée car elle fait appel à un modèle mathématique plus rapide à résoudre que le Modèle 6.3. Dans le cas d'une méthode de résolution, une telle méthode de minimisation permet une exploration plus rapide des solutions et peut permettre de trouver de meilleures solutions dans des laps de temps courts. Cette seconde méthode retourne la solution

$$
\begin{aligned}
&\text{Minimise } IC \\
&(6.5), (6.6) \\
&C_{j,i,l_1} + p_{i,l_1+1} \leq C_{j,i,l_1+1} \qquad \forall j \in J, \forall l_1 \in \{1, \ldots, o_j - 1\}, \forall i \in M \qquad (6.12) \\
&(6.8), (6.9), (6.10), (6.11)
\end{aligned}
$$

Modèle 6.3: Évaluation exacte d'un ensemble de séquences de production fixé

réalisable minimisant uniquement les coûts d'inventaire en fin de production de l'ensemble des producteurs. Cette méthode est donc une heuristique.

Cette méthode consiste à résoudre un modèle identique au Modèle 6.3 à l'exception de la fonction objectif qui minimise l'expression (6.13) ci-dessous :

$$\sum_{j \in J} \sum_{l \in L_j} (f_{j,l} - C_{j,m,l}) h_{j,l}^{FIN} \tag{6.13}$$

Une fois l'affectation déterminée, la méthode décale la production le plus tard possible en respectant les dates de départ des véhicules afin de réduire les coûts d'inventaire sur les machines 1 à $m - 1$.

Les méthodes proposées dans la section suivante pour résoudre le problème utilisent l'une des deux méthodes précédentes pour minimiser les coûts d'inventaire des ensembles de séquences de production à évaluer. La méthode utilisée dépend du paramètre $\tau^{eval} \in \{OPT, FIN\}$.

$\tau^{eval} = OPT$ correspond de la méthode d'évaluation qui minimise les coûts d'inventaire globaux. $\tau^{eval} = FIN$ correspond à la méthode qui minimise les coûts d'inventaire en fin de production.

6.3 Méthodes de résolution

Cette section présente un algorithme glouton pour générer des solutions initiales. Cet algorithme sera réutilisé dans la génération d'instances réalisables. Puis, un algorithme génétique est présenté pour résoudre le problème général.

6.3.1 Algorithme glouton

Cet algorithme glouton construit des solutions en utilisant des règles simples et de bon sens. Le but de cette heuristique est de produire une solution réalisable de qualité correcte.

Cet algorithme ordonnance les commandes d'après l'ordre croissant de leurs dates dues de livraison. Si plusieurs commandes possèdent la même date due de livraison, leur ordre de préparation est déterminé par l'heuristique NEH (minimisation de la date de fin de production, Section 5). Dans un premier temps, les coûts d'inventaire ne sont pas pris en compte, la production est donc planifiée au plus tôt.

Une affectation réalisable pour les séquences de production fixées est donnée une des méthodes de la Section 6.2. Si une telle affectation n'existe pas, l'instance concernée est jugée irréalisable.

Pour finir, si toutes les dates de production n'ont pas été déterminées par la méthode d'évaluation, la production est décalée au plus tard afin de réduire les coûts d'inventaire. Ce décalage

se fait en respectant l'ordre de préparation des commandes préétabli ainsi que les dates de départ des véhicules auxquels les commandes sont affectées.

6.3.2 Algorithme génétique

La population initiale est composée de solutions générées aléatoirement d'après un processus similaire à celui décrit dans la Section 4.3.2.1. La séquence de production est générée itérativement soit en commençant par ordonnancer des commandes en début de production, soit en commençant par ordonnancer des commandes en fin de production. Chaque nouvelle commande a ordonnancer est tirée aléatoirement dans une liste restreinte de solutions. Cette liste est constituée sur la base des dates dues de livraison des commandes. Pour un ordonnancement par le début, les commandes avec les dates due les plus urgentes sont d'abord intégrées dans la liste. Pour un ordonnancement par la fin, ce sont les commandes avec les dates dues les plus tardives qui sont d'abord intégrées. Le paramétrage choisi pour la génération des séquences de production de la population initiale est discuté dans la section expérimentation. Une solution ainsi créée est évaluée de manière exacte ou heuristique (d'après le paramètre τ^{eval}). Aucun couple de solutions dans la population n'a la même fonction objectif. Toute solution initiale déjà présente dans la population (c'est-à-dire avec la même fonction objectif qu'une autre) ou irréalisable n'est pas considérée.

Les nouveaux enfants sont générés par couple à partir du croisement d'un couple de solution parent sélectionné à la suite de deux tournois binaires. Pour procéder au croisement des deux parents, la séquence de production de chaque producteur des deux enfants est déterminée selon une des trois méthodes suivantes :

1. La séquence de l'enfant 1 est identique à celle du parent 1 et la séquence de l'enfant 2 est identique à celle du parent 2.
2. La séquence de l'enfant 1 est identique à celle du parent 2 et la séquence de l'enfant 2 est identique à celle du parent 1.
3. Les séquences des enfants 1 et 2 sont issues d'un croisement de celle des deux parents d'après la méthode décrite Section 4.3.3.2.

L'opération de mutation consiste à choisir aléatoirement un producteur et une commande dans sa production. Cette commande échange sa place dans la production avec la commande produite directement après elle. Cette opération est effectuée 2 fois pour les instances à 25 commandes et 4 fois pour les instances à 50 commandes.

La gestion des populations est similaire à celle décrite en Section 4.3.4.

Exemple : La paragraphe suivant illustre une opération de croisement entre deux solutions parents (parent 1 et 2) aboutissant à la création du couple d'enfants 1 et 2. L'instance considérée est composée de 3 producteurs j_1, j_2 et j_3 devant livrer respectivement 4, 4 et 6 commandes au clients. Lors du tirage aléatoire, les producteurs j_1, j_2 et j_3 sont placés respectivement dans la première, deuxième et troisième catégorie. Le producteur j_3 va donc subir un croisement de sa séquence de production, les indices tirés pour ce croisement sont 3 et 5. Le Tableau 6.3 présente l'ensemble des séquences de production initiales des deux parents ainsi que l'ensemble des séquences obtenues pour chaque enfant. En gras sont représentées les portions de séquences issues du parent 1 et en italique les portions issues du parent 2.

6.4 Génération et faisabilité des instances

Cette section décrit la méthode de génération des instances pour ce problème.

	Producteur j_1	Producteur j_2	Producteur j_3
Parent 1	{**1, 2, 3, 4**}	{**3, 1, 2, 4**}	{**1, 2, 3, 4, 5, 6**}
Parent 2	{*1, 3, 2, 4*}	{*2, 1, 4,* 3}	{*3, 2, 1, 4, 6, 5*}
Enfant 1	{**1, 2, 3, 4**}	{*2, 1, 4, 3*}	{*2, 1,* **3, 4, 5,** *6*}
Enfant 2	{*1, 3, 2, 4*}	{**3, 1, 2, 4**}	{**2, 3,** *1, 4, 6,* **5**}

Tab. 6.3: Exemple d'opération de croisement

Pour définir ces instances, nous nous inspirons des cas concrets décris en introduction (Section 1.6.2) de cette thèse. Dans le cas d'un approvisionnement de *cross-dock* de grande surface, les commandes sont attendues pour des créneaux précis. Nous considérons également que chaque tournée arrive au dépôt du client au début d'un de ces créneaux. Ainsi, les valeurs possibles des dates dues de livraison des commandes se restreignent aux dates de début des créneaux. Nous considérons que les différents itinéraires de livraison sont établis préalablement par les producteurs et le 3PL.

Les dates de passage des véhicules sur les différents sites sont calculées à partir des dates de livraison au client. Pour calculer la date de passage sur le site d'un producteur, nous considérons la date de passage du site suivant moins le temps de transport suffisant pour relier les deux sites.

Afin de générer des instances difficiles, nous considérons qu'un producteur peut produire (une fois la production amorcée) une commande par créneau en moyenne. Le nombre de commandes attendus par le client correspond quasiment à la capacité de production maximale de l'ensemble de producteurs. Ainsi, générer une instance avec deux fois plus de commandes pour le même nombre de producteurs implique des livraison étalées sur deux fois plus de créneaux et nécessitant deux fois plus de tournées.

6.4.1 Paramétrage

Pour générer les instances de ce problème, nous considérons que 25 commandes doivent être livrées par 6 tournées. Chaque producteur est doté de la même chaîne de production (*flow-shop* de permutation à 5 machines). La capacité de production totale variant avec le nombre de producteurs, le nombre de créneaux de livraison par instance varie avec le nombres de producteurs. Plus les producteurs sont nombreux, plus ils auront une capacité de production importante et ainsi, plus le nombre de créneaux de livraison sera restreint. Nous considérons que la taille d'une instance correspond à son nombre de commandes $o \in \{25, 50\}$ (soit 6 ou 12 tournées). Nous considérons également des instances avec un nombre de producteurs $j \in \{3, 6, 12\}$. Pour chaque instance, chaque producteur se voit attribuer entre $\lfloor o/n \rfloor$ et $\lceil o/n \rceil$ commandes à produire soit prêt de 8, 6 et 4 commandes pour 3, 6 et 12 producteurs respectivement pour des instances de 25 commandes. Ainsi pour 25 commandes, le client prévoit 8 créneaux de livraisons pour 3 producteurs, 4 créneaux pour 6 producteurs et 2 créneaux pour 12 producteurs. Ce nombre de créneaux est doublé pour les instances à 50 commandes. La date due de livraison de chaque commande est tirée aléatoirement parmi l'ensemble des dates de débuts de créneaux disponibles. La date de démarrage du premier créneau se situe à la date D_{min}. D_{min} est donc la première date due de livraison pouvant être attribuée à une commande et donc la première date d'arrivée d'une tournée chez le client. Cette date doit être assez élevée pour permettre la production et la livraison sans retard des commandes les plus urgentes. Nous discuterons la valeur de D_{min} dans la Section 6.4.2 suivante. Sans perte de généralité, nous considérons que la taille d'un créneau vaut 10 unités de temps. Nous considérons également que le temps de réalisation d'une tâche $p_{j,i,l}$ sur une machine est tiré aléatoirement entre 7 et 12 unités de temps. (Ces valeurs permettent d'approcher un taux de production par

producteur d'une commande toutes les 10 unités de temps). Les coûts d'inventaire d'une commande l sont générés de telle sorte qu'ils augmentent de 1 ou 2 unités d'une machine à l'autre : le coût d'inventaire initial $h_{j,l}^{START}$ est tiré aléatoirement dans $[1, 2]$, le coût d'inventaire $h_{j,1,l}^{WIP}$ est tiré aléatoirement dans $h_{j,l}^{START} + [1, 2]$, $h_{j,i+1,l}^{WIP}$ est tiré aléatoirement dans $h_{j,i,l}^{WIP} + [1, 2]$ et le coût d'inventaire final $h_{j,l}^{FIN}$ est tiré aléatoirement dans $h_{j,m,l}^{WIP} + [1, 2]$.

Concernant la livraison, nous considérons que le poids d'une commande $w_{j,l}$ est tiré aléatoirement entre 4 et 6. La capacité des véhicules est également discutée dans la Section 6.4.2. Les différents sites (producteurs, 3PL et client) sont positionnés dans un carré de 10 sur 10 unités de temps. La position de ces sites et les itinéraires des tournées restent identiques pour toutes les instances avec le même nombre de producteurs. Les dates de visite des véhicules sur les sites des producteurs et du client sont identiques pour les différents types d'instances et dépendes du nombre de producteurs (3 producteurs : Tableau 6.4, 6 producteurs : Tableau 6.5, 12 producteurs : Tableau 6.6). k indique l'indice d'une tournée, j l'indice du site producteurs visité. Le site client est livré à la fin de la tournée. Les instances à 25 commandes sont composées 6 tournées, les instances à 50 commandes sont composées de 12 tournées. Les tournées des instances à 25 commandes sont identiques aux 6 premières tournées de la première période des instances à 50 commandes. D_{min} désigne la première date de visite d'une tournée au client.

6.4.2 Test de faisabilité des instances

Le problème étudié pose un problème de faisabilité qui dépend directement de la construction des instances. Cette section cherche à évaluer la proportion d'instances réalisables en fonction de la première date due de livraison D_{min} des commandes et de la capacité des véhicules W.

Pour évaluer la faisabilité d'une instance, nous générons une solution où seul l'ensemble des séquences de production est fixé. Cet ensemble est généré à partir de la première partie de l'algorithme glouton (Section 6.3.1). Pour rappel, les commandes sont ordonnancées d'après l'ordre croissant de leurs dates dues de livraison. La faisabilité de l'instance est ensuite évaluée grâce à la méthode présentée Section 6.2.2.

6.4.2.1 Paramètres et jeux de test

Le problème étudié considère qu'entre la date 0 et sa date de livraison réelle au client, chaque commande doit être préparée et chargée dans un véhicule pour être livrée au client. Ce problème fait apparaître deux causes d'infaisabilité : 1) Pour un producteur donné, il n'existe pas d'ordonnancement permettant de produire toutes les commandes avant la date de passage d'un véhicule susceptible de les livrer à l'heure. 2) Du fait de la capacité des véhicules, il n'existe pas d'affectation réalisable des commandes à ces véhicules.

Ces deux critères sont liés à deux valeurs. D_{min} (la première date due de livraison) doit laisser un délais permettant la production et la livraison des premières commandes chez les différents producteurs et éventuellement de donner un délai supplémentaire aux producteurs ayant plus de commandes. En pratique, une augmentation de D_{min} n'a pas d'impact sur le 3PL. On considère que cette augmentation correspond à un démarrage des préparations des commandes plus tôt par rapport à la première date de livraison (le client doit seulement passer commande plus tôt). Ensuite, la capacité des véhicules W doit permettre de charger l'ensemble des commandes.

Une instance est construite pour une valeur de D_{min} et W. Cette instance est considérée comme réalisable si l'algorithme glouton (Section 6.3.1) permet d'y trouver une solution réalisable.

Des ensembles d'instances ont été générés pour un nombre de commandes $o \in \{25, 50\}$ et un nombre de producteurs $n \in \{3, 6, 12\}$. Pour chacun de ces ensembles, 50 instances sont gé-

	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7	k = 8	k = 9	k = 10	k = 11	k = 12
j_1	85.0	85.9	105.0	115.9	135.0	145.9	165.0	165.9	185.0	195.9	215.0	225.9
j_2	83.6	87.3	103.6	117.3	133.6	147.3	163.6	167.3	183.6	197.3	213.6	227.3
j_3	76.6	94.3	96.6	124.3	126.6	154.3	156.6	174.3	176.6	204.3	206.6	234.3
Client	90.0	100.0	110.0	130.0	140.0	160.0	170.0	180.0	190.0	210.0	220.0	240.0

Tab. 6.4: Date de visite des véhicules pour 3 producteurs et $D_{min} = 90$

	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7	k = 8	k = 9	k = 10	k = 11	k = 12
j_1	85.5	-	87.5	-	105.5	-	125.5	-	127.5	-	145.5	-
j_2	81.1	-	92.0	-	101.1	-	121.1	-	132.0	-	141.1	-
j_3	78.1	-	95.0	-	98.1	-	118.1	-	135.0	-	138.1	-
j_4	-	79.1	-	97.2	-	99.1	-	119.1	-	137.2	-	139.1
j_5	-	84.2	-	92.1	-	104.2	-	124.2	-	132.1	-	144.2
j_6	-	86.4	-	89.8	-	106.4	-	126.4	-	129.8	-	146.4
Client	90	90	100	100	110	110	130	130	140	140	150	150

Tab. 6.5: Date de visite des véhicules pour 6 producteurs et $D_{min} = 90$

	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7	k = 8	k = 9	k = 10	k = 11	k = 12
j_1	57.6	-	70.5	-	-	75.7	77.6	-	90.5	-	-	95.7
j_2	62.1	-	75.0	-	-	80.2	82.1	-	95.0	-	-	100.2
j_3	66.2	-	-	69.5	-	84.3	86.2	-	-	89.5	-	104.3
j_4	70.4	-	-	73.8	-	88.6	90.4	-	-	93.8	-	108.6
j_5	72.7	61.2	-	76.0	-	-	92.7	81.2	-	96.0	-	-
j_6	74.9	63.4	-	78.2	-	-	94.9	83.4	-	98.2	-	-
j_7	-	66.6	-	81.4	75.0	-	-	86.6	-	101.4	95.0	-
j_8	-	70.2	-	85.0	78.6	-	-	90.2	-	105.0	98.6	-
j_9	-	73.3	60.1	-	81.7	-	-	93.3	80.1	-	101.7	-
j_{10}	-	74.3	61.1	-	82.7	-	-	94.3	81.1	-	102.7	-
j_{11}	-	-	63.3	-	85.0	68.5	-	-	83.3	-	105	88.5
j_{12}	-	-	66.9	-	88.6	72.1	-	-	86.9	-	108.6	92.1
Client	80	80	80	90	90	90	100	100	100	110	110	110

Tab. 6.6: Date de visite des véhicules pour 12 producteurs et $D_{min} = 80$

nérées. Pour une valeur D_{min} de 50 jusqu'à 200 par pas de 10 et une capacité des véhicules $W \in \{25, 30, 35, 40\}$, la faisabilité de ces instances est testée. Pour information, à $W = 25$ le remplissage des véhicules est de 83.3% en moyenne.

Le choix du paramétrage (D_{min}, W) pour chaque type d'instance sera analysé après la présentation des résultats.

6.4.3 Résultats

6.4.3.1 Impact de la capacité des véhicules W

Dans un premier temps, l'impact de la capacité W des véhicules est testé grâce à une valeur de D_{min} arbitrairement grande (200) permettant à tous les producteurs de réaliser leurs commandes à temps. Dans ce contexte, nous considérons qu'une instance n'est pas réalisable si et seulement si le problème d'affectation des commandes aux véhicules n'a pas de solution. Pour chaque valeur de W, le nombre d'instances réalisables sur 50 est présenté dans les Tableaux 6.7, 6.8, 6.9 et 6.10.

o \ n	3	6	12
25	42	40	39
50	50	46	49

Tab. 6.7: Faisabilité : $W = 25$

o \ n	3	6	12
25	49	49	49
50	50	49	50

Tab. 6.8: Faisabilité : $W = 30$

o \ n	3	6	12
25	49	50	50
50	50	50	50

Tab. 6.9: Faisabilité : $W = 35$

o \ n	3	6	12
25	50	50	50
50	50	50	50

Tab. 6.10: Faisabilité : $W = 40$

D'après ces résultats, on constate que la capacité $W \geq 30$ ne pose quasiment pas de problème d'infaisabilité pour tous les types d'instances (au moins 49 instances sur 50 réalisables). Le cas $o = 25$ et $W = 25$ contient la plupart des cas non réalisables (entre 8 et 11 sur 50).

6.4.3.2 Impact de la première date de livraison D_{min}

Il est montré dans la section précédente que certaines instances ne sont jamais réalisables même pour une valeurs de D_{min} arbitrairement grandes. Cette section présente à partir de quelle valeur de D_{min} les autres instances deviennent réalisables. Le Tableau 6.11 présente ces résultats. Les colonnes n, o et W indiquent respectivement le nombre de producteurs et le nombre de commandes des instances testées ainsi que la capacité des véhicules utilisés. La colonne $\#F$ rappelle le nombre d'instances de l'ensemble testé qui sont réalisables pour $D_{min} = 200$. Les colonnes suivantes indiquent, pour des valeurs de D_{min} de 60 à 120, le pourcentage d'instances réalisables. Ce pourcentage se calcule sur la base du nombre d'instances indiqué par $\#F$. Pour une plus grande clarté, les valeurs supérieures ou égales à 90% ont été mises en gras et les autres valeurs supérieures à 0% en italique.

n	o	W	$\#F$	D_{min} = 60	D_{min} = 70	D_{min} = 80	D_{min} = 90	D_{min} = 100	D_{min} = 110	D_{min} = 120
3	25	25	42	0%	0%	*14%*	*61%*	**98%**	**100%**	**100%**
		30	49	0%	0%	*12%*	*57%*	**92%**	**98%**	**100%**
		35	49	0%	0%	*12%*	*57%*	**92%**	**98%**	**100%**
		40	50	0%	0%	*12%*	*56%*	**92%**	**98%**	**100%**
	50	25	50	0%	0%	*24%*	**94%**	**100%**	**100%**	**100%**
		30	50	0%	0%	*36%*	**94%**	**100%**	**100%**	**100%**
		35	50	0%	0%	*36%*	**94%**	**100%**	**100%**	**100%**
		40	50	0%	0%	*36%*	**94%**	**100%**	**100%**	**100%**
6	25	25	40	0%	0%	*50%*	**100%**	**100%**	**100%**	**100%**
		30	49	0%	0%	*43%*	**100%**	**100%**	**100%**	**100%**
		35	50	0%	0%	*42%*	**100%**	**100%**	**100%**	**100%**
		40	50	0%	0%	*42%*	**100%**	**100%**	**100%**	**100%**
	50	25	46	0%	0%	*13%*	*89%*	**100%**	**100%**	**100%**
		30	49	0%	0%	*12%*	*88%*	**100%**	**100%**	**100%**
		35	50	0%	0%	*12%*	*86%*	**100%**	**100%**	**100%**
		40	50	0%	0%	*12%*	*86%*	**100%**	**100%**	**100%**
12	25	25	39	0%	*23%*	**92%**	**100%**	**100%**	**100%**	**100%**
		30	49	0%	*30%*	**90%**	**100%**	**100%**	**100%**	**100%**
		35	50	0%	*30%*	**90%**	**100%**	**100%**	**100%**	**100%**
		40	50	0%	*30%*	**90%**	**100%**	**100%**	**100%**	**100%**
	50	25	49	0%	*8%*	*84%*	**100%**	**100%**	**100%**	**100%**
		30	50	0%	*8%*	*84%*	**100%**	**100%**	**100%**	**100%**
		35	50	0%	*8%*	*84%*	**100%**	**100%**	**100%**	**100%**
		40	50	0%	*8%*	*84%*	**100%**	**100%**	**100%**	**100%**

Tab. 6.11: Impact de D_{min} sur la faisabilité

Aucune instance n'est réalisable pour $D_{min} = 60$. L'intervalle de temps entre 0 et $D_{min} \leq 60$ ne permet pas de produire et transporter une commande. À part pour quelques instances à 12 producteurs, toutes les instances considérées ne sont pas réalisables pour $D_{min} = 70$. Pour $D_{min} = 80$, les résultats varient entre les ensembles, les instances à 3 et 6 producteurs n'ont pas plus de la moitié de leurs instances réalisables alors que neuf dixièmes des instances sont réalisables pour les instances à 12 producteurs. À $D_{min} = 90$, les instances des ensembles avec $n = 6$, $n = 12$ deviennent en grande partie réalisables (86% et plus). Pour les instances avec $n = 3$, plus de la moitié des instances sont maintenant réalisables pour $o = 25$ et 94% des instances sont réalisables $o = 50$. Pour $D_{min} = 100$, tous les ensembles ont au moins 92% d'instances réalisables. Pour D_{min} plus grand, quasiment toutes les instances considérées sont réalisables pour tous les ensembles avec toutes les instances réalisables dès $D_{min} = 120$.

En conclusion, afin d'évaluer les méthodes de résolution sur des instances avec des délais de production "serrés", nous utiliserons dans la suite des expérimentations de véhicules de capacité 25 ($W = 25$). Les dates de livraison D_{min} sont fixées à 90 pour les instances à 3, 6 producteurs et 80 pour les instances à 12 producteurs.

6.5 Expérimentations

Les résultats présentés dans cette section sont réalisés sur un ordinateur équipé d'un processeur Intel Core i7-7820HQ et de 16 Go RAM. Le code a été réalisé en C++ sous Visual Studio et compilé pour Windows 10. Le solveur commercial utilisé est Cplex 12.7.1.

6.5.1 Comparaison des méthodes de minimisation

Les performances des deux méthodes d'évaluation (Section 6.2) sont mesurées pour tous les types d'instance (10 instances de chaque type) sur la base de la séquence de production fournie par l'algorithme glouton (Section 6.3.1). Pour chaque méthode d'évaluation et chaque type d'instance, le Tableau 6.12 indique la durée moyenne d'évaluation d'une solution.

n	o	Durée d'évaluation	
		$\tau^{eval} = OPT$	$\tau^{eval} = FIN$
3	25	45.6 ms	24.7 ms
	50	413.9 ms	335.7 ms
6	25	22.2 ms	19.0 ms
	50	99.1 ms	60.6 ms
12	25	24.2 ms	17.5 ms
	50	135.0 ms	71.1 ms

Tab. 6.12: Durée moyenne d'évaluation d'une solution par méthode et type d'instance

Ce tableau montre plusieurs résultats. Pour les deux méthodes et tous les nombres de producteurs, le durée de résolution augmente avec le nombre de commandes. Les méthodes voient leurs temps de résolution baisser entre les instances à 3 producteurs et les instances à 6 et 12 producteurs. Cette diminution est due à la diminution du nombre de commandes par producteur et ainsi à la diminution de l'horizon de planification.

La méthode d'évaluation associée à $\tau^{eval} = OPT$ étant optimale pour des séquences de production données, ces performances sont prises comme référence pour évaluer la répartition des coûts des solutions trouvées par l'algorithme glouton. Le Tableau 6.13 est séparé en trois types de colonnes. La colonne "Total" indique la valeur moyenne des fonctions objectifs pour chaque type d'instance. Les colonnes "Répartition en pourcentage" indiquent le pourcentage occupé par chaque type de coût d'inventaire (IC^{START} : stockage avant production, IC^{WIP} : stockage en cours de production, IC^{FIN} : stockage après production). Les colonnes "Coûts par commandes" indiquent pour chacun des coûts sa valeur moyenne par commande.

n	o	Total	Répartition en pourcentage			Répartition moyenne par commande		
			IC^{START}	IC^{WIP}	IC^{FIN}	IC^{START}	IC^{WIP}	IC^{FIN}
3	25	3795.4	51.1%	11.4%	37.6%	77.4	17.2	57.2
	50	10374.3	65.3%	9.7%	25.0%	135.4	20.3	51.8
6	25	2967.9	43.5%	10.4%	46.1%	51.5	12.4	54.8
	50	7405.9	51.3%	12.6%	36.1%	75.9	18.7	53.5
12	25	1212.9	70.5%	6.1%	23.4%	33.8	3.1	11.6
	50	3378.3	69.9%	10.9%	19.2%	47.2	7.4	13.0

Tab. 6.13: Réparation des coûts d'inventaire en pourcentage pour $\tau^{eval} = OPT$ par instance

On peut d'abord noter que le coût d'inventaire par commande en fin de production reste constant pour les instances avec le même nombre de producteurs. Ce résultat s'explique par le

maintient du nombre de commandes par véhicule. En effet le nombre de véhicules est corrélé à la taille des instances. Le stockage en cours de production par instance augmente légèrement avec la taille des instances, mais reste toujours avec un ratio faible par rapport au total (moins de 13%). Enfin le stockage en début de production, qui représente près de la moitié des coûts d'inventaire totaux pour 25 commandes, augmente de manière importante (multiplié par une valeur entre 1.5 et 2) pour 50 commandes. Cette augmentation s'explique par l'allongement de l'horizon de production et explique entre autre la diminution en pourcentage des coûts d'inventaire en fin de production (alors que ce coût reste constant par commande). Les coûts de stockage avant production augmentent fortement avec le nombre de commandes par instance.

Enfin, le Tableau 6.14 fait une comparaison des valeurs des fonctions de coûts obtenues par la méthode FIN. Les colonnes "Total" indiquent l'écart à la solution exacte. Les colonnes suivantes indiquent l'écart moyen pour chaque type de coût avec la solution exacte.

n	l	Écart	IC^{START}	IC^{WIP}	IC^{FIN}
3	25	1.3%	1.8%	18.6%	-4.1%
	50	3.4%	1.4%	49.2%	-8.8%
6	25	0.9%	0.6%	18.5%	-1.4%
	50	1.8%	1.7%	29.3%	-7.3%
12	25	0.5%	1.8%	0.8%	-4.0%
	50	3.9%	3.8%	46.4%	-18.0%

Tab. 6.14: Écart moyen entre les coûts de la méthode $\tau^{eval} = OPT$ et de la méthode $\tau^{eval} = FIN$

La méthode associée à $\tau^{eval} = FIN$ affiche des écart à l'optimal ne dépassant pas 4%. La méthode $\tau^{eval} = FIN$ minimise les coûts d'inventaire en fin de production et ses coûts sont effectivement réduits entre -4% et -18% par rapport à l'optimal. Cependant, l'écart entre les deux méthodes se forme pour les coûts en cours de production (50%). En effet, ces coûts ne sont jamais considéré directement par la méthode $\tau^{eval} = FIN$ lors de l'affectation des commandes aux véhicules.

Ces résultats montrent que chacune des deux méthodes d'évaluation constitue un compromis entre la qualité de la solution trouvée et la durée de la résolution. La méthode optimale (associée à $\tau^{eval} = OPT$) fournit les meilleurs résultats mais a une durée de résolution plus longue. La méthode associée à $\tau^{eval} = FIN$ a des temps de résolution plus faibles que la méthode exacte tout en présentant des écarts raisonnables (pas plus de 4%).

6.5.2 Paramétrage de la méthode de génération de la séquence de production

Une heuristique semi-aléatoire est utilisée pour initialiser la population de l'algorithme génétique. Les séquences de production ainsi générées doivent être réalisables lorsque les commandes des différents producteurs sont ordonnancées au plus tôt. La faisabilité de la séquence est alors vérifiée à l'aide du Modèle 6.2. Dans ce chapitre l'heuristique est soumise à deux paramètres : $s^{point} \in \{Début, Fin\}$ si la séquence de production est définie par le début ($Début$) ou par la fin (Fin) et $r^{size} \in \{2, 3, 4\}$ définit la taille de la liste restreinte dans laquelle la prochaine commande ordonnancée est tirée.

Pour tester chacun de ces différents paramétrages, l'heuristique doit générer 100 séquences de production pour chacune des instances de différents types définis plus haut. La faisabilité de chacune de ces séquences est évaluée et le nombre moyen de séquences réalisables par type d'instance et par méthode est reporté dans le Tableau 6.15.

n	o	$r^{size} = 2$		$r^{size} = 3$		$r^{size} = 4$	
		Début	*Fin*	*Début*	*Fin*	*Début*	*Fin*
3	25	20	45.1	2.7	13	0.4	2.3
	50	58.4	91.8	20.4	72.4	5.1	48.9
6	25	58.3	77.3	36.5	57.5	31.2	34.8
	50	40.8	89.9	8	61.9	1.3	22.5
12	25	71	78.6	63	63.9	61.5	62.8
	50	9.7	50.7	1.7	7.1	0.8	1.3

Tab. 6.15: Nombre de solutions réalisables obtenues sur 100 solutions générées par l'heuristique semi-aléatoire

Ce tableau montre d'abord que les séquences de production générées par le début ont toujours moins de chance d'être réalisables (jusqu'à 50 solutions réalisables d'écart entre les deux paramétrages). Ce résultat s'explique par le fait que les commandes les plus urgentes peuvent facilement rendre la solution irréalisable si elles ne sont pas placées au début de la production. Or, lorsqu'une séquence est générée par le début, il existe une probabilité que la commande la plus urgente soit ordonnancée après de nombreuses autres commandes. Cependant, si la séquence est générée par la fin, cette commande sera obligatoirement ordonnancée dans les r^{size} premières commandes. C'est pourquoi le fait de commencer la séquence de production par la fin génère plus de solutions réalisables.

Ensuite, le nombre de solutions réalisables diminue rapidement avec l'augmentation de la taille de la liste restreinte. Dès $r^{size} = 4$, pour des commandes de 12 producteurs et 50 commandes, moins de 2 solutions réalisables sont générées en moyenne pour 100 solutions générées. Avec de tels résultats, pour certaines instances, il n'est pas possible de générer la population initiale d'un algorithme génétique avant la fin du temps de résolution qui lui est alloué. En effet, toute solution irréalisable est régénérée jusqu'à ce que la population initiale atteigne sa taille définie en paramètre.

Pour éviter une telle situation, le paramétrage utilisé pour l'heuristique semi-aléatoire dans la suite des expérimentations est $s^{point} = Fin$ et $r^{size} = 3$.

6.5.3 Résultat de l'algorithme génétique

L'algorithme génétique décrit en Section 6.3.2 est testé dans cette section. Le paramètre principal concerne la méthode utilisée pour évaluer une nouvelle solution $\tau^{eval} \in \{FIN, OPT\}$. Pour les autres paramètres, la taille de la population est fixée à 20, la probabilité de mutation est fixée à 0.5 et le paramétrage de l'heuristique servant à générer la population initiale est déterminé d'après les résultats de la section précédente ($s^{point} = FIN$ et $r^{size} = 3$). Pour chaque type d'instances, un ensemble de 10 instances est testé. La durée de résolution est fixée à 2 minutes 30 secondes pour les instances à 25 commandes et à 5 minutes pour les instances à 50 commandes.

Le Tableau 6.16 présente les résultats de l'algorithme génétique en fonction des deux méthodes de minimisation utilisées. Les colonnes "Nombre d'itérations" indiquent le nombre moyen d'itérations effectuées pour chaque méthode en fonction du types d'instances résolues. Les colonnes "Écart la meilleure" indique l'écart de la méthode évaluée à la meilleure solution trouvée par les deux méthodes. Enfin, les colonnes "Nombre de meilleures" indiquent le nombre de meilleures solutions trouvées par chaque méthode. La deuxième ligne du tableau indique la méthode concernée par les résultats de la colonne.

Le nombre d'itérations de chaque méthode est à mettre en perspective avec la durée d'évaluation moyenne de la méthode d'évaluation concernée (Tableau 6.12). Pour les deux méthodes, l'évolution du nombre d'itérations varie de manière importante. Ainsi, le nombre d'itérations de ces méthodes diminue avec le nombre de commandes mais augmente avec le nombre de producteurs.

n	o	Nombre d'itérations		Écart à la meilleure (%)		Nombre de meilleures	
		FIN	OPT	FIN	OPT	FIN	OPT
3	25	204.3	183.3	1.0%	0.5%	5	6
	50	84.6	62.2	1.1%	0.3%	3	7
6	25	156.2	160.2	0.1%	0.1%	8	8
	50	126.7	120.2	0.7%	0.4%	5	5
12	25	175.8	158.3	0.5%	0.0%	2	10
	50	163.9	113.9	1.8%	0.4%	3	7

Tab. 6.16: Résultats de l'algorithme génétique

Concernant le nombre de meilleures solutions trouvées et les écarts, on constate que la méthode $\tau^{eval} = OPT$, qui trouve les meilleurs résultats pour l'évaluation des séquences de production générées par l'algorithme glouton (Tableau 6.14), trouve également les meilleurs résultats pour l'algorithme génétique. Cependant, comparer aux précédents résultats, on peut noter que l'écart moyen entre les deux méthodes s'est réduit. Pour la méthode associée au paramétrage $\tau^{eval} = FIN$, l'écart à la meilleure solution connue passe de 3,8% au maximum à 1,8% au maximum. On peut même considérer que les méthodes $\tau^{eval} = OPT$ et $\tau^{eval} = FIN$ sont équivalentes pour résoudre les instances à 6 producteurs.

6.5.4 Comparaison avec l'algorithme glouton

Cette dernière section de résultats compare l'algorithme glouton (Section 6.3.1) avec l'algorithme génétique pour le paramétrage $\tau^{eval} = OPT$. L'algorithme glouton est également testé pour les deux méthodes de minimisation associées aux paramètres $\tau^{eval} = FIN$ et $\tau^{eval} = OPT$. Pour toutes les instances et toutes les méthodes, la durée d'exécution de l'algorithme glouton n'est significativement plus longue que la durée d'exécution la méthode de minimisation utilisé (Tableau 6.12). Ainsi, le temps d'exécution de l'algorithme glouton reste en dessous de la demi seconde en moyenne. L'algorithme glouton ne trouve jamais la meilleure solution comparée à l'algorithme génétique (avec un exception pour $\tau^{eval} = OPT$ pour une instance à 12 producteurs et 25 commandes). Le Tableau 6.17 montre l'écart moyen entre l'algorithme génétique et l'algorithme glouton.

Méthode d'évaluation du glouton	$n = 3$		$n = 6$		$n = 12$	
	$o = 25$	$o = 50$	$o = 25$	$o = 50$	$o = 25$	$o = 50$
$\tau^{eval} = FIN$	10.8%	12.9%	9.0%	15.3%	4.7%	17.1%
$\tau^{eval} = OPT$	9.3%	9.2%	7.9%	13.2%	4.1%	12.7%

Tab. 6.17: Écart moyen entre l'algorithme génétique ($\tau^{eval} = OPT$) et l'algorithme glouton

De manière logique, la paramètre $\tau^{eval} = OPT$ propose un écart plus faible que le paramètre $\tau^{eval} = FIN$. En règle générale, l'écart avec l'algorithme génétique augmente avec "la complexité des instances". Par exemple, pour les deux paramétrages, l'écart avec l'algorithme génétique augmente avec le nombre de commandes par instance. D'autre part, les écarts les plus faibles sont obtenus pour les instances à 12 producteurs et 25 commandes. Pour ces instances, où le nombre de commandes par producteur est le plus faible (au plus 4.7%), on peut s'attendre à ce qu'un classement des commandes par dates dues de livraison soit plus efficace.

Pour les autres instances et pour chaque type de paramétrage, l'écart entre l'algorithme génétique et l'algorithme glouton se situe entre 9.0% et 17.1%.

6.6 Conclusion

Ce chapitre présente un problème intégré se démarquant du problème abordé dans les chapitres précédents. Un ensemble de producteurs se coordonne avec un 3PL pour livrer un même client. Nous abordons un contexte où les itinéraires des tournées ainsi que les dates de passage sont déjà fixés. Il s'agit donc de déterminer la séquence de production de chaque producteur et l'affectation des commandes aux véhicules en respectant la capacité des véhicules et les délais de production et tout en minimisant les coûts d'inventaire liés à la production.

Ce nouveau modèle intégrant des contraintes de capacité des véhicules et interdisant les retards, la génération d'instances réalisables est plus complexe. Ce chapitre propose une méthode de génération d'instances ainsi qu'un protocole pour évaluer et ajuster leurs faisabilités.

Les méthodes de résolution proposées choisissent d'aborder ce problème en fixant d'abord les différentes séquences de production pour ensuite en évaluer la faisabilité et minimiser les coûts d'inventaires associées. Une méthode de minimisation exacte des coûts d'inventaire est proposée ainsi qu'une méthode de minimisation approchée. Cette dernière, avec un temps d'exécution plus rapide, ne minimise que les coûts d'inventaire en fin de production.

Ces méthodes sont intégrées dans un algorithme génétique. Les résultats des tests sont meilleurs lorsque l'affectation des commandes aux véhicules et la minimisation des coûts d'inventaire sont réalisées de manière optimale pour chaque solution explorée.

Ce chapitre est un travail préliminaire sur le problème abordé. Comme perspective, le champ des instances testées pourraient être élargi. Un plus grand nombre de commandes pourrait être considéré ainsi que des itinéraires de livraison différents de ceux présentés. Il serait intéressant de voir si les méthodes d'évaluation restent performantes pour des instances de plus grandes tailles. Concernant la résolution du modèle, les deux méthodes d'évaluation d'une solution proposées se basent sur la résolution d'un modèle mathématique, ce qui implique une temps de résolution élevé (de l'ordre de quelques millisecondes). Trouver un algorithme polynomial pour évaluer une solution en des temps beaucoup plus courts pourrait permettre de développer des algorithmes plus performants.

Enfin, le modèle lui même pourrait être élargi. Dans ce chapitre, les producteurs partagent l'ensemble de leurs informations (y compris les temps de préparation des commandes et les coûts d'inventaire sur la chaîne de production) afin de minimiser l'ensemble de leurs coûts. Dans un environnement concurrentiel, cette hypothèse n'est pas réaliste. Un modèle où les producteurs ne partagent que l'information des quantités à transporter et doivent se coordonner pour affecter leurs commandes aux véhicules pourrait être exploré.

Conclusion et perspectives

Dans cette thèse nous considérons un ensemble de problèmes intégrés de production et de distribution. Dans chacun de ces problèmes, un ensemble de commande doit être produite et acheminée à son client. La chaîne de production considérée est un *flow-shop* de permutation, chaque commande doit passer sur un ensemble de machines et toutes les machines doivent traiter les commandes dans le même ordre. De plus, des coûts d'inventaires sont pris en compte aux différentes étapes de la production. Une fois produites, les commandes sont regroupées en lot et acheminées à leurs clients lors d'une tournée. Nous considérons que la partie production des commandes est assurée par un ou plusieurs producteurs et que la livraison est assurée par un prestataire logistique (ou 3PL). Ces deux types d'agent peuvent faire partie de la même entité. Ils mettent alors leurs informations en commun et minimisent leurs coûts de manière conjointe. Ils peuvent également être indépendant. Ils doivent alors se coordonner tout en minimisant leurs coûts de manière individuelle.

Au Chapitre 1, nous introduisons des notions générales sur la recherche opérationnelle avec une présentation des différentes méthodes de résolution, exactes et approchées. Et également introduite la classification des problèmes d'ordonnancement et de tournées de véhicules. Un rapide aperçu de quelques cas concrets pouvant être rattachés à notre sujet est également présenté. Le Chapitre 2 fait un présentation de l'état de l'art découpé en trois parties : d'abord un état de l'art spécifique aux problèmes intégrés, avec une partie consacrée à l'évolution des modèles abordés dans la littérature, puis deux parties consacrées respectivement aux problèmes d'ordonnancement et de tournées en lien avec notre sujet de thèse. Le Chapitre 3 introduit le modèle générale du problème résolue dans les deux chapitres suivants. Dans ce problème, les commandes sont préparées par un unique producteur et livrées par une flotte de véhicules. Si les commandes sont livrées en retard, une pénalité doit être versée au client. Le Chapitre 4 aborde un cas où le producteur a la capacité d'imposer ses décisions au 3PL. Il choisit la séquence de production des commandes et leurs mise en lots tout en faisant des hypothèses sur la séquence de livraison utilisée par le 3PL. Ensuite, le 3PL livre les commandes en minimisant les coûts de transport et les pénalités dues aux retards de livraison. Un algorithme inspiré de l'algorithme Split de Christian Prins est proposé pour mettre les commandes en lot à partir d'une séquence de production connue. Différentes méthodes d'évaluation d'une solution sont proposées ainsi qu'une recherche locale. Ces différents composants sont utilisés pour concevoir un GRASP (*Greedy randomized adaptive search algorithm*) et un algorithme génétique. Ces deux méthodes sont testées pour différents paramètres et les résultats favorisent l'algorithme génétique. Dans le Chapitre 5, les deux agents (producteur et 3PL) font partie de la même entité et collabore de manière totale. De plus, nous considérons que la composition des lots lors de la livraison est déjà définie. La résolution de ce problème se fait en deux étapes. Pour chaque lot et chaque date de départ possible, une étape de prétraitement définit l'itinéraire de livraison minimisant les coûts de transport et les pénalités de retard du lot. Une PSE est définit pour réaliser cette étape de manière exacte pour des lots de petites tailles. Pour des lots plus grands, des heuristiques sont proposées. La seconde étape de résolution utilise les résultats de la première étape pour évaluer une solution. Cette méthode d'évaluation prend en entrée une séquence de production et minimise les différents coûts associés (inventaire, transport et retard) à l'aide d'un PLS (solvable théoriquement en temps polynomiale). Pour résoudre des instances de petite tailles,

cette méthode d'évaluation est insérée dans un modèle de programmation linéaire et un modèle de programmation par contraintes. Pour résoudre des instances de grande tailles, cette méthode d'évaluation est insérée dans une recherche locale s'appliquant sur la séquence de production, d'abord au niveau des lots, puis au niveau des commandes. Ces différentes méthodes sont ensuite testées et comparées pour différents jeux de paramètres. Enfin, le Chapitre 6 propose un problème différent où plusieurs producteurs doivent tous fournir un même client. Ces producteurs s'associent à un 3PL et ils déterminent ensemble, au préalable, les différentes tournées visitant les producteurs et finissant chez le client. Ainsi les horaires de passage des véhicules chez les producteurs sont déjà connues ainsi que les dates de livraison associées à chaque véhicule. De plus, pour ce problème, les véhicules ont une capacité limitée et le retard n'est pas permis. L'objectif est de trouver une solution réalisable minimisant les coûts d'inventaire liés à la production. Lors de la résolution, l'ensemble des séquences de production des différents producteurs sont déterminées avant l'évaluation d'une solution. Des méthodes d'évaluations, heuristiques et exactes, proposent alors une affectation des commandes aux véhicules minimisant les coûts d'inventaire lorsqu'une affectation réalisable existe. Ces méthodes d'évaluation sont utilisées au sein d'un algorithme génétique. Une étude pour définir des instances réalisables est également effectuée. L'algorithme génétique est évalué sur ces instances pour les différentes méthodes d'évaluation.

Plusieurs directions peuvent être explorées en vue de futures recherches. De nombreux modèles mathématiques présentés dans cette thèse sont de type PLS et sont actuellement résolus par appel à un solveur. Lorsque la séquence de production des commandes est fixée chez un producteur, ces modèles sont utilisés pour déterminer les différentes dates de production des tâches associées aux commandes sur les différentes machines. L'objectif est de minimiser les coûts d'inventaire. De plus, pour les instances utilisées dans cette thèse et pour une même commandes, les coûts d'inventaire sont croissants le long de la chaîne de production. En utilisant ces propriétés, des algorithmes polynomiaux *ad-hoc* pourraient être développés pour remplacer l'appel au solveur et ainsi accélérer les différents méthodes de résolutions. Une méthode heuristique peut être développée pour résoudre le problème général du Chapitre 3 dans son intégralité. Le Chapitre 5 montre qu'une fois la composition d'un lot connue, ses coûts de livraison peuvent être facilement calculés pour toutes dates de départ possibles pour être ensuite être intégrés sous forme d'une fonction de coûts dans une méthode de résolution plus générale. Il est alors envisageable de développer une méthode par génération de colonne ne travaillant qu'avec un sous-ensemble de lot. Plusieurs critères peuvent alors être explorées pour constituer les lots utilisés : la proximité des dates dues de livraison des commandes, la proximité géographique des sites de livraison clients et la capacité d'un lot à être produit rapidement et en générant peu de coûts d'inventaire. De même, dans le Chapitre 6, le modèle associé peut être modifié pour devenir un modèle multi-agents. En effet dans ce modèle, tout se passe comme si les différents producteurs étaient un seul et même agent. Ils partagent l'ensemble de leurs informations de production et planifient d'un commun accord l'affectation des commandes aux véhicules et la séquence de production sur les différents sites. Ces hypothèse ne sont pas réalistes dans un environnement concurrentiel. Un modèle où les producteurs ne partagent que l'information des quantités à transporter peut être envisagé. L'affectation des commandes aux véhicules devra tenir compte des relations de concurrence entre les producteurs. Enfin, des modélisations avec des chaînes de production plus simple peuvent être explorées. L'étude de la littérature à montrer que considérer des coûts d'inventaire en cours de production avec des pénalités de retard à verser aux clients dans le cadre de problèmes intégrés serait suffisant pour apporter une contribution.

Annexe A

Preuve

Dans le cadre d'un problème de livraison, un ensemble σ de commandes doit être livré pour une date de départ de tournée quelconque. Chaque commande l dans σ est associée a une date due de livraison d_l et des pénalités unitaires de retard π_l. Ainsi pour une date de livraison effective D_l de la commande l, les pénalités de retard associées sont données par l'expression suivantes : $\max(0, \pi_l(D_l - d_l))$.

Dans le cadre de cette preuve, nous avons connaissance de $|\sigma|$ dates, notées D_k^{min}. Pour tout $k \in \{1, \ldots, o\}$, la k^{eme} date de livraison possible au plus tôt pour les commandes de σ est supérieure ou égale à D_k^{min}.

Nous notons τ^{OPT} la séquence de livraison de toutes les commandes $l \in \sigma$ qui minimise la somme des pénalités de retard. τ_k^{OPT} représente l'index l de la k^{eme} commande livrée par cette séquence τ^{OPT}, $l \in \sigma$. Soit z_σ^{OPT} la pénalité de coût minimal pouvant être obtenue lors de la livraison des commandes de σ. Nous avons :

$$z_\sigma^{OPT} = \sum_{k=1}^{|\sigma|} \max\left(0, \pi_l(D_k^{OPT} - d_l)\right) \text{ avec } l = \tau_k^{OPT}$$

avec D_k^{OPT} la date de livraison de la k^{eme} commande de la séquence τ^{OPT}.

Soit lb_σ^P la valeur de la solution optimale d'un problème d'affectation de coûts minimum AP donné par la matrice $|\sigma| \times |\sigma|$ de coût $U = (u_{l,k})_{l,k}$ où $u_{l,k} = \max(0, \pi_l(D_k^{min} - d_l))$, $\forall l \in \sigma, \forall k \in \{1, \ldots, |\sigma|\}$.

Nous voulons montrer que lb_σ^P est une borne inférieur de z_σ^{OPT}:

Théorème 5.2

$$lb_\sigma^P \leq z_\sigma^{OPT}, \forall \sigma, \forall t \in \{0, \ldots, T\}$$

Démonstration. Définissons un second problème d'affectation AP^{OPT} avec une matrice de coût U^{OPT} de dimension $|\sigma| \times |\sigma|$ où $u_{l,k}^{OPT} = \max(0, \pi_l(D_k^{OPT} - d_l))$, $\forall l \in \sigma, \forall k \in \{1, \ldots, |\sigma|\}$.

Une solution X de AP^{OPT} représente une affectation des commandes aux dates de livraison D_k^{OPT}. Nous notons $f^{OPT}(X)$ la valeur de la fonction objectif de la solution X au problème d'affectation AP^{OPT}.

La solution X^{OPT} donnée par l'affectation de la solution optimale τ^{OPT} est réalisable pour le problème AP^{OPT} et :

$$z_\sigma^{OPT} = f^{OPT}(X^{OPT}) = \sum_{(l,k)\in X^{OPT}} u_{l,k}^{OPT} \tag{A.1}$$

Sachant que $D_k^{min} \leq D_k^{OPT}$ nous avons donc $u_{j,k} \leq u_{j,k}^{OPT}$, $\forall k \in \{1, \ldots, |\sigma|\}$ and $\forall j \in \tau^{OPT}$. Toutes solutions réalisable X de AP est aussi une solution réalisable de AP^{OPT}, donc :

$$\sum_{(i,k)\in X} u_{i,k} \leq \sum_{(i,k)\in X} u_{i,k}^{OPT} \tag{A.2}$$

$$f(X) \leq f^{OPT}(X) \quad \text{où } f(X) \text{ est la valeur de la fonction objectif de AP.} \tag{A.3}$$

$$lb_\sigma^P \leq f(X) \quad \text{car } lb_\sigma^P \text{ est la valeur de la solution optimale de AP}$$

$$lb_\sigma^P \leq f^{OPT}(X) \quad \text{d'après (A.3)}$$

$$lb_\sigma^P \leq f^{OPT}(X^{OPT}) = z_\sigma^{OPT} \quad \text{d'après (A.1)}$$

Le Théorème 5.2 est démontré. □

Bibliographie

[Aarts et al., 1985] Aarts, E. H., Van Laarhoven, P. J., et al. (1985). Statistical cooling : A general approach to combinatorial optimization problems. *Philips J. Res.*, 40(4) :193–226.

[Absi et al., 2015] Absi, N., Archetti, C., Dauzère-Pérès, S., and Feillet, D. (2015). A Two-Phase Iterative Heuristic Approach for the Production Routing Problem. *Transportation Science*, 49(4) :784–795.

[Absi et al., 2018] Absi, N., Archetti, C., Dauzère-Pérès, S., Feillet, D., and Speranza, M. G. (2018). Comparing sequential and integrated approaches for the production routing problem. *European Journal of Operational Research*, 269(2) :633 – 646.

[Agnetis et al., 2014] Agnetis, A., Ali, M., and Fu, L.-l. (2014). Coordination of production and interstage batch delivery with outsourced distribution. *European Journal of Operational Research*, 238(1) :130–142.

[Amorim et al., 2013] Amorim, P., Belo-Filho, M. A., Toledo, F. M., Almeder, C., and Almada-Lobo, B. (2013). Lot sizing versus batching in the production and distribution planning of perishable goods. *International Journal of Production Economics*, 146(1) :208–218.

[Armstrong et al., 2008] Armstrong, R., Gao, S., and Lei, L. (2008). A zero-inventory production and distribution problem with a fixed customer sequence. *Annals of Operations Research*, 159(1) :395–414.

[Bülbül et al., 2004] Bülbül, K., Kaminsky, P., and Yano, C. (2004). Flow shop scheduling with earliness, tardiness, and intermediate inventory holding costs. *Naval Research Logistics (NRL)*, 51(3) :407–445.

[Cakici et al., 2014] Cakici, E., Mason, S. J., Geismar, H. N., and Fowler, J. W. (2014). Scheduling parallel machines with single vehicle delivery. *Journal of Heuristics*, 20(5) :511–537.

[Chandra, 1994] Chandra, Pankaj et Fisher, M. (1994). Coordination of production and distribution planning. *European Journal of Operational Research*, 72 :503–517.

[Chang et al., 2014] Chang, Y.-C., Li, V. C., and Chiang, C.-J. (2014). An ant colony optimization heuristic for an integrated production and distribution scheduling problem. *Engineering Optimization*, 46(4) :503–520.

[Chen, 2010] Chen, Z.-l. (2010). Integrated Production and Outbound Distribution Scheduling : Review and Extensions. *Operations Research*, 58(1) :130–148.

[Chenevoy, 2016] Chenevoy, C. (2016). Carrefour mutualise ses approvisionnements avec les pme. *LSA - Libre service actualités*, Version en ligne.

[Cheng et al., 2019] Cheng, B., Leung, J. Y.-T., Li, K., and Yang, S. (2019). Integrated optimization of material supplying, manufacturing, and product distribution : Models and fast algorithms. *European Journal of Operational Research*, 277(1) :100 – 111.

[Cheng et al., 2015] Cheng, B.-y., Leung, J. Y., and Li, K. (2015). Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method. *Computers & Industrial Engineering*, 83 :217–225.

[Cheref et al., 2016] Cheref, A., Artigues, C., and Billaut, J.-C. (2016). A new robust approach for a production scheduling and delivery routing problem. *IFAC-PapersOnLine*, 49(12) :886–891.

[Chiang et al., 2009] Chiang, W. C., Russell, R., Xu, X., and Zepeda, D. (2009). A simulation/metaheuristic approach to newspaper production and distribution supply chain problems. *International Journal of Production Economics*, 121(2) :752–767.

[Condotta et al., 2013] Condotta, A., Knust, S., Meier, D., and Shakhlevich, N. V. (2013). Tabu search and lower bounds for a combined production–transportation problem. *Computers & Operations Research*, 40(3) :886–900.

[Cook, 1971] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158.

[Cordeau, 2000] Cordeau (2000). *The VRP with time windows*. Groupe d'études et de recherche en analyse des décisions Montréal.

[Cover, 1967] Cover, Thomas et Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1) :21–27.

[Devapriya et al., 2017] Devapriya, P., Ferrell, W., and Geismar, N. (2017). Integrated production and distribution scheduling with a perishable product. *European Journal of Operational Research*, 259(3) :906–916.

[Dong et al., 2013] Dong, J., Zhang, A., Chen, Y., and Yang, Q. (2013). Approximation algorithms for two-machine open shop scheduling with batch and delivery coordination. *Theoretical Computer Science*, 491 :94–102.

[Farahani et al., 2012] Farahani, P., Grunow, M., and Günther, H.-O. (2012). Integrated production and distribution planning for perishable food products. *Flexible Services and Manufacturing Journal*, 24(1) :28–51.

[Federgruen, 1984] Federgruen, Awi et Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. *Operations Research*, 32 :1019–1037.

[Fernandez-Viagas et al., 2017] Fernandez-Viagas, V., Ruiz, R., and Framinan, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan : State-of-the-art and computational evaluation. *European Journal of Operational Research*, 257(3) :707–721.

[Fu et al., 2018] Fu, L.-L., Aloulou, M. A., and Artigues, C. (2018). Integrated production and outbound distribution scheduling problems with job release dates and deadlines. *Journal of Scheduling*, 21(4) :443–460.

[Fu et al., 2017] Fu, L.-L., Aloulou, M. A., and Triki, C. (2017). Integrated production scheduling and vehicle routing problem with job splitting and delivery time windows. *International Journal of Production Research*, 55(20) :5942–5957.

[Fulconis et al., 2011] Fulconis, F., Gilles, P., Roveillo, G., et al. (2011). *La prestation logistique : origines, enjeux et perspectives*. EMS.

[Fumero, 1999] Fumero, Francesca et Vercellis, C. (1999). Synchronized development of production, inventory, and distribution schedules. *Transportation science*, 33(3) :330–340.

[Geismar et al., 2008] Geismar, J. H., Laporte, G., Lei, L., and Sriskandarajah, C. (2008). The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing*, 20(1) :21–33.

[Glover, 1997] Glover, F. (1997). Tabu search and adaptive memory programming—advances, applications and challenges. In *Interfaces in computer science and operations research*, pages 1–75. Springer.

[Goel, 2018] Goel, A. (2018). Legal aspects in road transport optimization in europe. *Transportation research part E : logistics and transportation review*, 114 :144–162.

[Golden et al., 1984] Golden, B., Assad, A., and Dahl, R. (1984). Analysis of a large scale vehicle routing problem with an inventory component. *Large scale systems*, 7(2-3) :181–190.

[Graham et al., 1979] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier.

[Gupta, 1979] Gupta, J. N. (1979). A review of flowshop scheduling research. In *Disaggregation*, pages 363–388. Springer.

[Han et al., 2019] Han, D., Yang, Y., Wang, D., Cheng, T. C. E., and Yin, Y. (2019). Integrated production , inventory , and outbound distribution operations with fixed departure times in a three-stage supply chain. *Transportation Research Part E*, 125(March) :334–347.

[Hassanzadeh et al., 2016] Hassanzadeh, A., Rasti-barzoki, M., and Khosroshahi, H. (2016). Two new meta-heuristics for a bi-objective supply chain scheduling problem in flow-shop environment. *Applied Soft Computing Journal*, 49 :335–351.

[Hezarkhani et al., 2016] Hezarkhani, B., Slikker, M., and Van Woensel, T. (2016). A competitive solution for cooperative truckload delivery. *Or Spectrum*, 38(1) :51–80.

[Hollande, 1975] Hollande, J. H. (1975). Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor*, 1 :8.

[Jackson, 1955] Jackson, J. (1955). Scheduling a production line to minimize maximum tardiness. *Research Report 43, Management Science Reasearch Project, University of California, Los Angeles*.

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

[Kergosien et al., 2017] Kergosien, Y., Gendreau, M., and Billaut, J.-C. (2017). Benders decomposition based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. *European Journal of Operational Research*, 262(1) :287–298.

[Koç, 2017] Koç, Utku et Sabuncuoglu, I. (2017). Coordination of inbound and outbound transportation schedules with the production schedule. *Computers & Industrial Engineering*, 103 :178–192.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2) :83–97.

[Laborie et al., 2018] Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints*, 23(2) :210–250.

[Lacomme et al., 2018] Lacomme, P., Moukrim, A., Quilliot, A., and Vinot, M. (2018). Supply chain optimisation with both production and transportation integration : multiple vehicles for a single perishable product. *International Journal of Production Research*, 56(12) :4313–4336.

[Lacomme et al., 2001] Lacomme, P., Prins, C., and Ramdane-Cherif, W. (2001). Algorithmes génétiques pour le CARP. In *proceedings of 3e Conférence Francophone de Modélisation et Simulation, Troyes, France*.

[Laporte, 1992a] Laporte, G. (1992a). The traveling salesman problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2) :231–247.

[Laporte, 1992b] Laporte, G. (1992b). The vehicle routing problem : An overview of exact and approximate algorithms. *European journal of operational research*, 59(3) :345–358.

[Laporte, 2009] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation science*, 43(4) :408–416.

[Lee et al., 1997] Lee, Y. H., Bhaskaran, K., and Pinedo, M. (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE transactions*, 29(1) :45–52.

[Lei et al., 2006] Lei, L., Liu, S., Ruszczynski, A., and Park, S. (2006). On the integrated production, inventory, and distribution routing problem. *IIE Transactions*, 38(11) :955–970.

[Li et al., 2005] Li, C.-L., Vairaktarakis, G., and Lee, C.-Y. (2005). Machine scheduling with deliveries to multiple customer locations. *European Journal of Operational Research*, 164(1) :39–51.

[Li et al., 2008] Li, K., Sivakumar, A. I., and Ganesan, V. K. (2008). Analysis and algorithms for coordinated scheduling of parallel machine manufacturing and 3PL transportation. *International Journal of Production Economics*, 115(2) :482–491.

[Lin, 2005] Lin, Bertrand MT et Cheng, T. E. (2005). Two-machine flowshop batching and scheduling. *Annals of Operations Research*, 133(1-4) :149–161.

[Marandi, 2018] Marandi, Fateme et Ghomi, S. M. T. F. (2018). Integrated multi-factory production and distribution scheduling applying vehicle routing approach. *International Journal of Production Research*, pages 1–27.

[Minella et al., 2008] Minella, G., Ruiz, R., and Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3) :451–471.

[Mishra, 2018] Mishra, Aseem et Shrivastava, D. (2018). A TLBO and a jaya heuristics for permutation flow shop scheduling to minimize the sum of inventory holding and batch delay costs. *Computers & Industrial Engineering*, 124 :509–522.

[Mohammadi et al., 2020] Mohammadi, S., Al-e Hashem, S. M., and Rekik, Y. (2020). An integrated production scheduling and delivery route planning with multi-purpose machines : A case study from a furniture manufacturing company. *International Journal of Production Economics*, 219 :347–359.

[Moons et al., 2017] Moons, S., Ramaekers, K., Caris, A., and Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level : A review and discussion. *Computers & Industrial Engineering*, 104 :224–245.

[Mouthuy et al., 2015] Mouthuy, S., Massen, F., Deville, Y., and Van Hentenryck, P. (2015). A multistage very large-scale neighborhood search for the vehicle routing problem with soft time windows. *Transportation Science*, 49(2) :223–238.

[Munkres, 1957] Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1) :32–38.

[Nawaz, 1984] Nawaz, M. (1984). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA*, 11.

[Neves-Moreira et al., 2019] Neves-Moreira, F., Almada-Lobo, B., Cordeau, J.-F., Guimarães, L., and Jans, R. (2019). Solving a large multi-product production-routing problem with delivery time windows. *Omega*, 86 :154 – 172.

[Nogueira et al., 2020] Nogueira, T. H., Bettoni, A. B., de Oliveira Mendes, G. T., dos Santos, A. G., and Ravetti, M. G. (2020). Problem on the integration between production and delivery with parallel batching machines of generic job sizes and processing times. *Computers & Industrial Engineering*, page 106573.

[Phouratsamay, 2017] Phouratsamay, S.-L. (2017). *Coordination des décisions de planification dans une chaîne logistique*. Theses, Université Pierre et Marie Curie - Paris VI.

[Potts, 2000] Potts, Chris N et Kovalyov, M. Y. (2000). Scheduling with batching : A review. *European journal of operational research*, 120(2) :228–249.

[Prins, 2004] Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12) :1985–2002.

[Prins et al., 2014] Prins, C., Lacomme, P., and Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems : A review. *Transportation Research Part C : Emerging Technologies*, 40 :179–200.

[Rohmer, 2015] Rohmer, Sonja et Billaut, J.-C. (2015). Production and outbound distribution scheduling : a two-agent approach. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 135–144.

[Ruiz, 2005] Ruiz, Rubén et Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2) :479 – 494. Project Management and Scheduling.

[Salani et al., 2016] Salani, M., Battarra, M., and Gambardella, L. M. (2016). Exact algorithms for the vehicle routing problem with soft time windows. In *Operations Research Proceedings 2014*, pages 481–486. Springer.

[Sarmiento, 1999] Sarmiento, Ana Maria et Nagi, R. (1999). A review of integrated analysis of production-distribution systems. *IIE transactions*, 31(11) :1061–1074.

[Scholz-Reiter et al., 2010] Scholz-Reiter, B., Frazzon, E. M., and Makuschewitz, T. (2010). Integrating manufacturing and logistic systems along global supply chains. *CIRP Journal of Manufacturing Science and Technology*, 2(3) :216–223.

[Shelbourne et al., 2017] Shelbourne, B. C., Battarra, M., and Potts, C. N. (2017). The vehicle routing problem with release and due dates. *INFORMS Journal on Computing*, 29(4) :705–723.

[Shen et al., 2014] Shen, L., Gupta, J. N., and Buscher, U. (2014). Flow shop batching and scheduling with sequence-dependent setup times. *Journal of scheduling*, 17(4) :353–370.

[Solomon, 1988] Solomon, Marius M et Desrosiers, J. (1988). Survey paper — Time window constrained routing and scheduling problems. *Transportation science*, 22(1) :1–13.

[Taillard et al., 1997] Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2) :170–186.

[Tavares-Neto, 2019] Tavares-Neto, Roberto F et Seido, M. (2019). An Iterated Greedy approach to integrate production by multiple parallel machines and distribution by a single capacitated vehicle. *Swarm and Evolutionary Computation*, 44(March 2018) :612–621.

[Tijs, 1986] Tijs, Stef H et Driessen, T. S. (1986). Game theory and cost allocation problems. *Management science*, 32(8) :1015–1028.

[Toth, 1990] Toth, Paolo et Martello, S. (1990). *Knapsack problems : Algorithms and computer implementations*. Wiley.

[Ullrich, 2013] Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research*, 227(1) :152–165.

[Viergutz, 2014] Viergutz, Christian et Knust, S. (2014). Integrated production and distribution scheduling with lifespan constraints. *Annals of Operations Research*, 213(1) :293–318.

[Viet et al., 2018] Viet, N. Q., Behdani, B., and Bloemhof, J. (2018). The value of information in supply chain decisions : A review of the literature and research agenda. *Computers & Industrial Engineering*, 120 :68–82.

[Wang et al., 2013] Wang, D., Grunder, O., and El Moudni, A. (2013). Single-item production–delivery scheduling problem with stage-dependent inventory costs and due-date considerations. *International Journal of Production Research*, 51(3) :828–846.

[Wang et al., 2019a] Wang, D., Zhu, J., Wei, X., Cheng, T., Yin, Y., and Wang, Y. (2019a). Integrated production and multiple trips vehicle routing with time windows and uncertain travel times. *Computers & Operations Research*, 103 :1–12.

[Wang et al., 2019b] Wang, S., Wu, R., Chu, F., and Yu, J. (2019b). Variable neighborhood search-based methods for integrated hybrid flow shop scheduling with distribution. *Soft Computing*, 24.

[Yağmur, 2020] Yağmur, Ece et Kesen, S. E. (2020). A memetic algorithm for joint production and distribution scheduling with due dates. *Computers & Industrial Engineering*, 142 :106342.