Université Lille Nord de France
Pôle de Recherche et d'Enseignement Supérieur

UNIVERSITÉ D'ARTOIS

Université d'Artois

École doctorale **SPI Lille Nord de France**
Unité de recherche **LGI2A**

Thèse présentée par **Nicolas Danloup**
Soutenue le **1er décembre 2016**
En vue de l'obtention du grade de docteur de l'Université d'Artois
Discipline **Sciences pour l'ingénieur**
Spécialité **Génie Informatique et Automatique**

Titre de la thèse

Les problèmes de collectes et livraisons avec collaboration et transbordements : modélisations et méthodes approchées

Thèse dirigée par	Gilles Goncalves	directeur
	Hamid Allaoui	co-directeur

Composition du jury

Rapporteurs	Pierre Dejax	professeur à l'Ecole des Mines de Nantes
	Aziz Moukrim	professeur à l'UTC
Examinateurs	Eric Ballot	professeur à l'Ecole des Mines ParisTech
	Feng Chu	professeur à l'Université d'Evry Val d'Essonne
	Laetitia Jourdan	professeur à l'Université de Lille 1
Directeurs de thèse	Gilles Goncalves	professeur à l'Université d'Artois
	Hamid Allaoui	professeur à l'Université d'Artois

Cette thèse a été préparée au

LGI2A
Laboratoire de Génie Informatique et d'Automatique de l'Artois - EA 3926 Faculté des Sciences Appliquées Technoparc Futura 62400 - BÉTHUNE Cedex France

Site `http://www.lgi2a.univ-artois.fr/spip/`

Les problèmes de collectes et livraisons avec collaboration et transbordements : modélisations et méthodes approchées

Résumé

La logistique collaborative est récemment devenue un élément important pour beaucoup d'entreprises afin d'améliorer l'efficacité de leur chaîne logistique. Dans cette thèse, nous étudions les possibilités offertes par les problèmes de collectes et livraisons pour améliorer les performances des chaînes logistiques grâce au transport collaboratif. La thèse est inscrite dans un projet européen nommé SCALE (Step Change in Agri-food Logistics Ecosystem). Dans un premier temps, deux métaheuristiques sont proposées et étudiées pour résoudre le problème de collectes et livraisons avec transbordements. Celles-ci sont comparées aux travaux de la littérature et permettent d'améliorer les résultats sur certaines instances. Dans un deuxième temps, un modèle pour un problème de collectes et livraisons (PDVRP) est proposé. Celui-ci est utilisé pour étudier les bénéfices de la collaboration sur le transport. Il est appliqué sur des données générées aléatoirement et sur des données réelles issues du projet SCALE. Enfin troisièmement, un modèle pour un PDVRP particulier est présenté. Dans ce modèle, les marchandises doivent passer par exactement deux points de transbordement entre les points de collecte et les points de livraison. Ce problème est inspiré d'une seconde étude de cas réalisée dans le cadre du projet SCALE. Ceci permet de mettre en évidence l'intérêt de la collaboration et du transbordement dans le domaine du transport de marchandises.

Mots clés : problèmes de collectes et livraisons ; transport collaboratif ; transbordements ; recherche à voisinage large ; algorithme génétique

Abstract

Collaborative logistics have become recently an important element for many companies to improve their supply chains efficiency. In this thesis, we study pickup and delivery problems to improve supply chains efficiency thanks to collaborative transportation. The thesis was part of the European project SCALE (Step Change in Agri-food Logistics Ecosystem). Firstly, two metaheuristics are proposed and studied to solve the Pickup and Delivery Problem with Transshipments. These metaheuristics are compared with literature works and the results of several instances are improved. Secondly, a mathematical model for a pickup and delivery problem (PDVRP) is proposed. This model is used to study the benefits of collaboration on transportation. It is applied on random data and on a case study from SCALE with real data. Finally, a model for a particular PDVRP is presented. In this model, the shipments have to cross exactly two transshipments nodes between their pickup and delivery points. This problem is inspired by a second case study made during the project SCALE. This allows to highlight the importance of collaboration and transshipment in the field of goods transportations.

Keywords: pickup and delivery problems; collaborative transportation; transshipments; large neighbourhood search; genetic algorithm

LGI2A
Laboratoire de Génie Informatique et d'Automatique de l'Artois - EA 3926
Faculté des Sciences Appliquées Technoparc Futura 62400 - BÉTHUNE Cedex
France

Remerciements

Premièrement je souhaite remercier mes deux directeurs de thèse, Prof. Gilles Goncalves et Prof. Hamid Allaoui pour m'avoir fait confiance et m'avoir laissé travailler de façon autonome. Je les remercie également pour leur disponibilité et pour avoir su me guider et m'aider quand il le fallait.

Je remercie également sincèrement Prof. Pierre Dejax et Prof. Aziz Moukrim pour avoir accepté d'être rapporteurs de ma thèse et pour le temps qu'ils ont bien voulu me consacrer. Je suis aussi très reconnaissant envers Prof. Eric Ballot, Prof. Feng Chu et Prof. Laetitia Jourdan pour avoir accepté d'évaluer mon travail de recherche.

Je remercie aussi tous mes collègues du LGI2A, permanents, doctorants, ingénieurs, stagiaires, grâce auxquels il est très agréable de venir travailler au laboratoire. Je remercie tout particulièrement Nathalie Morganti de nous apporter son soutien en permanence.

Je tiens aussi à remercier tous les membres du projet SCALE et toutes les personnes que j'ai croisé dans le cadre de ce projet, notamment Vahid, Denyse, Carlos, Marko, Andrew, Ilse, Roly et John.

Enfin je remercie tous mes amis et toute ma famille pour m'avoir toujours soutenu. Je remercie particulièrement ma conjointe, Caroline, qui m'a apporté tout son soutien durant toutes ces années et qui a toujours cru en moi.

Cette thèse a été réalisée dans le cadre du projet européen SCALE et financée par Interreg IVB North-West Europe.

Sommaire

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction

Les entreprises sont de plus en plus confrontées à des défis pour équilibrer leurs performances et leurs gains économiques avec les questions environnementales. Les législations, la pression sociale et sociétale sur les émissions et la conservation des ressources augmentent. Le développement durable à travers les chaînes logistiques n'est plus un fardeau mais plutôt une nouvelle stratégie pour améliorer la compétitivité des entreprises. Par conséquent, une gestion durable de la chaîne logistique, ainsi que l'intégration des aspects environnementaux dans tous les processus de la chaîne logistique deviennent de plus en plus importantes dans la conception et le fonctionnement des chaînes logistiques dans différentes industries (Sundarakani et al., 2010).

La thèse s'inscrit dans le cadre d'un projet européen nommé Step Change in Agri-food Logistics Ecosystem (SCALE)[1] qui s'est déroulé d'avril 2012 à septembre 2015. Le but de ce projet était d'aider à améliorer la compétitivité économique des chaînes logistiques agroalimentaires du nord-ouest de l'Europe, mais aussi d'améliorer l'impact social et environnemental de ces chaînes logistiques. Le projet a été financé en partie par INTERREG IVB North-West Europe, un instrument financier de la politique de cohésion de l'Union Européenne. Le reste du projet a été financé par les différents partenaires du projet :

— Cranfield School of Management, UK
— Université d'Artois, France
— Wageningen University, Pays-Bas
— DHL Supply Chain, UK
— European Food and Farming Partnerships, UK

1. **http://sfcplatform.eu/**.

Le projet a été élaboré afin de mettre en valeur la synergie entre l'expertise industrielle et la rigueur académique pour offrir un certain nombre de méthodes et d'outils d'aide à la décision que le secteur agroalimentaire pourrait utiliser pour améliorer l'efficacité et la durabilité de la logistique tout au long de la chaîne. Grâce à ce projet, ces outils ont été testés à travers un réseau d'entreprises agroalimentaires pour montrer comment de nouvelles approches peuvent améliorer la prise en compte du développement durable dans les activités de la logistique et sans altérer la part des bénéfices commerciaux réels.

Un facteur de durabilité dans les chaînes logistiques est le transport. Dans les pays développés, le transport routier de marchandises est le mode de transport dominant (85% du transport terrestre en France) et par conséquent il est responsable d'une part significative de l'impact environnemental global de la logistique (McKinnon et al., 2015). L'utilisation optimale du transport routier de marchandises dépend de plusieurs facteurs : le nombre de kilomètres parcourus, le coefficient de chargement des véhicules, le taux de retour à vide... Améliorer cette utilisation est devenu un objectif du ministère de l'environnement, de l'énergie et de la mer à l'aide du programme "objectif CO_2" [2]. Ce programme propose des guides et des outils aux transporteurs afin de réduire leurs émissions. Parmi les pistes proposées aux transporteurs, l'optimisation de la planification des tournées permet de réduire les kilomètres parcourus. Une autre piste proposée est d'augmenter le taux de remplissage des véhicules utilisés pour ces tournées. En effet, son augmentation est une méthode efficace pour améliorer l'aspect durable de la logistique en apportant des bénéfices économiques et environnementaux (Sarkar et Mohapatra, 2008 ; Van de Klundert et Otten, 2011). Les conséquences de l'amélioration de l'utilisation des véhicules sont la réduction des gaz à effets de serre, du trafic routier, des bruits et des congestions engendrés sur les routes (McKinnon et al., 2015). Une pratique pour augmenter le taux de remplissage des véhicules est d'utiliser le transbordement pour consolider les flux logistiques dans certains points du réseau. Le transbordement est l'action de faire passer des marchandises d'un véhicule à un autre via ces points de transbordement avec un stockage très limité dans le temps. Ceci peut être utilisé pour changer le mode de transport, pour séparer des marchandises

2. http ://www.objectifco2.fr/

qui ont différentes destinations ou au contraire pour regrouper des marchandises qui ont la même destination. Le transbordement est utilisé dans différents types d'industries comme la messagerie, l'industrie des pièces automobiles et la grande distribution (Kulwiec, 2004). Le partage de véhicules ainsi que le transbordement, en tant que manière de collaborer, permettent d'augmenter le taux de remplissage des véhicules (Newing, 2008).

Le terme « collaboration dans les chaînes logistiques » fait référence aux activités entre les partenaires de la chaîne logistique concernés par la création rentable, rapide et fiable et la circulation des matériaux pour satisfaire les besoins des clients (Muckstadt et al., 2001). Malgré les obstacles (économique, sociétal ...) qui limitent potentiellement la collaboration entre les entreprises au sein de nombreux secteurs au niveau mondial, la collaboration est vue de plus en plus comme une nécessité plutôt que comme une option. Différents types de collaboration (verticale, horizontale et latérale) peuvent être trouvés dans les différentes relations et partenariats dans les chaînes logistiques. La collaboration verticale se produit lorsque deux ou plusieurs acteurs tels que le fabricant, le distributeur, le transporteur et le détaillant partagent leurs ressources et/ou certaines informations pour servir les mêmes clients finaux (Bahinipati et Deshmukh, 2012). La collaboration horizontale est définie comme un accord commercial entre deux ou plusieurs entreprises situées au même niveau de la chaîne logistique afin de permettre une plus grande facilité de travail et de coopération en vue d'atteindre un objectif commun (Bahinipati et al., 2009). La collaboration latérale vise à acquérir une plus grande souplesse en combinant la collaboration horizontale et la collaboration verticale.

Deux piliers se distinguent dans le cadre de la collaboration dans les chaînes logistiques, portant sur la conception et la gouvernance des activités de la chaîne logistique, et la mise en place et le maintien des relations de la chaîne logistique (Matopoulos et al., 2007). Ces piliers peuvent être appliqués dans la collaboration horizontale, verticale ou latérale. Le premier pilier est lié à la conception et à la gouvernance des activités de la chaîne logistique et se compose de trois éléments. Le premier est de prendre la décision de choisir un partenaire approprié. Le deuxième élément est de sélectionner les activités pour lesquelles la collaboration sera établie. Après avoir sélectionné les activités, le troisième

élément consiste à identifier à quel niveau les entreprises vont collaborer. Plus la profondeur (du niveau opérationnel au niveau stratégique) et la largeur (des activités simples de la logistique aux activités plus complexes comme le développement de nouveaux produits) sont grandes, plus la collaboration est intense. Enfin, un autre élément important pour la conception et la gouvernance des activités de la chaîne logistique comprend la décision de choisir la technique et la technologie appropriées pour faciliter le partage de l'information. Le second pilier concerne la mise en place et le maintien des relations de collaboration. Les éléments essentiels pour ce pilier qui ont été cités dans la littérature sont la réciprocité des bénéfices et des risques avec le partage des profits et des risques (Barratt et Oliveira, 2001).

Malgré les avantages qui ont été identifiés grâce à la collaboration entre les entreprises, les pratiques de collaboration ne sont pas appropriées pour toutes les relations (Krause, 1999). En effet, à part les bénéfices, les risques sont également impliqués dans la collaboration. L'un des risques majeurs dans toutes les collaborations est le risque d'échec. Ce risque comprend la perte d'importants investissements en termes d'argent ou de temps, et des abandons de plans d'affaires, dans les cas où la collaboration est infructueuse. En outre, un risque inhérent est l'exposition d'informations à la concurrence. En effet, les entreprises doivent garder à l'esprit que le collaborateur potentiel peut devenir à un moment donné le partenaire d'un concurrent. Un autre risque tout aussi important est lié à la dépendance potentielle d'un acteur unique sur un des maillons de la chaîne. Une entreprise est, dans une plus ou moins grande mesure, tributaire d'une autre entreprise à travers un certain nombre de processus. En effet, de nombreux auteurs (Spekman et al., 1998 ; Adams et Goldsmith, 1999), ont fait valoir que dans le processus d'approvisionnement par exemple, plus un acheteur achète auprès d'un fournisseur, plus cet acheteur sera en mesure d'influencer ce fournisseur. Dans la plupart des cas de la littérature, la dépendance a été considérée comme un risque, qui est particulièrement élevé pour les petites entreprises collaborant avec les grandes entreprises. Finalement un risque inhérent à la collaboration est d'accroître la complexité opérationnelle. En effet la collaboration demande la mise en place de systèmes d'informations spécifiques ou l'homogénéisation des formats de données des entreprises.

La collaboration entre les organisations est censée présenter des opportunités pour l'amélioration du développement durable dans les chaînes logistiques. Dans un récent sondage, 90% de 3800 cadres et dirigeants de 113 pays ont indiqué qu'ils croyaient que la collaboration était nécessaire pour répondre aux défis du développement durable. Cependant, seulement 47% ont répondu que leurs entreprises étaient impliquées dans une collaboration en lien avec le développement durable. En outre, seulement 30% des personnes impliquées dans une collaboration estiment que celle-ci était réussie. Dans ce contexte, la collaboration est utilisée pour faire référence « à la capacité de travailler à travers les frontières organisationnelles pour construire et gérer des processus uniques à valeurs ajoutées pour mieux répondre aux besoins des parties prenantes ». Ces résultats sont également présentés dans une étude préliminaire réalisée dans le cadre du projet SCALE [3] par l'ensemble des partenaires dans laquelle 27 entreprises ont été interrogées au Royaume-Uni, aux Pays-Bas et en France. Les principaux goulots d'étranglement et les facteurs de succès ont été identifiés pour l'amélioration des indicateurs du développement durable. Il est intéressant de noter que la principale opportunité pour améliorer la durabilité des chaînes logistiques s'est avérée être la collaboration. Cependant, malgré les potentiels avantages de la collaboration, il existe de nombreux obstacles qui peuvent empêcher les entreprises de tirer pleinement parti d'une initiative de collaboration tels qu'un partage d'information inadapté, un manque de confiance, des indicateurs et des objectifs incompatibles et des préoccupations concernant le partage de gains.

Dans cette thèse, nous étudions les possibilités offertes par les problèmes de routage pour améliorer les performances des chaînes logistiques grâce au transport collaboratif. Le transport collaboratif est le partage de véhicules ou de tâches de transport par plusieurs entreprises. Le but du transport collaboratif est d'améliorer le taux d'utilisation des véhicules pour réduire les coûts de transport. Le transport de marchandises génère une part importante des émissions de CO_2, en raison du grand volume de marchandises transportées et de la fréquence croissante des voyages effectués pour livrer les marchandises aux clients finaux. Le taux de remplissage des véhicules pouvant être amélioré, la collaboration de

3. http ://sfcplatform.eu/articles/sustainability-in-food-supply-chains-scoping-study/

différentes entreprises dans le domaine du transport logistique permettrait de réduire les coûts de transport et les émissions de CO_2. Une telle collaboration pourrait être sous la forme de partage de véhicules entre plusieurs fournisseurs afin d'augmenter leur taux de remplissage. Plusieurs études de cas ont été réalisées dans lesquelles différents scénarios ont été simulés et optimisés dans le but de réduire les coûts de transport et les émissions de CO_2 potentielles suite à la mise en œuvre d'une collaboration.

La figure 1 présente la structure de la thèse. Pour les chapitres 2, 3 et 4, les caractéristiques des modèles étudiés sont résumées ainsi que leurs contributions. Cette thèse est divisée en quatre chapitres. Dans le premier chapitre, un état de l'art sur les problèmes de collectes et livraisons est effectué. Les notions générales des problèmes de routage sont présentées. Une classification de ces problèmes venant de la littérature est présentée. Différents travaux sur les problèmes de collectes et livraisons sans puis avec transbordements sont présentés. Ensuite, des travaux sur les problèmes de transport collaboratif sont décrits. Ces travaux présentent des problèmes pouvant se rapporter à des problèmes de collectes et livraisons. Enfin, les travaux réalisés dans le cadre de cette thèse sont positionnés par rapport aux travaux présentés dans ce chapitre.

Le deuxième chapitre propose deux méthodes approchées pour résoudre le problème de collectes et livraisons avec transbordements. Les deux méta-heuristiques choisies sont la recherche à large voisinage ou Large Neighbourhood Search (LNS) et l'algorithme génétique ou Genetic Algorithm (GA). Ces deux méthodes sont d'abord testées sur des instances issues de la littérature pour le problème de collectes et livraisons sans transbordement puis sur des instances issues de la littérature pour le problème de collectes et livraisons avec transbordements. Ceci permet de comparer ces deux méthodes avec les meilleurs résultats obtenus de la littérature.

Dans le troisième chapitre, un modèle linéaire en nombres entiers mixtes est proposé pour résoudre le Pickup and Delivery Vehicle Routing Problem (PDVRP) sans fenêtre de temps. Les caractéristiques du problème sont énoncées puis le modèle est décrit. Ce problème est utilisé afin de faciliter la collaboration entre plusieurs entreprises dans le domaine du transport. Il est d'abord testé sur des données générées aléatoirement. Ceci permet de faire varier des paramètres

tels que le nombre de fournisseurs, le nombre de clients ou le nombre de types de produits. Les résultats obtenus en appliquant un routage collaboratif sont comparés avec les résultats obtenus sans collaboration pour évaluer l'impact de la collaboration sur le transport. Ensuite le problème est testé dans un cadre réel. En effet, dans le cadre du projet SCALE, une entreprise a accepté de partager ses données afin d'améliorer la distribution de ses produits à ses clients dans le but de réduire les coûts et l'impact environnemental de leurs transports. Les résultats obtenus sont comparés avec la situation existante. Ce PDVRP est ensuite étendu pour prendre en compte les fenêtres de temps. Ce nouveau modèle est ensuite testé avec les données de la même entreprise que précédemment.

Dans le quatrième chapitre, un autre modèle linéaire en nombres entiers mixtes est proposé pour résoudre le PDVRP avec fenêtres de temps. La différence avec le précédent est l'ajout de nœuds de transbordement dans lesquels les marchandises peuvent être transférées d'un véhicule à l'autre. Il est également testé dans un cadre réel. Cette étude de cas est réalisée dans le cadre du projet SCALE avec trois entreprises voulant collaborer pour mutualiser leurs transports. Plusieurs scénarios sont testés et les résultats obtenus sont comparés avec la situation existante.

Les principales contributions de cette thèse sont résumées par les points suivants :

— Deux méthodes approchées efficaces sont proposées pour résoudre le Pickup and Delivery Problem with Transhipments (PDPT). Parmi ces deux méthodes, l'une d'elles, le LNS est déjà utilisé dans la littérature pour résoudre ce problème. Dans cette thèse, le LNS diffère des autres approches LNS par les fonctions d'insertion utilisées et par le critère d'acceptation. En revanche, la deuxième, le GA n'a jamais été utilisé à notre connaissance pour résoudre ce problème. Les deux méthodes proposées permettent d'obtenir de meilleurs résultats en termes de nombres de véhicules utilisés et de coûts que ceux de la littérature.

— Deux nouvelles formulations sont proposées pour le PDVRP. Une première formulation sans fenêtre de temps est proposée puis une formulation avec fenêtres de temps souples est proposée. Une formulation pour ce

problème prenant en compte à la fois plusieurs types de produits et une flotte hétérogène de véhicules, ainsi qu'une fonction objectif minimisant les émissions de CO_2 n'a pas encore été proposée, à notre connaissance, que ce soit sans ou avec fenêtres de temps.

— A l'aide de données générées aléatoirement et de données réelles, l'utilité et l'efficacité du PDVRP en préalable à la mise en place d'une planification du transport collaboratif sont montrées.

— La formulation du PDVRP est étendue pour autoriser le transbordement. A l'aide de données réelles, l'utilité et l'efficacité du transbordement dans la planification du transport collaboratif sont montrées également.

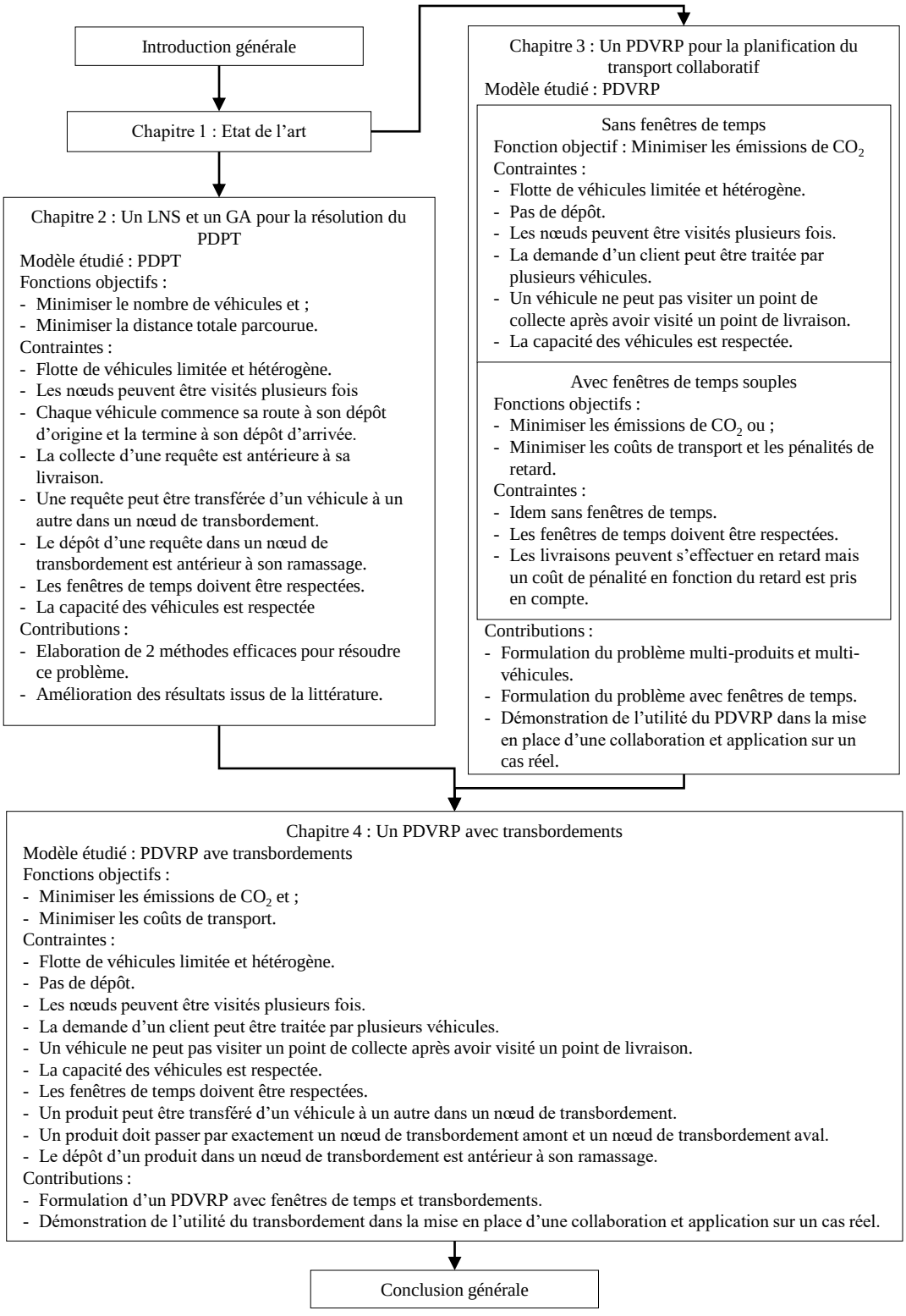

FIGURE 1 – Structure de la thèse

Chapitre 1

Etat de l'art

1.1 Introduction

Dans ce chapitre, nous présentons les notions et les travaux relatifs au cadre de cette thèse. Le but de cette thèse étant, rappelons le, de montrer l'intérêt que peuvent avoir les problèmes de routage et plus spécifiquement les problèmes de collectes et livraisons pour faciliter la collaboration dans la planification du transport logistique, ce chapitre se concentre donc sur cette thématique. Il est découpé en trois sections. La première section présente les problèmes de collectes et livraisons. Les notions relatives à ces problèmes sont d'abord introduites et une classification des problèmes de routage est ensuite présentée. Ensuite, après avoir formulé le problème de collectes et livraisons, plusieurs travaux représentatifs sont présentés. La deuxième section présente les problèmes de collectes et livraisons avec transbordement, ou transfert, lorsqu'on cherche à mutualiser les ressources ou à massifier les flux. Ce problème est défini puis une formulation est proposée. Ensuite les travaux relatifs à ce problème sont présentés. Enfin la troisième section comprend un état de l'art sur le problème de planification collaborative du transport de marchandises. Chaque article étudié dans cette section, sera lié à un problème présenté dans la classification. Pour pouvoir comparer les travaux étudiés, ceux-ci sont regroupés dans des tableaux qui résument leurs principales caractéristiques.

1.2 Les problèmes de collectes et livraisons

1.2.1 Présentation générale des problèmes de routage

Les problème de tournées de véhicules ou Vehicle Routing Problem (VRP) ont été depuis longtemps traités dans la littérature. Le premier d'entre eux a été défini dans (Dantzig et Ramser, 1959) sous le nom "The truck dispatching problem". Il y est présenté comme une généralisation du problème du voyageur de commerce ou Traveling Salesman Problem (TSP). Il s'agit de trouver un ensemble de routes, partant d'un ou plusieurs dépôts, qui satisfait les demandes de tous les clients ainsi que certaines contraintes opérationnelles. Dans le cas général, ces routes doivent être construites de telle sorte qu'elles minimisent la distance totale parcourue, mais d'autres objectifs existent pour ce problème (minimiser le nombre de véhicules, les temps de trajet, ...). Depuis, de nombreuses variantes sont apparues. Ces variantes peuvent différer de par leurs contraintes ou par de leurs objectifs. Ces problèmes possèdent cependant des caractéristiques communes. Trois éléments principaux sont communs à tous les problèmes de routage. Ces éléments sont les clients, les véhicules et le graphe associé au réseau routier.

Les clients

Les clients possèdent les caractéristiques suivantes :

— Une localisation. Celle-ci est souvent donnée par ses coordonnées géodésiques (latitude, longitude).

— Une demande. Cette demande correspond à la quantité de produits traités chez le client. Si le client est un point de collecte de cette marchandise, la demande est négative. Si le client est un point de livraison de cette marchandise, la demande est positive. Dans certains problèmes, plusieurs types de marchandises sont prises en compte et un client peut à la fois être un point de collecte et un point de livraison.

— Une fenêtre de temps. Le client peut être servi uniquement durant ce laps de temps. Cette fenêtre peut-être une contrainte dure, c'est-à-dire qu'elle

doit strictement être respectée. Elle peut aussi être une contrainte souple, le client peut alors être servi en dehors de cette fenêtre mais dans ce cas une pénalité sera attribuée. Cette pénalité peut être fixe ou proportionnelle à l'avance ou au retard par rapport à la fenêtre de temps.

— Un temps de service. C'est le temps nécessaire pour traiter la demande du client. Cela peut correspondre au temps de chargement ou de déchargement de la marchandise dans le véhicule. Dans certains problèmes, ce temps de service peut différer selon les véhicules.

Les deux dernières caractéristiques sont utilisées uniquement dans les problèmes prenant en compte les fenêtres de temps.

Les véhicules

Dans les problèmes de routage, les tournées sont réalisées par des véhicules. Il faut donc qu'elles respectent les caractéristiques des véhicules. Un véhicule réalise une ou plusieurs tournées. Selon les problèmes, la flotte de véhicules peut être de taille infinie ou finie. Elle peut également être homogène (tous les véhicules possèdent les mêmes caractéristiques) ou hétérogène (les véhicules possèdent des caractéristiques différentes). Les véhicules possèdent les caractéristiques suivantes :

— Un dépôt. Un véhicule doit commencer et terminer sa tournée par un dépôt fixé à l'avance. Les dépôts peuvent être aussi des points de collecte et de livraison dans certains problèmes. De plus, le dépôt de départ et celui d'arrivée peuvent être différents.

— Une capacité. Elle correspond à la quantité maximale de marchandises qu'un véhicule peut transporter. Elle peut être exprimée en volume, en poids ou en nombre de palettes. Dans certains problèmes, la capacité considérée peut être infinie (messagerie ou distribution de courriers . . .) car celle ci est non bloquante pour les objets transportés.

— Nombre de compartiments. Chaque compartiment est caractérisé par une quantité et un type de marchandises qu'il peut contenir (distribution de carburants par exemple). En règle générale, le nombre de compartiments des véhicules est de un.

— Un coût. L'utilisation d'un véhicule engendre un coût. Ce coût peut être fixe ou variable en fonction de la distance parcourue ou du temps de trajet.

Le réseau

Le réseau routier est souvent représenté par un graphe qui possède les caractéristiques suivantes :

— Ensemble de sommets. Les sommets du graphe représentent les dépôts et les sites des clients à servir.

— Ensemble d'arcs. Les arcs représentent les liaisons qui séparent deux sites clients sur un mode de transport particulier. Ces arcs peuvent être orientés dans le cas où ils ne peuvent être traversés que dans une seule direction, ou non orientés dans le cas où ils peuvent être parcourus dans les deux sens.

— Coût des arcs. Chaque arc possède un coût. Ce coût est en général la distance entre les deux sommets reliés par l'arc.

1.2.2 Classification des problèmes de routage

Une classification est donnée par Parragh et al. (2008). Ils généralisent le VRP en deux problèmes différents. Dans le premier problème, le Vehicle Routing Problem with Backhauls (VRPB), les transferts de marchandises se font depuis les dépôts vers les clients et depuis les clients vers les dépôts. Dans le second problème, le Vehicle Routing Problem with Pickups and Deliveries (VRPPD), les transferts se font entre les clients. Le VRPPD peut lui-même être divisé en deux catégories ; une première où les clients ne sont pas appariés, et une deuxième où les clients sont appariés, cette dernière trouvant des applications dans les transports postaux ou dans les transports de passagers. Avec le développement des sites de transbordements (transfert de marchandises d'un véhicule à un autre), nous pouvons également ajouter les problèmes de collectes et livraisons avec transbordement comme une généralisation des VRPPD. Berbeglia et al. (2007) donnent une autre nomenclature pour ces problèmes et distinguent trois catégories différentes. Tout d'abord, les problèmes "one-to-many-to-one" correspondent aux VRPB. Ensuite les problèmes "many-to-many" correspondent aux VRPPD

avec clients non appariés. Et enfin les problèmes "one-to-one" correspondent aux VRPPD avec clients appariés. La figure 1.1 résume cette classification. Dans cette figure, chaque classe est une extension de la classe située juste au dessus. Nous utiliserons la nomenclature donnée par Parragh et al. (2008) dans tout le reste du document. Nous présentons dans la suite les trois classes de problèmes, mais dans le cadre de nos travaux, nous nous concentrerons essentiellement sur les VRPPD.

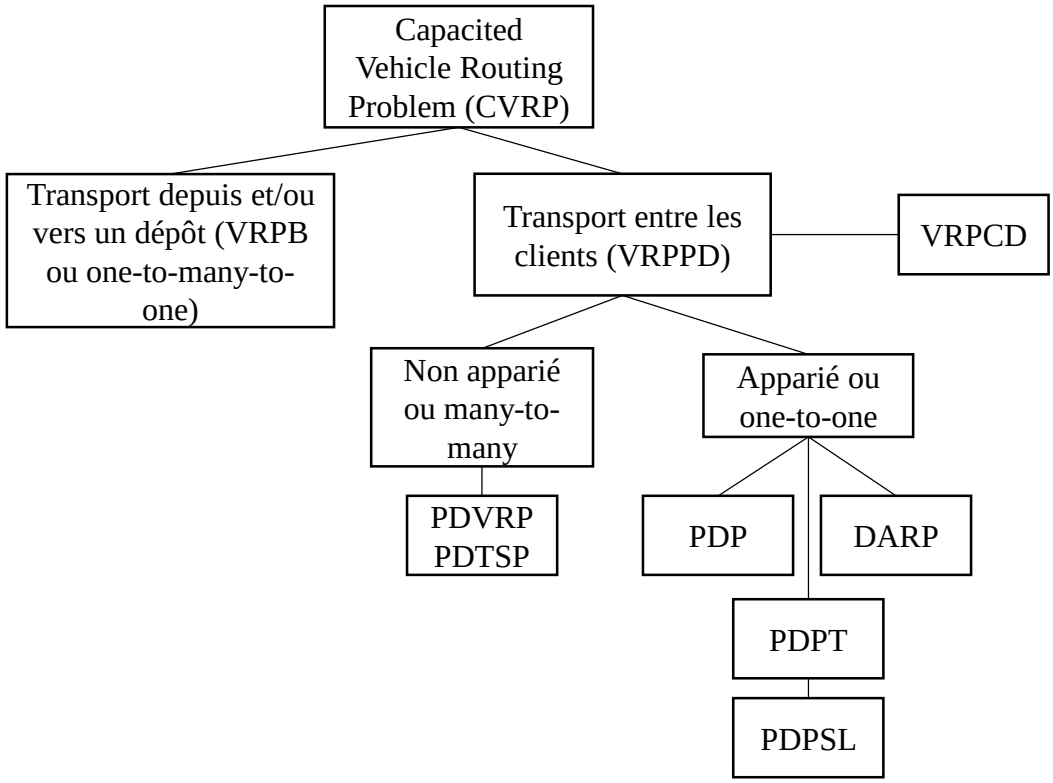

Figure 1.1 – Classification des problèmes de VRP (Parragh et al., 2008)

Le VRPB

Le VRPB est une extension du VRP où les transferts de marchandises se font depuis les dépôts vers les clients et depuis les clients vers les dépôts. Le développement du VRPB a été motivé par le fait qu'une réduction significative

des coûts pouvait être obtenue en combinant les livraisons chez les clients avec les collectes, et ce afin de réduire les retours à vide des véhicules. Il a été introduit par Goetschalckx et Jacobs-Blecha (1989). L'ensemble des clients est divisé en deux groupes. Le premier groupe contient les clients appelés clients "linehaul" qui réclament chacun une certaine quantité de marchandises. Le deuxième groupe contient les clients appelés clients "backhaul" qui ont chacun une certaine quantité de marchandises à collecter. Dans le cadre classique du VRPB, il existe une contrainte de précédence sur les clients. Si un véhicule sert à la fois des clients "linehaul" et des clients "backhaul", tous les clients "linehaul" doivent être servis avant les clients "backhaul". Mais des variantes existent où cette contrainte peut être relâchée et où les clients peuvent être à la fois de type "linehaul" et "backhaul".

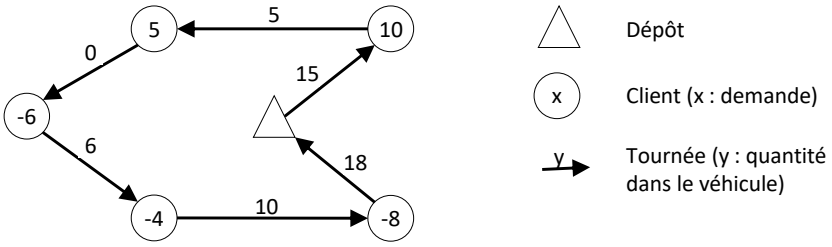

Figure 1.2 – Exemple de tournée du VRPB classique

La figure 1.2 présente un exemple de tournée de VRPB. Les clients avec une demande positive sont des clients "linehaul" et les clients avec des demandes négatives sont des clients "backhaul". Le véhicule démarre sa tournée du dépôt avec assez de marchandises pour satisfaire les demandes des deux clients "linehaul". Il livre ces deux clients puis part collecter la marchandise chez les trois clients "backhaul". Il rentre ensuite au dépôt avec les marchandises collectées.

Le VRPPD avec clients non appariés

Dans ce problème, le transfert de marchandises se fait entre les clients qui sont des points de livraison ou des points de collecte. Dans le cas où une seule tournée doit être réalisée, le problème est appelé Pickup and Delivery Trave-

ling Salesman Problem (PDTSP), tandis que lorsque plusieurs véhicules sont disponibles, le problème est appelé PDVRP. Dans ces problèmes, un point de livraison peut recevoir la marchandise venant de n'importe quel point de collecte. De la même manière, un point de collecte peut livrer n'importe quel point de livraison. La quantité de marchandises disponibles aux points de collecte doit donc être supérieure ou égale à la demande totale des points de livraison pour que celle-ci puisse être satisfaite. Contrairement au problème précédent, il n'y a pas de contrainte de précédence sur les points de collecte et de livraison. Un véhicule peut visiter un point de collecte puis un point de livraison puis un point de collecte à nouveau. Il faut tout de même veiller à ce que la capacité du véhicule soit suffisante lors de la visite d'un point de collecte et que la quantité de marchandises dans le véhicule soit suffisante pour satisfaire la demande lors de la visite d'un point de livraison.

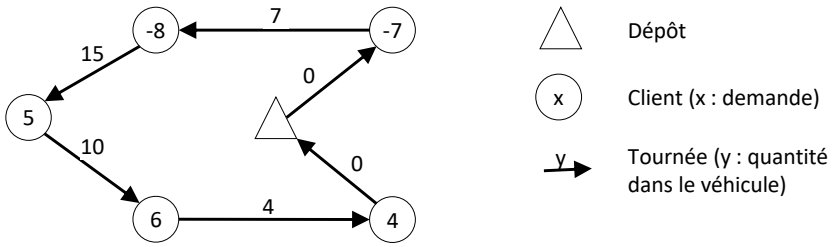

Figure 1.3 – Exemple de tournée du PDVRP

La figure 1.3 présente un exemple de tournée de PDVRP. Les clients avec une demande négative sont les points de collecte et les clients avec une demande positive sont les points de livraison. Le véhicule démarre vide du dépôt et part collecter la marchandise dans les deux points de collecte. Il part ensuite livrer cette marchandise dans les trois points de livraison. Enfin il rentre vide au dépôt.

Le VRPPD avec clients appariés

Dans ce problème, le transfert de marchandises se fait entre les clients également. Mais contrairement au PDTSP et au PDVRP, les clients sont associés par paires : un point de collecte est associé avec un point de livraison et la

marchandise provenant du point de collecte doit obligatoirement aller dans le point de livraison associé. Une nouvelle notion apparait pour ces problèmes : les requêtes. Une requête est composée d'un couple de clients (un point de collecte et un point de livraison) et d'une quantité. Résoudre ces problèmes revient donc à traiter l'ensemble des requêtes. Ces problèmes sont classés en deux groupes. Les problèmes traitant de transport de marchandises sont appelés les Pickup and Delivery Problem (PDP) (messageries express de colis, ...) et les problèmes traitant de transport de personnes sont appelés les Dial-A-Ride Problem (DARP) (transport d'handicapés, taxi à la demande, ...). La différence entre ces deux problèmes est l'ajout dans le DARP de contraintes ou d'objectifs qui prennent en compte les souhaits des personnes transportées.

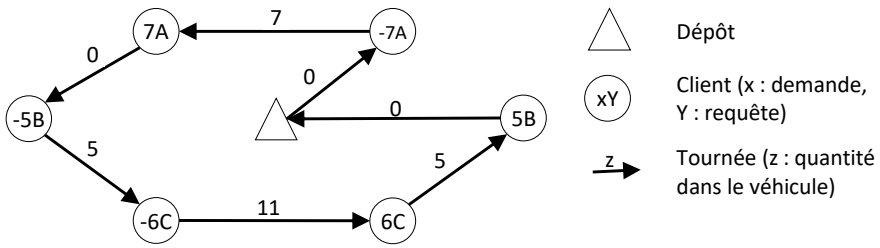

FIGURE 1.4 – Exemple de tournée du PDP

La figure 1.4 présente un exemple de tournée du PDP. Les lettres indiquent une requête et donc les clients qui sont appariés. Les demandes négatives désignent les points de collecte et les demandes positives désignent les points de livraison. Le véhicule part vide du dépôt pour collecter la requête A. Il livre cette requête puis part collecter les requêtes B et C. Il livre les requêtes C et B et rentre vide au dépôt.

1.2.3 Modélisation du VRPPD

Nous introduisons dans cette section les modèles mathématiques issus de la littérature pour les deux classes du VRPPD. Ces problèmes peuvent être définis sur un graphe $G = \{N, A\}$ où N est l'ensemble des nœuds et A est l'ensemble des arcs. L'ensemble N est composé de plusieurs sous-ensembles tel que $N =$

$V^+ \cup V^- \cup O \cup D$ avec $v^+ \in V^+$ et $v^- \in V^-$ le dépôt de départ et le dépôt d'arrivée du véhicule v, O l'ensemble des points de collecte et D l'ensemble des points de livraison et $P \cap D = \varnothing$. L'ensemble des arcs peut être défini par $A = \{(i,j) : i,j \in N, i \notin V^-, j \notin V^+, i \neq j\}$. Le but du problème est de trouver un ensemble de routes partant du dépôt de départ et arrivant au dépôt d'arrivée et qui satisfait toutes les demandes des clients. Nous disposons pour cela d'une flotte hétérogène V de véhicules ayant chacun une capacité limitée C^v. Ces véhicules doivent partir vides et arriver vides au dépôt. Un seul type de produit est considéré, chaque nœud est visité une seule fois et les visites chez les clients doivent respecter des fenêtres de temps. Nous utilisons la notation suivante :

P ensemble des points de collecte

D ensemble des points de livraison

V ensemble des véhicules

V^+ ensemble des dépôts de départ

V^- ensemble des dépôts d'arrivée

v^+ dépôt de départ du véhicule v

v^- dépôt d'arrivée du véhicule v

q_i demande au nœud i ; les demandes négatives sont associées aux nœuds de collecte et les demandes positives sont associées aux nœuds de livraison ; la demande aux dépôts est nulle

e_i heure minimum de début de service au nœud i

l_i heure maximum de début de service au nœud i

d_i temps de service au nœud i

c_{ij}^v coût pour traverser l'arc (i,j) avec le véhicule v

t_{ij}^v temps pour traverser l'arc (i,j) avec le véhicule v

C^v capacité du véhicule v

M une grande constante

Les variables de décision sont définies comme suit :

x_{ij}^v vaut 1 si l'arc (i,j) est traversé par le véhicule v ; 0 sinon

Q_i^v la charge du véhicule v quand il quitte le nœud i

B_i^v début de temps de service du véhicule v au nœud i

Nous présentons d'abord le modèle pour le PDVRP puis le modèle pour le PDP

Modèle pour le PDVRP

Le modèle pour les PDVRP est issu du modèle proposé dans Parragh et al. (2008) et peut être formulé comme suit :

$$\min \sum_{v \in V} \sum_{(i,j) \in A} c_{ij}^v x_{ij}^v \tag{1.1}$$

sc.

$$\sum_{v \in V} \sum_{j:(i,j) \in A} x_{ij}^v = 1 \qquad \forall i \in P \cup D, \tag{1.2}$$

$$\sum_{j:(v^+,j) \in A} x_{v^+j}^v = 1 \qquad \forall v \in V, \tag{1.3}$$

$$\sum_{i:(i,v^-) \in A} x_{iv^-}^v = 1 \qquad \forall v \in V, \tag{1.4}$$

$$\sum_{i \in V^+ \setminus \{v^+\}} \sum_{j:(i,j) \in A} x_{ij}^v = 0 \qquad \forall v \in V, \tag{1.5}$$

$$\sum_{i:(i,j) \in A} x_{ij}^v - \sum_{i:(j,i) \in A} x_{ji}^v = 0 \qquad \forall j \in P \cup D, \forall v \in V, \tag{1.6}$$

$$B_j^v \geq B_i^v + d_i + t_{ij}^v - M(1 - x_{ij}^v) \qquad \forall (i,j) \in A, \forall v \in V, \tag{1.7}$$

$$e_i \leq B_i^v \leq l_i \qquad \forall i \in N, \forall v \in V, \tag{1.8}$$

$$Q_j^v \geq Q_i^v + q_j - M(1 - x_{ij}^v) \qquad \forall (i,j) \in A, \forall v \in V, \tag{1.9}$$

$$Q_j^v \leq Q_i^v + q_j + M(1 - x_{ij}^v) \qquad \forall (i,j) \in A, \forall v \in V, \tag{1.10}$$

$$\max\{0, q_i\} \leq Q_i^v \leq \min\{C^v, C^v + q_i\} \qquad \forall i \in N, \forall v \in V, \tag{1.11}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall v \in V, \tag{1.12}$$

La fonction objectif 1.1 minimise le coût total de toutes les tournées effectuées par les véhicules. La contrainte 1.2 assure que les nœuds sont visités une et une seule fois. La contrainte 1.3 assure que chaque véhicule commence sa route au dépôt de départ et la contrainte 1.4 assure que chaque véhicule finit sa route au dépôt d'arrivée. Si le véhicule visite uniquement le dépôt de départ et le dépôt d'arrivée, cela signifie qu'il n'est pas utilisé. La contrainte 1.5 interdit à un véhicule de partir d'un dépôt autre que le sien. La contrainte 1.6 est la contrainte de conservation des flux. Si un véhicule arrive à un nœud qui n'est pas un dépôt, ce véhicule doit absolument quitter le nœud. Les contraintes 1.7 et 1.8 sont les contraintes liées au temps. La contrainte 1.7 assure que les temps de trajet et les temps de service sont respectés. De plus, cette contrainte permet d'éliminer les sous-tours. La contrainte 1.8 assure que les fenêtres de temps sont respectées. Les contraintes 1.9 à 1.11 sont les contraintes de capacité. Elles assurent que la charge d'un véhicule ne dépasse pas sa capacité pendant la tournée.

Modèle pour le PDP

Pour modéliser le PDP, nous avons besoin d'une information supplémentaire dans la notation. En effet, les clients étant appariés il est nécessaire de connaître pour chaque nœud de collecte i appartenant à O le nœud correspondant de livraison j appartenant à D. Nous désignons donc par n la taille de l'ensemble O. Avec cette notation supplémentaire, nous pouvons établir les hypothèses suivantes :

— Si $i \in O$ alors $i + n \in D$ et i est apparié à $i + n$

— Les clients étant appariés alors $|O| = |D| = n$

— $\forall i \in O$, $q_i = -q_{i+n}$

— Une requête est composée d'un point de collecte $i \in O$, d'un point de livraison $i + n \in D$ et d'une quantité q_i

Pour modéliser le PDP, nous pouvons reprendre la fonction objectif 1.1 ainsi que les contraintes 1.2 à 1.12. De plus, nous ajoutons l' ensemble de contraintes suivantes :

$$\sum_{j:(i,j)\in A} x_{ij}^{v} - \sum_{j:(i+n,j)\in A} x_{i+n,j}^{v} = 0 \qquad \forall i \in P, \forall v \in V, \tag{1.13}$$

$$B_i^v \leq B_{i+n}^v \qquad \forall i \in N, \forall v \in V, \tag{1.14}$$

La contrainte 1.13 assure que le point de collecte et le point de livraison d'une requête sont servis par le même véhicule. La contrainte 1.14 assure que la collecte d'une requête est réalisée avant sa livraison.

1.2.4 Etude de différents travaux sur les VRPPD

Dans cette section nous présentons les travaux réalisés sur les VRPPD. Ce sont des problèmes assez présents dans la littérature. Plusieurs états de l'art ont déjà été réalisés sur ce sujet. Berbeglia et al. (2007) proposent une étude des VRPPD et des VRPB dans leur ensemble. Parragh et al. (2008) ont étudié les VRPPD. Ces deux travaux ont chacun proposé une classification des VRPPD. De leur côté, Cordeau et al. (2008) ont réalisé une revue de littérature uniquement sur les VRPPD avec clients appariés. Enfin récemment Battarra et al. (2014) ont proposé une étude des VRPB et VRPPD dans le cadre du transport de marchandises uniquement. Les travaux étudiés sont résumés dans le tableau 1.1.

La première colonne rappelle les noms des auteurs et l'année de la publication. La deuxième colonne donne le type de problème étudié selon la terminologie qui sera introduite dans la section 1.2.2. La troisième colonne précise la ou les fonctions objectifs utilisées. La quatrième colonne donne la méthode utilisée pour résoudre le problème. La cinquième colonne donne la taille en nombre de nœuds de collecte et de livraison de la plus grande instance résolue. La sixième colonne indique si le problème est mono ou multi-produits. La septième colonne indique le nombre de dépôts du problème. La huitième colonne donne les caractéristiques des véhicules comme le nombre de véhicules disponibles, la capacité ou le type de flotte. Une flotte homogène ne comporte que des véhicules identiques, tandis qu'une flotte hétérogène comporte des véhicules qui peuvent avoir des capacités, des dépôts ou des coûts différents. La neuvième colonne

indique si les fenêtres de temps sont prises en compte ou non. La dixième colonne indique le nombre de fois maximum qu'un nœud peut accueillir un véhicule. Enfin la dernière colonne indique si une requête peut être éclatée et donc si celle-ci peut être transportée par plusieurs véhicules simultanément.

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule							TW	Nb visites par client		Split
					1	Plusieurs	1	Plusieurs	Nb.		Cap.		Flotte				1	Plusieurs	
									1	Limité	Illimité	Limité	Illimité	Homogène	Hétérogène				
Benavent et al. (2015)	PDP	Min temps	Heuristique	60	X		X			X		X		X		X	X		
Bent et Van Hentenryck (2006)	PDP	Min nb véh + min coût	Recuit simulé + LNS	600	X		X			X		X		X		X	X		
Bettinelli et al. (2014)	PDP	Min cost + retard + avance	Branch & price	150	X			X		X		X			X	X	X		
Chen et al. (2014)	PDVRP	Min coût	Heuristique + VNS	10		X	X			X		X		X				X	X
Créput et al. (2004)	PDP	Min nb véh + min coût + min temps trajet	GA	100	X		X			X		X			X	X	X		
Dridi et al. (2011)	PDP	Min cost + retard	GA	100	X		X			X		X			X	X	X		
Dror et al. (1998)	PDVRP	Min distance	Exacte et heuristique	8	X			X		X		X		X			X		X
Dumas et al. (1991)	PDP	Min coût	Génération de colonnes	60	X			X		X		X			X	X	X		
Han et al. (2015)	PDTSP	Min distance	Heuristique	1000	X		X		X			X		X			X		
Hernández-Pérez et Salazar-González (2004a)	PDTSP	Min distance	Branch & cut	50	X		X			X		X		X			X		
Hernández-Pérez et SalazarGonzález (2004b)	PDTSP	Min distance	Heuristiques	500	X		X			X		X		X			X		
Hernández-Pérez et Salazar-González (2007)	PDTSP	Min distance	Branch & cut	100	X		X			X		X		X			X		
Hernández-Pérez et Salazar-González (2014)	PDTSP	Min distance	Branch & cut	50		X	X			X		X		X			X		

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule								TW	Nb visites par client		Split
					1	Plusieurs	1	Plusieurs	Nb.			Cap.		Flotte				1	Plusieurs	
									1	Limité	Illimité	Limité	Illimité	Homogène	Hétérogène					
Hernández-Pérez et al. (2009)	PDTSP	Min distance	GRASP & VND	500	X		X		X			X		X				X		
Hernández-Pérez et al. (2016)	PDTSP	Min distance	Recherche locale	500		X	X		X			X		X				X		
Hosny et Mumford (2010)	PDTSP	Min distance	ILS/SA	500	X		X		X			X		X				X		
Kalantari et al. (1985)	PDP	Min distance	Branch & bound	30	X		X		X	X		X	X	X				X		
Lau et Liang (2001)	PDP	Min nb véh + min distance	Recherche tabou	100	X		X			X		X		X			X	X		
Li et Lim (2003)	PDP	Min nb véh + min distance + min temps de trajet + min temps d'attente	Recherche tabou et recuit simulé	100	X		X			X		X		X			X	X		
Lu et Dessouky (2004)	PDP	Min cost	Branch & cut	50	X			X		X		X			X		X	X		
Lu et Dessouky (2006)	PDP	Min nb véh + coût	Heuristique	100	X		X			X		X		X			X	X		
Mladenovic et al. (2012)	PDTSP	Min distance	VNS	1000	X		X		X			X		X				X		
Nagata et Kobayashi (2010)	PDP	Min nb véh + min distance	Algorithme mémétique	1000	X		X			X		X		X			X	X		
Nanry et Barnes (2000)	PDP	Min temps	Recherche tabou		X		X			X		X		X			X	X		
Nowak et al. (2008)	PDP	Min coût	Heuristique	30	X		X			X		X		X					X	X
Pankratz (2005)	PDP	Min distance	GA	100	X		X			X		X		X			X	X		

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule							TW	Nb visites par client		Split
					1	Plusieurs	1	Plusieurs	Nb.		Cap.		Flotte				1	Plusieurs	
									1	Limité	Illimité	Limité	Illimité	Homogène	Hétérogène				
Ritzinger et al. (2016)	DARP	Min temps de trajet	Programmation dynamique + LNS		×		×			×		×		×		×	×		
Ropke et Cordeau (2009)	PDP	Min cost	Branch & cut & price	1000	×		×			×		×		×		×	×		
Ropke et Pisinger (2006)	PDP	Min coût	ALNS	1000	×			×		×		×			×	×	×		
Ropke et al. (2007)	PDP, DARP	Min cost	Branch & cut	194	×		×			×		×			×	×	×		
Salazar-González et Santos-Hernández (2015)	PDTSP	Min distance	Branch & cut	50	×		×		×			×		×				×	×
Savelsbergh et Sol (1995)	PDP	Min distance, temps, ou nombre de véhicule			×			×		×		×			×	×	×		
Shi et al. (2009)	PDVRP	Min distance	GA	500	×			×		×		×			×		×		
Zhao et al. (2009)	PDTSP	Min distance	GA & recherche locale	500	×		×		×			×		×			×		

TABLEAU 1.1 – Résumé des travaux sur le VRPPD

PDVRP avec clients non appariés

Dans cette catégorie nous distinguons les problèmes avec un seul véhicule appelés PDTSP et les problèmes avec plusieurs véhicules appelés PDVRP. Le PDTSP a été introduit par Hernández-Pérez et Salazar-González (2003) et Hernández-Pérez et Salazar-González (2004a) et est nommé "one commodity pickup-and-delivery traveling salesman problem" (1-PDTSP). Ils proposent une formulation mathématique pour ce problème. Chaque client doit être visité une seule fois et l'objectif est de réduire la distance totale parcourue par le véhicule. Les fenêtres de temps ne sont pas prises en compte. Ils proposent ensuite une méthode exacte pour résoudre ce problème : un algorithme "branch-and-cut". Pour améliorer l'efficacité de cette méthode ils ont développé également quelques heuristiques afin d'obtenir de bonnes solutions faisables. Avec cet algorithme, ils sont capables de trouver les solutions optimales sur des instances comprenant jusqu'à 50 clients en quelques minutes (exception faite pour une instance résolue en deux heures).

Pour résoudre des instances plus grandes, Hernández-Pérez et Salazar-González (2004b) proposent deux heuristiques pour le PDTSP avec les mêmes contraintes. La première est basée sur un algorithme glouton. La seconde est une méthode exacte de type" branch-and-cut" appliquée sur un espace de recherche restreint prédéfini dans le but de trouver un optimum local. Ces deux méthodes sont d'abord testées sur des petites instances inférieures à 60 clients pour lesquelles la solution optimale est connue. La première méthode permet d'avoir des solutions avec un écart inférieur à 2% par rapport à la solution optimale pour un temps de calcul inférieur à une seconde. La seconde méthode permet de trouver des solutions avec un écart inférieur à 1% par rapport à la solution optimale pour un temps de calcul inférieur à une minute. Ils ont ensuite testé ces méthodes sur des instances plus larges jusqu'à 500 clients. La seconde méthode est en moyenne plus performante.

Dans Hernández-Pérez et Salazar-González (2007) le "branch-and-cut" est amélioré en ajoutant des contraintes valides pour renforcer la formulation. De nouvelles instances avec une taille de 100 clients sont résolues en moins de deux heures.

Depuis 2009, plusieurs méthodes approchées ont été étudiées pour ce problème. Ainsi, Hernández-Pérez et al. (2009) proposent une nouvelle méta-heuristique pour ce problème. Celle-ci est composée de deux méthodes. A chaque itération, une phase de construction est réalisée grâce à la méthode Greedy Randomized Adaptive Search Procedure (GRASP). Ensuite un algorithme Variable Neighborhood Descent (VND) est utilisé pour améliorer la solution. Enfin, à la fin de toutes les itérations un autre algorithme VND est appliqué. Grâce à cette méta-heuristique, 113 solutions sur 150 sont améliorées par rapport aux méthodes utilisées dans Hernández-Pérez et Salazar-González (2004b). Zhao et al. (2009) proposent un GA combiné à une recherche locale pour ce problème. La recherche locale est ajoutée dans le GA pour accélérer la convergence. Ils représentent leurs solutions comme une liste de nœuds à visiter, les nœuds étant classés par ordre de visite. Ils testent leur méthode sur les instances de Hernández-Pérez et Salazar-González (2004b). Leurs résultats sont légèrement meilleurs en moyenne que ceux de Hernández-Pérez et al. (2009) pour un temps de calcul réduit. Hosny et Mumford (2010) proposent une méthode hybride Iterative Local Search (ILS)/recuit simulé. Ils utilisent les mêmes instances que les articles précédents et améliorent encore un peu les résultats mais le temps de calcul explose pour les grandes instances (jusqu'à 42 heures). Mladenovic et al. (2012) proposent un Variable Neighborhood Search (VNS) qui donnent de bons résultats sur les grandes instances en quelques minutes. Ils utilisent leur méthode pour résoudre également des nouvelles instances comprenant 1000 clients.

Salazar-González et Santos-Hernández (2015) introduisent la possibilité de réaliser des collectes ou des livraisons partielles et donc de passer plusieurs fois par le même nœud. Ils proposent une formulation et un algorithme "branch-and-cut" pour ce problème. Ils résolvent des instances avec des tailles allant jusqu'à 50 clients. Dans les articles précédents traitant du PDTSP, le dépôt était considéré comme un point de collecte et un point de livraison, c'est-à-dire que le véhicule peut partir et retourner au dépôt non vide. Han et al. (2015) proposent une formulation et une heuristique pour le PDTSP avec le dépôt non considéré comme point de collecte et de livraison et résolvent des instances avec 1000 clients.

Hernández-Pérez et Salazar-González (2014) introduisent le PDTSP avec plusieurs types de produit. Ils proposent une formulation et un algorithme "branch-and-cut" pour ce problème et résolvent des instances avec 50 clients. Dans Hernández-Pérez et al. (2016), les auteurs proposent une heuristique basée sur des recherches locales pour ce problème et résolvent des instances avec 500 clients.

Contrairement au PDTSP, le PDVRP (avec plusieurs véhicules) a été beaucoup moins étudié dans la littérature. Les premiers à traiter ce problème sont Dror et al. (1998). Leur problème est appliqué à la redistribution de voitures électriques en libre-service. Ils autorisent la livraison partielle de la demande et la flotte des véhicules qui transportent les voitures électriques peut être hétérogène. Ils proposent une modélisation du problème et le résolvent de manière exacte à l'aide de solveurs pour des tailles allant jusqu'à 8 clients. Ils proposent aussi une heuristique pour la résolution approchée. Shi et al. (2009) proposent un GA pour résoudre ce problème avec des instances à 500 clients. Ils ajoutent une contrainte pour limiter la taille des routes. Chen et al. (2014) proposent une formulation et une heuristique pour résoudre le PDVRP avec livraison éclatée, c'est-à-dire avec la possibilité de livrer un client avec plusieurs véhicules différents.

PDVRP avec clients appariés

Cette catégorie peut être séparée en deux types de problèmes : premièrement le PDP qui concerne le transport de marchandises et deuxièmement le DARP qui concerne le transport de personnes. Ces deux problèmes sont assez présents dans la littérature et plusieurs revues de littérature ont été réalisées spécifiquement sur ces problèmes (Desaulniers et al., 2001 ; Cordeau et al., 2007 ; Cordeau et al., 2008). Savelsbergh et Sol (1995) proposent une formulation uniformisée du PDP avec un ou plusieurs véhicules, avec différents types de fonctions objectifs et différents types de contraintes. Parmi les fonctions nous pouvons citer la minimisation de la distance, du nombre de véhicules, du temps de trajet total... Les différents types de contraintes concernent les contraintes de fenêtres de temps. Ces contraintes peuvent être appliquées aux clients, aux requêtes ou aux véhicules. Ils donnent aussi un rapide aperçu des solutions existantes.

Plusieurs méthodes exactes ont été développées pour résoudre le PDP. Kalantari et al. (1985) proposent un algorithme "branch-and-bound" pour le PDP avec un seul et plusieurs véhicules. Ils traitent les cas pour lesquels les véhicules ont une capacité finie ou infinie. Avec un seul véhicule de capacité infinie, ils sont capables de résoudre des instances jusqu'à 30 clients. Dumas et al. (1991) donnent la modélisation du PDP avec un seul dépôt, une flotte homogène et des fenêtres de temps. Ils proposent également une méthode exacte basée sur la génération de colonnes qui gère les cas avec plusieurs dépôts et une flotte hétérogène. Grâce à leur méthode, ils résolvent de manière optimale des instances de 60 clients. Une première méthode "branch-and-cut" est proposée par Lu et Dessouky (2004). Ils résolvent de manière optimale des instances avec 50 clients. Une méthode "branch-and-cut" est également proposée par Ropke et al. (2007). Deux formulations différentes sont données pour résoudre le PDP et le DARP. Dans la première formulation, des variables temporelles sont utilisées. Dans la deuxième formulation, le nombre de variables est réduit en remplaçant les contraintes de temps par l'ajout de routes infaisables. La méthode "branch-and-cut" permet de résoudre des instances de 150 clients pour le PDP et de 194 clients pour le DARP. Cette méthode est améliorée par Ropke et Cordeau (2009) en utilisant une méthode "branch-and-cut-and-price". Trois instances à 1000 clients sont résolues de manière optimale en une dizaine de minutes. Plus récemment, Bettinelli et al. (2014) proposent un algorithme "branch-and-price" pour le PDP avec plusieurs dépôts, une flotte hétérogène de véhicules et des fenêtres de temps souples. Si les fenêtres de temps ne sont pas respectées, une pénalité proportionnelle à l'avance ou au retard est incluse dans la fonction objectif. Cette méthode permet de résoudre les instances de Ropke et al. (2007) pour ce problème.

Le PDP étant NP-complet et difficile à résoudre pour les instances de grande taille, plusieurs méthodes approchées ont été proposées. Nanry et Barnes (2000) proposent une des premières méthodes approchées basée sur une recherche tabou. Pour explorer le voisinage d'une solution, trois opérateurs sont proposés. Dans le premier opérateur, une requête est enlevée d'un véhicule puis placée dans un autre. Dans le deuxième opérateur, deux requêtes de deux véhicules différents sont échangées. Dans le troisième opérateur, une requête est placée

ailleurs dans sa propre route. La violation des contraintes de fenêtres de temps et/ou de capacité est autorisée mais une pénalité est introduite dans la fonction objectif dans ce cas. Avec cette méthode, des solutions sont obtenues en quelques minutes sur des instances de 100 clients. Cette méthode est améliorée par Lau et Liang (2001) en proposant une autre heuristique pour construire la solution initiale. L'amélioration de cette solution est toujours réalisée par une recherche tabou avec les trois mêmes opérateurs. Li et Lim (2003) proposent également une recherche tabou reprenant les trois opérateurs. Cependant, à chaque itération de la recherche tabou, la solution est améliorée à l'aide d'une heuristique de type recuit simulé. Pour tester leur méthode, 56 instances de 100 clients ont été générées à partir des instances du VRP. Les temps d'exécution varient de quelques minutes à plus d'une heure. Un GA est proposé par Créput et al. (2004). Un individu est simplement représenté par une liste de routes. La méthode est testée sur les instances de Li et Lim (2003). Un GA est également proposé par Pankratz (2005). Dans leur approche, un gène ne représente pas une requête mais plutôt un groupe de requêtes. Toutes les requêtes comprises dans un groupe sont traitées par le même véhicule. L'ordre de visite des requêtes n'est donc pas déterminé par le GA mais par une heuristique qui est appelée lors du décodage d'une solution. Les solutions de quelques instances de Li et Lim (2003) sont améliorées avec des temps de calcul de quelques minutes. Lu et Dessouky (2006) proposent une heuristique constructive et testent leur méthode sur les instances de Li et Lim (2003). Bent et Van Hentenryck (2006) proposent un algorithme hybride en deux étapes. Dans la première étape, un recuit simulé est utilisé dans le but de réduire le nombre de véhicules. Dans la deuxième étape, un LNS est utilisé pour réduire les coûts de transports. De bons résultats sont obtenus sur les instances de Li et Lim (2003) jusqu'à 600 clients. Peu après, Ropke et Pisinger (2006) proposent un LNS adaptatif. Après avoir généré une solution faisable, un certain nombre de requêtes sont enlevées et réinsérées suivant plusieurs méthodes à chaque itération. Cet algorithme permet d'améliorer les solutions de plusieurs instances de Li et Lim (2003) jusqu'à 1000 clients. Un autre GA est proposé par Dridi et al. (2011) pour résoudre le PDP avec plusieurs objectifs. Le premier objectif est de minimiser le coût total du transport et le deuxième objectif est de minimiser le retard des véhicules. Dans leur approche,

une solution est représentée par deux vecteurs. Le premier vecteur contient l'ordre de visite des nœuds et le deuxième contient le nombre de nœuds à visiter pour chaque véhicule. Toujours sur les algorithmes évolutionnaires, Nagata et Kobayashi (2010) proposent un algorithme mémétique. La partie évolutionnaire de l'algorithme repose sur un crossover qui échange des routes entre deux parents. Les enfants ainsi générés sont améliorés à l'aide d'une recherche locale. A chaque itération de cette recherche locale, une requête est enlevée de la solution et replacée à un endroit moins coûteux. Cette étape est répétée tant que la solution peut être améliorée. De meilleurs résultats sont obtenus sur plusieurs instances par rapport à Ropke et Pisinger (2006) mais avec des temps de calcul plus longs.

Il existe également des travaux traitant du PDP mais avec des contraintes supplémentaires. Parmi ces travaux, le DARP est la variante la plus traitée. Le DARP traitant du transport de personnes, la majeure différence avec le PDP est l'ajout d'une contrainte qui permet de limiter le temps de trajet d'une personne. Un état de l'art sur ce problème est proposé par Cordeau et Laporte (2007). Plus récemment, une méthode exacte de programmation dynamique et une méthode approchée hybride basée sur la programmation dynamique est proposée par Ritzinger et al. (2016). Une autre variante est le PDP avec contraintes de chargement. Dans le PDP classique, l'ordre dans lequel les requêtes sont chargées et déchargées dans un véhicule n'a pas d'importance. Dans le PDP avec contraintes de chargement, la dernière requête chargée dans un véhicule doit être la première à être décharger (Benavent et al., 2015). Nowak et al. (2008) introduisent une variante dans laquelle les requêtes peuvent être divisées et traitées par plusieurs véhicules.

1.3 Le PDPT

Le PDPT est une variante du PDP. Dans le PDP, une requête est servie par un et un seul véhicule. Dans le PDPT, nous ajoutons la possibilité de traiter une requête avec plusieurs véhicules. Une requête peut être transférée d'un véhicule à un autre dans des nœuds spécifiques appelés points de transbordement. La figure 1.5 montre un exemple dans lequel la requête C est transférée d'un véhicule à

un autre. Cette pratique peut permettre de générer des réductions du coût de transport. Dans cette section, nous présentons les différences existantes entre le PDP et le PDPT ainsi que les caractéristiques du PDPT. Nous donnons ensuite une formulation du PDPT et présentons les travaux traitant de ce problème.

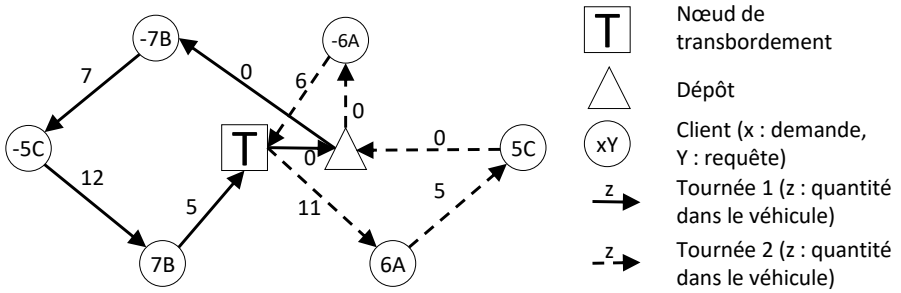

Figure 1.5 – Exemple de tournée du PDPT

La principale différence entre les deux problèmes est donc l'utilisation d'un nouvel ensemble de nœuds appelés points de transbordement ou de transfert. Ces nœuds comportent plusieurs différences avec les nœuds de collecte et de livraison. Tout d'abord, un nœud de collecte ou de livraison ne peut être visité qu'une seule et unique fois par un véhicule (sauf dans le cas du split delivery) alors qu'un nœud de transbordement peut recevoir la visite de plusieurs véhicules et même dans certains cas plusieurs visites d'un même véhicule. Ensuite un nœud de collecte (respectivement de livraison) ne sert qu'à la collecte (respectivement qu'à la livraison) alors qu'un nœud de transbordement accueille à la fois des collectes et des livraisons. Une autre différence est qu'une requête n'est pas forcément servie par un seul véhicule mais peut être transportée par plusieurs véhicules successivement. Ces différences apportent des caractéristiques propres au PDPT liées aux fenêtres de temps. En effet, il y a une contrainte de précédence à respecter au niveau des points de transbordement. Si un véhicule dépose une requête dans un point de transbordement, le véhicule qui la récupère doit forcément collecter la requête après le dépôt de celle-ci. Il peut arriver avant mais il devra alors attendre. Ceci entraîne une dépendance du deuxième véhicule envers le premier et toute modification apportée au premier véhicule avant le point de transbordement affectera donc le deuxième véhicule. De plus cet effet

est transitif puisque tous les véhicules dépendant du deuxième seront aussi impactés par cette modification. Enfin si deux véhicules s'échangent plusieurs requêtes, ils doivent être au même instant au point de transbordement et sont donc dépendants l'un de l'autre.

1.3.1 Modèle mathématique

La modélisation de ce problème a été introduite par Cortès et al. (2010). Pour cette modélisation, nous reprenons la plupart des notations introduites pour le PDP et introduisons de nouvelles notations propres aux PDPT. Nous ajoutons tout d'abord l'ensemble des points de transbordements nommé T. Ainsi l'ensemble des nœuds N est défini par $N = M^+ \cup M^- \cup O \cup D \cup T$. Pour faciliter la formulation, un nœud physique de transbordement $t \in T$ est divisé en deux nœuds virtuels : t^+ dédié à la collecte et t^- dédié à la livraison. Nous ajoutons également l'ensemble R des requêtes à traiter. Une requête r est composée d'un nœud de collecte $r^+ \in P$, d'un nœud de livraison $r^- \in D$ et d'une quantité q_{r^+}. Comme pour le PDP, le but du problème est de trouver un ensemble de routes partant du dépôt de départ et arrivant au dépôt d'arrivée et qui satisfassent toutes les demandes des clients. Nous disposons pour cela d'une flotte hétérogène V de véhicules ayant chacun une capacité limitée C^v. Ces véhicules doivent partir vides et arriver vides au dépôt. Un seul type de produit est considéré, chaque nœud est visité une seule fois et les visites chez les clients doivent respecter des fenêtres de temps. Nous ajoutons en plus la possibilité de transférer une requête d'un véhicule à l'autre dans n'importe quel nœud de l'ensemble T. Nous utilisons donc les notations supplémentaires suivantes :

T ensemble des points de transbordement

R ensemble des requêtes à traiter

t^+ nœud virtuel de collecte du point de transbordement t. Quand un véhicule récupère une requête dans t, il passe par t^+.

t^- nœud virtuel de livraison du point de transbordement t. Quand un véhicule dépose une requête dans t, il passe par t^-.

r^+ nœud de collecte de la requête r.

r^- nœud de livraison de la requête r.

z_i^{vr} variable binaire valant 1 si la requête r est dans le véhicule v en arrivant au nœud i et 0 sinon.

La formulation du PDPT peut être définie comme suit :

$$\min \omega \sum_{v \in V} (1 - x^v_{v^+v^-}) + \sum_{v \in V} \sum_{(i,j) \in A} c^v_{ij} x^v_{ij} \tag{1.15}$$

Les contraintes 1.2 à 1.5, 1.7 et 1.8 sont toujours valides pour le PDPT.

$$\sum_{v \in V} \sum_{j:(i,j) \in A} x^v_{ij} = 1 \qquad \forall i \in P \cup D, \tag{1.2}$$

$$\sum_{j:(v^+,j) \in A} x^v_{v^+j} = 1 \qquad \forall v \in V, \tag{1.3}$$

$$\sum_{i:(i,v^-) \in A} x^v_{iv^-} = 1 \qquad \forall v \in V, \tag{1.4}$$

$$\sum_{i \in V^+ \setminus \{v^+\}} \sum_{j:(i,j) \in A} x^v_{ij} = 0 \qquad \forall v \in V, \tag{1.5}$$

$$B^v_j \geq B^v_i + d_i + t^v_{ij} - M(1 - x^v_{ij}) \qquad \forall (i,j) \in A, \forall v \in V, \tag{1.7}$$

$$e_i \leq B^v_i \leq l_i \qquad \forall i \in N, \forall v \in V, \tag{1.8}$$

A ces contraintes nous ajoutons les contraintes suivantes :

$$\sum_{i:(i,j) \in A} x^v_{ij} - \sum_{i:(j,i) \in A} x^v_{ji} = 0 \qquad \forall j \in P \cup D \cup T, \forall v \in V, \tag{1.16}$$

$$\sum_{i:(i,j) \in A} x^v_{ij} \leq 1 \qquad \forall j \in T, \forall v \in V, \tag{1.17}$$

$$B^v_j \leq M x^v_{ij} \qquad \forall (i,j) \in A, \forall v \in V, \tag{1.18}$$

$$\sum_{v \in V} B^v_{r^+} \leq \sum_{v \in V} B^v_{r^-} \qquad \forall r \in R, \tag{1.19}$$

$$z^{vr}_{v^+} = z^{vr}_{v^-} = 0 \qquad \forall v \in V, \forall r \in R, \tag{1.20}$$

$$z_i^{vr} \leq (1 - x_{ij}^v) + z_j^{vr} \qquad \forall v \in V, \forall r \in R, \forall (i,j) \in A, \; i \neq r^+, i \neq r^-, j \notin T, \tag{1.21}$$

$$z_i^{vr} \geq -(1 - x_{ij}^v) + z_j^{vr} \qquad \forall v \in V, \forall r \in R, \forall (i,j) \in A, \; i \neq r^+, i \neq r^-, j \notin T, \tag{1.22}$$

$$z_j^{vr} \leq (1 - x_{r^+j}^v) + 1 \qquad \forall v \in V, \forall r \in R, \forall j : (r^+, j) \in A, \tag{1.23}$$

$$z_j^{vr} \geq -(1 - x_{r^+j}^v) + 1 \qquad \forall v \in V, \forall r \in R, \forall j : (r^+, j) \in A, \tag{1.24}$$

$$z_j^{vr} \leq (1 - x_{r^-j}^v) \qquad \forall v \in V, \forall r \in R, \forall j : (r^-, j) \in A, \tag{1.25}$$

$$z_j^{vr} \geq -(1 - x_{r^-j}^v) \qquad \forall v \in V, \forall r \in R, \forall j : (r^-, j) \in A, \tag{1.26}$$

$$\sum_{v \in V} z_{t^+}^{vr} - \sum_{v \in V} z_{t^-}^{vr} = 0 \qquad \forall t \in T, \forall r \in R, \tag{1.27}$$

$$\sum_{r \in R} z_j^{vr} \leq \sum_{i:(i,j) \in A} x_{ij}^v \qquad \forall v \in V, \forall j \in N \setminus \{v^+, v^-\}, \tag{1.28}$$

$$B_{t^-}^{v_1} + d_{t^-} \leq M(2 - (z_{t^-}^{v_1 r} + z_{t^+}^{v_2 r})) + B_{t^+}^{v_2} \qquad \forall t \in T, \forall v_1, v_2 \in V, \forall r \in R, \tag{1.29}$$

$$\sum_{r \in R} q_{r^+} z_j^{vr} \leq C^v \qquad \forall v \in V, \forall j \in N, \tag{1.30}$$

La fonction objectif 1.15 est composée de deux termes. Le premier terme correspond à la minimisation du nombre de véhicules utilisés. Le deuxième terme correspond à la minimisation des coûts de transport. ω est un coefficient qui donne plus ou moins d'importance à la minimisation du nombre de véhicules. Si ce coefficient est suffisamment grand, alors l'objectif prioritaire est de minimiser le nombre de véhicules, puis de minimiser les coûts. Si ce coefficient est nul, seuls les coûts sont minimisés. Dans la contrainte 1.16, la conservation des flux est ajoutée au niveau des nœuds de transbordements. La contrainte 1.17 interdit à un véhicule de passer plus d'une fois par nœud de transbordement. La contrainte 1.18 initialise à zéro le temps d'arrivée d'un véhicule dans un nœud si ce véhicule ne passe pas par ce nœud. La contrainte 1.19 permet d'assurer l'antériorité de la collecte d'une requête par rapport à la livraison. Cette contrainte diffère de son équivalent PDP par le fait que la livraison ne se fait pas forcément dans le même véhicule que la collecte. Il faut donc vérifier la date d'arrivée au point de livraison de tous les véhicules pour vérifier que la date d'arrivée du véhicule qui livre est bien postérieure à la date de collecte.

C'est pour cette raison que les dates d'arrivée des véhicules non concernés sont mises à zéro par la contrainte 1.18. Les contraintes 1.20 à 1.28 représentent les processus de chargement et de déchargement des requêtes aussi bien aux nœuds de collecte et de livraison qu'aux nœuds de transbordement. Elles permettent le suivi des requêtes qui est nécessaire pour identifier les véhicules par lesquels les requêtes sont transportées. La contrainte 1.20 assure que les véhicules partent vides de leur dépôt et reviennent vides. Les contraintes 1.21 et 1.22 permettent d'assurer que les requêtes ne sont pas chargées ou déchargées ailleurs que dans leur point de collecte ou de livraison ou que dans un nœud de transbordement. Les contraintes 1.23 et 1.24 indiquent que si un véhicule passe par un nœud de collecte, alors la requête correspondante est chargée dans le véhicule. De la même manière, les contraintes 1.25 et 1.26 indiquent que si un véhicule passe par un nœud de livraison, alors la requête correspondante doit être déchargée du véhicule. La contrainte 1.27 assure que si une requête est déposée dans un nœud de transbordement, alors cette requête doit forcément quitter ce nœud de transbordement. La contrainte 1.28 initialise les variables z à zéro si un véhicule ne passe pas par le nœud correspondant. La contrainte 1.29 permet d'assurer que si un véhicule dépose une requête dans un nœud de transbordement, le véhicule qui récupère cette requête arrive après le déchargement. La contrainte 1.30 assure que la capacité des véhicules est respectée.

1.3.2 Etude des différents travaux sur les problèmes de routage avec transbordements

Le PDPT a reçu un intérêt accru ces dernières années mais reste moins étudié que les VRPPD. Dans ce chapitre nous étudions des travaux traitant du PDPT mais aussi des travaux ne traitant pas spécifiquement du PDPT mais pouvant s'y rapporter. Nous pouvons citer les PDP avec crossdocking, qui diffèrent des PDPT par le fait que le passage dans un nœud de transbordement est souvent obligatoire. Nous pouvons aussi citer les PDP with scheduled lines qui sont des PDP dans lesquelles les requêtes peuvent être transbordées des véhicules vers des lignes préétablies comme par exemple des bus. Ces travaux sont résumés dans le tableau 1.2.

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule						TW	Nb visites par client		Split
					1	Plusieurs	1	Plusieurs	Nb. 1	Nb. Limité	Nb. Illimité	Cap. Limité	Cap. Illimité	Flotte Homogène	Flotte Hétérogène		1	Plusieurs	
Cortès et al. (2010)	PDPT	Min coût	Branch & cut	12	X			X		X		X			X	X	X		
Dondo et Cerdá (2013)	PDPCD	Min coût ou min temps plus longue tournée	Heuristique à base d'échanges	100	X		X			X		X		X		X	X		
Ghilas et al. (2016b)	PDPSL	Min coût	Solver	22	X			X		X		X			X	X	X		
Ghilas et al. (2016a)	PDPSL	Min coût	LNS	200	X			X		X		X			X	X	X		
Gørtz et al. (2009)	PDPT	Min temps plus longue tournée	Heuristiques		X			X		X		X	X		X	X	X		
Kerivin et al. (2008)	PDPT	Min coût	Branch & cut	30		X	X			X		X		X		X		X	X
Lee et al. (2006)	VRPCD	Min nb véh + min coût	Recherche tabou	50	X		X			X		X		X		X	X		
Liao et al. (2010)	VRPCD	Min nb véh + min coût	Recherche tabou	400	X		X			X		X		X		X	X		
Masson et al. (2013)	PDPT	Min coût	ALNS	130	X					X			X	X		X		X	
Masson et al. (2014)	PDPT	Min distance	ALNS	198	X			X		X		X			X	X		X	
Mitrovic-Minic et Laporte (2006)	PDPT	Min coût	Heuristique	200	X						X	X		X		X	X		
Morais et al. (2014)	VRPCD	Min coût	ILS	400	X		X			X		X		X		X	X		
Musa et al. (2010)	PDPCD	Min coût	Colonies de fourmis	200	X					X		X		X				X	
Oertel (2000)	PDPT	Min nb véh + min coût	Recherche tabou	43	X		X			X		X		X		X		X	
Petersen et Ropke (2011)	PDPCD	Min coût (fixe + temps + km + manutention)	LNS	628	X		X			X		X		X		X		X	

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule							TW	Nb visites par client		Split
									Nb.			Cap.		Flotte					
					1	Plusieurs	1	Plusieurs	1	Limité	Illimité	Limité	Illimité	Homogène	Hétérogène		1	Plusieurs	
Qu et Bard (2012)	PDPT	Min nb véh + min coût	GRASP + LNS	50	X			X		X		X		X		X	X		
Rais et al. (2014)	PDPT	Min coût	Solver	14	X			X		X		X			X	X		X	
Santos et al. (2013)	PDPCD	Min coût transport + crossdock	Branch & price	60	X		X			X		X		X			X		
Santos et al. (2011)	VRPCD	Min coût transport + crossdock	Branch & price	100	X		X			X		X		X			X		
Shang et Cuff (1996)	PDPT	Min coût + retard + temps trajet	Heuristique	9	X						X		X	X		X		X	
Tchapnga Takoudjou et al. (2012)	PDPT	Min nb véh + min coût	VND	50	X		X			X		X		X		X	X		
Thangiah et al. (2007)	PDPT	Min nb véh + min coût	Recherche locale	9	X			X		X		X		X		X		X	X
Wen et al. (2009)	VRPCD	Min coût	Recherche tabou	400	X		X			X		X		X		X	X		

Tableau 1.2 – Résumé des travaux sur les problèmes de routage avec transbordements

La possibilité de transférer des marchandises d'un véhicule à un autre dans un PDP est pour la première fois évoquée par Shang et Cuff (1996). Ils ont développé une heuristique multi-objectif pour résoudre le PDPT. Les objectifs à minimiser sont les coûts, les retards et les temps de trajet. Dans leur problème, deux requêtes peuvent avoir le même point de collecte ou le même point de livraison. Un ensemble de nœuds de transbordement n'est pas encore défini mais une requête peut être transférée d'un véhicule à l'autre dans n'importe quel nœud de collecte ou de livraison. De plus la taille de la flotte de véhicules n'est pas connue à l'avance et il n'y a pas de contrainte de capacité. Une heuristique à base d'insertion de mini-routes est utilisée. L'idée de cette heuristique est d'insérer plusieurs requêtes dans une route simultanément. Pour ce faire, des mini-routes sont créées avec plusieurs requêtes. Ensuite ces minis-routes sont insérées dans les véhicules. Ce problème a été créé spécifiquement pour résoudre la gestion du transport d'une "Health maintenance organization" . L'heuristique est donc testée sur des données fournies par cette organisation. Leur réseau est composé de 9 points de collecte ou livraison. Ils comparent leur méthode avec la planification manuelle réalisée par l'organisation. L'heuristique permet de réduire les coûts, les retards et les temps de trajet. Oertel (2000) propose une formulation et une recherche tabou pour résoudre le PDPT. Pour explorer le voisinage, l'auteur utilise des opérateurs à base d'échanges d'arcs et de nœuds. La méthode est basée sur des données réelles avec 43 clients. Les solutions avec transbordements sont comparées avec les solutions sans transbordement pour démontrer l'intérêt d'une telle pratique. Mitrovic-Minic et Laporte (2006) proposent une heuristique à base d'insertion pour résoudre le PDPT. Cette heuristique est composée de deux phases. Dans la première phase, plusieurs solutions sont générées aléatoirement en insérant les requêtes dans des ordres différents. Si une requête ne peut pas être insérée dans un véhicule existant, un nouveau véhicule est utilisé. A la fin de cette phase, la meilleure solution est choisie pour la seconde phase. Dans cette seconde phase une requête est enlevée puis réinsérée dans la solution à chaque itération. Pour tester la méthode, plusieurs jeux d'instances avec 100 et 200 nœuds ont été créés. Ces instances sont divisées en quatre groupes : trois groupes où les nœuds sont placés dans des clusters et un groupe où les nœuds sont placés aléatoirement. Pour mener

leurs expériences, ils font varier le nombre de points de transbordement de 0 à 4, ainsi que la taille des fenêtres de temps. Leurs résultats montrent que le transbordement est bénéfique que ce soit sur les instances avec ou sans clusters. Thangiah et al. (2007) proposent également une heuristique à base d'insertions. Cette heuristique est étendue en ajoutant une recherche locale pour améliorer les solutions initiales. Deux opérateurs principaux de recherche locale sont utilisés pour améliorer une solution. Le premier opérateur enlève toutes les requêtes contenues dans le véhicule en contenant le moins, et les replace dans les autres véhicules utilisés. Cet opérateur permet de réduire le nombre de véhicules utilisés dans la solution. Le second opérateur échange des requêtes entre des véhicules sélectionnés au hasard. Dans leur approche, ils permettent d'éclater une requête pour que celle-ci soit transportée simultanément par plusieurs véhicules. Leur méthode est testée sur les données de Shang et Cuff (1996) et celle-ci permet d'améliorer les résultats. Kerivin et al. (2008) proposent une formulation pour le PDPT dans lequel une requête peut être déposée dans n'importe quel nœud, partiellement ou totalement, et ramassée par un autre véhicule jusqu'à ce que la requête atteigne sa destination finale. Ils utilisent un algorithme "branch-and-cut" pour résoudre ce problème et sont capables de résoudre des instances générées aléatoirement avec 30 clients. Gørtz et al. (2009) proposent des heuristiques pour résoudre le PDPT avec et sans contrainte de capacité en minimisant la durée de la tournée la plus longue.

Une méthode exacte est proposée par Cortès et al. (2010). Dans la formulation proposée, une variable binaire supplémentaire par rapport au PDP est utilisée pour suivre les requêtes et savoir dans quels véhicules celles-ci sont transportées. La méthode "branch-and-cut" est testée sur des instances générées aléatoirement avec 12 clients et ils comparent le temps de calcul de leur méthode avec le temps de calcul mis par un "branch-and-bound" classique. Tchapnga Takoudjou et al. (2012) proposent une heuristique VND pour résoudre le PDP. Ils utilisent ensuite une heuristique pour améliorer leur solution en ajoutant des transbordements. Des instances de 50 clients sont résolues par cette méthode. Qu et Bard (2012) proposent une méthode hybride GRASP et LNS. Dans leur problème, un véhicule peut venir déposer une requête dans un nœud de transbordement et revenir plus tard la reprendre. La méthode GRASP est utilisée

pour générer des solutions initiales. La méthode LNS est utilisée pour améliorer les solutions. Ils utilisent également une mémoire tampon qui sauvegarde les insertions déjà testées afin d'économiser du temps de calcul. Leur méthode est testée sur des instances de 50 nœuds construites de telle sorte que la solution optimale est connue. Ainsi ils peuvent comparer leurs résultats avec la solution optimale. Ils arrivent à obtenir des résultats très proches de l'optimum. Un LNS adaptatif est proposé dans Masson et al. (2013) et Masson et al. (2014) pour résoudre le PDPT et ils appliquent leur méthode sur des instances réelles de transport scolaire et sur les instances de Mitrovic-Minic et Laporte (2006). La taille maximale des instances résolues est de 130 clients. Rais et al. (2014) proposent une nouvelle formulation du PDPT en considérant plusieurs variantes. Ils utilisent le solveur GUROBI pour résoudre ce problème en prenant en compte les fenêtres de temps, une flotte hétérogène de véhicules et des requêtes non éclatables. La plus grande instance résolue possède 14 nœuds.

Le transbordement est aussi autorisé dans les classes de problèmes appelés Vehicle Routing Problem with Cross-Docking (VRPCD) et Pickup and Delivery Problem with Cross-Docking (PDPCD). Dans ces problèmes le point de transbordement est appelé crossdock et il est aussi le dépôt des véhicules. Dans les VRPCD, le passage des marchandises dans un crossdock est obligatoire contrairement au PDPCD. Ce dernier se rapproche donc du PDPT. Lee et al. (2006) proposent une formulation et une recherche tabou pour le VRPCD. Dans ce problème, les véhicules exécutent soit des tournées de collecte, soit des tournées de livraison. Des instances avec 50 clients sont résolues. Wen et al. (2009) et Liao et al. (2010) proposent également une recherche tabou pour le même problème. Ils prennent en compte en plus les fenêtres de temps. Ils résolvent des instances contenant jusqu'à 400 clients. Musa et al. (2010) proposent un algorithme de colonie de fourmis. Dans ce problème, une requête peut passer ou non par un cross-dock . De plus, les collectes et les livraisons ne sont pas organisées en tournées. Les véhicules ne peuvent parcourir que les arcs entre les points de collecte et les cross-docks, entre les cross-docks et les points de livraison et entre les points de collecte et les points de livraison. La méthode est testée sur des instances contenant jusqu'à 200 nœuds. Petersen et Ropke (2011) proposent un LNS pour résoudre le PDPCD. La méthode est testée sur des instances de

données réelles contenant jusqu'à 628 nœuds. Un algorithme "branch-and-price" est proposé par Santos et al. (2011b) et Santos et al. (2011a). Le coût de manutention des marchandises dans le cross-dock est pris en compte. Le problème est résolu sur des instances contenant jusqu'à 100 clients. Ce "branch-and-price" est étendu dans Santos et al. (2013). Ce problème est résolu sur des instances de 60 clients. Pour prendre en compte la possibilité de livrer une requête sans passer par le cross-dock, Dondo et Cerdá (2013) proposent une nouvelle formulation et une heuristique à base d'échanges. Ils autorisent les véhicules à collecter et livrer une requête sans passer par le cross-dock. Les instances résolues contiennent jusqu'à 100 nœuds. Morais et al. (2014) proposent une recherche locale itérative pour résoudre le VRPCD dans laquelle les tournées sont soit des tournées de collecte soit des tournées de livraison. Leur méthode permet d'améliorer les instances de Wen et al. (2009).

Enfin, Ghilas et al. (2016b) introduisent le Pickup and Delivery Problem with Scheduled Lines (PDP-SL). Ils définissent ce problème comme une extension du PDP dans lequel les requêtes peuvent être transférées dans des lignes de transport public. Une requête peut donc être collectée par un véhicule de transport de marchandises, déchargée dans un point de transfert pour être transportée par un véhicule qui effectue des lignes régulières (bus, train, métro, ...). Cette requête est ensuite ramassée dans un autre point de transfert en fin de ligne par un autre véhicule de transport de marchandises pour être livrée à son point de livraison. Nous pouvons donc considérer ce problème comme une généralisation du PDPT puisque le PDPT est un PDP-SL dans lequel les lignes régulières ont une longueur nulle. Après avoir formulé ce problème, un LNS est développé dans Ghilas et al. (2016a) pour le résoudre. Des instances aléatoires comprenant jusqu'à 200 clients sont générées pour tester le LNS.

1.4 Planification du transport collaboratif

La collaboration dans les chaînes logistiques a reçu un gain d'attention cette dernière décennie à la fois dans le milieu industriel et dans le milieu académique. Nous nous concentrons dans cette section sur les travaux réalisés sur le transport collaboratif ou Collaborative Transport Planning (CTP) et ayant un rapport avec

les problèmes de routage. En effet, les problèmes décrits précédemment peuvent être des facilitateurs d'une telle collaboration. En ce sens, pour chaque article sur la collaboration étudié, nous nous efforcerons de le classer parmi un des problèmes présentés dans les sections précédentes. Les travaux étudiés sont résumés dans le tableau 1.3.

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule							TW	Nb visites par client		Split
					1	Plusieurs	1	Plusieurs	Nb.		Cap.		Flotte				1	Plusieurs	
									1	Limité	Illimité	Limité	Illimité	Homogène	Hétérogène				
Dai et Chen (2012)	PDP	Min coût	Relaxation lagrangienne	50		×				×		×		×				×	×
Dias et Tsuzuki (2010)	PDP	Min coût	Branch & cut	7	×						×	×		×		×		×	
Ergun et al. (2007)	PDP	Min coût	Algorithme glouton	500	×						×	×		×				×	
Ergun et al. (2007)	PDP	Min coût	Algorithme glouton	500	×						×	×		×		×		×	
Hernández et al. (2011)	PDPT	Min coût	Solver	20		×		×		×		×		×		×		×	
Li et al. (2016)	PDP	Max profit	LNS	200	×		×			×		×		×		×	×		
Liu et al. (2010)	PDP	Min coût	Algorithme glouton	500	×			×		×		×		×				×	×
Pan et al. (2013)	PDPCD	Min emissions	CPLEX	140		×				×		×			×			×	
Wang et Kopfer (2014)	PDP	Min nbvéh + min coût	Heuristique	500	×			×		×		×		×		×	×		

TABLEAU 1.3 – Résumé des travaux sur le CTP

Le CTP est un type de collaboration horizontale et celle-ci peut être vue selon deux points de vue : celui des fournisseurs et des clients, et celui des transporteurs (Liu et al., 2010). Tout d'abord, les fournisseurs ou les clients peuvent collaborer entre eux pour minimiser leurs coûts de transport. En effet, ceux-ci sont exposés à des délais d'approvisionnement de plus en plus courts, et les marchandises sont livrées plus souvent mais avec des quantités moindres. Dans ces cas, il peut être intéressant pour des entreprises de mutualiser leurs moyens de transport pour réaliser des économies.

Il existe très peu de travaux traitant de la collaboration des fournisseurs ou des clients dans le cadre du CTP. Ergun et al. (2007b) introduisent ce problème comme un Lane Covering Problem (LCP). Le but de ce problème est de couvrir un ensemble d'arcs prédéfini à l'aide d'un ou plusieurs cycles de telle sorte que le coût de ces cycles soit minimal. Ce problème peut être rapproché du PDP. Les arcs à couvrir représentent les requêtes et les cycles représentent les tournées. Cependant dans ce problème, un véhicule ne peut transporter qu'une seule requête à la fois, c'est-à-dire que le point de collecte d'une requête précède toujours immédiatement le point de livraison correspondant. Ils définissent également une variante du LCP dans laquelle la taille des cycles doit être inférieure à un nombre prédéfini. Le LCP est étendu dans Ergun et al. (2007a) pour prendre en compte les fenêtres de temps. Ces deux problèmes sont testés sur des instances générées de manière aléatoire et sur des données réelles. Dias et Tsuzuki (2010) proposent un "branch-and-cut" pour résoudre le LCP dans le cadre de la collaboration entre fournisseurs. La méthode est testée sur des données réelles contenant 7 nœuds. Dans Pan et al. (2013) les auteurs proposent un modèle dans lequel les flux de plusieurs fournisseurs peuvent être mutualisés dans deux types de points de transbordement : les points de transbordement amont, qui sont situés après les fournisseurs, et les points de transbordement aval, qui sont situés avant les clients. Les flux directs entre les fournisseurs et entre les clients ne sont pas autorisés. Ils considèrent les deux modes de transport routier et ferroviaire, ce dernier étant réservé aux flux entre les points de transbordement amont et aval. Le but est de minimiser les émissions de CO_2 produites par les véhicules. Ce modèle est utilisé sur des données réelles contenant jusqu'à 140 nœuds et résolu à l'aide du solver CPLEX.

Ensuite ce sont les transporteurs qui peuvent également collaborer. En effet ceux-ci sont également confrontés aux cadences de livraison plus élevées mais de plus faible quantité. Ils peuvent donc également mutualiser leurs moyens de transport, par exemple en s'échangeant des requêtes afin d'optimiser le taux d'utilisation des véhicules ou en minimisant les retours à vide. Liu et al. (2010) proposent une formulation pour le problème de collaboration entre transporteurs. Ce problème est semblable à un LCP dans le sens où les véhicules ne peuvent transporter qu'une seule requête à la fois. Cependant, contrairement au LCP, les véhicules doivent partir et revenir à leur dépôt. De plus, une requête peut être divisée et être transportée par plusieurs véhicules en même temps. Un algorithme glouton est développé pour résoudre ce problème et il est testé sur des instances générées aléatoirement comprenant jusqu'à 500 clients. Hernández et al. (2011) proposent une formulation pour le CTP qui peut se rapporter à un PDPT. Ils testent leur modèle sur des instances comprenant 20 nœuds et montrent le gain que peut apporter la collaboration en termes de coûts pour les transporteurs. Dans Dai et Chen (2012), le CTP est décrit comme un PDP dans lequel il n'y a pas de dépôt et dans lequel les requêtes peuvent être éclatées et servies par plusieurs véhicules à la fois. Une méthode basée sur la relaxation lagrangienne est proposée et testée sur des instances générées aléatoirement comprenant 50 nœuds. Wang et Kopfer (2014) proposent un modèle apparenté au PDP dans lequel les transporteurs peuvent s'échanger des requêtes. Le but est de trouver les meilleures requêtes à échanger dans le but de minimiser les coûts fixes et variables du transport. Leur méthode est testée sur des instances basées sur la fusion de plusieurs instances de Li et Lim (2003) comprenant jusqu'à 500 nœuds et donne des meilleurs résultats lorsque la collaboration entre les transporteurs est prise en compte. Li et al. (2016) proposent un PDP pour la collaboration entre transporteurs. Dans leur problème, chaque transporteur a des requêtes réservées et transportées par lui-même, et des requêtes qui peuvent être transportées par d'autres. Ces requêtes non réservées ne sont pas obligatoirement traitées. Un profit est associé à chaque requête, l'objectif du problème est donc de maximiser le profit. Une formulation et un LNS sont proposés pour ce problème. Le LNS est testé sur des instances comprenant jusqu'à 200 clients.

1.5 Conclusion

Dans ce chapitre, nous avons présenté divers travaux sur VRPPD. Nous pouvons remarquer que les problèmes de collectes et livraisons classiques sont très étudiés, excepté les PDVRP. Les problèmes de collectes et livraisons avec transbordements sont moins étudiés mais le nombre de travaux sur le sujet croît de plus en plus. La faible présence de ces derniers peut s'expliquer par la complexité d'intégrer la notion de transbordement. En effet, cette notion amène une forte contrainte d'interdépendance entre les routes qui n'existait pas avant. Enfin nous pouvons noter que les travaux sur le CTP sont également très peu abordés. Pourtant la collaboration peut être un moyen efficace pour les entreprises d'améliorer la gestion de leur transport. De plus, quel que soit le problème étudié, les aspects environnementaux sont rarement pris en compte. A notre connaissance, seuls Pan et al. (2013) prennent en compte ces aspects dans les problèmes étudiés.

A partir de ces études, nous pouvons dégager trois contributions apportées par cette thèse, chacune étant présentée par un chapitre dédié. Dans le chapitre deux, deux méthodes approchées sont proposées pour résoudre le PDPT. De par les caractéristiques choisies, nous nous sommes efforcés de traiter le cas le plus général possible. Les caractéristiques sont résumées dans le tableau 1.4. Parmi les travaux étudiés, Rais et al. (2014) sont les seuls à prendre en compte ces caractéristiques. Cependant, ils proposent uniquement une formulation et ne proposent pas de méthode pour résoudre le problème sur des grandes instances. Les deux méthodes que nous proposons sont un LNS et un GA. Le LNS a été proposé plusieurs fois que ce soit pour le PDP ou le PDPT et son efficacité à résoudre ces problèmes a été démontrée. Cependant il va servir de point de comparaison pour le GA qui n'est pas encore utilisé pour résoudre le PDPT et qui est peu utilisé dans la résolution du PDP. De plus nous utilisons une représentation innovante des individus par rapport à ce qui existe déjà dans la littérature.

Dans le chapitre trois, une formulation de type Mixed-Integer Linear Programming (MILP) pour le PDVRP est proposée et utilisée pour favoriser la collaboration entre fournisseurs ou entre clients. Les caractéristiques sont ré-

sumées dans le tableau 1.4. Ce problème est très peu étudié et n'a, à notre connaissance, pas encore été utilisé dans le cadre du transport collaboratif. De plus, nous prenons en compte les aspects environnementaux pour résoudre ce problème.

Dans le chapitre quatre, une formulation pour le PDVRP avec transbordement est proposée et utilisée pour la collaboration entre fournisseurs. Les caractéristiques sont résumées dans le tableau 1.4. Ce problème n'a pas encore été étudié dans la littérature. Il est inspiré d'une étude de cas réelle réalisée dans le cadre du projet SCALE.

Références	Type	Obj	Méthode	Taille	Nb produits		Nb dépôts		Véhicule						TW	Nb visites par client		Split
					1	Plusieurs	1	Plusieurs	Nb.		Cap.		Flotte			1	Plusieurs	
									1	Limité	Illimité	Limité	Homogène	Hétérogène				
Chapitre 2	PDPT	Min nb véh + coût	LNS et GA	100	X			X		X		X		X	X		X	
Chapitre 3	PDVRP	Min emissions et min coût	Solver			X				X		X		X	X		X	X
Chapitre 4	PDVRPT	Min emissions et min coût	Solver			X		X		X		X		X	X		X	X

TABLEAU 1.4 – Résumé des caractéristiques des problèmes étudiés dans les chapitres deux, trois et quatre

Chapitre 2

Un LNS et un GA pour la résolution du PDPT

2.1 Introduction

Nous avons vu dans le chapitre précédent que le PDPT est une généralisation du PDP (Cortès et al., 2010) et que le transbordement pouvait permettre de réduire les coûts de transport (Oertel, 2000). Cependant, le PDPT est moins étudié que le PDP. De plus, le PDPT est un problème NP-difficile (Rais et al., 2014) et ne peut pas être résolu de manière exacte en un temps raisonnable pour des grandes instances, d'où la nécessité de développer des méthodes approchées. Dans ce chapitre, nous proposons deux méthodes approchées pour résoudre le PDPT. La première méthode est un LNS et la deuxième est un GA. Le LNS a déjà été proposé pour résoudre le PDPT et semble être une méthode efficace (Qu et Bard, 2012 ; Masson et al., 2013). Par contre, bien que des GA aient été proposés pour le PDP (Créput et al., 2004 ; Dridi et al., 2011), à notre connaissance, aucun GA n'a été proposé pour résoudre le PDPT. Ces deux méthodes sont testées sur deux types d'instances de la littérature. Le premier type d'instances vient de Li et Lim (2003) et ces instances sont spécifiques au PDP. En effet, le PDPT étant une généralisation du PDP, un algorithme permettant de résoudre le PDPT doit également résoudre le PDP. Le deuxième type d'instances vient de Qu et Bard (2012) et ces instances sont spécifiques au PDPT. Les résultats des deux

méthodes sont ainsi comparés entre eux et avec les meilleurs résultats connus dans la littérature.

Ce chapitre est décomposé en cinq sections. Dans la première section, quelques méthodes de résolution (exactes et approchées) sont rapidement présentées. La description des méthodes LNS et GA est plus longuement détaillée. Dans la deuxième section, les caractéristiques du PDPT étudié sont détaillées. De plus, les notions communes aux deux méthodes sont également présentées. Dans la troisième section, la méthode LNS utilisée est décrite. Dans la quatrième section, la méthode GA utilisée est décrite. Enfin dans la dernière section, les résultats sur les instances testées sont présentés puis analysés.

2.2 Les méthodes de résolution

Pour résoudre les problèmes de collectes et livraisons, plusieurs méthodes ont été proposées aux cours des dernières années. Ces méthodes de résolution peuvent être classées en deux grandes catégories : les méthodes exactes et les méthodes approchées.

2.2.1 Méthodes exactes

Les méthodes exactes sont des algorithmes qui renvoient la solution optimale du problème à la fin de leur exécution. Pour les problèmes NP-difficiles comme les problèmes de collectes et livraisons, l'utilisation de tels algorithmes est difficile pour résoudre les problèmes de grande taille car le temps de calcul devient très long. Les méthodes exactes utilisées pour résoudre les problèmes de collectes et livraisons sont le "branch-and-bound", le "branch-and-cut" et le "branch-and-price".

Le "branch-and-bound" a été proposé pour la première fois par Land et Doig (1960). Dans cette méthode un arbre est utilisé pour énumérer les solutions du problème. L'arbre partant de la racine représente l'ensemble des solutions et chaque sous-arbre partant d'un nœud représente un sous-ensemble. Le principe est d'éliminer l'exploration de sous-ensembles dans lesquels l'absence de la solution optimale est certaine. Pour savoir si un sous-ensemble peut être éliminé,

il faut chercher une borne inférieure (pour un problème de minimisation) à la solution optimale du sous-ensemble. Si cette borne inférieure est plus grande que la valeur de la meilleure solution trouvée jusqu'à présent, alors le sous-ensemble ne contient pas la solution optimale et n'est donc pas exploré. Le calcul des bornes inférieures se fait en règle générale à l'aide de relaxation continue ou lagrangienne.

Le "branch-and-cut" est mentionné pour la première fois par Padberg et Rinaldi (1987). Cette méthode est semblable au "branch-and-bound". Cependant, à chaque nœud de l'arbre d'énumération, des contraintes sont ajoutées dans le problème ou le sous-problème afin d'éliminer des solutions réelles de la relaxation continue. Si après l'ajout de telles contraintes, la solution n'est toujours pas entière, le problème est divisé en deux sous-problèmes et ce procédé est répété.

Le "branch-and-price" est défini par Johnson (1989). Cette méthode est semblable au branch-and-bound. Cependant, à chaque nœud de l'arbre d'énumération, une méthode à génération de colonnes est utilisée. L'idée de la génération de colonnes est de commencer par résoudre le problème avec un nombre restreint de variables , puis d'ajouter au fur et à mesure des variables (colonnes) qui sont susceptibles d'améliorer l'objectif.

2.2.2 Méthodes approchées

Contrairement aux méthodes exactes, les méthodes approchées ne garantissent pas que la solution trouvée à la fin de l'algorithme soit optimale. Cependant, ces méthodes permettent de trouver des bonnes solutions en des temps raisonnables pour des problèmes de grande taille. Ces méthodes sont divisées en deux groupes : les heuristiques et les méta-heuristiques. Les heuristiques sont des algorithmes qui sont spécifiques à un problème donné. Ils ont généralement été conçus pour résoudre uniquement un problème ou une classe de problèmes. Les méta-heuristiques sont des méthodes plus globales qui peuvent être adaptées à différents problèmes. Les méta-heuristiques peuvent également être divisées en deux catégories (Talbi, 2009) : les méthodes à parcours et les méthodes à population. Dans les méthodes à parcours, une seule solution évolue au fil des

itérations. Dans les méthodes à population, plusieurs solutions sont traitées en parallèle.

Dans les méthodes à parcours, nous pouvons classer :

— Recherche locale (Papadimitriou et Steiglitz, 1982) : cette méthode part d'une solution initiale et la fait évoluer en passant d'une solution voisine à une autre de manière itérative. Le voisinage de la solution courante est défini par un opérateur qui transforme la solution d'origine en un ensemble de solutions voisines. Une solution est sélectionnée parmi ces voisines et le processus est répété jusqu'à ce que le critère d'arrêt soit atteint.

— Recuit simulé (Kirkpatrick et al., 1983) : le recuit simulé est une méthode de recherche locale dans laquelle une solution voisine qui n'améliore pas la solution courante, est acceptée selon une probabilité liée à une température. Plus la température est élevée, plus la probabilité d'acceptation est grande. La température décroît doucement au fil des itérations comme dans le traitement thermique du même nom.

— Recherche tabou (Glover, 1986) : la recherche tabou est également une recherche locale dans laquelle la solution sélectionnée n'améliore pas forcément la solution courante. Les solutions déjà explorées sont enregistrées dans une liste tabou pour éviter d'y revenir.

— GRASP (Feo et Resende, 1989) : le GRASP est constitué de deux phases qui sont une phase de construction et une phase d'amélioration. Une solution est construite de manière itérative par partie lors de la première phase. A chaque étape, un ensemble de morceaux à insérer est trié selon leur qualité. Le morceau à insérer est choisi au hasard mais les meilleurs morceaux ont une plus grande probabilité d'être choisis. Quand la création de la solution est terminée, une recherche locale est effectuée pour améliorer la solution. Ces deux phases sont exécutées plusieurs fois afin de créer plusieurs solutions et la meilleure est conservée.

— VNS (Mladenović et Hansen, 1997) : le VNS est une recherche locale dans laquelle plusieurs opérateurs sont disponibles. Il y a donc le choix entre plusieurs voisinages lors du passage d'une solution à une autre, ce qui permet ainsi de s'échapper d'un optimum local.

— LNS (Shaw, 1998) : le LNS est présenté plus en détail par la suite.

— ILS (Lourenço et al., 2003) : le ILS est une recherche locale itérative dans laquelle la solution courante peut être perturbée afin de sortir d'un optimum local pour repartir vers la recherche d'un optimum local plus prometteur.

Dans les méthodes à population, nous pouvons classer :

— GA (Holland, 1975) : le GA est présenté plus en détail par la suite.

— Algorithme mémétique (Moscato, 1989) : un algorithme mémétique est un GA combiné avec une recherche locale. Une population de solutions est générée au départ de façon aléatoire pour couvrir l'espace de recherche. Plusieurs individus parmi cette population sont sélectionnés et croisés. Une recherche locale est ensuite appliquée sur les nouveaux individus ainsi obtenus et le processus de sélection, croisement et recherche locale est de nouveau répété.

— Algorithme de colonies de fourmis (Dorigo et al., 1996) : cet algorithme est inspiré du comportement des fourmis. Les solutions sont représentées par le chemin qu'empruntent les fourmis pour aller de leur nid à une source de nourriture. Les fourmis empruntent différents chemins et laissent des phéromones le long du chemin. Plus le chemin est intéressant, plus la quantité de phéromones laissées sur ce chemin sera importante. Les fourmis suivantes auront tendance à suivre le chemin avec le plus de phéromones. Les phéromones étant volatiles, les chemins les moins intéressants auront tendance à avoir de moins en moins de phéromones. A la fin de l'algorithme, le chemin possédant le plus de phéromones sera sélectionné comme le plus intéressant.

Dans ce chapitre, deux méthodes approchées ont été implémentées pour résoudre le PDPT. La première méthode est le LNS, une méthode à parcours, la seconde méthode est une méthode à population, le GA.

Le LNS a été introduit par Shaw (1998). Le principe de cet algorithme est de détruire et de reconstruire la solution de manière itérative. Le principe est assez similaire à la recherche locale mais le voisinage exploré est beaucoup plus vaste car défini implicitement par des heuristiques de destruction et de réparation.

L'algorithme 2.1 présente le LNS général tel qu'il est décrit dans Pisinger et Ropke (2010).

Algorithm 2.1 LNS général

Input: a feasible solution x

1: $x^b = x$
2: $bestSolution \leftarrow currentSolution$
3: **repeat**
4: $x^t = repair(destroy(x))$
5: **if** accept(x^t, x) **then**
6: $x = x^t$
7: **end if**
8: **if** $c(x^t) \leq c(x^b)$ **then**
9: $x^b = x^t$
10: **end if**
11: **until** stop criterion is met
12: **return** x^b

Une solution est d'abord initialisée, généralement à l'aide d'une heuristique constructive. Ensuite la solution est détruite en partie et réparée. Si la solution est acceptée, elle est conservée pour l'itération suivante. La meilleure solution est sauvegardée. Ces étapes sont répétées jusqu'à ce que les critères d'arrêt soient atteints. Le LNS est utilisé par Bent et Van Hentenryck (2006) et Ropke et Pisinger (2006) pour résoudre le PDP, par Qu et Bard (2012) et Masson et al. (2013) pour résoudre le PDPT, par Petersen et Ropke (2011) pour résoudre le PDPCD et par Ghilas et al. (2016a) pour résoudre le PDP-SL.

Le principe du GA a été introduit par Holland (1975) puis a été popularisé par Goldberg (1989). Il est inspiré des théories évolutionnistes et de la sélection naturelle. Il reprend donc des termes issus de la biologie. C'est une méthode à base de population. Une population est composée de plusieurs individus ou chromosomes. Chaque individu est composé de plusieurs gènes. L'ensemble de ces gènes permet de coder une solution particulière au problème traité. L'algorithme 2.2 présente le GA général.

Tout d'abord, une population initiale est créée. Ensuite, chaque individu est évalué pour déterminer la qualité (i.e. fitness) de la solution qu'il représente. Des paires d'individus (i.e. parents) sont sélectionnées pour participer aux croi-

Algorithm 2.2 GA général

```
1: generate initial population P
2: evaluate individuals in P
3: repeat
4:    cross some pairs of individuals and put new individuals in P
5:    mutate some individuals
6:    evaluate individuals in P
7:    select individuals for the next generation
8: until stop criterion is met
9: return best individual in P
```

sements et créer ainsi de nouveaux individus (i.e. fils). Chaque paire donne naissance à deux nouveaux individus. Certains individus peuvent ensuite muter, c'est-à-dire avoir un gène qui est modifié. Enfin les individus sont ensuite évalués et certains individus sont sélectionnés pour faire partie de la génération suivante. Ces étapes sont répétées jusqu'à ce que les critères d'arrêt soient respectés. Le GA est utilisé par Zhao et al. (2009) pour résoudre le PDTSP, par Shi et al. (2009) pour résoudre le PDVRP et par Créput et al. (2004), Pankratz (2005) et Dridi et al. (2011) pour résoudre le PDP. Selon notre connaissance, il n'a pas encore été utilisé pour résoudre le PDPT.

2.3 Le PDPT

Le problème étudié dans ce chapitre est le PDPT. Ce problème est une généralisation du PDP dans lequel une requête peut être transférée d'un véhicule à un autre au moyen de nœuds spécifiques appelés nœuds de transbordement. Dans ce chapitre nous considérons les hypothèses suivantes :

— Les fenêtres de temps sont prises en compte. Chaque véhicule doit commencer le service au client dans cette fenêtre. Un véhicule peut arriver dans un nœud avant le début de la fenêtre de temps, cependant il devra attendre le début de la fenêtre pour commencer son service.

— Chaque véhicule possède un dépôt de départ et un dépôt d'arrivée. Ces deux dépôts peuvent être confondus.

— Le cas multi-dépôts est considéré. Les véhicules n'ont pas forcément les mêmes dépôts de départ ou les mêmes dépôts d'arrivée.

— Le nombre de véhicules disponibles est limité. Tous les véhicules ne sont pas obligatoirement utilisés.

— La flotte de véhicules est hétérogène. Les véhicules peuvent avoir des capacités, des coûts ou des vitesses différentes.

— Un véhicule ne peut visiter un nœud qu'une et une seule fois, même chose pour les nœuds de transbordement.

— A l'inverse, un nœud de transbordement peut être visité par plusieurs véhicules.

— Un nœud peut jouer plusieurs rôles à la fois. Il peut être nœud de collecte, nœud de livraison et nœud de transbordement en même temps. Un nœud peut également être la source ou la destination de plusieurs requêtes. Ainsi nous n'avons pas forcément $|P| = |D| = |R|$ (P est l'ensemble des nœuds de collecte, D est l'ensemble des nœuds de livraison et R est l'ensemble des requêtes).

— Une requête peut être transbordée au maximum une seule fois.

2.3.1 Représentation et évaluation de la qualité d'une solution

Une solution est représentée par un ensemble de routes $S = v_1, v_2, v_3, \ldots, v_m$ avec $m = |V|$. Une route $v_i = x_1, x_2, x_3, \ldots, x_p$ correspond à la tournée du véhicule i, et représente une succession de nœuds à visiter qui commence par le dépôt de départ du véhicule v_i, noté v_i^+, et qui termine par le dépôt d'arrivée du véhicule v_i, noté v_i^-. Pour chaque nœud x_j de la route v_i, $x_j \in N$ et l'indice j correspond à l'ordre du nœud dans la route. Si le véhicule v_i n'est pas utilisé alors la route est vide ($v_i = \varnothing$). La figure 2.1 représente un exemple de représentation d'une solution avec deux véhicules. Le véhicule 1 passe par les nœuds 1, 5, 6 et 2 dans cet ordre et le véhicule 2 passe par les nœuds 3 et 4.

Pour évaluer la qualité d'une solution, nous comparons deux objectifs distincts. Le premier objectif f_1 à minimiser est le nombre de véhicules utilisés,

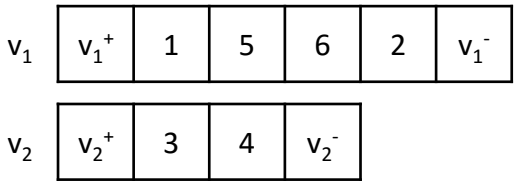

FIGURE 2.1 – Exemple de représentation d'une solution avec deux véhicules

autrement dit, le nombre de routes non vides. Le deuxième objectif f_2 à minimiser est la somme des distances parcourues par tous les véhicules. Si un véhicule n'est pas utilisé, la distance parcourue par celui-ci est nulle. Ainsi une solution S_1 sera meilleure qu'une solution S_2 si et seulement si $f_1(S_1) < f_1(S_2)$ ou, dans le cas ou $f_1(S_1) = f_1(S_2)$, $f_2(S_1) < f_2(S_2)$. Dans cette thèse, pour évaluer une solution S nous utilisons une fonction pondérée $f(S) = P.f_1(S) + f_2(S)$ avec P un coefficient suffisamment élevé pour favoriser f_1 dans l'évaluation de la solution.

2.3.2 Les opérateurs d'insertion

Que ce soit pour le LNS ou pour le GA, la construction d'une solution est basée sur l'insertion successive de requêtes dans des solutions partielles. L'insertion d'une requête dans un véhicule correspond à l'ajout du nœud de collecte et du nœud de livraison dans la route du véhicule. Des opérateurs d'insertion sont utilisés pour insérer une requête dans une solution. Nous en utilisons deux différents. Le premier insère une requête dans un seul véhicule, donc sans transbordement. Il est décrit par l'algorithme 2.3. Le deuxième insère une requête dans deux véhicules différents, donc avec un transbordement. Il est décrit par l'algorithme 2.4. Les deux opérateurs prennent deux arguments en entrée :

— Une requête $r(p \rightarrow d)$ avec p le point de collecte et d le point de livraison de r.

— La solution partielle S dans laquelle la requête doit être insérée.

L'algorithme 2.3 présente l'opérateur d'insertion sans transbordement. Cet opérateur insère une requête à la position avec le coût d'insertion le plus faible. Pour chaque véhicule, nous testons l'insertion de p et de d dans toutes les positions possibles. p peut être inséré n'importe où entre v^+ et v^-. d peut être

Algorithm 2.3 bestSingleInsertion

Input: $r(p \rightarrow d), S$
Output: S

```
1: bestRoute ← ∅
2: C(bestRoute) ← MAX_VALUE
3: for all Route v do
4:    for i from 2 to nb(v) do
5:       for j from i + 1 to nb(v) do
6:          currentRoute ← v
7:          if currentRoute = ∅ then
8:             insert v+ and v− in currentRoute
9:          end if
10:         if p is not in currentRoute then
11:            insert p at position i in currentRoute
12:         end if
13:         if d is not in currentRoute then
14:            insert d at position j in currentRoute
15:         end if
16:         if C(currentRoute) < C(bestRoute) then
17:            if currentRoute feasible then
18:               bestRoute ← currentRoute
19:            end if
20:         end if
21:      end for
22:   end for
23: end for
24: replace appropriate v in S with bestRoute
```

inséré n'importe où entre p et v^- (le point de livraison doit forcément être après le point de collecte). Les nœuds p et d sont insérés dans la route uniquement s'ils n'y sont pas déjà (un nœud ne peut être présent qu'une seule fois dans une route). Si le coût de *currentRoute* après avoir inséré la requête est inférieur au meilleur coût actuel, alors nous vérifions si *currentRoute* est faisable. Si c'est le cas, *currentRoute* devient la nouvelle meilleure insertion. A la fin de l'algorithme, le véhicule v adéquat de S est remplacé par *bestRoute*. Par exemple, si la meilleure insertion a lieu dans le véhicule 2, alors v_2 est remplacé par *bestRoute*. Nous avons une complexité de $O(|V|.|N|^2)$ pour tester toutes les possibilités et une

complexité de $O(|N|)$ pour vérifier la faisabilité. La complexité de l'opérateur est donc de $O(|V|.|N|^3)$.

Algorithm 2.4 BestDoubleInsertion

Input: $r(p \rightarrow d), S$
Output: S

```
1: allRoutes1 ← ∅
2: allRoutes2 ← ∅
3: bestSolution ← ∅
4: f(bestSolution) ← MAX_VALUE
5: for all t ∈ T do
6:    for all Route v do
7:       for i from 2 to nb(v) do
8:          for j from i + 1 to nb(v) do
9:             currentRoute1 ← v
10:            currentRoute2 ← v
11:            insert p at position i in currentRoute1
12:            insert t at position j in currentRoute1
13:            put currentRoute1 in allRoutes1 ordered by ascending cost
14:            insert t at position i in currentRoute2
15:            insert d at position j in currentRoute2
16:            put currentRoute2 in allRoutes2 ordered by ascending cost
17:         end for
18:      end for
19:   end for
20:   currentSolution ← getBestFeasibleSolution(allRoutes1, allRoutes2, S)
21:   if f(currentSol) < f(bestSol) then
22:      bestSol ← currentSol
23:   end if
24: end for
25: S ← bestSol
```

L'opérateur de double insertion est décrit par l'algorithme 2.4. L'insertion d'une requête $r(p \rightarrow d)$ dans deux routes distinctes fonctionne de la manière suivante (figure 2.2) :

— Deux véhicules différents v et v' et un point de transbordement t sont sélectionnés.

— Le nœud de collecte p est inséré dans le premier véhicule v.

— Le nœud de transbordement t est inséré dans v après p.

— Le nœud de transbordement t est inséré dans le deuxième v'.

— Le nœud de livraison d est inséré dans v' après t.

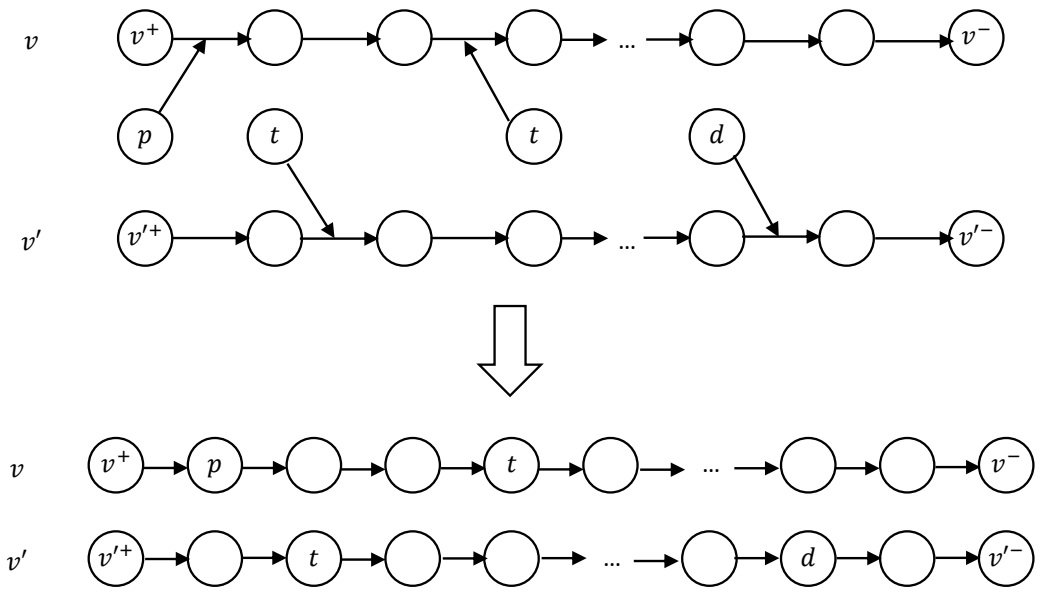

FIGURE 2.2 – Insertion d'une requête $r(p \rightarrow d)$ dans les véhicules v et v' en utilisant le point de transbordement t

Chaque nœud est inséré dans une route si et seulement si ce nœud n'est pas déjà présent dans la route. Contrairement à l'algorithme d'insertion sans transbordement (algorithme 2.3), tester toutes les insertions possibles pour la double insertion pour chaque point de transbordement et pour chaque paire de routes serait trop coûteux en termes de temps de calcul. En effet la complexité serait de $O(|T|.|V|^2.|N|^5)$. Pour éviter de tester toutes les possibilités, nous procédons de la manière suivante pour chaque point de transbordement t :

— Toutes les insertions possibles du couple de nœud $(p \rightarrow t)$ dans chacun des véhicules sont rangées dans la liste $allRoutes1$ par ordre croissant du coût d'insertion.

— Toutes les insertions possibles du couple de nœud $(t \rightarrow d)$ dans chacun des véhicules sont rangées dans la liste $allRoutes2$ par ordre croissant du coût d'insertion.

— A partir de ces deux listes, nous testons les paires d'insertions de $(p \rightarrow t)$ et de $(t \rightarrow d)$ dans l'ordre croissant des coûts.

— Dès que nous trouvons une paire d'insertions faisable, alors c'est la meilleure insertion possible en termes de coût dans le point de transbordement t, puisque les paires sont testées dans l'ordre croissant des coûts d'insertion.

— Nous répétons ce procédé pour chaque point de transbordement et récupérons la meilleure solution.

Avec cette méthode, dans le meilleur des cas, lorsque la première paire d'insertions testée est faisable, la complexité de l'algorithme est de $O(|T|.|V|.|N|^2)$. Dans le pire des cas, c'est-à-dire que seule la paire d'insertions la plus coûteuse est faisable ou aucune paire d'insertions n'est faisable, la complexité de l'algorithme est de $O(|T|.|V|^2.|N|^5)$.

L'algorithme 2.5 décrit la façon dont les paires d'insertions sont parcourues et comment la meilleure paire d'insertions faisable est trouvée.

La variable *index* représente la meilleure paire d'insertions possible actuelle. Son premier élément correspond à l'indice de l'insertion de $(p \rightarrow t)$ rangée dans $allRoutes1$. Son deuxième élément correspond à l'indice de l'insertion de $(t \rightarrow d)$ rangée dans $allRoutes2$. Comme $allRoutes1$ et $allRoutes2$ sont classés dans l'ordre croissant des coûts d'insertion, la paire d'insertion avec le coût le plus faible est la paire d'indice (0;0). Elle est rangée dans *routePair* qui est une liste dans laquelle nous stockons les paires à tester. *routePair* est trié dans l'ordre croissant des coûts d'insertion également, donc son premier élément est toujours la meilleure paire d'insertion en terme de coût. Tant que *routePair* est non vide, c'est-à-dire tant que nous n'avons pas testé toutes les paires, nous vérifions si l'insertion de la première paire contenue dans *routePair* est faisable ou non. Si elle est faisable, alors c'est la meilleure insertion possible et nous renvoyons la solution. Si ce n'est pas le cas, nous ajoutons les deux paires suivantes (une paire avec seulement le premier élément incrémenté et une paire avec seulement le deuxième élément incrémenté) dans *routePair*.

Maintenant que les opérateurs d'insertion utilisés par le LNS et le GA ont été décrits, les deux métaheuristiques peuvent être présentées.

Algorithm 2.5 getBestFeasibleSolution

Input: $allRoutes1, allRoutes2, solution$

```
1: index ← (0;0)
2: routePair ← {index}
3: while routePair ≠ ∅ do
4:    index ← routePair[0]
5:    v1 ← allRoutes1[index.first]
6:    v2 ← allRoutes2[index.second]
7:    if v1(i) ≠ v2(i) then
8:       S ← solution
9:       replace appropriate route in S with v1 and v2
10:      if S feasible then
11:         return S
12:      end if
13:   end if
14:   erase first element in currentRoutePair
15:   if index.first + 1 < sizeof(allRoutes1) then
16:      put (index.first+1; index.second) in routePair sorted by ascending cost
      of the corresponding pair of routes
17:   end if
18:   if index.second + 1 < sizeof(allRoutes2) then
19:      put (index.first; index.second+1) in routePair sorted by ascending cost
      of the corresponding pair of routes
20:   end if
21: end while
22: return solution
```

2.4 LNS pour le PDPT

Dans cette section, nous allons décrire une première méta-heuristique basée sur la LNS pour résoudre le problème du PDPT. Après avoir décrit le fonctionnement global du LNS, nous présentons les composants principaux de notre LNS qui sont :

— La construction d'initialisation de la solution.

— Les opérateurs de destruction.

— Les opérateurs de réparation.

2.4.1 Fonctionnement général du LNS

Le LNS a tout d'abord été introduit par Shaw (1998) pour résoudre le problème de VRP. Le principe général du LNS est de détruire et de réparer de manière itérative une solution dans le but de l'améliorer. Le fonctionnement de cette méthode est décrit dans l'algorithme 2.6. Les paramètres de l'algorithme sont les suivants :

— R : l'ensemble des requêtes à traiter.

— $nbIterMax$: le nombre d'itérations maximum à réaliser.

— $acceptCriterion$: le pourcentage de dégradation par rapport à la meilleure solution pour que la solution courante soit acceptée.

Algorithm 2.6 LNS

Input: $R, nbIterMax, acceptCriterion$

1: $currentSolution \leftarrow initiateSol(R)$ {random initialization}
2: $bestSolution \leftarrow currentSolution$
3: $RTM = \varnothing$
4: **for** $iter$ from 1 to $nbIterMax$ **do**
5: $S \leftarrow currentSolution$
6: choose uniformly randomly $destroy$ operator
7: $S \leftarrow destroy(S, RTM)$
8: choose uniformly randomly $repair$ operator
9: $S \leftarrow repair(S, RTM)$
10: **if** $f(S) \leq f(bestSolution) * acceptCriterion$ **then**
11: $currentSolution \leftarrow S$
12: **end if**
13: **if** $f(S) \leq f(bestSolution)$ **then**
14: $bestSolution \leftarrow S$
15: **end if**
16: **end for**
17: **return** $bestSolution$

Premièrement, nous initialisons la solution courante à l'aide d'une heuristique constructive décrite dans la section suivante. Ensuite, la meilleure solution est initialisée avec la solution initiale, puis l'étape itérative démarre.

A chaque itération, un opérateur de destruction est sélectionné au hasard parmi les quatre proposés. Cet opérateur est appliqué à la solution courante S.

Dans ce problème, une fonction de destruction enlève un certain nombre de requêtes d'une solution. La liste des requêtes enlevées est stockée dans RTM. S est donc une solution partielle. Un opérateur de réparation est ensuite choisi aléatoirement parmi les trois fournis. Le but de cet opérateur de réparation est de réinsérer les requêtes de l'ensemble RTM dans la solution S. A noter que les requêtes peuvent être réinsérées à la même position, dans ce cas l'itération a été inutile. Si S respecte le critère d'acceptation, alors S devient la nouvelle solution courante. Ce critère d'acceptation permet de dégrader la solution courante afin de ne pas rester dans un optimum local. Ainsi, si $acceptCriterion = 1.05$, la solution S est acceptée si la différence entre $f(S)$ et $f(bestSolution)$ est inférieure à 5%. Puis si S est de meilleure qualité que $bestSolution$ alors $bestSolution$ est remplacée par S. A la fin de l'exécution, l'algorithme renvoie la meilleure solution trouvée.

2.4.2 Initialisation de la solution

L'algorithme 2.7 décrit la création de la solution initiale. C'est une heuristique de type constructive, c'est-à-dire que nous partons d'une solution vide et que nous insérons les requêtes une à la fois jusqu'à ce qu'elles aient été toutes insérées. L'algorithme prend un seul paramètre, la liste des requêtes à traiter.

Algorithm 2.7 initiateSol

Input: R

1: $S \leftarrow \varnothing$
2: $LR \leftarrow R$ {LR = list of non inserted requests}
3: **while** $LR \neq \varnothing$ **do**
4: choose randomly r in LR
5: $bestSingleInsertion(r, S)$
6: $LR \leftarrow LR \setminus \{r\}$
7: **end while**
8: **return** S

La solution S est initialement nulle. Toutes les requêtes de R sont placées dans la liste des requêtes à traiter LR. Cette liste contient toutes les requêtes qui n'ont pas encore été insérées. Tant que cette liste n'est pas vide, les opérations

suivantes sont effectuées :

— Une requête r est choisie au hasard dans LR.

— r est insérée dans S grâce à l'opérateur $bestSingleInsertion$ introduit précédemment.

— r est supprimée de la liste LR.

La requête est insérée dans la route dans laquelle le coût d'insertion est le plus faible. Il est à noter que lors de l'initialisation, les requêtes sont insérées sans transbordement. La solution initiale S est donc également une solution pour le PDP. A la fin de l'algorithme, la solution S est renvoyée.

2.4.3 Les opérateurs de destruction

Dans le LNS pour le PDPT, les opérateurs de destruction ont pour but de choisir un certain nombre de requêtes et de les enlever de la solution S. Nous utilisons quatre opérateurs différents. Ils fonctionnent tous les quatre de la même manière. Ils diffèrent dans la façon de sélectionner les requêtes à enlever. Les trois opérateurs sont décrits dans les algorithmes 2.8 à 2.11. Ils prennent tous trois deux arguments en entrée, la solution S à détruire et la liste des requêtes à enlever RTM.

Algorithm 2.8 destroy1

Input: S, RTM
Output: RTM
1: $LR \leftarrow R$
2: choose randomly $nbRTM$ between 1 and $|R|$
3: **for** i from 1 to $nbRTM$ **do**
4: choose randomly r in LR
5: remove r from S
6: $RTM \leftarrow RTM \cup \{r\}$
7: $LR \leftarrow LR \setminus \{r\}$
8: **end for**
9: **return** S

Dans l'opérateur $destroy1$ (algorithme 2.8) les requêtes à supprimer de la

solution S sont choisies aléatoirement. Le nombre de requêtes à supprimer est lui aussi choisi aléatoirement.

Algorithm 2.9 destroy2

Input: S, RTM
Output: RTM
1: $LR \leftarrow R$
2: choose randomly $v \neq \varnothing$ in S
3: **for all** r in v **do**
4: remove r from S
5: $RTM \leftarrow RTM \cup \{r\}$
6: **end for**
7: **return** S

Dans l'opérateur *destroy*2 (algorithme 2.9), un véhicule v est choisi aléatoirement parmi tous les véhicules non vides. Toutes les requêtes situées dans ce véhicule sont supprimées de la solution S. Une requête r est considérée dans un véhicule v dans l'un des trois cas suivants :

— Le point de collecte et le point de livraison de r sont tous les deux situés dans v.

— Seul le point de collecte de r est situé dans v.

— Seul le point de livraison de r est situé dans v.

Après application de l'opérateur *destroy*2, le véhicule v est donc vide. L'intérêt de supprimer toutes les requêtes d'un véhicule est de donner la possibilité d'insérer ces requêtes dans d'autres véhicules afin d'en utiliser moins et ainsi satisfaire l'objectif f_1.

De la même manière que dans l'opérateur *destroy*2 (algorithme 2.9), l'opérateur *destroy*3 (algorithme 2.10) supprime toutes les requêtes contenues dans un véhicule v. Cependant, v n'est pas choisi au hasard mais c'est le véhicule non vide qui contient le moins de nœuds. L'avantage de choisir v de cette manière est que le nombre de requêtes à réinsérer est plus petit, donc la probabilité pour que le nombre de véhicules utilisés soit réduit est plus grande.

L'opérateur *destroy*4 a été proposé par Shaw (1998). Cet algorithme tente de sélectionner les requêtes qui sont similaires par leur quantité, par la position

Algorithm 2.10 destroy3

Input: S, RTM
Output: RTM
1: $LR \leftarrow R$
2: $route \leftarrow$ route with the smallest number of nodes
3: **for all** r in $route$ **do**
4: remove r from S
5: $RTM \leftarrow RTM \cup \{r\}$
6: **end for**
7: **return** S

ou par la fenêtre de temps de leur point de collecte et de livraison. D'après Shaw (1998), si les requêtes enlevées sont similaires, la probabilité pour qu'elles soient réinsérées dans une autre route ou à une autre position est plus élevée. En effet, si les requêtes enlevées sont trop similaires, elles auront tendance à être réinsérées à la même position que précédemment pour garantir la faisabilité de la solution. Et dans ce cas, nous n'aurons pas d'amélioration de la solution. L'opérateur consiste en trois opérations :

— Une requête choisie aléatoirement est enlevée de la solution S puis est placée dans la liste RTM.

— Les requêtes encore présentes dans la solution S (présentes dans LR) sont triées dans l'ordre croissant de la mesure de leur ressemblance RM_r définie par l'équation 2.1.

— Une requête est de nouveau choisie au hasard et enlevée de la solution S, mais les requêtes avec le plus petit RM ont une plus grande probabilité d'être sélectionnées.

Ces trois opérations sont répétées jusqu'à ce que le nombre de requêtes enlevées ait atteint le nombre voulu $nbRTM$. Shaw (1998) a proposé la fonction suivante pour calculer la ressemblance d'une requête $r(i \rightarrow j)$:

Algorithm 2.11 destroy4

Input: S, RTM
Output: RTM

$LR \leftarrow R$
choose randomly $nbRTM$ between 1 and $|R|$
$i = 1$
while $i \leq nbRTM$ **do**
  choose randomly r in LR
  remove r from S
  $RTM \leftarrow RTM \cup \{r\}$
  $LR \leftarrow LR \setminus \{r\}$
  for all $r \in LR$ **do**
    $RM_r = \sum_{r'(k \rightarrow l) \in RTM} \theta_1 |Q_r - Q_{r'}| + \theta_2 \left| d_{r_p r'_p} + d_{r_d r'_d} \right| + \theta_3 \left(\left| t_{r_p} - t_{r'_p} \right| + \left| t_{r_d} - t_{r'_d} \right| \right)$
  end for
  sort request in LR by ascending RM
  generate randomly a number $p \in [0, 1[$
  $r = LR[p^{\alpha}.|LR|]$
  remove r from S
  $RTM \leftarrow RTM \cup \{r\}$
  $LR \leftarrow LR \setminus \{r\}$
  $i = i + 2$
end while
return S

$$RM_{r(r_p \rightarrow r_d)} = \sum_{r'(k \rightarrow l) \in RTM} \theta_1 |Q_r - Q_{r'}| + \theta_2 \left| d_{r_p r'_p} + d_{r_d r'_d} \right| + \theta_3 \left(\left| t_{r_p} - t_{r'_p} \right| + \left| t_{r_d} - t_{r'_d} \right| \right) \tag{2.1}$$

où RTM est la liste des requêtes enlevées, Q_r et $Q_{r'}$ sont les quantités des requêtes r et r', $d_{r_p r'_p}$ et $d_{r_d r'_d}$ sont les distances entre les nœuds r_p et r'_p et r_d et r'_d, et t_{r_p}, t_{r_d}, $t_{r'_p}$ et $t_{r'_d}$ sont les temps de début de service de la route pour les nœuds r_p, r'_p, r_d et r'_d dans la solution courante. θ_1, θ_2 et θ_3 sont les poids pour chaque composante de la formule. Plus la valeur de RM_r est petite, plus la requête r est

similaire aux requêtes dans RTM.

Pour sélectionner une requête dans la troisième opération, la liste des requêtes LR est triée dans l'ordre croissant des valeurs de RM. Un nombre aléatoire $p \in (0,1)$ est généré, et pour un paramètre prédéfini $\alpha \geq 1$, nous sélectionnons la requête $LR[p^{\alpha}.|LR|]$. Si $\alpha = \infty$ la requête avec la plus faible mesure de ressemblance est toujours sélectionnée. Si $\alpha = 1$, la requête est choisie aléatoirement de manière uniforme.

Ces quatre algorithmes de destruction (algorithmes 2.8 à 2.11) stockent les requêtes supprimées dans la liste RTM. La solution S en sortie est une solution partielle. Pour enlever une requête, il faut effectuer les opérations suivantes :

— Parcourir le ou les véhicules dans lesquels la requête est présente.

— Trouver les nœuds qui contiennent la requête (le nœud de collecte, le nœud de livraison et éventuellement un point de transbordement).

— Supprimer la requête de ces nœuds. Si un nœud ne contient aucune autre requête, alors il peut être enlevé de la route.

— Si un véhicule ne contient aucun nœud autre que son dépôt de départ et son dépôt d'arrivée, alors les dépôts sont supprimés du véhicule.

2.4.4 Les opérateurs de réparation

Après avoir enlevé des requêtes, il faut les réinsérer pour réparer la solution. Il y a plusieurs choses à prendre en compte lors de la réparation d'une solution. Tout d'abord, il faut déterminer l'ordre dans lequel les requêtes seront insérées. Ensuite, pour chaque requête à réinsérer, il faut déterminer si elle sera réinsérée en utilisant ou non le transbordement. Nous utilisons pour cela deux opérateurs de réparation principaux décrits par les algorithmes 2.12 et 2.13 qui diffèrent par leur manière de choisir l'ordre d'insertion des requêtes. Ils prennent tous les deux en arguments la solution S à réparer et la liste RTM des requêtes à réinsérer. Ces deux algorithmes fonctionnent de la manière suivante : tant qu'il reste des requêtes à réinsérer, une requête r est choisie parmi celle-ci et est remise dans S à l'aide d'un opérateur d'insertion. Cet opérateur diffère selon la règle d'insertion choisie. Ces règles d'insertion sont au nombre de trois et sont

décrites par les algorithmes 2.14, 2.15 et et 2.16. La solution S est retournée à la fin de chacun des trois algorithmes. Les opérateurs d'insertion sont présentés dans la section suivante.

Algorithm 2.12 repair1

Input: S, RTM

$LR \leftarrow RTM$
choose randomly an insertion rule IR
while $LR \neq \varnothing$ **do**
    choose randomly r in LR
    $S = IR(S, r)$
    $LR \leftarrow LR \setminus \{r\}$
end while
return S

Dans l'opérateur $repair1$ (algorithme 2.12), l'ordre d'insertion des requêtes est choisi aléatoirement.

Algorithm 2.13 repair2

Input: S, RTM

$LR \leftarrow RTM$
choose randomly an insertion rule IR
for all request $r \in LR$ **do**
    $IE_{r(i \rightarrow j)} = \beta_1 D_{ij} - \beta_2(|b_i - a_i| + |b_j - a_j|) + \beta_3 Q_r$
end for
sort request in LR by descending Q_r
while $LR \neq \varnothing$ **do**
    generate randomly a number $p \in [0, 1[$
    $r = LR[p^{\alpha}.|LR|]$
    insert r applying the insertion rule IR
    $LR \leftarrow LR \setminus \{r\}$
end while
return S

Dans l'opérateur $repair2$ (algorithme 2.13), l'ordre des requêtes est basé sur la facilité d'insertion des requêtes. En effet, plus une requête aura une quantité de cargaison importante, des fenêtres de temps réduites ou une distance entre son point de collecte et son point de livraison importante, plus cette requête sera

difficile à insérer. L'idée est donc de les insérer en premier. Cet opérateur est inspiré de Qu et Bard (2012). Pour calculer la facilité d'insertion d'une requête r en fonction de sa distance entre son point de collecte et de livraison, sa fenêtre de temps et sa quantité de cargaison, nous utilisons la formule suivante :

$$IE_{r(i\to j)} = \beta_1 d_{ij} - \beta_2(|l_i - e_i| + |l_j - e_j|) + \beta_3 Q_r \tag{2.2}$$

où d_{ij} est la distance entre les nœuds i et j, e_i et e_j sont les heures d'ouverture des nœuds i et j, l_i et l_j sont les heures de fermeture des nœuds i et j, et Q_r est la quantité de cargaison de la requête r. β_1, β_2 et β_3 sont les poids de chaque terme de l'équation 2.2. Plus IE_r est grand, plus la requête sera difficile à insérer, et donc plus elle aura de chance d'être sélectionnée pour l'insertion. En effet si une requête est difficile à insérer, il peut être plus intéressant de la traiter dans les premières.

De la même manière que dans l'opérateur de destruction 4, pour sélectionner les requêtes à insérer, la liste des requêtes LR est triée dans l'ordre décroissant des valeurs de IE. Un nombre aléatoire $p \in (0,1)$ est généré, et pour un paramètre prédéfini $\alpha \geq 1$, nous sélectionnons la requête $LR[p^\alpha.|LR|]$. Si $\alpha = \infty$ la requête avec la plus faible facilité d'insertion est toujours sélectionnée. Si $\alpha = 1$, la requête est choisie aléatoirement de manière uniforme.

Quand une requête est sélectionnée pour être réinsérée, il faut choisir l'opérateur d'insertion. Pour cela nous avons établi trois règles d'insertion décrites par les algorithmes 2.14, 2.15 et et 2.16.

Algorithm 2.14 IR1

Input: S, r
1: $bestSingleInsertion(r, S)$
2: **return** S

Dans la règle d'insertion 1 (algorithme 2.14), toutes les requêtes sont insérées sans transbordement, c'est-à-dire en utilisant l'opérateur d'insertion $BestSingleInsertion$ uniquement.

Dans la règle d'insertion 2 (algorithme 2.15), nous testons l'insertion sans

Algorithm 2.15 IR2

Input: S, r

1: $currentSolution1 \leftarrow S$
2: $currentSolution2 \leftarrow S$
3: $bestSingleInsertion(r, currentSolution1)$
4: $bestDoubleInsertion(r, currentSolution2)$
5: **if** $f(currentSolution2) \leq f(currentSolution1)$ **then**
6: $\quad S \leftarrow currentSolution2$
7: **else**
8: $\quad S \leftarrow currentSolution1$
9: **end if**
10: **return** S

transbordement ainsi que l'insertion avec transbordement et nous prenons l'insertion avec le coût le plus faible.

Algorithm 2.16 IR3

Input: S, r

1: choose randomly $op \in [0, 1[$
2: **if** $op < \gamma$ **then**
3: $\quad bestDoubleInsertion(r, S)$
4: **else**
5: $\quad IR2(S, r)$
6: **end if**
7: **return** S

Dans la règle d'insertion 3 (algorithme 2.16), nous utilisons la règle d'insertion 2, mais nous donnons une probabilité γ pour chaque requête d'être insérée uniquement en utilisant le transbordement. L'avantage de cette règle est que nous pouvons forcer l'utilisation d'un point de transbordement même dans les cas où une insertion de ce type peut paraître à priori désavantageuse. En effet, dans la plupart des cas, l'utilisation d'un point de transbordement n'est bénéfique que lorsque plusieurs requêtes sont transférées à travers ce point. La règle $IR2$ (algorithme 2.15) ne permet pas de trouver ces solutions puisque le transfert d'une requête peut avoir un coût plus élevé lorsque les requêtes sont insérées une à une.

La figure 2.3 permet de comprendre l'intérêt de la règle d'insertion 3. Dans cette figure, deux véhicules sont disponibles, e_1 et e_2 sont les dépôts respectifs des véhicules 1 et 2. Il y a deux requêtes à satisfaire, p_1 et d_1 sont les points de collecte et de livraison de la requête 1 et p_2 et d_2 sont les points de collecte et de livraison de la requête 2. Un véhicule ne peut prendre qu'une seule requête à la fois et nous supposons que les fenêtres de temps ne permettent pas à un seul véhicule de traiter les deux requêtes successivement. Les valeurs situées sur chaque arête sont les coûts de ces arêtes. Le point T est un point de transbordement. Prenons le cas où pour chaque requête nous prenons la meilleure insertion en terme de coût entre l'insertion sans transbordement et l'insertion avec transbordement. Lorsque nous voulons insérer la requête 1, la meilleure solution sans transbordement est la suivante : $e_1 \rightarrow p_1 \rightarrow d_1 \rightarrow e_1$ pour le véhicule 1. Le coût de cette insertion est de 22 (3+10+9) et un seul véhicule est utilisé. La meilleure insertion avec transbordement est la suivante : $e_1 \rightarrow p_1 \rightarrow T \rightarrow e_1$ pour le véhicule 1 et $e_2 \rightarrow T \rightarrow d_1 \rightarrow e_2$ pour le véhicule 2. Le coût de cette insertion est de 24 (3+5+4+4+5+3) et deux véhicules sont utilisés. Si nous appliquons la règle d'insertion 2, nous devons sélectionner l'insertion sans transbordement. Une fois que la requête 1 est dans la solution, nous insérons ensuite la requête 2. De la même manière que précédemment, nous testons l'insertion sans transbordement et l'insertion avec transbordement. La meilleure solution sans transbordement est la suivante : $e_2 \rightarrow p_2 \rightarrow d_2 \rightarrow e_2$ pour le véhicule 2. Le coût de cette insertion est de 22 (3+10+9) et un véhicule supplémentaire est utilisé. L'insertion de la requête 2 avec transbordement n'est pas possible. En effet le véhicule 1 gère déjà la collecte et la livraison de la requête 1. Or si nous voulons insérer la requête 2 avec transbordement, cette requête sera collectée par le véhicule 2 puis déposée au point T. Le véhicule 1 sera alors en charge de collecter la requête 2 au point T après avoir livré la requête 1. Or nous supposons que les fenêtres de temps ne permettent pas le traitement successif des deux requêtes par un seul véhicule. La seule insertion possible pour la requête 2 est donc l'insertion sans transbordement. Le coût total de la solution est donc de 44 (22+22) et deux véhicules sont utilisés.

Prenons maintenant le cas où nous forçons l'utilisation des points de transbordement. Lorsque nous voulons insérer la requête 1, la meilleure insertion

avec transbordement est la suivante : $e_1 \rightarrow p_1 \rightarrow T \rightarrow e_1$ pour le véhicule 1 et $e_2 \rightarrow T \rightarrow d_1 \rightarrow e_2$ pour le véhicule 2. Le coût de cette insertion est de 24 (3+5+4+4+5+3) et deux véhicules sont utilisés. Lorsque nous voulons insérer la requête 2 dans la solution existante, la meilleure insertion avec transbordement est la suivante : $e_1 \rightarrow p_1 \rightarrow T \rightarrow d_2 \rightarrow e_1$ pour le véhicule 1 et $e_2 \rightarrow p_2 \rightarrow T \rightarrow d_1 \rightarrow e_2$ pour le véhicule 2. Le coût de cette insertion est de 8 (5+3-4+3+5-4). Le coût total de la solution est donc 32 (24+8).

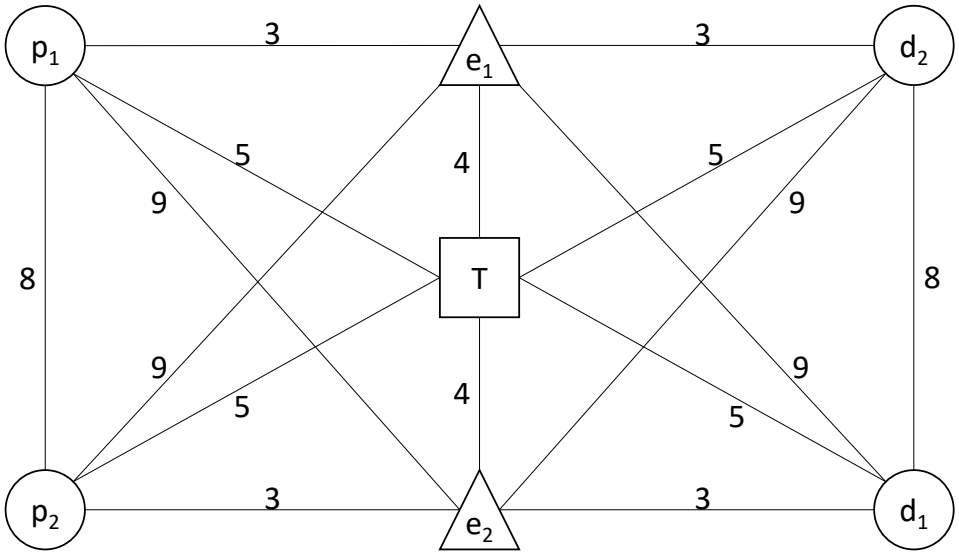

FIGURE 2.3 – Exemple où forcer l'utilisation d'un point de transbordement améliore la solution

Cette dernière solution est bien meilleure que la solution précédente et n'aurait pas pu être trouvée en appliquant les règles d'insertion 1 et 2 uniquement. Il est donc nécessaire de forcer l'utilisation du transbordement sur certaines requêtes pour trouver des solutions plus intéressantes.

2.5 GA pour le PDPT

Dans cette section, nous décrivons une deuxième méta-heuristique proposée pour résoudre le problème du PDPT. Cette deuxième méthode est un algorithme génétique. Après avoir décrit le fonctionnement global du GA, nous présentons

les composants principaux de ce GA qui sont :

— La représentation, le décodage et le calcul de fitness des individus.

— La fonction d'initialisation de la population.

— Les opérateurs de croisement.

— Les opérateurs de mutation.

— La sélection des individus pour la génération suivante.

— La mise en cache de solutions partielles pour économiser du temps de calcul.

2.5.1 Fonctionnement général du GA

Le principe général du GA est de modifier des individus représentant chacun une solution et de les mélanger entre eux afin d'avoir des individus toujours plus performants. Cette méta-heuristique est une méthode à base de population inspirée des croisements génétiques du monde du vivant. Le fonctionnement de cette méthode est décrite dans l'algorithme 2.17. Les paramètres de l'algorithme sont les suivants :

— R : l'ensemble des requêtes à traiter.

— $nbIterMax$: le nombre d'itérations maximum à réaliser.

— $popSize$: la taille de la population d'individus.

— $nbCrossover$: le nombre de croisements à réaliser pour chaque opérateur de croisement à chaque itération.

Premièrement, nous initialisons la population en créant le nombre voulu $popSize$ d'individus. Les individus sont générés en donnant des valeurs aléatoires à leurs gènes. Ensuite, l'étape itérative démarre. A chaque itération, nous effectuons un certain nombre de croisements à l'aide de deux opérateurs de croisement différents. Un croisement est une opération qui consiste à créer deux nouveaux individus enfants à partir de deux individus parents. Ces nouveaux individus sont systématiquement ajoutés à la population. Ensuite, un individu est sélectionné pour lui appliquer une mutation. Cette mutation transforme l'individu pour en créer un nouveau qui est ajouté à la population. L'opérateur

Algorithm 2.17 GA

Input: $R, nbIterMax, popSize, nbCrossover$

```
population ← initiatePop(R, popSize)
for iter from 1 to nbIterMax do
    crossover1Point(population, nbCrossover)
    crossover2Points(population, nbCrossover)
    mutation(population)
    sort population by ascending fitness
    keep only the popSize best individuals in population
end for
return solution of the best individual
```

de mutation sert essentiellement à diversifier les individus de la population afin de ne pas converger trop rapidement vers un optimum local. Après application des croisements et de la mutation, la taille de la population s'est donc agrandie. Il faut ensuite sélectionner les individus qui participeront à la prochaine génération, c'est-à-dire à la prochaine itération. En effet, pour éviter que la taille de la population explose, il est préférable de garder le même nombre $popSize$ d'individus dans la population à chaque itération. Pour cela, la population est triée par ordre croissant des fitness des individus, et les meilleurs sont sélectionnés pour participer à la génération suivante (élitisme). A la fin de l'exécution, l'algorithme renvoie l'individu avec le meilleur fitness.

2.5.2 Représentation et décodage d'un individu

Dans les GA, le choix de la représentation des individus est crucial. L'efficacité d'un GA est en effet souvent très dépendante de ce choix. La représentation d'un individu est aussi très fortement dépendante du problème à résoudre et n'est souvent utilisable que pour un problème donné. De plus, un problème peut posséder plusieurs représentations possibles plus ou moins efficaces. Un individu peut être représenté de manière assez explicite de façon à ce que son fitness soit facile à calculer. Mais en contrepartie, lors des opérations de croisement et de mutation, il peut être très difficile d'obtenir des solutions réalisables et dans ce cas il faut effectuer des opérations de réparation coûteuses en temps de calcul. A l'inverse, un individu peut être représenté de manière peu explicite

mais le décodage et le calcul du fitness peuvent être coûteux en temps. Dans ce cas, il ne sera pas forcément nécessaire de réparer les individus après les croisements et les mutations. Pour coder les individus pour le problème du PDPT, nous sommes partis du constat que lors de la construction d'une solution, l'ordre dans lequel les requêtes étaient insérées au fur et à mesure avait une importance capitale. En effet trouver le bon ordre d'insertion peut mener à trouver la solution optimale. Dans cette optique, nous avons donc décidé de représenter les individus en fonction de l'ordre d'insertion des requêtes. La première idée était de représenter un individu par un vecteur avec une taille correspondante au nombre de requêtes. Chaque gène représente une requête différente et la place du gène dans le vecteur correspond au numéro d'insertion de la requête. Par exemple, dans la figure 2.4, l'individu est représenté par la deuxième ligne. Dans cet exemple, la requête 5 sera insérée en première, puis ce sera la requête 2, puis la 6, etc... La dernière requête insérée sera la 4.

Ordre	1	2	3	4	5	6	7	8	9
Requête	5	2	6	8	9	1	3	7	4

Figure 2.4 – Exemple de la première représentation envisagée d'un individu

Cette représentation n'a pas été retenue principalement car les croisements et les mutations entraînent trop de solutions non faisables. En effet, après une opération de croisement ou de mutation, il est très fréquent qu'une ou plusieurs requêtes soient présentes plusieurs fois dans le même individu et que d'autres requêtes par contre ne le soient pas. Il faut alors supprimer les doublons et replacer les requêtes manquantes. De plus ces opérations de réparation ne sont pas compensées par un décodage facile, puisque le calcul de fitness est aussi assez coûteux. En effet, pour chaque individu, il faut construire la solution entièrement en insérant toutes les requêtes. Cette représentation n'a donc pas été retenue.

Ensuite, nous avons gardé la représentation d'un individu par un vecteur avec une taille correspondante au nombre de requêtes. Mais les gènes sont maintenant un poids qui est un réel compris entre 0 et 1 et la place du gène dans le vecteur correspond au numéro de la requête. Ce sont ces poids qui vont

déterminer l'ordre d'insertion des requêtes. Plus une requête aura un poids faible, plus cette requête sera insérée tôt. Plus une requête aura un poids élevé, plus cette requête sera insérée tard. Par exemple, dans la figure 2.5, l'individu est représenté par la deuxième ligne. Dans cet exemple, la requête 8 sera insérée en première puisqu'elle a le poids le plus faible. Ensuite ce sera la requête 5, puis la 2, etc... La dernière requête insérée sera la requête 3.

Requête	1	2	3	4	5	6	7	8	9
Poids	0.21	0.14	0.7	0.59	0.1	0.64	0.57	0.05	0.26

FIGURE 2.5 – Exemple de représentation d'un individu

L'avantage de cette représentation par rapport à la précédente est que malgré les croisements et les mutations, toutes les requêtes sont toujours représentées dans les individus et donc aucune réparation n'est nécessaire. Cependant, comme pour la représentation précédente, le calcul de fitness est assez coûteux. En effet, une fois que l'ordre d'insertion des requêtes a été établi, pour chaque requête nous testons l'insertion sans et avec transbordement décrite par les algorithmes 2.3 et 2.4 et nous appliquons la meilleure insertion. Cependant, cette méthode de décodage possède l'inconvénient d'écarter des meilleures solutions si nous ne forçons pas parfois l'utilisation d'un point de transbordement comme décrit dans l'exemple de la figure 2.3. Pour éviter cela, nous ajoutons un deuxième vecteur à l'individu dont le rôle est d'indiquer le type d'insertion à utiliser lors du décodage. Ce vecteur est un vecteur binaire de taille $|R|$. La place d'un gène dans le vecteur correspond toujours à une requête donnée, et la valeur du gène indique le type d'insertion à réaliser. Si le gène vaut 0, alors l'insertion à réaliser pour cette requête est l'insertion la moins coûteuse entre l'insertion sans transbordement et l'insertion avec transbordement. Si le gène vaut 1, alors c'est l'insertion avec transbordement qui est réalisée.

La figure 2.6 représente un individu complet. Le nombre de requêtes est m. Chaque indice correspond à une requête donnée. Le premier vecteur de l'individu est celui qui contient les poids qui détermineront l'ordre des requêtes. Chaque poids w_i est un réel compris entre 0 et 1. Plus w_i est faible et plus l'ordre d'insertion de la requête r_i sera faible. Le deuxième vecteur est celui qui déter-

Requête	r_1	r_2	r_3	r_4	...	r_{m-3}	r_{m-2}	r_{m-1}	r_m
Poids	p_1	p_2	p_3	p_4	...	p_{m-3}	p_{m-2}	p_{m-1}	p_m
Insertion	u_1	u_2	u_3	u_4	...	u_{m-3}	u_{m-2}	u_{m-1}	u_m

FIGURE 2.6 – Représentation d'un individu complet

Algorithm 2.18 evaluation

Input: *individual*

put requests sorted by ascending w of *individual* in LR
$solution = \emptyset$
for all request r in LR **do**
  if $u_r = 0$ **then**
    $currentSolution1 \leftarrow solution$
    $currentSolution2 \leftarrow solution$
    $bestSingleInsertion(r, currentSolution1)$
    $bestDoubleInsertion(r, currentSolution2)$
    if $f(currentSolution2) \leq f(currentSolution1)$ **then**
      $solution \leftarrow currentSolution2$
    else
      $solution \leftarrow currentSolution1$
    end if
  else
    $bestDoubleInsertion(r, solution)$
  end if
end for
return $f(solution)$

mine le type d'insertion de la requête. Chaque gène u_j est un entier qui vaut 0 ou 1. Si u_i vaut 0, l'insertion réalisée pour la requête r_j est la moins coûteuse entre l'insertion sans et avec transbordement. Si u_j vaut 1, l'insertion réalisée est celle avec transbordement. L'algorithme 2.18 présente le calcul de fitness d'un individu. Premièrement, les requêtes sont triées selon les poids de l'individu. Ensuite, pour chaque requête, en partant de celle avec le poids le plus faible jusqu'à celle ayant le poids le plus élevé, nous effectuons l'insertion en fonction de la valeur de u. A la fin, nous obtenons une solution correspondante à l'individu. Le fitness de cet individu est donc le coût de la solution correspondante. Nous pouvons noter que les algorithmes d'insertion étant déterministes, deux

individus possédant le même ordre de requêtes ainsi que les mêmes valeurs de u auront toujours la même solution correspondante et par conséquent le même fitness.

2.5.3 Les opérateurs de croisement

Dans les GA, les opérateurs de croisement ont pour but de créer de nouveaux individus en croisant des individus existants de la population. Ces opérateurs ne sont généralement pas dépendants du problème à résoudre. Nous utilisons ici deux opérateurs classiques des GA, l'opérateur de croisement à un point et l'opérateur de croisement à deux points.

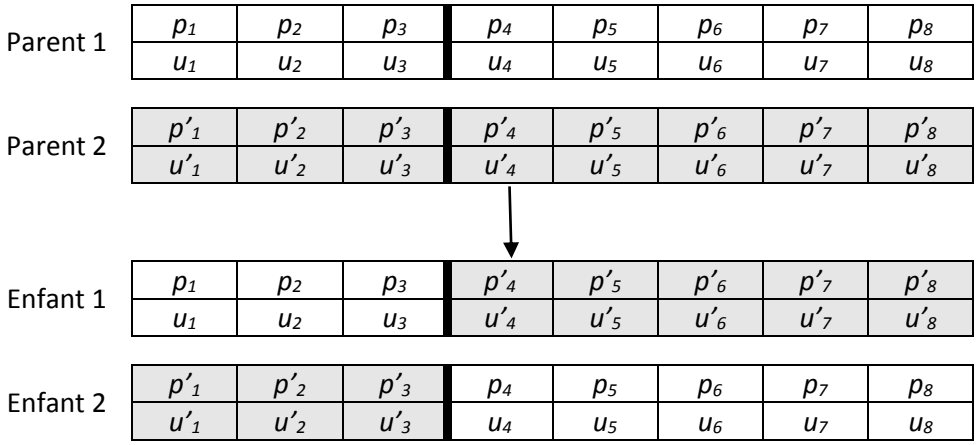

FIGURE 2.7 – Opérateur de croisement à un point

La figure 2.7 représente l'opérateur de croisement à un point. Dans cet exemple le point de croisement se situe au point 3. L'enfant 1 est composé des trois premiers gènes du parent 1 et des cinq gènes suivants du parent 2. L'enfant 2 est lui composé des trois premiers gènes du parent 2 et des cinq gènes suivants du parent 1.

La figure 2.8 représente l'opérateur de croisement à deux points. Dans cet exemple, les points de croisement se situent aux points 3 et 5. L'enfant 1 est composé des deux premiers gènes du parent 1, des trois gènes suivants du parent 2 et des trois derniers gènes du parent 1. L'enfant 2 est composé des deux

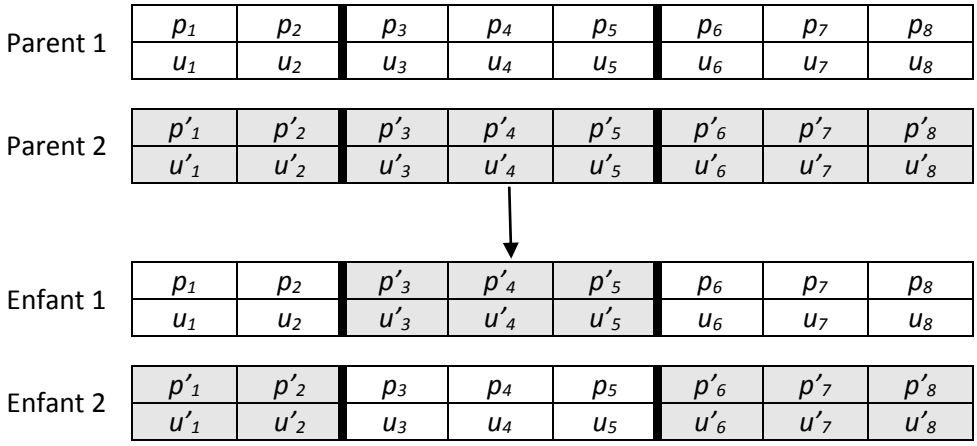

FIGURE 2.8 – Opérateur de croisement à deux points

premiers gènes du parent 2, des trois gènes suivants du parent 1 et des trois derniers gènes du parent 2.

Dans notre GA, la fonction de croisement à 1 et 2 points fonctionne de la même manière et est décrite par l'algorithme 2.19. Tout d'abord, la population est copiée dans un tampon. Ensuite deux individus sont sélectionnés au hasard. La probabilité de tirer les meilleurs individus et plus élevée que celle de tirer les individus les plus faibles. Comme pour les opérateurs de destruction 4 et de réparation 2 du LNS, cette probabilité est déterminée par le paramètre α. Plus *alpha* est grand, plus la probabilité de tirer les meilleurs individus est grande. Un individu ne peut pas être sélectionné plus d'une fois par génération par opérateur. Ensuite, selon la fonction, un ou deux points de croisement sont sélectionnés de manière aléatoire. L'opérateur de croisement à 1 point ou à 2 points est appliqué aux deux individus sélectionnés pour créer deux nouveaux enfants. Ces enfants sont évalués et ajoutés à la population. Ces étapes sont répétées jusqu'à ce que le nombre de croisements voulu soit atteint.

2.5.4 Les opérateurs de mutation

Contrairement aux opérateurs de croisements, les opérateurs de mutation sont appliqués sur un seul individu à la fois. Ils ont pour but de diversifier la population et d'éviter de converger trop rapidement vers un optimum local. Ces

Algorithm 2.19 crossover1point (resp. 2points)

Input: $population, nbCrossover$
Output: $population$
1: $TP = population$
2: **for** i from 1 to $nbCrossover$ **do**
3: generate randomly a number $p \in [0,1[$
4: $parent1 = TP[p^{\alpha}.|TP|]$
5: remove $parent1$ from TP
6: generate randomly a number $p \in [0,1[$
7: $parent2 = TP[p^{\alpha}.|TP|]$
8: remove $parent2$ from TP
9: randomly choose 1 (resp. 2) crossover point(s)
10: apply 1-point (resp. 2-points) crossover operator on $parent1$ and $parent2$ to create $child1$ and $child2$
11: $evaluation(child1)$
12: $evaluation(child2)$
13: add $child1$ and $child2$ in $population$
14: **end for**

opérateurs sont en général indépendants du problème à résoudre, mais certains opérateurs de mutation peuvent être spécifiques à un problème donné. Nous utilisons cinq opérateurs de mutation, dont trois opérateurs classiques des GA et deux opérateurs qui sont plus spécifiques à notre problème.

Le premier opérateur de mutation (figure 2.9) est un opérateur qui prend un gène p au hasard dans l'individu et le remplace par une autre valeur déterminée au hasard également. Par exemple dans la figure 2.9, le gène p_6 devient p'_6. Le deuxième opérateur de mutation (figure 2.10) prend au hasard deux gènes et échange leur place. Dans la figure 2.10, les gènes p_1 et p_5 sont inversés. Le troisième opérateur (figure 2.11) modifie quant à lui non pas le poids mais la méthode d'insertion d'une requête. Un gène u est choisi au hasard dans l'individu. Si ce gène vaut 0 alors il est transformé en 1 et s'il vaut 1 alors il est transformé en 0. Dans la figure 2.11, le gène u_7 est modifié. Ces trois opérateurs sont des opérateurs classiques des GA.

Les deux opérateurs suivants sont plus spécifiques à notre problème et inspirés de la méthode LNS. Le quatrième opérateur (figure 2.12) est inspiré par

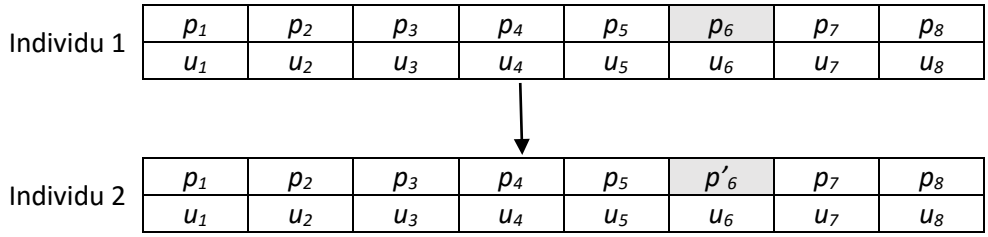

FIGURE 2.9 – Premier opérateur de mutation

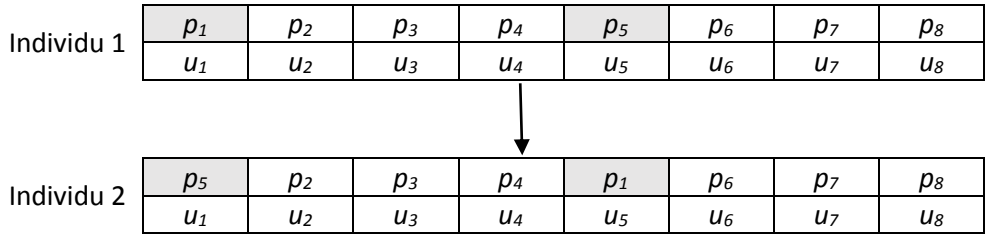

FIGURE 2.10 – Deuxième opérateur de mutation

le principe du LNS de modifier une grande partie de la solution. L'idée ici est de modifier plusieurs gènes afin de fortement changer l'individu. Le principe est similaire à l'opérateur de mutation 1, plusieurs gènes p sont sélectionnés au hasard dans l'individu, et sont remplacés par des valeurs déterminées au hasard. Dans la figure 2.12, les gènes p_2, p_3 et p_8 sont ainsi modifiés.

Le cinquième opérateur a été implémenté dans le but de réduire le nombre de véhicules utilisés. En effet, dans le but de minimiser le nombre de véhicules, les opérateurs d'insertion favorisent l'ajout des requêtes dans des véhicules déjà utilisés. Si l'insertion dans un véhicule utilisé n'est pas possible (à cause des fenêtres de temps ou de la capacité du véhicule), alors la requête est insérée dans le premier véhicule libre. Le dernier véhicule utilisé contient donc les requêtes qui n'ont pas pu être insérées dans les premiers véhicules. L'idée est donc de réduire le poids de ces requêtes afin qu'elles puissent être insérées plus tôt et donc dans des véhicules déjà utilisés, ceci afin de réduire le nombre de véhicules utilisés. Cet opérateur est décrit par l'algorithme 2.20. Tout d'abord, nous récupérons la liste des requêtes présentes dans le dernier véhicule utilisé. Ensuite, pour chacune de ces requêtes, nous générons aléatoirement un nombre

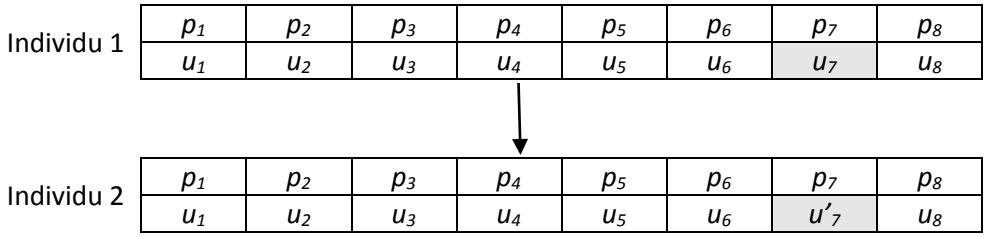

FIGURE 2.11 – Troisième opérateur de mutation

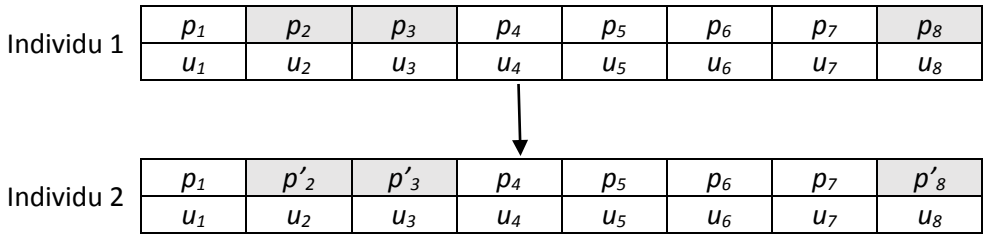

FIGURE 2.12 – Quatrième opérateur de mutation

réel compris entre ω et 1 (ω est un paramètre compris entre 0 et 1). Le poids correspondant à la requête est ensuite multiplié par ce nombre, ce qui va réduire le poids d'origine.

Algorithm 2.20 mutation5

Input: $individual$

1: put in LR all the requests from the last vehicle used
2: $newIndividual = individual$
3: **for all** r in LR **do**
4: generate randomly a number $p \in [\omega, 1[$
5: $newIndividual[p_r] = individual[p_r].p$
6: **end for**
7: $evaluation(newIndividual)$
8: **return** $newIndividual$

La fonction générale de mutation est décrite par l'algorithme 2.21. A chaque génération un individu est sélectionné au hasard dans la population. Cet individu peut venir de la génération précédente ou être issu des croisements. L'opérateur de mutation est choisi au hasard parmi les cinq. Cet opérateur est appliqué à

l'individu pour en créer un nouveau qui sera ajouté à la population.

Algorithm 2.21 mutation

Input: *population*
Output: *population*
1: choose randomly *individual* in *population*
2: *newIndividual* = *individual*
3: choose randomly a mutation operator
4: apply operator to *newIndividual*
5: *evaluation*(*newIndividual*)
6: add *newIndividual*) in *population*

2.5.5 Sélection des individus

Après avoir appliqué les croisements et les mutations et avoir obtenu une population élargie, il faut sélectionner les individus qui participeront à la génération suivante dans le but d'avoir une population qui ne change pas de taille au fur et à mesure des itérations. Les individus sélectionnés sont les meilleurs de chaque génération. Cependant, afin d'éviter que la population contienne plusieurs fois le même individu, si plusieurs individus ont le même fitness, le premier d'entre eux dans la liste est sélectionné. Ceci permet de garder une population variée et hétérogène.

2.5.6 Amélioration des performances

Dans notre approche GA, l'évaluation d'un individu est un processus assez coûteux. En effet, pour chaque requête, il faut tester toutes les insertions possibles sans transbordement et toutes les insertions possibles avec transbordement. Cependant, lorsque deux individus possèdent le même ordre de requêtes (mais pas forcément le même génotype) ainsi que le même vecteur d'insertion, leur solution est identique, puisque les opérateurs d'insertion utilisés sont déterministes. Il n'est donc pas nécessaire de calculer la solution pour le deuxième individu mais de reprendre celle du premier. De même, lorsque l'ordre des premières requêtes à insérer est le même qu'un autre individu, il n'est pas non

plus nécessaire de calculer la solution partielle, il suffit de reprendre la solution partielle existante. Par exemple dans la figure 2.13, l'ordre des requêtes à insérer de l'individu 1 est 6, 4, 8, 1, 3, 5, 7 et 2 et celui de l'individu 2 est 6, 4, 8, 3, 5, 7, 2 et 1. Nous pouvons remarquer que jusqu'à la troisième requête incluse, l'ordre est le même. Pour évaluer l'individu 2, nous pouvons donc reprendre la solution partielle contenant les trois requêtes 6, 4 et 8 de l'individu 1, et démarrer l'évaluation de l'individu 2 à partir de la requête 3. Ceci nous permet ainsi d'éviter de tester les insertions pour trois requêtes.

Individu 1	0.36	0.78	0.46	0.23	0.6	0.05	0.71	0.29
	0	0	0	0	0	0	0	0
Individu 2	0.89	0.75	0.4	0.18	0.59	0.12	0.63	0.34
	0	0	0	0	0	0	0	0

FIGURE 2.13 – Exemple de deux individus dont l'ordre des trois premières requêtes est le même

Pour implémenter ceci, nous avons recours à une structure arborescente pour sauvegarder les solutions partielles. La hauteur de l'arbre est de $|R|+1$. La racine possède $2.|R|$ fils, la moitié pour la méthode d'insertion sans et avec transbordement, et l'autre moitié pour la méthode d'insertion avec transbordement. Chaque nœud de profondeur 1 possède $2.(|R|-1)$ fils et plus la profondeur est élevée, moins les nœuds ont de fils, cette diminution correspondant au fait que le nombre de requêtes à insérer diminue avec le niveau de la profondeur. Les nœuds de l'avant-dernière profondeur ont chacun deux fils. Dans chaque nœud est stockée une solution partielle, plus la profondeur est grande, plus cette solution contient de requêtes. Dans chaque feuille est stockée une solution complète. Le nombre maximal de feuilles de cet arbre est de $(2.|R|)!!$, soit $2^{|R|}.|R|!$. Cet arbre est construit au fur et à mesure des itérations du GA.

La figure 2.14 représente un arbre incomplet de parcours de ces solutions partielles pour un problème à quatre requêtes. Dans cet exemple, les nœuds blancs correspondent à l'insertion des requêtes sans ou avec transbordement, et les nœuds gris correspondent à l'insertion des requêtes avec transbordement uniquement. Le premier nœud blanc noté 1 contient la solution partielle de

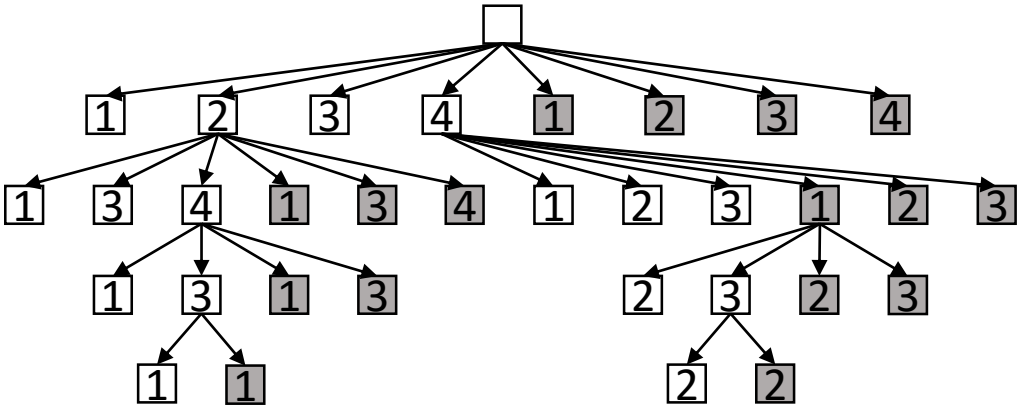

FIGURE 2.14 – Exemple d'un arbre de parcours de solutions partielles pour un problème à quatre requêtes

l'insertion de la requête 1 en premier (sans ou avec transbordement), le deuxième nœud blanc noté 2 contient la solution partielle de l'insertion de la requête 2 en premier, etc... Le premier nœud gris noté 1 contient la solution partielle de l'insertion de la requête 1 en premier avec transbordement, etc... La première feuille blanche contient donc la solution complète correspondant à l'insertion des requêtes dans cet ordre : requête 2, 4, 3 et 1 (sans ou avec transbordement pour toutes les requêtes). La deuxième feuille blanche contient la solution complète correspondant à l'insertion des requêtes dans cet ordre : requête 4, 1 (avec transbordement), 3 et 2.

Avec cet arbre, nous pouvons mettre à jour la fonction d'évaluation d'un individu. L'algorithme 2.22 présente cette nouvelle fonction d'évaluation. Premièrement, les requêtes sont toujours triées selon les poids de l'individu. Ensuite, en partant de la racine de l'arbre, nous vérifions si l'insertion de la requête a déjà été testée. L'arbre est ainsi parcouru en profondeur. Lorsque nous tombons sur une requête non testée, nous récupérons la solution partielle et testons les insertions possibles des requêtes restantes. A chaque nouvelle insertion, l'arbre est complété en conséquence.

Algorithm 2.22 evaluation

Input: $individual$

put requests sorted by ascending w of $individual$ in LR
$solution = \emptyset$
$tree$ = complete search tree from root
$isintree = true$
$i = 1$
while $isInTree$ **do**
  $r = LR[i]$
  if r is in $tree$ **then**
    $solution$ = solution store in node r
    $tree$ = subtree starting from node r
    $i = i + 1$
  else
    $isintree = false$
  end if
end while
while $i \leq |LR|$ **do**
  $r = LR[i]$
  if $u_r = 0$ **then**
    $currentSolution1 \leftarrow solution$
    $currentSolution2 \leftarrow solution$
    $bestSingleInsertion(r, currentSolution1)$
    $bestDoubleInsertion(r, currentSolution2)$
    if $f(currentSolution2) \leq f(currentSolution1)$ **then**
      $solution \leftarrow currentSolution2$
    else
      $solution \leftarrow currentSolution1$
    end if
  else
    $bestDoubleInsertion(r, solution)$
  end if
  put r and $solution$ in $tree$
  $tree$ = subtree starting from node r
  $i = i + 1$
end while
return $f(solution)$

2.6 Expérimentations et résultats

Les deux algorithmes présentés précédemment ont été testés sur deux types d'instances différentes. Premièrement, ils ont été testés sur les instances de PDP de Li et Lim (2003). En effet, le PDPT étant une généralisation du PDP, un algorithme permettant de résoudre le PDPT doit également résoudre le PDP. Deuxièmement, ils ont été testés sur les instances de PDPT de Qu et Bard (2012). Les algorithmes ont été codés en C++ sur un PC disposant d'un processeur core i7 2,7GHz, de 8Go de RAM et tournant sous Windows 10. Les paramètres pour le LNS sont les suivants :

- $nbIterMax = 5000$
- $acceptCriterion = 1.01$
- $\beta_1 = 0.5$ (Poids de l'équation 2.2 utilisée dans l'algorithme 2.13 $repair2$)
- $\beta_2 = 0.2$ (Poids de l'équation 2.2 utilisée dans l'algorithme 2.13 $repair2$)
- $\beta_3 = 0.3$ (Poids de l'équation 2.2 utilisée dans l'algorithme 2.13 $repair2$)

Les paramètres pour le GA sont les suivants :

- $nbIterMax = 500$
- $popSize = 20$
- $nbCrossover = 5$
- $\omega = 0.5$ (paramètre utilisé dans l'algorithme 2.20 $mutation5$)

Les paramètres communs au LNS et au GA sont les suivants :

- $P = 10000$ (coût fixe d'utilisation d'un véhicule)
- $\alpha = 2$ (paramètre qui favorise ou non la sélection des premiers éléments dans une liste ordonnée, utilisé dans les algorithmes 2.10, 2.13 et 2.19)

2.6.1 Expérimentations sur les instances du PDP

Les méthodes LNS et GA ont tout d'abord été testées sur les instances du PDP provenant de Li et Lim (2003). Le PDPT étant une généralisation du PDP, il est intéressant de tester rapidement l'efficacité de ces deux méthodes sur le PDP avant de réaliser une étude plus poussée sur des instances du PDPT. Les

instances du PDP ont été créées à partir des instances de Solomon (1987) pour le VRP. Ces instances sont regroupées en six catégories. Dans les catégories lr1x et lr2x, les nœuds sont répartis uniformément. Dans les catégories lc1x et lc2x, la répartition des nœuds est clusterisée. Dans les catégorie lrc1x et lrc2x, la répartition des nœuds est semi-clusterisée. Pour toutes les catégories, le "1" désigne les instances avec des fenêtres de temps étroites et le "2" désigne les instances avec des fenêtres de temps larges. Les valeurs qui sont inférieures ou égales à celles de la "BKS" (Best Known Solution) sont surlignées en gris dans les tableaux résultats.

	BKS		LNS			GA		
Instances	NbV	Coût	NbV	Coût	Tps (s)	NbV	Coût	Tps (s)
lc101	10	829	10	829	129	10	829	511
lc102	10	829	10	829	120	10	829	402
lc103	9	1035	10	828	121	9	1150	295
lc104	9	860	9	860	191	9	860	437
lc105	10	829	10	829	136	10	829	360
lc106	10	829	10	829	139	10	829	318
lc107	10	829	10	829	122	10	829	279
lc108	10	826	9	1880	126	9	1896	292
lc109	9	1001	10	958	129	10	1372	346
lc201	3	592	3	592	951	3	592	2154
lc202	3	592	3	592	1013	3	592	1434
lc203	3	591	3	591	930	3	591	1810
lc204	3	591	3	585	1167	3	591	1396
lc205	3	589	3	589	949	3	589	1845
lc206	3	588	3	589	1116	3	588	2021
lc207	3	588	3	588	1032	3	588	1275
lc208	3	588	3	588	1067	3	588	2061
lr101	19	1651	19	1651	65	19	1651	318
lr102	17	1488	17	1490	105	17	1488	356
lr103	13	1293	13	1293	108	13	1315	256
lr104	9	1013	10	1049	150	10	1074	434
lr105	14	1377	14	1377	106	14	1384	361

	BKS		LNS			GA		
Instances	NbV	Coût	NbV	Coût	Tps (s)	NbV	Coût	Tps (s)
lr106	12	1253	12	1253	104	12	1253	268
lr107	10	1111	10	1111	420	10	1111	771
lr108	9	969	9	969	134	10	1042	284
lr109	11	1209	12	1239	97	11	1209	220
lr110	10	1159	11	1166	98	10	1159	680
lr111	10	1109	10	1109	145	11	1183	563
lr112	9	1004	9	1004	134	9	1004	515
lr201	4	1253	4	1255	537	4	1294	658
lr202	3	1198	3	1198	801	3	1198	682
lr203	3	949	3	975	1027	3	949	978
lr204	2	849	2	849	1821	2	849	1524
lr205	3	1054	3	1054	949	3	1054	840
lr206	3	932	3	932	1028	3	932	1115
lr207	2	903	2	1076	3401	2	903	4739
lr208	2	735	2	735	2861	2	735	1648
lr209	3	931	3	938	1185	3	931	1164
lr210	3	964	3	964	1184	3	1063	2173
lr211	2	912	2	912	1321	2	912	3989
lrc101	14	1709	14	1709	94	14	1709	351
lrc102	12	1558	12	1558	110	12	1558	308
lrc103	11	1259	11	1259	107	11	1259	324
lrc104	10	1128	10	1128	133	10	1128	136
lrc105	13	1638	13	1638	93	13	1638	296
lrc106	11	1425	11	1425	101	11	1425	172
lrc107	11	1230	11	1230	113	11	1230	234
lrc108	10	1147	10	1147	108	10	1147	259
lrc201	4	1407	4	1432	473	4	1407	547
lrc202	3	1374	3	1374	564	3	1374	1181
lrc203	3	1089	3	1089	1085	3	1089	782
lrc204	3	819	3	819	1125	3	819	852
lrc205	4	1302	4	1302	476	4	1302	499

	BKS		LNS			GA		
Instances	NbV	Coût	NbV	Coût	Tps (s)	NbV	Coût	Tps (s)
lrc206	3	1159	3	1159	1022	3	1159	976
lrc207	3	1062	3	1062	1052	3	1062	601
lrc208	3	853	3	853	1134	3	853	1034
Total	402	58061	407	58114	35009	406	58924	50324

Tableau 2.1 – Comparaison des résultats de la littérature avec les résultats du LNS et du GA sur les instances du PDP

Les deux méthodes ont été testées sur 56 instances comprenant 100 nœuds. Le tableau 2.1 résume les résultats obtenus. Pour ces instances, les métaheuristiques ont été exécutées une fois par instance, le but étant de vérifier rapidement leur efficacité. La colonne "BKS" donne les résultats des meilleures solutions connues jusqu'à maintenant dans la littérature. Pour chaque méthode, la colonne "NbV" représente le nombre de véhicules utilisés, la colonne "Coût" représente le coût de la solution (distance totale parcourue) et la colonne "Tps" représente le temps d'exécution en secondes. Ce temps n'est pas connu pour les "BKS". Parmi les 56 instances le LNS réussit à faire aussi bien sur 46 instances et le GA réussit à faire aussi bien sur 47 instances. L'écart moyen entre le LNS et la BKS est de 1,2% pour le nombre de véhicules utilisés et il est de 1% entre le GA et la BKS. L'écart moyen entre le LNS et la BKS est de 0,1% pour le coût et il est de 1,5% entre le GA et la BKS Ces deux méthodes sont donc relativement efficaces sur les instances du PDP. En effet, elles permettent de trouver la meilleure solution connue dans la plupart des cas et les écarts moyens entre les métaheuristiques et la BKS sont très faibles.

Nous pouvons également noter un écart entre les temps de calcul des instances de type "1" et des instances de type "2". Celui-ci est beaucoup plus élevé pour les instances de type "2". Ceci est dû à la taille des fenêtres de temps. En effet, la fonction de vérification de la faisabilité d'une solution est très consommatrice de temps de calcul. Si une solution est faisable, cette fonction est exécutée en entier, tandis que si une solution n'est pas faisable, celle-ci peut très vite être éliminée et la fonction de faisabilité s'arrête avant la fin de son exécution.

Dans les instances de type "1", les fenêtres de temps sont courtes, le nombre de solutions infaisables évaluées est donc important et la fonction de faisabilité est rarement exécutée jusqu'à son terme. Par contre, dans les instances de type "2", les fenêtres de temps sont grandes et donc les solutions infaisables sont plus rares. La fonction de faisabilité est donc plus souvent exécutée jusqu'à son terme, ce qui rallonge la durée totale d'exécution.

2.6.2 Expérimentations sur les instances du PDPT

Les deux méthodes précédentes ont été testées sur les instances du PDPT provenant de Qu et Bard (2012). Cinq jeux d'instances nommés "pdpt1" à "pdpt5" ont été créés. Chaque jeu contient 10 instances. Toutes les instances créées comportent 25 requêtes, 25 points de collecte, 25 points de livraison, un nœud de transbordement et un dépôt qui fait office de dépôt de départ et d'arrivée à la fois. Ces instances ont été créées de manière à ce que la solution optimale ainsi que sa valeur soient connues. Pour créer un jeu, une instance comprenant 6 requêtes est créée aléatoirement. Cette instance est résolue de manière optimale. Ensuite des requêtes sont ajoutées une à une tant qu'il n'y en a pas 25 et de manière à ce que la valeur de la solution optimale reste inchangée. Pour ce faire, les points de collecte et livraison des requêtes ajoutées sont placés directement sur les routes de la solution optimale. En faisant attention à ce que les contraintes de capacité et de fenêtres de temps soient respectées, ceci permet d'ajouter des nœuds sans que les routes de la solution optimale soient modifiées et donc la valeur optimale ne change pas que ce soit pour 6 ou pour 25 requêtes. De plus, les 10 instances d'un jeu possèdent la même valeur de solution optimale.

Le LNS et le GA ont été exécutés trente fois sur chaque instance. Le tableau 2.2 compare les résultats obtenus avec le LNS et le GA avec les résultats obtenus dans Qu et Bard (2012). La colonne "Instances" indique à quel jeu d'instances correspondent les résultats. La colonne "Méthode" indique à quelle méthode correspondent les résultats. "QB" sont les résultats obtenus par Qu et Bard (2012). la colonne "NbV" représente le nombre de véhicules utilisés. La colonne "MC" les meilleurs coûts trouvés sur les trente exécutions. Les résultats sont présentés

par jeu d'instances, ce sont donc les moyennes des résultats des 10 instances. La colonne "Tps" représente le temps moyen d'exécution en secondes. La colonne "gap" représente la différence entre le résultat trouvé par la méthode et la solution optimale. Elle est calculée de la manière suivante : $Gap = 100(MC - opt)/opt$. Qu et Bard (2012) ont limité le nombre de véhicules disponibles au nombre de véhicules utilisés dans la solution optimale. De ce fait, leur méthode ne permet pas toujours de trouver une solution faisable avec ce nombre de véhicule limité. La colonne "NbSNT" représente donc le nombre d'instances pour lesquelles ils n'ont pas pu trouver de solution faisable. Pour toujours avoir une solution faisable durant nos tests, nous avons systématiquement multiplié le nombre de véhicules disponibles par deux. Ainsi, pour les méthodes LNS et GA, la colonne "NbSNT" représente le nombre d'instances pour lesquelles le nombre de véhicules utilisés est supérieur au nombre de véhicules utilisés dans la solution optimale. De plus, pour le calcul de "MC", seules les solutions utilisant le même nombre de véhicules que la solution optimale sont prises en compte. La colonne "opt" contient les valeurs des solutions optimales des cinq jeux d'instances. Les meilleures valeurs pour chaque jeu pour les colonnes "MC", "Tps", "Gap" et "NbSNT" sont surlignées en gris.

Pour le jeu d'instances "pdpt1", le LNS et le GA ont trouvé la solution optimale pour les dix instances alors que QB ne trouve pas de solution pour trois instances. De plus, sur les sept solutions trouvées, la solution optimale n'est pas trouvée à chaque fois. Le gain des méthodes LNS et GA par rapport à QB est de 0.27%. Pour le jeu d'instances "pdpt2", Les trois méthodes trouvent une solution pour les dix instances mais les meilleures solutions sont trouvées par le LNS et le GA. La différence entre ces deux méthodes et l'optimum n'est que de 0.01%. Pour le jeu "pdpt3", le nombre de solutions trouvées pour le LNS n'est que d'un et n'est que de deux pour le GA, alors que le QB trouve neuf solutions. De plus les solutions trouvées par le LNS et le GA sont moins bonnes que celles trouvées par le QB. Ces mauvais résultats s'expliquent par les hypothèses prises en compte pour résoudre le PDPT. En effet, dans nos hypothèses, un véhicule ne peut passer qu'une seule fois dans un nœud de transbordement alors que la méthode QB autorise un véhicule à passer plusieurs fois par un nœud de transbordement. Les solutions optimales de ce jeu requérant deux fois le passage d'un véhicule

Instances	Méthode	NbV	MC	Tps (s)	Gap (%)	NbSNT	opt
pdpt1	LNS	2	2510.9	245	0	0	2510.9
	GA	2	2510.9	428	0	0	
	QB	2	2571.7	408	2.42	3	
pdpt2	LNS	2	2360.6	377	0.07	0	2359
	GA	2	2360.6	493	0.07	0	
	QB	2	2374.7	295	0.67	0	
pdpt3	LNS	2	2678.9	225	12.33	9	2384.8
	GA	2	2642.6	427	10.81	8	
	QB	2	2404.9	477	0.84	1	
pdpt4	LNS	2	2405.8	410	0	0	2405.8
	GA	2	2405.8	529	0	0	
	QB	2	2405.8	55	0	0	
pdpt5	LNS	3	3479	120	0	2	3479
	GA	3	3479	296	0	0	
	QB	3	3479	238	0	1	

Tableau 2.2 – Comparaison des résultats de la littérature avec les résultats du LNS et du GA sur les instances du PDPT

par le nœud de transbordement, les méthodes LNS et GA ne sont pas en mesure de trouver des bonnes solutions pour ce jeu. Cette hypothèse a été choisie pour respecter le modèle mathématique présenté dans le chapitre 2. Elle présente l'avantage de faciliter la vérification de la contrainte de précédence dans les nœuds de transbordement. Pour le jeu "pdpt4", les trois méthodes trouvent à chaque fois la solution optimale. Pour le jeu "pdpt5", la solution optimale est toujours trouvée par le GA. Elle est trouvée neuf fois par le QB et huit fois par le LNS.

En termes de nombre de solutions trouvées, le GA fait mieux que le QB sur trois jeux, fait pareil sur un jeu et fait moins bien sur un jeu. Le LNS fait mieux que le QB sur deux jeux, pareil sur un jeu et moins bien sur deux jeux. En termes de qualité de solution, le GA fait mieux que le QB sur deux jeux. Les méthodes LNS et GA sont donc efficaces sur ces jeux d'instances puisqu'elles permettent d'obtenir des solutions de bonne qualité et font au moins aussi bien que le QB dans quatre jeux sur cinq avec des temps de calcul à peu près similaires.

Le tableau 2.3 compare l'efficacité du LNS et du GA sur les instances du

	Taux de succès (%)		Moyenne	
Instances	LNS	GA	LNS	GA
pdpt1	59	49	26773	27037
pdpt2	53	78	22444	22422
pdpt3	0	0	31770	31574
pdpt4	95	91	22406	22934
pdpt5	50	87	38521	34645
Toutes	51	61	28382	27722

Tableau 2.3 – Comparaison de l'efficacité du LNS et du GA sur les instances du PDPT

PDPT. La colonne "Instances" indique à quel jeu d'instances correspondent les résultats. La colonne "Taux de succès" représente le pourcentage du nombre de fois où la solution optimale a été trouvée sur les trente exécutions par groupe d'instances. La colonne "Moyenne" représente la moyenne des solutions trouvées sur les trente exécutions par groupe d'instances. Le LNS a le meilleur taux de succès sur deux groupes d'instances et le GA également. Au total, c'est le GA qui a le meilleur taux de succès. La moyenne des solutions trouvées par le GA est meilleure que celle obtenue par le LNS sur trois jeux d'instances. La moyenne générale des solutions trouvées par le GA est également meilleure.

Pour ne pas répéter toujours les mêmes tests d'insertion lors du décodage d'une solution, un cache en forme d'arbre a été implémenté. Pour tester l'efficacité de ce cache, nous avons exécuté chaque instance une fois sans cache et une fois avec cache. Le nombre d'itérations a été fixé à 2000. La figure 2.15 montre le temps de calcul moyen du GA sans et avec cache en fonction du nombre d'itérations. Nous pouvons remarquer que les performances avec le cache sont largement supérieures à celles sans cache.

2.7 Conclusion

Dans ce chapitre, deux méthodes approchées ont été proposées pour résoudre le PDPT. Les hypothèses prises en compte permettent de résoudre le problème dans un cas très général. Tout d'abord, un LNS a été proposé. Le LNS est déjà utilisé dans Qu et Bard (2012) et dans Masson et al. (2013) pour résoudre le

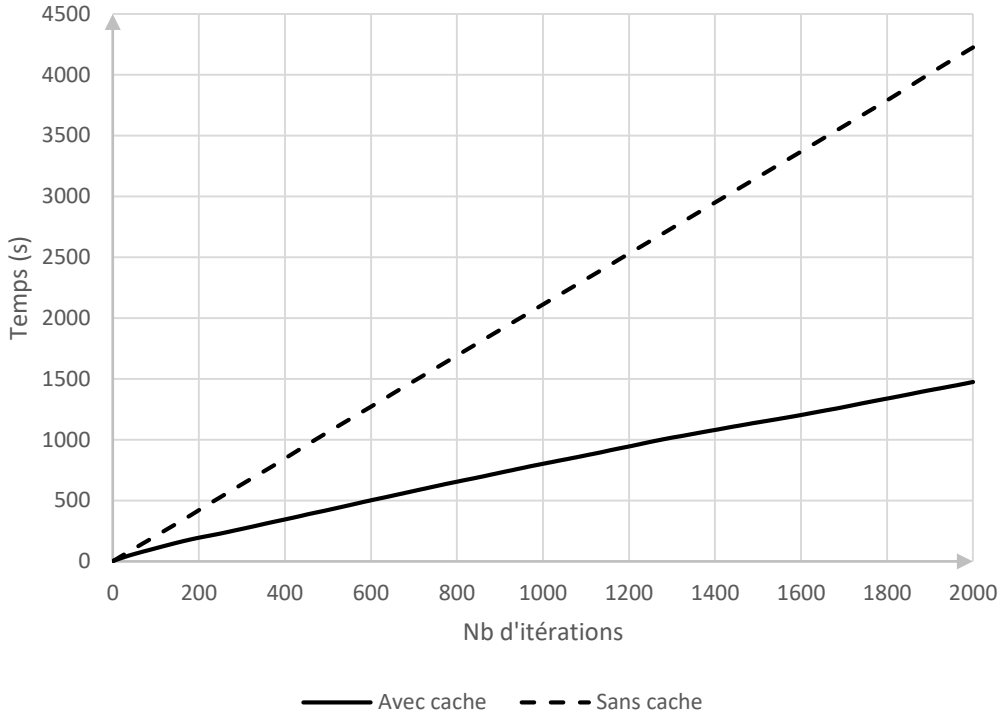

Figure 2.15 – Comparaison des performances du GA sans et avec cache

PDPT et s'est montré efficace. Dans nos travaux, le LNS se distingue des autres approches précédentes par plusieurs points. Tout d'abord, l'algorithme principal du LNS proposé diffère par le critère d'acceptation de la solution courante. Ensuite, la fonction d'insertion des requêtes avec transbordement est améliorée. Dans les travaux précédents, soit elle ne permet pas de trouver forcément la meilleure insertion, soit toutes les insertions possibles sont testées pour trouver la meilleure. La fonction d'insertion utilisée dans ce chapitre permet de trouver la meilleure insertion possible sans tester toutes les possibilités. Ensuite, un GA a été proposé. Ce type de méthode n'a pas encore été utilisé pour résoudre le PDPT. Dans ce GA, le codage retenu pour les individus permet de toujours obtenir des solutions réalisables, même après un croisement ou une mutation. Pour réduire le temps de décodage, une mémoire cache en forme d'arbre est utilisée.

Ces deux méthodes ont été testées sur deux types d'instances de la littérature,

des instances de PDP et des instances de PDPT. Les résultats montrent que le LNS et le GA sont des méthodes efficaces pour résoudre à la fois le PDP et surtout le PDPT. Pour ce dernier, en comparaison avec les méthodes existantes dedans la littérature, elles donnent de meilleures performances sur deux jeux d'instances sur cinq pour le LNS et sur trois jeux d'instances pour le GA. Le LNS trouve la solution optimale dans 51% des cas et le GA dans 61% des cas.

Chapitre 3

Un PDVRP pour la planification du transport collaboratif

3.1 Introduction

Le PDVRP a été peu abordé dans la littérature. A notre connaissance, seuls trois travaux ont été réalisés sur ce problème (Dror et al., 1998 ; Shi et al., 2009 ; Chen et al., 2014). Dans ce chapitre, nous proposons deux formulations pour ce problème : une formulation ne prenant pas en compte les fenêtres de temps et une formulation prenant en compte des fenêtres de temps souples. Ces deux formulations sont utilisées dans le cadre d'une collaboration entre fournisseurs ou entre clients pour la planification du transport. En effet, ce type de problème n'a pas encore été considéré pour traiter la planification du transport collaboratif.

Nos formulations diffèrent de celles déjà proposées dans la littérature de par les hypothèses considérées. Les hypothèses pour notre PDVRP sont les suivantes :

— Les dépôts ne sont pas considérés. Un véhicule commence sa tournée depuis n'importe quel point de collecte et la termine sur n'importe quel point de livraison.

— Le nombre de véhicules disponibles est limité. Tous les véhicules ne sont pas obligatoirement utilisés.

— La flotte de véhicules est hétérogène. Les véhicules peuvent avoir des capacités, des coûts ou des vitesses différentes.

— Les nœuds peuvent être visités par plusieurs véhicules.

— La demande d'un client peut être traitée par plusieurs véhicules (split delivery).

— Lors d'une tournée, un véhicule doit d'abord visiter les points de collecte, puis les points de livraison.

Ces hypothèses ont été choisies car elles permettent de simuler un réseau de transport collaboratif.

La suite de ce chapitre est décomposée en quatre sections. Dans la section 2, le PDVRP sans fenêtre de temps est décrit puis une formulation mathématique est proposée. Dans la section 3, le PDVRP est appliqué sur des données générées aléatoirement. Dans la section 4, le PDVRP est appliqué sur des données réelles issues d'une étude de cas (pilote) issue du projet SCALE. Dans la section 5, une variante du modèle précédent est formulée pour prendre en compte des fenêtres de temps souples. Ce nouveau modèle PDVRP est appliqué dans le cadre de la même étude de cas.

3.2 Présentation du problème

Le problème présenté dans ce chapitre est un PDVRP. Il est utilisé pour traiter le CTP dans le cadre du transport de charges partielles (les fournisseurs peuvent envoyer des véhicules peu remplis et les clients peuvent recevoir des véhicules peu remplis). Une solution pour réduire le nombre de véhicules utilisés, la distance totale parcourue et le total des émissions de CO_2, est de regrouper les marchandises de plusieurs fournisseurs et de plusieurs clients. Donc, à la place d'utiliser un véhicule par fournisseur et par client, un véhicule peut charger des marchandises chez plusieurs fournisseurs et ensuite livrer chez plusieurs clients. Les participants de la collaboration sont les fournisseurs et/ou les clients. Nous avons donc modélisé un PDVRP pour traiter ce problème qui peut être défini sur un graphe $G = O \in D, A$ où l'ensemble des nœuds est l'union de deux ensembles : l'ensemble des fournisseurs ou des nœuds de collecte (O) et l'ensemble des clients ou des nœuds de livraison (D). Un nœud ne peut pas être à la fois un fournisseur et un client ($O \cap D = \varnothing$). Comme nous nous concentrons sur la

collaboration entre fournisseurs et/ou entre clients, nous ne considérons pas de dépôt dans ce problème. L'ensemble des arcs A est composé de trois sous-ensembles : l'ensemble des arcs entre les fournisseurs, l'ensemble des arcs partant des fournisseurs et allant aux clients, et l'ensemble des arcs entre les clients. Nous considérons dans ce problème une flotte finie hétérogène de véhicules qui ont une capacité limitée. Chaque véhicule peut démarrer sa tournée chez n'importe quel fournisseur et peut la terminer chez n'importe quel client. Nous considérons également la possibilité d'avoir plusieurs types de produits. Un point de collecte peut fournir plusieurs types de produits et un type de produits peut être fourni par plusieurs points de collecte. De même, un point de livraison peut requérir plusieurs types de produits et un type de produit peut être requis par plusieurs points de collecte. Les différents fournisseurs prenant part à la collaboration fournissent généralement des types de produits différents. Chaque point de collecte peut fournir une certaine quantité d'un type de produits. De même, chaque point de livraison demande une certaine quantité d'un type de produits. La disponibilité totale d'un type de produits sur tous les fournisseurs est supposée strictement égale à la demande totale de ce type de produits

Le but du problème présenté dans ce chapitre est d'étudier les bénéfices du transport collaboratif en termes d'émissions de CO_2 et en termes de coûts en préalable à toute collaboration éventuelle. Cette section est découpée en trois parties : premièrement nous introduisons les notations utilisées, ensuite nous, expliquons les méthodes de calcul des émissions de CO_2 et de coûts, et enfin nous présentons une formulation pour le modèle linéaire en nombres entiers mixtes du problème étudié.

3.2.1 Notations

Les notations utilisées dans ce chapitre sont les suivantes :

O ensemble des fournisseurs

D ensemble des clients

V ensemble des véhicules

K ensemble des types de produits

E_{start} émissions au démarrage du moteur ($kgCO_2$)

E_{hot} émissions d'un moteur en fonctionnement ($kgCO_2$/km)

E_{evap} émissions dues aux évaporations du carburant ($kgCO_2$/km)

$E_{average}$ émissions moyennes d'un véhicule ($kgCO_2$/km)

E_{full} émissions d'un véhicule rempli ($kgCO_2$/km)

E^v_{empty} émissions du véhicule v à vide ($kgCO_2$/km)

E^v_{pallet} émissions par palette ajoutée dans le véhicule v ($kgCO_2$/km)

Fc consommation moyenne de carburant des véhicules (l/km)

Fcf quantité de CO2 émis par litre de carburant utilisé ($kgCO_2$/l)

C^v capacité du véhicule v

C_{KM} coût par kilomètre par véhicule (€)

C_{Hour} coût par heure par véhicule (€)

C_{Fixed} coût fixe d'utilisation d'un véhicule (€)

d_{ij} distance entre le nœud i et le nœud j

t_{ij} temps de trajet entre le nœud i et le nœud j

q^+_{ik} capacité du fournisseur i pour le type de produits k

q^-_{jk} demande du client j pour le type de produits k

Les variables utilisées sont définies comme suit :

x^v_{ij} vaut 1 si le véhicule v traverse l'arc ij

y^{vk}_{ij} quantité de produits de type k dans le véhicule v entre les nœuds i et j

3.2.2 Estimation des émissions de CO2 et des coûts

Dans ce problème, nous cherchons à minimiser les émissions de CO_2 induits par le transport de marchandises. Cependant, à titre d'information, nous calculons également les coûts de transport. En ce qui concerne les émissions de CO_2, les industriels sont de plus en plus soumis aux pressions législatives et sociétales pour réduire leur empreinte carbone. Cependant, le calcul exact des

émissions d'un véhicule est un travail complexe à cause du nombre de paramètres à prendre en compte comme la vitesse, la consommation du moteur, le comportement du chauffeur, l'entretien du véhicule, la météo, le dénivelé, etc. Notre méthode de calcul de ces émissions est donc simplifiée et est basée sur plusieurs articles et rapports (Hickman et al., 1999 ; Jancovici, 2007 ; Ubeda et al., 2011). La formule générale pour calculer les émissions est :

$$E_{start} + E_{hot} + E_{evaporation} \tag{3.1}$$

Pour simplifier, nous ne considérons que les émissions dues au moteur en fonctionnement Ehot. Pour calculer ces émissions, nous utilisons les formules suivantes de Jancovici (2007) :

$$E_{hot} = d * (E_{empty} + y * E_{pallet}) \tag{3.2}$$

$$E_{pallet} = \frac{E_{full} - E_{empty}}{C} \tag{3.3}$$

$$E_{full} = E_{empty} * 1.44 \tag{3.4}$$

$$E_{empty} = \frac{E_{average}}{1 + 0.44 * (1 - Tdv) * Trm} \tag{3.5}$$

Où Tdv est le taux moyen de transport à vide, Trm est le taux moyen de transport chargé et 0.44 correspond au pourcentage de surconsommation entre un véhicule plein et un véhicule vide. Finalement, pour calculer les émissions

moyennes nous utilisons la formule suivante :

$$E_{average} = Fc * Fcf \tag{3.6}$$

A titre d'information, nous calculons également les coûts de transport. Selon le Comité National Routier (CNR [1]), les coûts de transport dépendent de trois valeurs : le coût par kilomètre (carburant, usure du véhicule, ...), le coût par heure (coût du chauffeur) et le coût fixe d'utilisation (assurances, taxes, ...). Le coût de transport total est donc la somme de ces trois termes :

$$d * C_{KM} + t * C_{Hour} + C_{Fixed} \tag{3.7}$$

3.2.3 Modélisation

Nous proposons la formulation suivante pour le problème :

$$\min \sum_{i \in O \cup D} \sum_{j \in O \cup D} \sum_{v \in V} (E^v_{Empty} d_{ij} x^v_{ik} + \sum_{k \in K} E^v_{Pallet} d_{ij} y^{vk}_{ij}) \tag{3.8}$$

$$\sum_{v \in V} (\sum_{j \in O} y^{vk}_{ij} + \sum_{j \in D} y^{vk}_{ij} - \sum_{j \in O} y^{vk}_{ji}) = q^+_{ik} \qquad \forall i \in O, \forall k \in K, \tag{3.9}$$

$$\sum_{v \in V} (\sum_{i \in O} y^{vk}_{ij} + \sum_{i \in D} y^{vk}_{ij} - \sum_{i \in D} y^{vk}_{ji}) = q^-_{jk} \qquad \forall j \in D, \forall k \in K, \tag{3.10}$$

$$\sum_{j \in O} y^{vk}_{ij} + \sum_{j \in D} y^{vk}_{ij} \geq \sum_{j \in O} y^{vk}_{ji} \qquad \forall i \in O, \forall v \in V, \forall k \in K, \tag{3.11}$$

$$\sum_{i \in O} y^{vk}_{ij} + \sum_{i \in D} y^{vk}_{ij} \geq \sum_{i \in D} y^{vk}_{ji} \qquad \forall j \in D, \forall v \in V, \forall k \in K, \tag{3.12}$$

$$\sum_{j \in O} x^v_{ij} + \sum_{j \in D} x^v_{ij} \leq 1 \qquad \forall i \in O, \forall v \in V, \tag{3.13}$$

1. **http://www.cnr.fr/**.

$$\sum_{j \in D} x_{ij}^{v} \leq 1 \qquad \forall i \in D, \forall v \in V, \tag{3.14}$$

$$\sum_{j \in O} x_{ji}^{v} \leq 1 \qquad \forall i \in O, \forall v \in V, \tag{3.15}$$

$$\sum_{i \in O} x_{ij}^{v} + \sum_{i \in D} x_{ij}^{v} \leq 1 \qquad \forall j \in D, \forall v \in V, \tag{3.16}$$

$$C^{v} x_{ij}^{v} \geq \sum_{k \in K} y_{ij}^{vk} \qquad \forall i, j \in O \cup D, \forall v \in V, \tag{3.17}$$

$$x_{ij}^{v} = \{0; 1\} \qquad \forall i, j \in O \cup D, \forall v \in V, \tag{3.18}$$

$$y_{ij}^{vk} \geq 0 \qquad \forall i, j \in O \cup D, \forall v \in V, \forall k \in K, \tag{3.19}$$

La fonction objectif 3.8 du modèle est de minimiser les émissions totales de CO_2. Cette fonction objectif est composée de deux termes : le premier terme correspond aux émissions de tous les véhicules sans prendre en compte leur chargement, et le deuxième terme correspond aux émissions dues aux chargements des véhicules. La contrainte 3.9 assure que la quantité d'un type de produit qu'un fournisseur doit fournir est respectée. De la même manière, la contrainte 3.10 assure que la demande d'un client pour un type de produit est respectée. La contrainte 3.11 assure que la quantité de produits dans le véhicule en quittant un fournisseur est supérieure ou égale à la quantité initiale présente à l'arrivée du véhicule chez ce fournisseur. La différence de quantité correspond à la quantité chargée dans ce véhicule. De même manière, la contrainte 3.12 assure que la quantité de produits dans le véhicule en quittant un client est inférieure ou égale à la quantité initiale présente à l'arrivée du véhicule chez ce client. La différence de quantité correspond à la quantité déchargée dans ce véhicule. Les contraintes 3.13 et 3.14 assurent qu'un véhicule ne se dirige que vers un seul nœud au maximum en quittant un nœud. Les contraintes 3.15 et 3.16 assurent qu'un véhicule ne vient que d'un seul nœud maximum en arrivant dans un autre nœud. La contrainte 3.17 assure que la capacité des véhicules est respectée.

3.3 Etudes expérimentales

3.3.1 Présentation des instances

Pour tester ce modèle, nous avons généré cinq ensembles de dix instances. L'organisation de ces cinq ensembles est présentée dans le tableau 3.1.

Set	Number of suppliers	Number of customers	Number of product types	Number of nodes
1	2	8	2	10
2	3	17	3	20
3	4	26	4	30
4	6	34	6	40
5	8	42	8	50

Tableau 3.1 – Organisation des cinq ensembles

Chaque fournisseur fournit un produit différent, donc le nombre de type de produits est égal au nombre de fournisseurs. Pour chacune des dix instances des cinq jeux, les nœuds sont placés aléatoirement dans un carré de 250 sur 250, c'est-à-dire que les coordonnées horizontales et verticales d'un nœud sont générées aléatoirement dans l'intervalle $[0;250]$ de manière uniforme. La distance d_{ij} entre deux nœuds i et j est la distance Euclidienne. Pour chaque client, la demande q_{ik}^{-} est générée de manière aléatoire dans l'intervalle $[0;50]$. La probabilité que la demande pour un type de produit soit de zéro est de 50% pour les ensembles 1 et 2, et 75% pour les ensembles 3, 4 et 5. Les probabilités pour que la demande soit générée de manière aléatoire uniforme dans $]0;10]$, $]10;25]$ et $]25;50]$ sont respectivement de 12,5%, 25% et 12,5%.

3.3.2 Expérimentations

Nous proposons quatre différents scénarios pour cette étude, dans le but de comparer ces scénarios avec le scénario "as is". Le scénario "as is" est le scénario dans lequel un fournisseur livre directement un de ses clients en utilisant un seul véhicule (figure 3.1).

Ensuite, dans les scénarios 1 et 2 (sans collaboration entre fournisseurs), nous

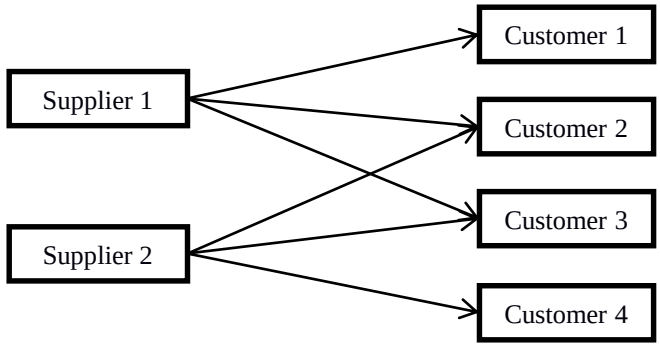

FIGURE 3.1 – Réseau de distribution du scénario "as is"

autorisons un fournisseur à livrer plusieurs de ses clients avec le même véhicule (figure 3.2).

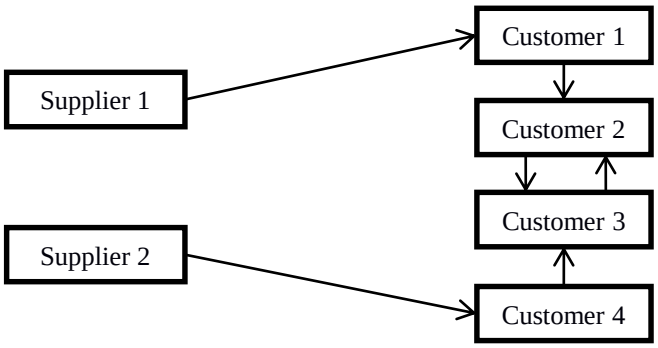

FIGURE 3.2 – Réseau de distribution sans collaboration entre les fournisseurs

La différence entre les deux scénarios 1 et 2 est que dans le premier, nous utilisons uniquement des véhicules de type 1 et que dans le second nous utilisons des véhicules de type 1 et 2. Finalement dans les scénarios 3 et 4, nous considérons la collaboration entre les fournisseurs, donc un véhicule peut charger chez plusieurs fournisseurs et ensuite livrer chez plusieurs clients (figure 3.3).

Comme pour les scénarios 1 et 2, la différence entre les scénarios 3 et 4 est que dans le scénario 3, nous utilisons uniquement des véhicules de type 1, et que dans le scénario 4 nous utilisons des véhicules de types 1 et 2. Les caractéristiques des véhicules sont données dans le tableau 3.2. Pour les scénarios 2 et 4, la proportion de véhicules de type 1 disponibles est de 2/3 et la proportion de véhicules de type 2 disponibles est de 1/3.

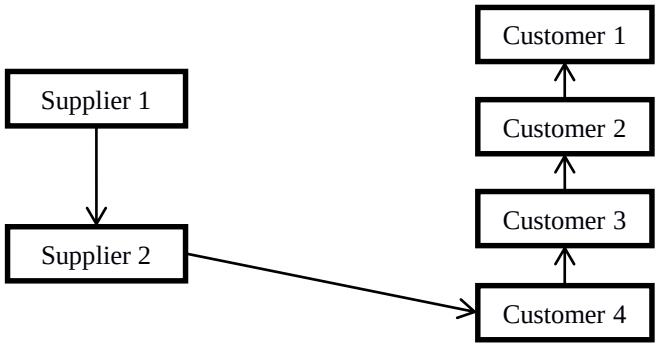

FIGURE 3.3 – Réseau de distribution avec collaboration entre les fournisseurs

Type	Capacité (pallettes)	Consommation (l/km)	E^v_{Empty} (kg/km)	E^v_{Pallet} (kg/km)	C_{KM} (€/km)	C_{Hour} (€/h)	C_{Day} (€/day)
1	26	0.342	0.767	1.104	0.525	21.67	156.81
2	11	0.25	0.561	0.808	0.358	19.85	195.98

TABLEAU 3.2 – Caractéristiques des différents types de véhicules utilisés

Nous utilisons le modèle présenté dans la section 2 pour résoudre les instances. Dans ce modèle, ce sont les émissions qui sont minimisées, mais les coûts correspondants sont calculés à titre indicatif pour comparer les scénarios sur le plan économique. Le modèle est résolu avec le solver CPLEX 12.5 sur un PC disposant d'un processeur core i7 2,7GHz, de 8Go de RAM et tournant sous Windows 8.1. Une limite de temps de 3600 secondes est fixée pour résoudre les instances. Le gap entre la meilleure solution trouvée par CPLEX (la borne supérieure) et la borne inférieure dépend du jeu d'instance utilisé. Plus les instances sont grandes, plus les solutions sont difficiles à trouver. Pour l'ensemble 1, le gap moyen pour les quatre scénarios et les dix instances est de 0.9%, pour l'ensemble 2 ce gap est de 5.6%, pour l'ensemble 3 ce gap est de 6.2%, pour l'ensemble 4 ce gap est de 16%, et pour l'ensemble 5 ce gap est de 20%. Nous faisons tourner le modèle une fois pour chaque instance et pour chaque scénario. Le résultat d'un ensemble est la somme des résultats de toutes les instances de l'ensemble considéré.

3.3.3 Résultats et discussions

Regrouper les livraisons présente des bénéfices en termes d'émissions de CO_2 et de coûts de transport. Dans la figure 3.4, nous pouvons voir que le taux d'utilisation moyen des véhicules pour le scénario "as is" est d'environ 61%, ce qui est approximativement le taux d'utilisation moyen en Europe. La mutualisation des moyens de transport permet d'améliorer ce taux à 72% pour le scénario 1, 81% pour le scénario 2, 82% pour le scénario 3 et 90% pour le scénario 4.

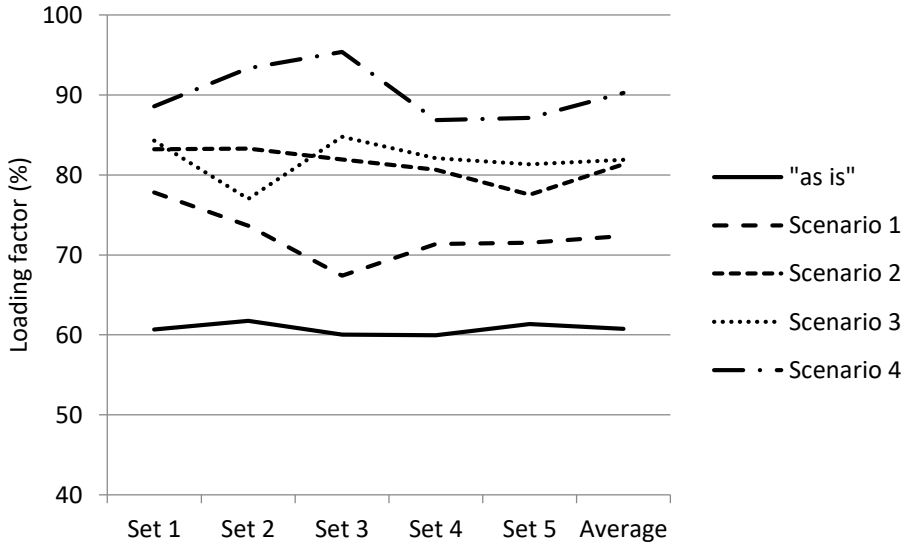

FIGURE 3.4 – Taux d'utilisation des véhicules par ensemble d'instances

Pour les émissions de CO_2 (figure 3.5), pour le scénario 1, la réduction varie de 6% à 13%. La réduction moyenne des émissions pour tous les ensembles est de 9%. Pour le scénario 2, la réduction varie de 10% à 14%. La réduction moyenne des émissions pour tous les ensembles est de 12%. Pour le scénario 3, la réduction varie de 15% à 20%. La réduction moyenne des émissions pour tous les ensembles est de 18%. Pour le scénario 4, La réduction varie de 16% à 22%. La réduction moyenne des émissions pour tous les ensembles est de 20%.

En ce qui concerne les coûts (figure 3.6), pour le scénario 1, la réduction varie de 8% à 16%. La réduction moyenne des coûts pour tous les ensembles est de

12%. Pour le scénario 2, la réduction varie de 11% à 17%. La réduction moyenne des coûts pour tous les ensembles est de 14%. Pour le scénario 3, la réduction varie de 20% à 23%. La réduction moyenne des coûts pour tous les ensembles est de 22%. Pour le scénario 4, la réduction varie de 20% à 24%. La réduction moyenne des coûts pour tous les ensembles est de 23%.

Comme nous pouvons le voir dans les scénarios 1 et 2, regrouper les commandes d'un fournisseur vers plusieurs clients est très bénéfique en termes d'utilisation des véhicules, de coûts et d'émissions de CO_2. En utilisant aussi une part de véhicules de capacité plus petite, nous pouvons encore améliorer ces indicateurs. Cependant, en utilisant la collaboration entre les fournisseurs et en regroupant les commandes de plusieurs clients, les coûts et les émissions peuvent être réduits davantage. Pour conclure, optimiser les émissions CO_2 permet donc de réduire l'impact environnemental des entreprises et permet également des gains économiques.

3.4 Etude de cas

L'étude de cas, en tant que méthode de recherche dans le domaine de la gestion des opérations, est reconnue comme une méthode appropriée à utiliser, en raison de sa capacité à explorer un phénomène dans le contexte de la vie réelle (Yin, 2013 ; Karlsson, 2010). D'un autre côté, la simulation en tant que méthode appliquée pour imiter le comportement d'un processus réel (Banks, 1998), permet aux chercheurs d'explorer différents scénarios d'un processus (Law et al., 1991). Adopter l'étude de cas et la simulation en tant que deux perspectives méthodologiques dans la recherche de solution logistique fournit une vue étendue d'un phénomène de la logistique. L'étude de cas donne des interprétations subjectives individuelles de la compréhension du phénomène et la simulation donne ses mesures et quantifications (Hellström et Nilsson, 2006).

Dans ce chapitre une étude de cas issue du projet Scale est présentée pour fournir des estimations sur les réductions possibles des émissions de CO_2 et des coûts de transport dans le cadre d'une collaboration potentielle entre clients.

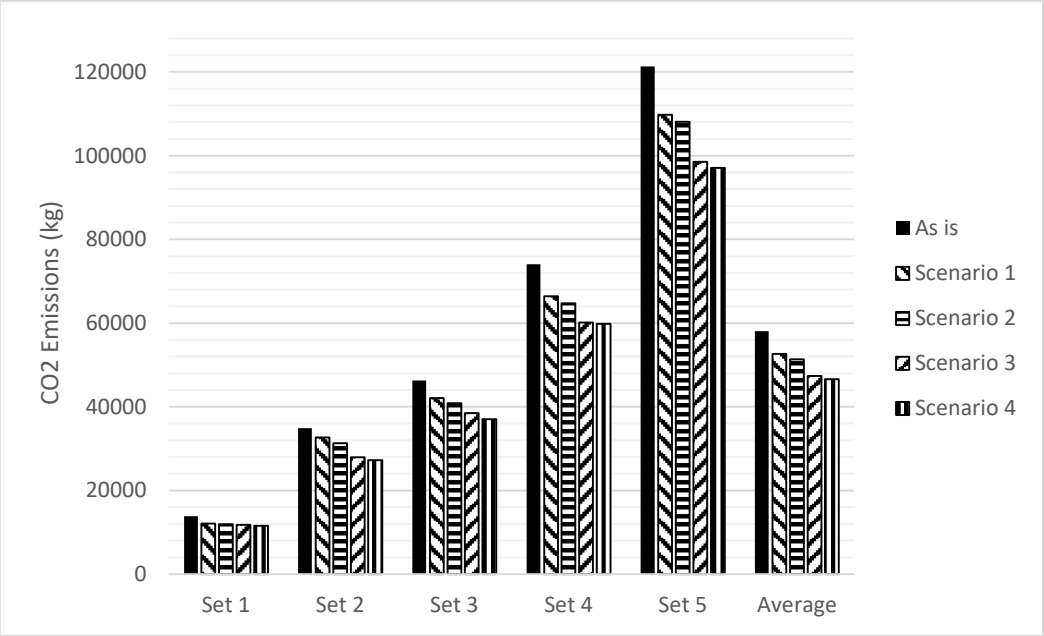

FIGURE 3.5 – Emissions totales par ensemble d'instances

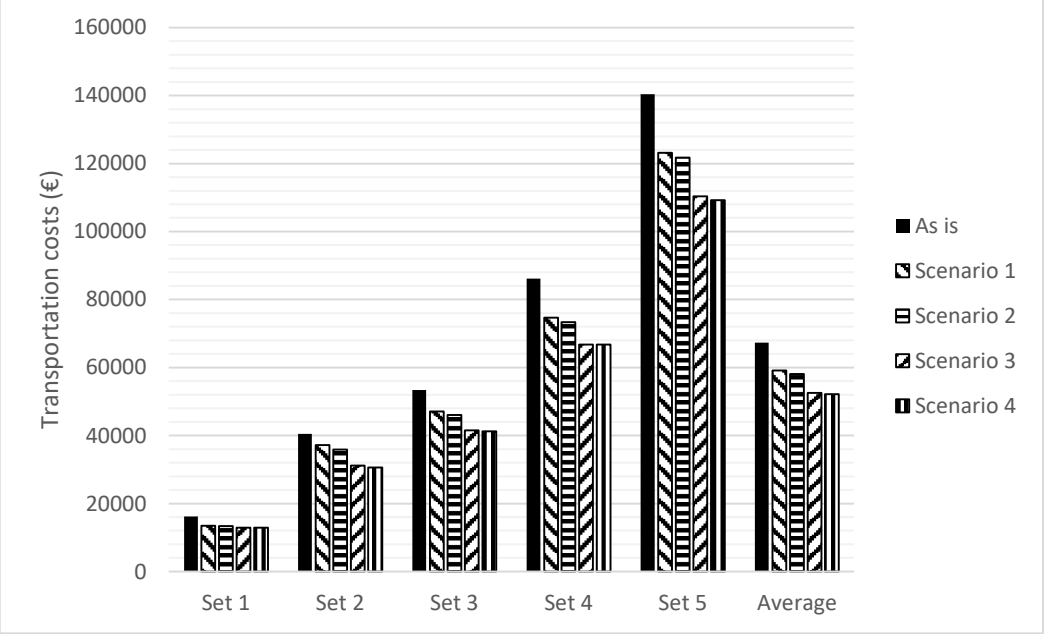

FIGURE 3.6 – Coûts totaux de transport par ensemble d'instances

3.4.1 Données du réseau de distribution

Les données ont été collectées lors de plusieurs visites à l'entreprise, en réalisant des interviews avec les responsables logistiques de l'entreprise, et aussi en analysant leurs données réelles, incluant des fichiers Excel contenant les données des commandes. En cas de besoins, des conversations par email ont eu lieu pour avoir plus d'éclaircissement sur les données. Ces données, enregistrées dans les fichiers Excel fournis, indiquent les données de ventes par type de produits commandés ainsi que la quantité en nombre ou en poids, la localisation des entrepôts des distributeurs clients de l'entreprise, et des informations sur les types de palettes utilisées et la place prise par les produits sur la palette afin de déterminer le nombre de palettes requise pour chaque commande.

L'entreprise étudiée, appelée entreprise A durant ce chapitre, distribue des fruits et légumes à partir de son centre de distribution situé au Royaume-Uni. Les produits proviennent principalement d'Espagne pendant les mois d'hiver et de sources locales durant les mois d'été. L'entreprise A procède à l'emballage et au stockage des produits, les rendant ainsi prêts à être livrés aux 27 sites des trois plus grands distributeurs du Royaume-Uni. L'entreprise A fournit un volume significatif de fruits et légumes sur le marché de la distribution britannique et a également le potentiel pour améliorer l'utilisation du réseau de distribution étudié, faisant de l'entreprise A un cas d'étude très approprié. La figure 3.7 illustre le réseau de distribution étudié, incluant le centre de distribution de l'entreprise A, désigné par DC dans la figure, et les 27 magasins des distributeurs, désignés par R1 à R27 dans la figure.

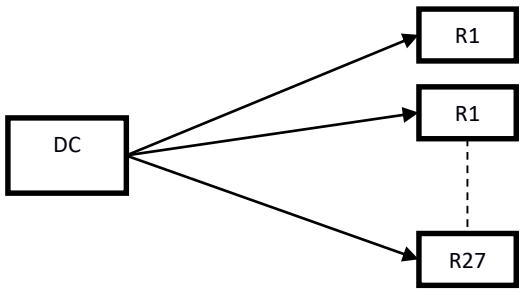

Figure 3.7 – Le réseau de distribution étudié

Cette étude a été conduite pour identifier les opportunités de réduire l'impact environnemental de ce réseau de distribution, à travers la collaboration des trois différents distributeurs. Autrement dit quand les trois distributeurs acceptent de partager des véhicules pour la distribution de leurs commandes. Le modèle présenté précédemment consiste à optimiser la quantité des émissions de CO_2 produits par les véhicules dans ce réseau de distribution en partageant les véhicules entre distributeurs. L'étude utilise un historique de données provenant de l'entreprise A comprenant la période allant de septembre 2012 à avril 2013. La figure 3.8 illustre comment les opérations de distribution sont modifiées de la livraison directe vers chaque distributeur (figure 3.7) à la distribution collaborative vers plusieurs points de livraison.

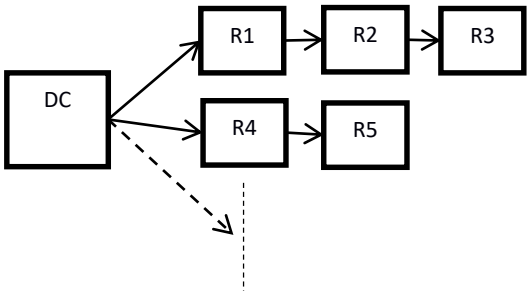

Figure 3.8 – Distribution collaborative en visitant plusieurs distributeurs avec le même véhicule

Dans les fichiers de données reçues de la part de l'entreprise A, montrant l'historique des livraisons du centre de distribution vers les sites des distributeurs, les quantités demandées sont exprimées en termes de nombre de têtes, de caissettes ou de poids. Nous avons donc utilisé les descriptions des produits pour déterminer le nombre de produits contenus dans une palette, afin de déterminer pour chaque commande et pour chaque produit, le nombre de palettes nécessaires au transport de la commande. Si la description d'un produit ne contient pas assez d'informations pour effectuer la conversion en palettes, les commandes correspondantes à ce produit ne sont pas prises en compte, en accord avec les responsables logistiques. Tous les produits étant conditionnés de la même manière, plusieurs produits différents peuvent être placés sur la même palette. Ce faisant, tous les produits sont considérés comme appartenant au même type.

3.4.2 Expérimentations

Nous proposons deux scénarios pour étudier ce problème. Dans le premier scénario, nous utilisons seulement un type de véhicule, le type 1. Dans le deuxième scénario, nous utilisons deux types de véhicules différents, le type 1 et le type 2. Dans ce scénario, la proportion de véhicules de type 1 disponibles est de 2/3 et la proportion de véhicules de type 2 disponibles est de 1/3. Les caractéristiques des différents types de véhicules sont les mêmes que dans la section 3 (tableau 3.2). Nous comparons ces deux scénarios avec la situation existante, appelée dans le reste du chapitre scénario "as is". Les caractéristiques du scénario "as is" sont les suivantes :

— Le type de véhicules utilisés est le type 1.

— Quand l'entreprise A doit expédier plusieurs commandes vers le même site distributeur durant la journée, les demandes de ces commandes sont additionnées et les produits sont expédiés en utilisant le nombre minimum de véhicules pour satisfaire toutes les commandes vers ce site.

Les distances utilisées entre les différents sites sont les distances à vol d'oiseau. En général, la plupart des distributeurs veulent être livrés durant une petite fenêtre de temps durant la matinée mais le modèle ne prend pas en compte les fenêtres de temps. Donc en théorie, un véhicule peut livrer un grand nombre de sites en une seule tournée. Ceci n'est pas forcément compatible avec le cas pratique car un véhicule n'a pas le temps de livrer beaucoup de sites en respectant les fenêtres de temps. Nous avons donc décidé dans un premier temps d'ajouter deux contraintes pour limiter la longueur des tournées. Tout d'abord, nous limitons la distance maximum qu'il peut y avoir entre deux sites livrés consécutivement par un même véhicule. Cette distance est fixée à 50 km. Deuxièmement, nous limitons le nombre de sites qu'un véhicule peut livrer à trois maximum. Ces deux contraintes supplémentaires permettent de livrer plusieurs sites avec le même véhicule en des temps raisonnables. Ces deux contraintes sont exprimées par les équations 3.20 et 3.21.

$$d_{ij} x_{ij}^{v} \leq 50 \qquad \forall i, j \in O \cup D, \forall v \in V, \tag{3.20}$$

$$\sum_{i \in D} \sum_{j \in D} x_{ij}^{v} \leq 3 \qquad \forall v \in V, \tag{3.21}$$

Nous utilisons le modèle présenté dans la section 2 pour résoudre cette étude de cas. Dans ce modèle, ce sont les émissions qui sont minimisées, mais les coûts correspondants sont calculés à titre indicatif. Le modèle est résolu avec le solver CPLEX 12.5 sur un PC disposant d'un processeur core i7 2,7GHz, de 8Go de RAM et tournant sous Windows 8.1. Ce problème étant NP-difficile (Chen et al., 2014), une limite de temps de 180 secondes est fixée. Le gap entre la meilleure solution trouvée par CPLEX (la borne supérieure) et la borne inférieure varie entre 0 et 5% et le gap moyen est de 1%. L'étude de cas est un cas particulier du modèle présenté puisqu'il n'y a qu'un seul fournisseur et qu'un seul type de produits. La simulation est réalisée jour par jour. Nous lançons le modèle une fois pour chaque jour. Après avoir fait tourner le modèle, des tournées sont créées pour livrer plusieurs sites et pour améliorer le taux d'utilisation des véhicules pour une journée définie. L'expérimentation est réalisée sur la saison hivernale 2012-2013. Les résultats journaliers sont additionnés pour agréger les résultats par mois.

3.4.3 Résultats et discussions

Nous pouvons voir que regrouper les livraisons peut être extrêmement bénéfique pour l'entreprise A, Dans la figure 3.9, nous pouvons voir que, excepté pour septembre et octobre 2012 (deux mois avec une forte demande), l'entreprise A n'a pas une planification du transport très efficace en termes de taux d'utilisation des véhicules. Le taux d'utilisation moyen de la période étudiée pour le scénario "as is" est de 47.63%. Grâce au regroupement, nous pouvons améliorer le taux d'utilisation de 27% en utilisant un type de véhicules, et de 37% en utilisant deux types de véhicules.

Pour les émissions de CO_2 (figure 3.10) pour le scénario 1, la réduction des émissions par rapport au scénario "as is" varie de 18.33% en septembre 2012 à 30.85% en janvier 2013. La réduction moyenne pour la période est de 26.44%. Pour le scénario 2, la réduction des émissions par rapport au scénario "as is" varie de 19.84% en septembre 2012 à 34.52% en février 2013. La réduction moyenne

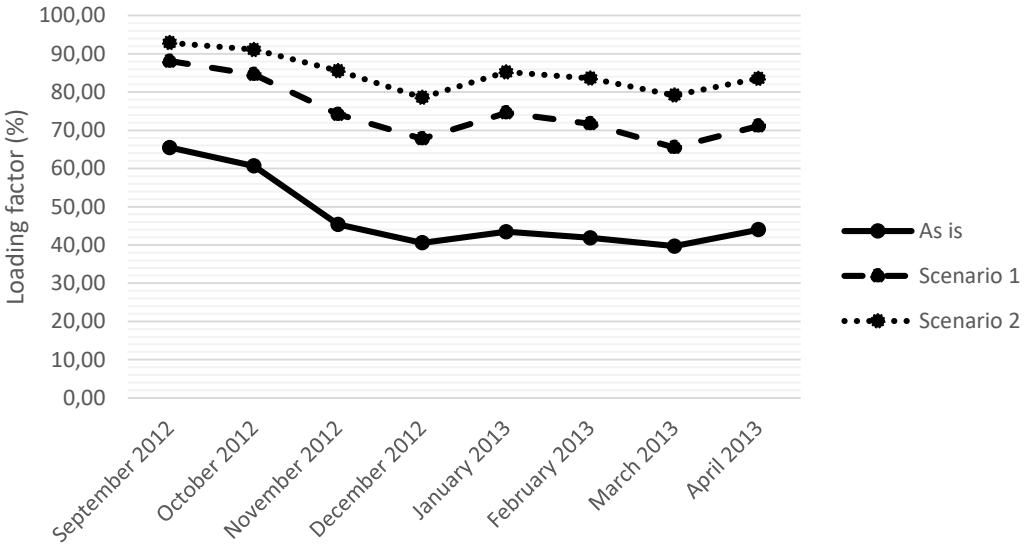

FIGURE 3.9 – Taux d'utilisation des véhicules par mois

pour la période entière est de 29.49%. La différence entre les deux scénarios est plus importante pour les émissions que pour les coûts.

Pour les coûts de transport (figure 3.11), pour le scénario 1, la réduction des coûts par rapport au scénario "as is" varie de 23.5% en septembre 2012 à 36.64% en janvier 2013. La réduction moyenne des coûts est de 32.03% pour la période entière. Pour le scénario 2, la réduction des coûts par rapport au scénario "as is" varie de 24.42% en septembre 2012 à 38.51% en février 2013. La réduction moyenne des coûts est de 33.83% pour la période entière. Nous pouvons voir que la différence entre les deux scénarios est très basse car la différence de coûts d'utilisation entre les deux types de véhicules n'est pas très significative.

Après avoir obtenu ces résultats, une réunion a été organisée entre les membres de l'équipe de recherche et les responsables logistiques et produits de l'entreprise A pour présenter et valider le modèle et les résultats obtenus. Les résultats des différents scénarios ont été présentés et leurs impacts pour l'entreprise ont été discutés longuement. Des retours ont notamment porté sur l'intégration des fenêtres de temps dans un nouveau modèle. Cette intégration est étudiée dans la section 5 de ce chapitre.

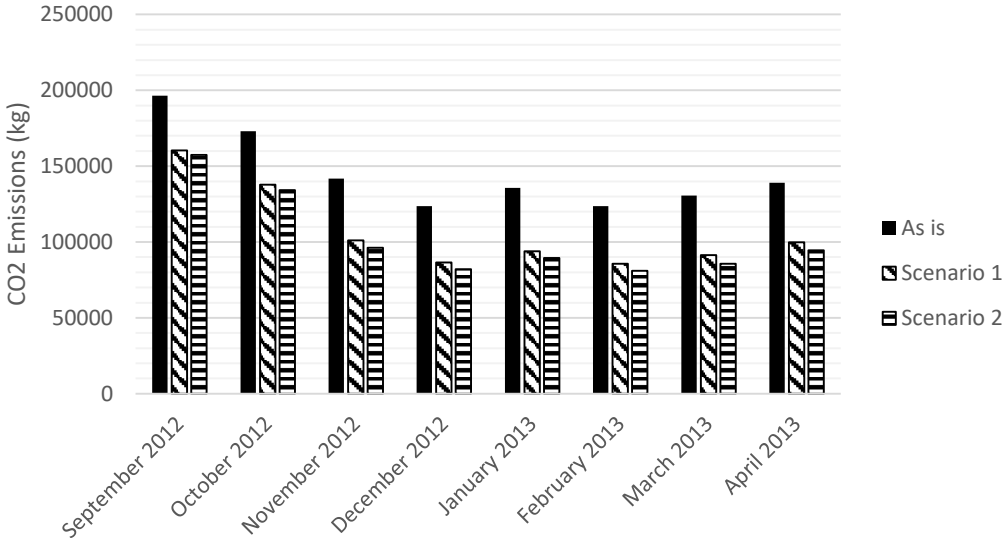

FIGURE 3.10 – Emissions de CO_2 par mois

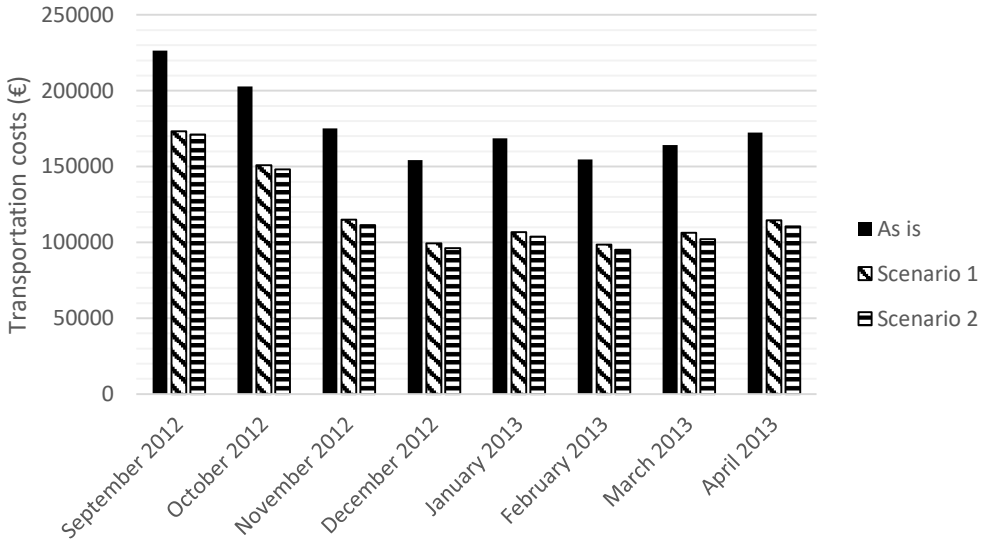

FIGURE 3.11 – Coûts de transport totaux par mois

3.5 Le PDVRP avec fenêtres de temps souples

Le problème présenté ici est le même que celui des sections précédentes. Cependant, pour respecter davantage les contraintes du terrain dans le cadre de l'étude de cas, les fenêtres de temps sont maintenant prises en compte. Deux types de fenêtres de temps sont à distinguer. Premièrement, du côté des fournisseurs, les fenêtres de temps sont dures. Un véhicule doit arriver dans un délai prédéfini pour chaque fournisseur. Si le véhicule arrive trop tôt, il peut attendre jusqu'à l'ouverture de la fenêtre de temps. Par contre le véhicule doit arriver absolument avant la fin de la fenêtre de temps. Du côté des clients, les fenêtres de temps sont souples. Comme pour les fournisseurs, un véhicule peut arriver chez un client avant le début de la fenêtre de temps et attendre. Par contre, contrairement aux fournisseurs, un véhicule peut arriver après la fin de la fenêtre de temps mais une pénalité de retard doit être payée. Le coût de pénalité maximum correspond ici au coût de transport entre le fournisseur et le client concernés. Ce coût de pénalité est proportionnel au coefficient de retard de véhicule (Flamini et al., 2011). Ce coefficient de retard est compris entre 0 et 1 et augmente de manière linéaire avec le retard (figure 3.12). Après un certain retard, le coefficient est de 1. Pour cela, nous considérons deux bornes supérieures aux fenêtres de temps, l et l^{max} avec $l \leq lmax$. Si le véhicule arrive avant l, il est dans la fenêtre de temps et donc le coefficient de retard est nul et il n'y a pas de pénalité. Si le véhicule arrive entre l et l^{max}, le coefficient de retard croît de manière linéaire avec le retard. Et si le véhicule arrive après l^{max}, le coefficient est de 1 et la pénalité est donc maximale. Le temps de service chez un fournisseur ou chez un client est proportionnel à la quantité de marchandises à charger ou à décharger.

Le but du problème de cette section est de comparer les bénéfices de la collaboration quand nous minimisons les coûts ou les émissions de CO_2 dues au transport. Les émissions sont en relation avec la distance parcourue et la quantité de marchandises dans le véhicule. Les coûts sont l'addition des coûts de transports et des pénalités de retard.

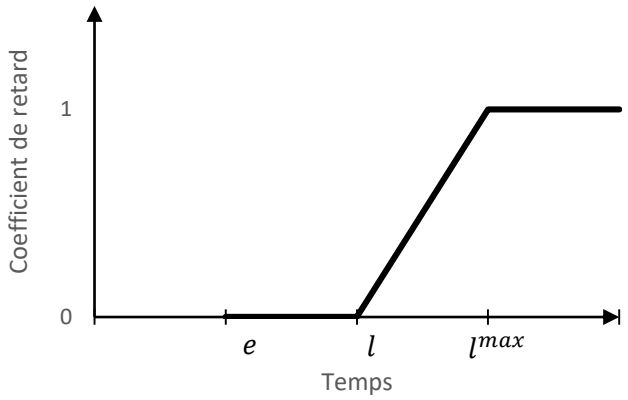

FIGURE 3.12 – Coefficient de retard d'un véhicule en fonction de l'heure d'arrivée

3.5.1 Formulation du problème

Pour modéliser le PDVRP avec fenêtres de temps, nous avons besoin des notations supplémentaires suivantes :

P pénalité de retard

M une grande constante

e_i heure minimum de début de service au nœud i

l_i heure maximum de début de service au nœud i

l_i^{max} heure maximum après laquelle la probabilité de refuser le véhicule est égale à 1

h_i^v temps de chargement au site i pour le véhicule v

Et des variables supplémentaires suivantes :

s_{ij}^v vaut 1 si le véhicule v transfère des marchandises du fournisseur i au client j

z_{ij}^v quantité de produits de type k chargés à i et livrés à j par le véhicule v

a_i^v heure de début de service du véhicule v au site i

b_i^v heure de fin de service du véhicule v au site i

w_i^v temps d'attente du véhicule v au site i

m_i^v vaut 1 si le véhicule v arrive après l_i

u_i^v vaut 1 si le véhicule v arrive après l_i^{max}

f_i^v coefficient de retard du véhicule v au site i

g_{ij}^v coefficient du coût de pénalité

Nous présentons maintenant la formulation de notre modèle mono-objectif. Nous envisageons deux alternatives de ce modèle afin d'obtenir des bornes minorantes pour chacune d'entre elles. Nous voulons minimiser soit les émissions de CO_2 ou sinon les coûts financiers.

$$\min \sum_{i\in O\cup D}\sum_{j\in O\cup D}\sum_{v\in V}(E^v_{Empty}d_{ij}x^v_{ik}+\sum_{k\in K}E^v_{Pallet}d_{ij}y^{vk}_{ij}) \tag{3.22}$$

$$\min \sum_{i\in O\cup D}\sum_{j\in O\cup D}\sum_{v\in V}c^v_{KM}d_{ij}x^v_{ij}+\sum_{i\in O}\sum_{j\in D}g^v_{ij}d_{ij}P \tag{3.23}$$

La première alternative objectif du modèle (Équation 3-22) est de minimiser les émissions totales de CO2. Cette fonction objectif est composée de deux termes, le premier terme correspond aux émissions de tous les véhicules sans leur chargement, et le second terme est les émissions additionnelles dues aux chargements des véhicules. La seconde alternative objectif (Équation 3-23) est de minimiser les coûts des kilomètres parcourus et les coûts de pénalité.

Ce problème comprend les contraintes 3.9 à 3.19 plus les contraintes suivantes :

$$\sum_{j\in D}\sum_{v\in V}z^{vk}_{ij}=q^+_{ik} \qquad \forall i\in O,\forall k\in K, \tag{3.24}$$

$$\sum_{i\in O}\sum_{v\in V}z^{vk}_{ij}=q^-_{jk} \qquad \forall j\in D,\forall v\in V, \tag{3.25}$$

$$\sum_{j\in D}z^{vk}_{ij}\leq\sum_{j\in O}y^{vk}_{ij}+\sum_{j\in D}y^{vk}_{ij}-\sum_{j\in O}y^{vk}_{ji} \qquad \forall i\in O,\forall v\in V,\forall k\in K, \tag{3.26}$$

$$\sum_{i\in O}z^{vk}_{ij}\leq\sum_{i\in O}y^{vk}_{ij}+\sum_{i\in D}y^{vk}_{ij}-\sum_{i\in D}y^{vk}_{ji} \qquad \forall j\in D,\forall v\in V,\forall k\in K, \tag{3.27}$$

$$Ms_{ij}^{v} \geq \sum_{k \in K} z_{ij}^{vk} \qquad \forall i \in O, \forall j \in D, \forall v \in V, \tag{3.28}$$

$$s_{ij}^{v} \leq M \sum_{k \in K} z_{ij}^{vk} \qquad \forall i \in O, \forall d \in D, \forall v \in V \tag{3.29}$$

Les contraintes 3.24 à 3.29 sont les contraintes qui permettent de déterminer si un véhicule livre de la marchandise depuis un fournisseur vers un client, et aussi la quantité livrée. Les contraintes 3.24 à 3.27 permettent de connaître la quantité d'un type de marchandises prélevées chez un fournisseur et livrées chez un client. Cette quantité est représentée par la variable z_{ij}^{vk}. Cette variable nous permet de calculer le temps de chargement ou de déchargement en fonction de la quantité. Les contraintes 3.28 et 3.29 permettent de connaître si un véhicule charge chez un fournisseur i pour livrer un client j. Cette information nous permet de calculer les frais de pénalité le cas échéant.

$$b_i^v + t_{ij} + w_j^v - a_j^v \leq M(1 - x_{ij}^v) \qquad \forall i, j \in O \cup D, \forall v \in V, \tag{3.30}$$

$$b_i^v + t_{ij} + w_j^v - a_j^v \geq -M(1 - x_{ij}^v) \qquad \forall i, j \in O \cup D, \forall v \in V, \tag{3.31}$$

$$a_i^v + h_i^v \sum_{j \in D} \sum_{k \in K} z_{ij}^{vk} = b_i^v \qquad \forall i \in O, \forall v \in V, \tag{3.32}$$

$$a_j^v + h_j^v \sum_{i \in O} \sum_{k \in K} z_{ij}^{vk} = b_j^v \qquad \forall j \in D, \forall v \in V, \tag{3.33}$$

$$a_i^v \geq e_i \qquad \forall i \in O \cup D, \forall v \in V, \tag{3.34}$$

$$a_i^v \leq l_i \qquad \forall i \in O, \forall v \in V, \tag{3.35}$$

Les contraintes 3.30 à 3.35 sont les contraintes relatives aux fenêtres de temps. Les contraintes 3.30 et 3.31 assurent le respect des temps de trajet et des temps d'attente. Les contraintes 3.32 et 3.33 assurent le respect des temps de chargement et déchargement. Les contraintes 3.34 et 3.35 assurent que les fenêtres de temps dures sont respectées.

$$Mm_i^v \geq a_i^v - l_i \qquad \forall i \in D, \forall v \in V, \tag{3.36}$$

$$m_i^v \leq a_i^v / l_i \qquad \forall i \in D, \forall v \in V, \tag{3.37}$$

$$M u_i^v \geq a_i^v - l_i^{max} \qquad \forall i \in D, \forall v \in V, \tag{3.38}$$

$$u_i^v \leq a_i^v / l_i^{max} \qquad \forall i \in D, \forall v \in V, \tag{3.39}$$

$$f_i^v \geq u_i^v \qquad \forall i \in D, \forall v \in V, \tag{3.40}$$

$$f_i^v \geq \frac{a_i^v - l_i}{l_i^{max} - l_i} - M u_i^v \qquad \forall i \in D, \forall v \in V, \tag{3.41}$$

$$f_i^v \leq m_i^v \qquad \forall i \in D, \forall v \in V, \tag{3.42}$$

$$g_{ij}^v \geq f_j^v - (1 - s_{ij}^v) \qquad \forall i \in O, \forall j \in D, \forall v \in V \tag{3.43}$$

$$g_{ij}^v \leq s_{ij}^v \qquad \forall i \in O, \forall j \in D, \forall v \in V, \tag{3.44}$$

$$g_{ij}^v \leq f_j^v \qquad \forall i \in O, \forall j \in D, \forall v \in V, \tag{3.45}$$

Les contraintes 3.36 à 3.45 sont les contraintes qui permettent de calculer les pénalités de retard. Les contraintes 3.36 et 3.37 sont des contraintes de linéarisation qui permettent de savoir si un véhicule arrive en retard ou non chez un client mais avant l_i^{max}. Les contraintes 3.38 et 3.39 sont des contraintes de linéarisation qui permettent de savoir si un véhicule arrive en retard ou non chez un client mais après l_i^{max}. Les contraintes 3.40 à 3.42 permettent de connaître le coefficient de retard f_i^v du véhicule v chez le client i. Ce coefficient est de 0 lorsque le véhicule arrive avant l_i, il est compris entre 0 et 1 et est proportionnel au retard lorsque le véhicule arrive entre l_i et l_i^{max}, et il est de 1 lorsque le véhicule arrive après l_i^{max}. Les contraintes 3.43 à 3.45 permettent de connaître le coefficient de pénalité d'un véhicule en fonction du coefficient de retard lorsqu'on cherche à optimiser les coûts logistiques.

$$s_{ij}^v \in \{0;1\} \qquad \forall i \in O, \forall j \in D, \forall v \in V, \tag{3.46}$$

$$z_{ij}^{vk} \geq 0 \qquad \forall i \in O, \forall j \in D, \forall v \in V, \forall k \in K, \tag{3.47}$$

$$a_i^v, b_i^v, w_i^v \geq \qquad \forall i \in O \cup D, \forall v \in V, \tag{3.48}$$

$$m_i^v, u_i^v \in \{0;1\} \qquad \forall i \in D, \forall v \in V, \tag{3.49}$$

$$0 \leq f_i^v \leq 1 \qquad \forall i \in D, \forall v \in V, \tag{3.50}$$

$$0 \leq g_{ij}^v \leq 1 \qquad \forall i \in O, \forall j \in D, \forall v \in V, \tag{3.51}$$

Les contraintes 3.46 à 3.51 représentent les restrictions imposées aux variables.

3.5.2 Expérimentations et résultats

Ce modèle a été testé sur des données réelles provenant de l'entreprise A mentionnée dans la section 3. Les données fournies par cette entreprise sont toujours de la même forme que celle décrite précédemment. Cependant, l'horizon de temps étudié n'est plus le même. Les données utilisées dans cette section ne couvrent que la période allant de janvier 2013 à avril 2013. De plus, le modèle avec les fenêtres de temps étant plus difficile à résoudre, nous nous sommes concentrés non plus sur trois clients distributeurs mais sur uniquement deux clients possédant 19 sites au total.

Deux scénarios sont étudiés dans cette étude de cas. Dans le premier scénario, seules les émissions de CO_2 sont minimisées et donc la fonction objectif 3.22 est utilisée. Dans le deuxième scénario, seuls les coûts (comprenant les coûts de transport et les pénalités) sont minimisés et donc la fonction objectif 3.23 est utilisée. Un seul type de véhicule est utilisé ici. Les caractéristiques des véhicules sont les suivantes :

— La capacité des véhicules est de 26 palettes.

— Les émissions de CO_2 d'un véhicule à vide sont de 0,767 kgCO_2/km.

— Chaque palette dans un véhicule émet 12,96 gCO_2/km.

— Les coûts et les pénalités par km par véhicule sont de £0,91.

— La vitesse moyenne des véhicules utilisée pour calculer les temps de trajet est de 55 km/h.

Les caractéristiques de temps pour les différents sites sont les suivantes :

— Le centre de distribution de l'entreprise A ouvre à 3h et ne possède pas d'heure de fermeture

— Tous les sites clients ouvrent à 6h.

— La première borne maximale de début de service des clients (l_i) est 9h.

— La seconde borne maximale de début de service des clients (l_i^{max}) est 10h.

— Le temps de chargement et de déchargement d'une palette est de 3 minutes.

— Nous avons également fixé une durée maximale de 9 heures pour les tournées.

Comme pour la section 3, le problème est résolu avec le solver CPLEX 12.5 sur un PC disposant d'un processeur core i7 2,7GHz, de 8Go de RAM et tournant sous Windows 8.1. Le modèle est exécuté pour chaque jour de la période étudiée et les résultats sont agrégés par mois. Le temps limite pour la résolution d'une journée est de 600 secondes. Le gap entre la meilleure solution trouvée et la borne inférieure varie entre 1% et 20%. Le gap est toujours beaucoup plus faible pour le scénario 1 que pour le scénario 2. Ceci s'explique par le fait que dans le calcul de la fonction de minimisation des émissions, le retard ne rentre pas en compte, et donc les variables de temps et de retard n'ont pas d'influence sur la fonction objectif.

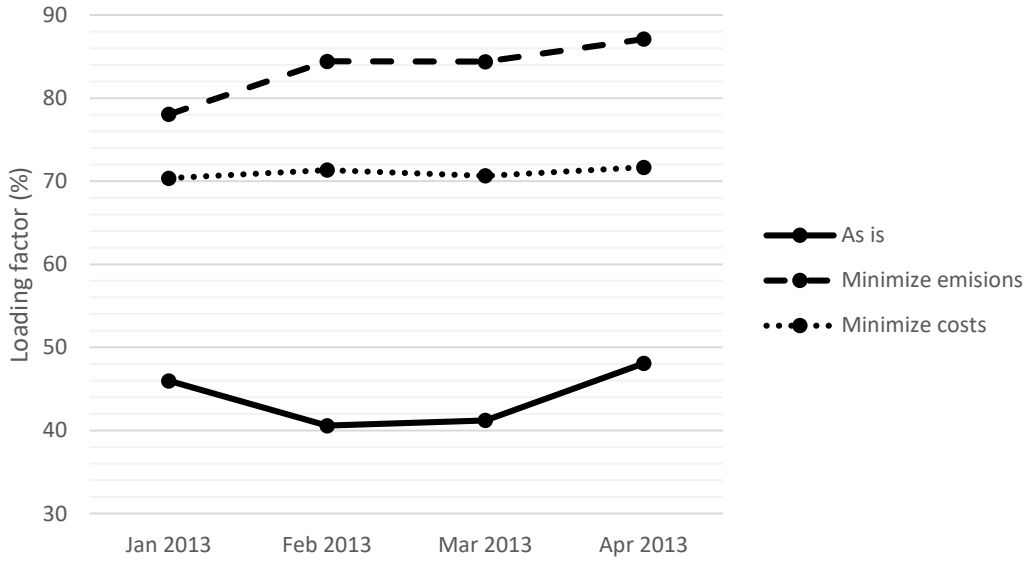

Figure 3.13 – Taux d'utilisation des véhicules avec les fenêtres de temps

Avec la prise en compte des fenêtres de temps, le regroupement des livraisons est toujours bénéfique pour l'entreprise A. La figure 3.13 représente le taux d'utilisation des véhicules par mois. Lorsque les émissions sont minimisées, l'amélioration des taux d'utilisation des véhicules varie de 32% à 43%. Lorsque

les coûts sont minimisés, l'amélioration des taux d'utilisation des véhicules varie de 24% à 30%.

La figure 3.14 représente les émissions de CO_2 par mois. Lorsque les émissions sont minimisées, l'amélioration varie de 29% à 35%. Lorsque les coûts sont minimisés, l'amélioration varie de 19% à 26%.

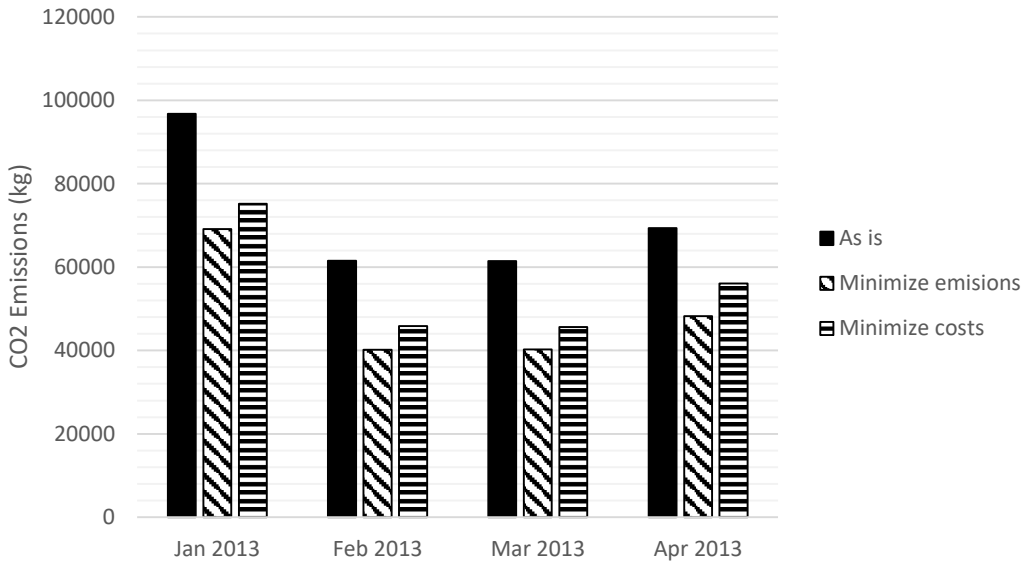

FIGURE 3.14 – Emissions de CO_2 par mois avec les fenêtres de temps

La figure 3.15 représente les coûts de transport et de pénalités par mois. Lorsque les émissions sont minimisées, l'amélioration varie de 11% à 28%. Lorsque les coûts sont minimisés, l'amélioration varie de 24% à 31%.

Ces résultats (sections 3 et 5) permettent de mettre en évidence les avantages que peut apporter la collaboration entre les clients de l'entreprise A. Effectuer ces simulations est important pour montrer aux clients de l'entreprise A les bénéfices d'une telle collaboration, puisque ceux-ci sont réticents à collaborer car ce sont des concurrents. En effet, les différentes raisons qui peuvent empêcher une collaboration entre ces clients sont les suivantes :

— Avoir des objectifs différents.

— Avoir des perceptions différentes de la collaboration.

— Le manque de confiance envers les autres collaborateurs.

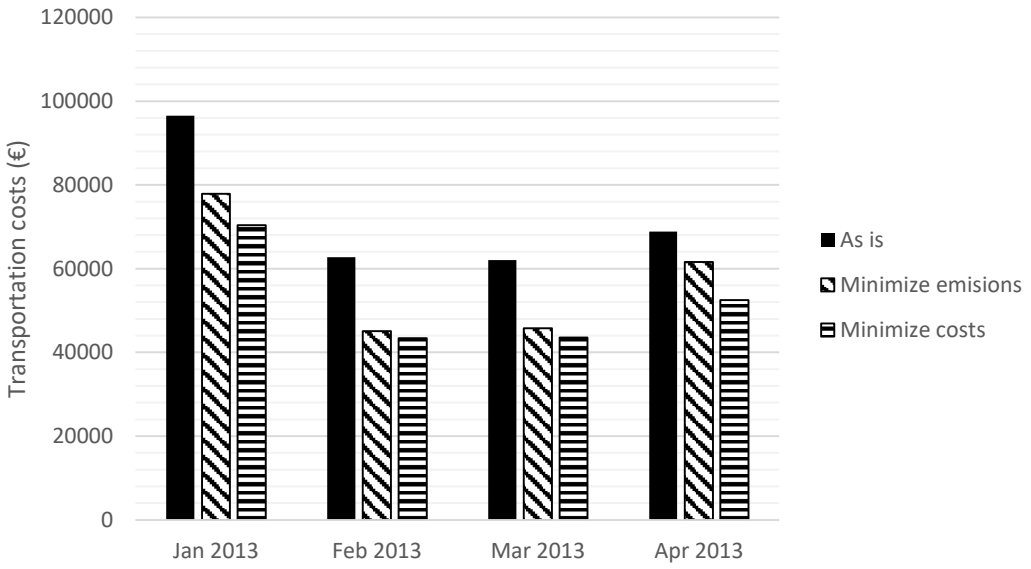

FIGURE 3.15 – Coûts de transport totaux et de pénalités par mois avec les fenêtres de temps

— Ne pas vouloir mélanger les marchandises de différents clients dans le même camion.

— Le manque de quantification des bénéfices apportés par la collaboration.

Les travaux réalisés dans ce chapitre permettent de répondre à ce dernier point et ont engendré des discussions entre l'entreprise A et ses clients pour permettre à cette entreprise d'améliorer son efficacité logistique au niveau du transport de marchandises.

3.6 Conclusion

Nous avons présenté dans ce chapitre un modèle pour résoudre le PDVRP sans et avec fenêtre de temps. Ce modèle est utilisé pour illustrer les bénéfices du transport collaboratif. Tout d'abord, la version sans fenêtre de temps du modèle a été appliquée à une étude de cas avec des données réelles provenant d'une entreprise, puis avec des données générées aléatoirement. Ensuite la version avec fenêtres de temps du modèle a été appliquée sur les données réelles de

l'entreprise uniquement.

Ce chapitre apporte une contribution dans l'élaboration du modèle pour le PDVRP. En effet, les articles traitant ce problème sont très peu nombreux et ne prennent pas en compte ni fenêtres de temps, ni certaines caractéristiques prises en compte dans ce chapitre. Il contribue également à la planification du transport collaboratif en démontrant l'apport que peut avoir le PDVRP que ce soit en termes de coûts ou en termes d'émissions de CO_2. En effet, nous avons montré dans ce chapitre que les coûts et les émissions globaux sont réduits lorsque les entreprises collaborent. En amont d'une mise en place de collaboration, l'application du PDVRP à la collaboration peut permettre aux entreprises d'avoir une quantification des gains potentiels et donc de faciliter la mise en place de cette collaboration. En aval, le PDVRP peut permettre de maintenir ces gains et les objectifs des entreprises afin de continuer la collaboration.

Chapitre 4

Un PDVRP avec transbordements

4.1 Introduction

Dans le chapitre précédent, nous avons vu que le PDVRP collaboratif pouvait permettre d'améliorer l'efficacité de la planification du transport. Dans ce chapitre, nous présentons un PDVRP particulier inspiré d'un cas réel qui a été réalisé dans le cadre du projet SCALE. Dans ce cas réel, les fournisseurs sont séparés des clients par la mer et plusieurs choix sont possibles pour la traverser. Les marchandises doivent donc transiter par un port parmi plusieurs ports du côté des fournisseurs et par un port parmi plusieurs ports du côté des clients. Le PDVRP est donc mis à jour pour prendre en compte ces contraintes et ces choix supplémentaires. De plus, contrairement au chapitre précédent, les fenêtres de temps prises en compte ici sont toutes des contraintes dures, il n'y a donc pas de coûts de pénalités en cas de non-respect de ces fenêtres. Les hypothèses sont les mêmes que celles du chapitre précédent. Dans ce chapitre, nous ajoutons également la possibilité de pouvoir transférer les marchandises d'un véhicule à un autre dans certains nœuds spécifiques (les nœuds de transbordement). Nous évaluons les bénéfices que peuvent en tirer les entreprises lors d'une collaboration sur les transports à réaliser.

La suite de ce chapitre est décomposée en deux sections. Dans la section 2, nous présentons le PDVRP avec transbordements à l'aide d'un exemple illustratif, puis nous proposons une formulation pour ce problème. Dans la section 3, ce

problème est appliqué sur les données réelles fournies par trois entreprises agroalimentaires.

4.2 Présentation du problème

Le problème présenté ici est un PDVRP avec fenêtres de temps et avec deux niveaux de transbordements. Ce problème est appliqué à la gestion du transport collaboratif dans le cadre du transport de charges partielles. Les partenaires impliqués dans la collaboration sont les fournisseurs et/ou les clients. L'objectif des partenaires de la collaboration et de grouper leurs cargaisons dans le but de proposer des chargements complets aux transporteurs. Ce problème peut être défini sur un graphe $G = O \cup USH \cup DSH \cup D, A$ où l'ensemble des nœuds est l'union de quatre ensembles : l'ensemble des fournisseurs ou des nœuds origine (O), l'ensemble des nœuds de transbordement amont (USH), l'ensemble des nœuds de transbordement aval (DSH), et l'ensemble des clients ou des nœuds destination (D). Chaque nœud appartient à un et un seul ensemble. Comme nous nous concentrons sur la collaboration entre fournisseurs et/ou clients, nous ne considérons pas les dépôts des transporteurs. L'ensemble des arcs est composé de cinq sous-ensembles : les arcs entre les fournisseurs, les arcs entre les fournisseurs et les nœuds de transbordement amont, les arcs entre les nœuds de transbordement amont et les nœuds de transbordement aval, les arcs entre les nœuds de transbordement aval et les fournisseurs, et les arcs entre les fournisseurs. Nous considérons dans ce problème une flotte finie hétérogène de véhicules avec une capacité limitée. Nous considérons également la possibilité d'avoir plusieurs types de produits. Un fournisseur peut fournir plusieurs types de produits et un type de produits peut être fourni par plusieurs fournisseurs. De même, un client peut requérir plusieurs types de produits et un type de produit peut être requis par plusieurs clients. Chaque nœud d'origine peut fournir une certaine quantité d'un type de produits. De même, chaque nœud de destination demande une certaine quantité d'un type de produits. La disponibilité totale d'un type de produits sur tous les fournisseurs est supposée strictement égale à la demande totale de ce type de produits. Les demandes peuvent être éclatées, i.e. un client peut être servi par plusieurs véhicules pour un même type de produits.

Finalement, nous considérons également les fenêtres de temps. Un véhicule doit servir chaque nœud dans une fenêtre de temps prédéfinie . S'il arrive avant, il peut attendre jusqu'à l'ouverture de la fenêtre, mais il ne peut pas arriver après la borne supérieure de cette fenêtre. Le temps de service est fixé pour chaque nœud.

Comme il n'y a pas de dépôt, chaque véhicule peut commencer ou terminer sa route sur n'importe quel nœud. Les contraintes de flux sont en fait appliquées sur la marchandise en transit. Chaque produit doit partir de son nœud d'origine via un véhicule. Ensuite il peut passer par plusieurs autres fournisseurs. Il doit passer par exactement un nœud de transbordement amont et un nœud de transbordement aval. Cette contrainte s'avère utile par exemple dans les cas où les fournisseurs sont séparés des clients par la mer. Finalement, il peut visiter plusieurs clients avant d'arriver à son point de destination. Le produit peut être transféré uniquement d'un véhicule à un autre dans les nœuds de transbordement amont et aval. Pour simplifier et parce que nous nous concentrons sur le transport, nous ne considérons pas de coûts additionnels de manutention dans les nœuds de transbordement, si transbordement il y a. La différence entre les nœuds de transbordement en amont et en aval est que les nœuds en amont sont situés près des fournisseurs et les nœuds en aval sont situés près des clients. La livraison directe d'un fournisseur vers un client sans passer par un nœud de transbordement amont et un nœud de transbordement aval n'est pas autorisée.

Le but du problème présenté dans ce chapitre est d'étudier les bénéfices des problèmes de tournées de véhicules collaboratifs avec transbordements en termes de réduction de coûts et de réduction d'émissions de CO_2. Dans cette section, nous illustrons tout d'abord le problème avec un exemple simple, puis nous introduisons les notations utilisées et enfin nous présentons une nouvelle formulation en nombre entiers mixtes pour le problème étudié.

4.2.1 Un exemple illustratif

Nous illustrons le problème avec un exemple simple. Dans notre exemple, il y a trois fournisseurs, cinq clients et trois types de produits. Les fournisseurs sont nommés A, B et C, les clients sont nommés D, E, F, G et H, le noeud de

transbordement amont est nommé I, le noeud de transbordement aval est noté J et les différents types de produits sont nommés a, b et c. Chaque fournisseur fournit un type de produits, le fournisseur A fournit le produit a, le fournisseur B fournit le produit b et le fournisseur C fournit le produit c. Les capacités des fournisseurs et les demandes des clients sont les suivantes :

— Le fournisseur A doit fournir 15 unités du produit a.

— Le fournisseur B doit fournir 10 unités du produit b.

— Le fournisseur C doit fournir 25 unités du produit c.

— Le client D requiert 4 unités du produit a et 13 unités du produit c.

— Le client E requiert 6 unités du produit a.

— Le client F requiert 2 unités du produit b.

— Le client G requiert 5 unités du produit a et 12 unités du produit c.

— Le client H requiert 8 unités du produit b.

FIGURE 4.1 – Position des nœuds dans le réseau de l'exemple illustratif

La figure 4.1 montre le réseau de cet exemple. Les nombres entre parenthèses sont les coordonnées $(x;y)$ des nœuds. Les distances utilisées sont les distances euclidiennes et sont résumées dans le tableau 1. La capacité des véhicules disponibles pour traiter cet exemple est de 25 unités. Pour simplifier, nous ne considérons pas les fenêtres de temps dans cet exemple, mais elles sont considérées dans le modèle. Aussi, dans cet exemple, nous comparons uniquement les distances, mais nous nous concentrons sur les coûts de transport et les

émissions de CO_2 dans le modèle proposé. Nous considérons trois scénarios dans cet exemple. Dans le premier scénario, chaque fournisseur optimise ses tournées indépendamment des autres fournisseurs. Ceci peut correspondre à dérouler un VRP ou un PDP classique pour chaque fournisseur. L'organisation des tournées dans ce scénario est présentée dans la figure 4.2. Le premier véhicule est représenté en traits continus. Il charge chez le fournisseur A pour livrer les clients E, D et G. Le deuxième véhicule est représenté en pointillés. Il charge chez le fournisseur B et livre les clients F et H. Le troisième véhicule est représenté en traits mixtes (points et tirets). Il charge chez le fournisseur C et livre les clients G et D. La distance totale de cette solution est de 464 km.

Distances	B	C	D	E	F	G	H	I	J
A	22,4	100	80	70,7	92,2	114	106,3	59,3	78,1
B		80,6	72,8	60,8	80	94,3	84,9	36,1	58,3
C			128,1	114	120,4	94,9	72,8	58,3	78,1
D				14,1	22,4	70,7	80,6	70,7	53,9
E					22,4	63,2	70	56,6	41,2
F						50	63,2	67,1	42,4
G							22,4	63,2	36,1
H								50	31,6
I									30

Tableau 4.1 – Distance entre les différents noeuds du réseau en kilomètres

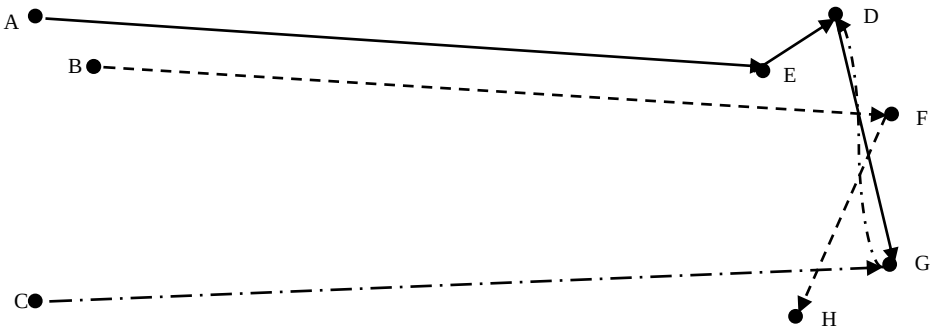

Figure 4.2 – Organisation des tournées sans collaboration

Dans le deuxième scénario, les trois fournisseurs collaborent pour optimiser leurs tournées. Ceci correspond à la majorité des travaux traitant de la planifica-

tion du transport collaboratif. L'organisation des tournées dans ce scénario est présentée dans la figure 4.3. Le premier véhicule est représenté en trait continu. Il charge chez les fournisseurs A et B et livre les clients E, D, F, G et H. Le second véhicule est représenté en pointillés. Il charge chez le fournisseur C et livre les clients G et D. Contrairement au premier scénario, seulement deux véhicules sont utilisés et la distance totale de cette solution est de 358 km, soit une réduction de 106 km par rapport au premier scénario.

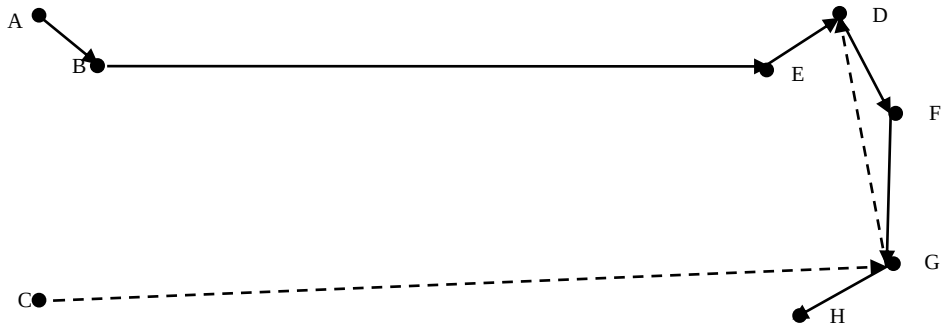

FIGURE 4.3 – Organisation des tournées avec collaboration

Dans le dernier scénario, les trois fournisseurs collaborent toujours, mais un nœud de transbordement est ajouté en amont (I) près des fournisseurs, et un nœud de transbordement est ajouté en aval (J) près des clients. Dans ces nœuds de transbordement, les produits peuvent être déplacés d'un véhicule à un autre. De plus, tous les produits doivent passer par le nœud de transbordement en amont puis par le nœud de transbordement en aval. Ce scénario correspond à notre modèle. L'organisation des tournées dans ce scénario est présentée dans la figure 4.4. Le premier véhicule est représenté en trait continu. Il charge chez les fournisseurs A et B et livre les clients E, D et F. Le second véhicule est représenté en pointillés. Il charge chez le fournisseur C et livre les clients H et G. Dans les nœuds de transbordement, 5 unités du produit a et 8 unités du produit b sont transférées du premier véhicule au second, et 13 unités du produits c sont transférées du second véhicule au premier. La distance totale de ce scénario est de 308 km. Ce scénario améliore la distance totale de 50 km par rapport au scénario 2, et de 156 km par rapport au scénario 1.

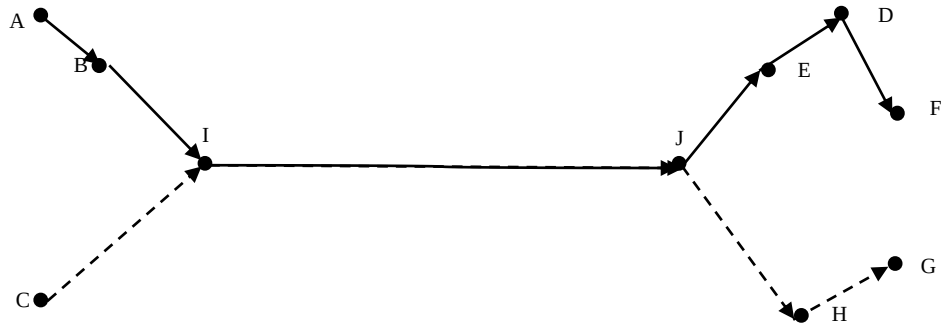

FIGURE 4.4 – Organisation des tournées avec collaboration et transbordements

Les avantages de l'ajout d'un ensemble de nœuds de transbordement en amont et en aval surviennent quand les fournisseurs sont situés proches les uns des autres mais éloignés des clients. Les avantages sont d'autant plus importants lorsque plusieurs types de véhicules sont disponibles. Par exemple, des petits véhicules peuvent visiter les fournisseurs pour charger les marchandises et ensuite les consolider dans un nœud de transbordement en amont. A ce niveau, des véhicules plus grands peuvent être utilisés pour transporter les marchandises entre les nœuds de transbordement amont et aval où les marchandises sont alors déconsolidées. Enfin, de nouveau des petits véhicules peuvent effectuer les livraisons chez les clients. Les nœuds de transbordements peuvent aussi jouer le rôle de plateformes hubs multimodales, où les marchandises sont transférées des camions à d'autres modes de transport comme le train ou l'avion dans les nœuds en amont, et du train ou de l'avion vers des camions dans les nœuds en aval. Dans notre étude de cas SCALE, les fournisseurs et les clients sont séparés par la mer, et les marchandises doivent passer par des ports qui peuvent aussi jouer le rôle de nœuds de transbordement.

4.2.2 Notations

Dans ce chapitre nous utilisons les notations suivantes :

O Ensemble des fournisseurs

D Ensemble des clients

USH Ensemble des nœuds de transbordement en amont

DSH Ensemble des nœuds de transbordement en aval

V Ensemble des véhicules

K Ensemble des types de produits

E^v_{Empty} Emissions du véhicule v à vide ($kgCO_2/km$)

E^v_{Pallet} Emissions par palette ajoutée dans le véhicule v ($kgCO_2/km$)

C^v_{KM} Coût du véhicule v par km (€)

C^v_{Hour} Coût du véhicule v par heure (€)

C^v_{Fixed} Coût fixe du véhicule v (€)

C^v Capacité du véhicule v

M Une grande constante

d_{ij} Distance entre i et j (km)

t_{ij} Temps de trajet entre i et j (h)

q^+_{ik} Capacité du fournisseur i pour le produit de type k

q^-_{ik} Quantité de produit k que le client i demande

e_i Borne inférieure de la fenêtre de temps du nœud i

l_i Borne supérieure de la fenêtre de temps du nœud i

u^v_i Temps de service du véhicule v au nœud i

α Poids des émissions de CO_2 dans la fonction objectif

β Poids des coûts dans la fonction objectif

Les variables utilisées sont définies comme suit :

x^v_{ij} Vaut 1 si le véhicule v traverse l'arc ij

y^{vk}_{ij} Quantité de produits de type k dans le véhicule v entre les nœuds i et j

a^v_i Début de temps de service du véhicule v au nœud i

b^v_i Fin de temps de service du véhicule v au nœud i

w^v_i Temps d'attente du véhicule v au nœud i

$z^{vv'}_{ik}$ Quantité de produits k transférés du véhicule v au véhicule v' dans le nœud de transbordement i

s_{ik}^{v} Vaut 1 si le véhicule v ramasse des produits k dans le nœud de transbordement i

$r_{i}^{vv'}$ Vaut 1 s'il y a un transfert du véhicule v au véhicule v' dans le nœud de transbordement i

4.2.3 Modélisation

$$\text{Min} \sum_{i\in O\cup D}\sum_{j\in O\cup D}\sum_{v\in V}\left(E_{Empty}^{v}d_{ij}x_{ij}^{v}+\sum_{k\in K}E_{Pallet}^{v}d_{ij}x_{ij}^{vk}\right) \tag{4.1}$$

$$\text{Min} \sum_{i\in O\cup D}\sum_{j\in O\cup D}\sum_{v\in V}\left(C_{KM}^{v}d_{ij}x_{ij}^{v}+C_{Hour}^{v}t_{ij}x_{ij}^{v}\right)+\sum_{v\in V}C_{Fixed}^{v}\left(\sum_{i\in O}\sum_{j\in USH}x_{ij}^{v}\right) \tag{4.2}$$

Ce modèle est composé de deux fonctions objectives. La première fonction objectif 3.44 du modèle est de minimiser les émissions de CO_2. Cette fonction objectif de deux termes. Le premier terme correspond aux émissions de tous les véhicules à vide, et le second terme correspond aux émissions additionnelles dues aux chargements des véhicules. La seconde fonction objective 3.24 correspond aux coûts et est composé de trois termes. Le premier terme correspond aux coûts par kilomètres, le deuxième terme correspond aux coûts par heures et le troisième terme correspond aux coûts fixes.

$$\begin{aligned}\text{Min } \alpha&\left(\sum_{i\in O\cup D}\sum_{j\in O\cup D}\sum_{v\in V}\left(E_{Empty}^{v}d_{ij}x_{ij}^{v}+\sum_{k\in K}E_{Pallet}^{v}d_{ij}x_{ij}^{vk}\right)\right)\\&+\beta\left(\sum_{i\in O\cup D}\sum_{j\in O\cup D}\sum_{v\in V}\left(C_{KM}^{v}d_{ij}x_{ij}^{v}+C_{Hour}^{v}t_{ij}x_{ij}^{v}\right)+\sum_{v\in V}C_{Fixed}^{v}\left(\sum_{i\in O}\sum_{j\in USH}x_{ij}^{v}\right)\right)\end{aligned} \tag{4.3}$$

Pour résoudre le problème, les émissions et les coûts sont agrégés dans une seule fonction objectif avec chacun un poids (α et β) compris entre 0 et 1 pour que nous puissions donner plus d'importance aux émissions ou aux coûts. Cette fonction agrégée est définie par la fonction objectif 4.3. Les contraintes du

problème sont les suivantes :

$$\sum_{v \in V} \left(\sum_{j \in O} y_{ij}^{vk} + \sum_{j \in USH} y_{ij}^{vk} - \sum_{j \in O} y_{ji}^{vk} \right) = q_{ik}^{+} \qquad \forall i \in O, \forall k \in K, \tag{4.4}$$

$$\sum_{v \in V} \left(\sum_{i \in DSH} y_{ij}^{vk} + \sum_{i \in D} y_{ij}^{vk} - \sum_{i \in D} y_{ji}^{vk} \right) = q_{jk}^{-} \qquad \forall j \in D, \forall k \in K, \tag{4.5}$$

La contrainte 4.4 assure que la capacité d'un type de produits d'un fournisseur est respectée. De la même manière, la contrainte 4.5 assure que la demande d'un client pour un type de produits est respectée. Dans la contrainte 4.4, le premier et le deuxième terme correspondent à la quantité de produits de type k quittant le fournisseur i, et le troisième terme correspond à la quantité de produits de type k déjà présents dans le véhicule en arrivant chez le fournisseur i. Dans la contrainte 4.5, le premier et le deuxième terme correspondent à la quantité de de produits de type k présents dans le véhicule quand il arrive chez le client j, et le troisième terme correspond à la quantité de produits de type k encore présents dans le véhicule lorsque celui-ci a quitté le client j.

$$\sum_{j \in O} y_{ij}^{vk} + \sum_{j \in USH} y_{ij}^{vk} \geq \sum_{j \in O} y_{ji}^{vk} \qquad \forall i \in O, \forall v \in V, \forall k \in K, \tag{4.6}$$

$$\sum_{i \in DSH} y_{ij}^{vk} + \sum_{i \in D} y_{ij}^{vk} \geq \sum_{i \in D} y_{ji}^{vk} \qquad \forall j \in D, \forall v \in V, \forall k \in K, \tag{4.7}$$

Les contraintes 4.6 et 4.7 respectent le fait qu'un véhicule doit être chargé chez un fournisseur et déchargé chez un client. En effet, quand il quitte un fournisseur, la quantité de produits dans le véhicule doit être supérieure ou égale à celle quand il arrive. Cette différence correspond à la quantité chargée dans le véhicule (4.6). De la même manière, quand il quitte un client, la quantité de produits présents dans un véhicule doit être inférieure ou égale à celle quand il arrive. Cette différence correspond à la quantité déchargée du véhicule (4.7).

$$\sum_{j\in O}\sum_{v\in V} y_{ji}^{vk} = \sum_{j\in DSH}\sum_{v\in V} y_{ij}^{vk} \qquad \forall i \in USH, \forall k \in K, \tag{4.8}$$

$$\sum_{j\in USH}\sum_{v\in V} y_{ji}^{vk} = \sum_{j\in D}\sum_{v\in V} y_{ij}^{vk} \qquad \forall i \in DSH, \forall k \in K, \tag{4.9}$$

Les contraintes 4.8 et 4.9 assurent que la quantité de produits quittant un nœud de transbordement est égale à la quantité de produits qui est entrée dans ce nœud. Ces contraintes autorisent le transfert de marchandises d'un véhicule à un autre dans les nœuds de transbordements.

$$\sum_{j\in O} x_{ij}^{v} + \sum_{j\in USH} x_{ij}^{v} \leq 1 \qquad \forall i \in O, \forall v \in V, \tag{4.10}$$

$$\sum_{j\in D} x_{ij}^{v} \leq 1 \qquad \forall i \in D, \forall v \in V, \tag{4.11}$$

$$\sum_{j\in O} x_{ji}^{v} \leq 1 \qquad \forall i \in O, \forall v \in V, \tag{4.12}$$

$$\sum_{i\in DSH} x_{ij}^{v} + \sum_{i\in D} x_{ij}^{v} \leq 1 \qquad \forall j \in D, \forall v \in V, \tag{4.13}$$

$$\sum_{i\in O}\sum_{j\in USH} x_{ij}^{v} \leq 1 \qquad \forall v \in V, \tag{4.14}$$

$$\sum_{i\in USH}\sum_{j\in DSH} x_{ij}^{v} \leq 1 \qquad \forall v \in V, \tag{4.15}$$

$$\sum_{i\in DSH}\sum_{j\in D} x_{ij}^{v} \leq 1 \qquad \forall v \in V, \tag{4.16}$$

Les contraintes 4.10 à 4.16 assurent qu'un véhicule passe dans un nœud une fois au maximum. La contrainte 4.10 (respectivement 4.11) assure qu'un véhicule ne se dirige que vers un seul nœud au maximum lorsqu'il quitte un fournisseur (respectivement un client). De la même manière, la contrainte 4.12 (respectivement 4.13) assure qu'un véhicule vient d'une seule direction au maximum en arrivant chez un fournisseur (respectivement un client). Les contraintes 4.14 à 4.16 assurent qu'un véhicule passe par un nœud de transbordement en

amont et un nœud de transbordement en aval au maximum.

$$C^v x_{ij}^v \geq \sum_{k \in K} y_{ij}^{vk} \qquad \forall i, j \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.17}$$

La contrainte 4.17 est la contrainte de capacité des véhicules. La quantité de produits transportés dans un véhicule sur un arc donné ne doit pas dépasser la capacité du véhicule.

$$b_i^v + t_{ij} + w_j^v - a_j^v \leq M\left(1 - x_{ij}^v\right) \qquad \forall i, j \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.18}$$
$$b_i^v + t_{ij} + w_j^v - a_j^v \geq -M\left(1 - x_{ij}^v\right) \qquad \forall i, j \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.19}$$
$$a_i^v + u_i^v = b_i^v \qquad \forall i \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.20}$$
$$a_i^v \geq e_i \qquad \forall i \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.21}$$
$$a_i^v \leq l_i \qquad \forall i \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.22}$$

Les contraintes 4.18 à 4.22 sont les contraintes relatives aux fenêtres de temps. Les contraintes 4.18 à 4.20 assurent le respect des temps de trajet et de chargement/déchargement. Les contraintes 4.18 et 4.19 assurent que si un véhicule utilise l'arc ij, le début du temps de service à j du véhicule est égal à la fin de son temps de service à i plus le le temps de trajet entre i et j plus le temps d'attente à j. La contrainte 4.20 assure que l'heure de fin de service d'un véhicule au nœud i est égale à l'heure de début de service plus le temps de service. Les contraintes 4.21 et 4.22 assurent que le début de service d'un véhicule dans un nœud i est comprise dans la fenêtre de temps de i.

$$\sum_{i \in O} y_{ij}^{vk} - \sum_{i \in DSH} y_{ji}^{vk} \leq \sum_{v' \in V, v' \neq v} z_{jk}^{vv'} \qquad \forall j \in USH, \forall v \in V, \forall k \in K, \tag{4.23}$$
$$\sum_{i \in USH} y_{ij}^{vk} - \sum_{i \in D} y_{ji}^{vk} \leq \sum_{v' \in V, v' \neq v} z_{jk}^{vv'} \qquad \forall j \in DSH, \forall v \in V, \forall k \in K, \tag{4.24}$$

$$z_{jk}^{vv'} + \sum_{i \in DSH} y_{ji}^{vk} - \sum_{i \in O} y_{ij}^{vk} \leq M\left(1 - s_{jk}^{v'}\right) \quad \forall j \in USH, \forall v, v' \in V, v \neq v', \forall k \in K, \tag{4.25}$$

$$z_{jk}^{vv'} + \sum_{i \in D} y_{ji}^{vk} - \sum_{i \in USH} y_{ij}^{vk} \leq M\left(1 - s_{jk}^{v'}\right) \quad \forall j \in DSH, \forall v, v' \in V, v \neq v', \forall k \in K, \tag{4.26}$$

$$z_{ik}^{vv'} \leq M s_{ik}^{v'} \quad \forall i \in USH \cup DSH, \forall v, v' \in V, v \neq v', \forall k \in K, \tag{4.27}$$

$$\sum_{i \in DSH} y_{ji}^{vk} - \sum_{i \in O} y_{ij}^{vk} \leq M s_{jk}^{v'} \quad \forall j \in USH, \forall v, v' \in V, v \neq v', \forall k \in K, \tag{4.28}$$

$$\sum_{i \in D} y_{ji}^{vk} - \sum_{i \in USH} y_{ij}^{vk} \leq M s_{jk}^{v'} \quad \forall j \in DSH, \forall v, v' \in V, v \neq v', \forall k \in K, \tag{4.29}$$

$$z_{ik}^{vv'} \leq M r_{i}^{vv'} \quad \forall i \in USH \cup DSH, \forall v, v' \in V, v \neq v', \forall k \in K, \tag{4.30}$$

$$a_i^v - b_i^{v'} \leq M\left(1 - r_i^{vv'}\right) \quad \forall i \in USH \cup DSH, \forall v, v' \in V, v \neq v', \tag{4.31}$$

Les contraintes 4.23 à 4.31 sont les contraintes relatives aux transbordements. Les contraintes 4.23 4.29 permettent de calculer la quantité de produits transférés d'un véhicule à un autre véhicule dans un nœud de transbordement. Les contraintes 4.23 et 4.24 permettent de calculer la quantité minimum de produits qu'un véhicule doit déposer dans un nœud de transbordement. Les contraintes 4.26 et 4.26 permettent de calculer la quantité maximale de produits qui peuvent être transférés d'un véhicule à un autre dans un nœud de transbordement. La contrainte 4.27 assure que si un véhicule ne récupère pas de produits dans un nœud de transbordement, la quantité de produits transférés dans ce véhicule est nulle. Les contraintes 4.28 et 4.29 assurent que si la quantité de produits dans un véhicule quittant un nœud de transbordement est plus grande que lorsqu'il est arrivé dans ce nœud, alors le véhicule doit récupérer des produits d'un autre véhicule. Les contraintes 4.30 et 4.31 assurent que s'il y a un transfert d'un véhicule v vers un véhicule v' dans un nœud de transbordement, le véhicule v doit arriver dans ce nœud avant que le véhicule v' ne le quitte.

$$x_{ij}^{v} \in \{0;1\} \qquad \forall i \in O, \forall j \in O \cup USH, \forall v \in V, \tag{4.32}$$

$$x_{ij}^{v} \in \{0;1\} \qquad \forall i \in USH, \forall j \in DSH, \forall v \in V, \tag{4.33}$$

$$x_{ij}^{v} \in \{0;1\} \qquad \forall i \in DSH \cup D, \forall j \in D, \forall v \in V, \tag{4.34}$$

$$y_{ij}^{vk} \geq 0 \qquad \forall i, j \in O \cup USH \cup DSH \cup D, \forall v \in V, \forall k \in K, \tag{4.35}$$

$$a_i^v, b_i^v, w_i^v \geq 0 \qquad \forall i, j \in O \cup USH \cup DSH \cup D, \forall v \in V, \tag{4.36}$$

$$z_{ik}^{vv'} \geq 0 \qquad \forall i \in USH \cup DSH, \forall k \in K, \forall v, v' \in V, v \neq v', \tag{4.37}$$

$$s_{ik}^{v} \in \{0;1\} \qquad \forall i \in USH \cup DSH, \forall k \in K, \tag{4.38}$$

$$r_{i}^{vv'} \in \{0;1\} \qquad \forall i \in USH \cup DSH, \forall v, v' \in V, v \neq v', \tag{4.39}$$

Les contraintes 4.32 à 4.2 représentent l'aspect binaire ou positif des variables de décisions. Les contraintes 4.32 à 4.34 forcent les véhicules à passer à travers les nœuds de transbordement en amont et en aval de par l'absence d'arcs entre les fournisseurs (O) et les clients (D).

4.3 Etude de cas

Dans cette étude de cas, nous utilisons notre modèle pour optimiser les coûts et les émissions totales. Nous simulons différents scénarios pour calculer les émissions de CO_2, les coûts de transport et les distances parcourues par la distribution de produits agroalimentaire, avec une collaboration entre fournisseurs et transbordements possibles.

4.3.1 Données du réseau

L'étude de cas porte sur la collaboration au niveau du transport entre trois entreprises agroalimentaires (nommées ici A, B et C). Ces entreprises sont toutes les trois basées au Royaume-Uni et possèdent des clients à travers le nord-ouest de l'Europe continentale. De ce fait les marchandises doivent traverser la mer pour aller des fournisseurs au Royaume-Uni vers les clients dans le reste de l'Europe. Nous avons identifié quatre ports au Royaume-Uni qui jouent le rôle

de nœuds de transbordement en amont, et quatre ports en Europe continentale qui jouent le rôle de nœuds de transbordement en aval. Les trois fournisseurs ont au total 72 clients répartis principalement en France, en Belgique, en Allemagne, aux Pays-Bas et au Danemark. Dans le scénario actuel, ou scénario "as is", ces trois fournisseurs envoient leurs produits séparément à chacun de leur client (figure 4.5) en utilisant un véhicule par client, même si le véhicule n'est pas totalement rempli. Les traits continus représentent les véhicules partant du fournisseur A, les traits en pointillés représentent les véhicules partant du fournisseur B, et les traits mixtes (points et tirets) représentent les véhicules partant du fournisseur C. Toutes les connexions entre les ports britanniques et les ports présents sur le continent ne sont pas autorisées. Par simplification, tous les ports et tous les clients ne sont pas représentés dans les figures.

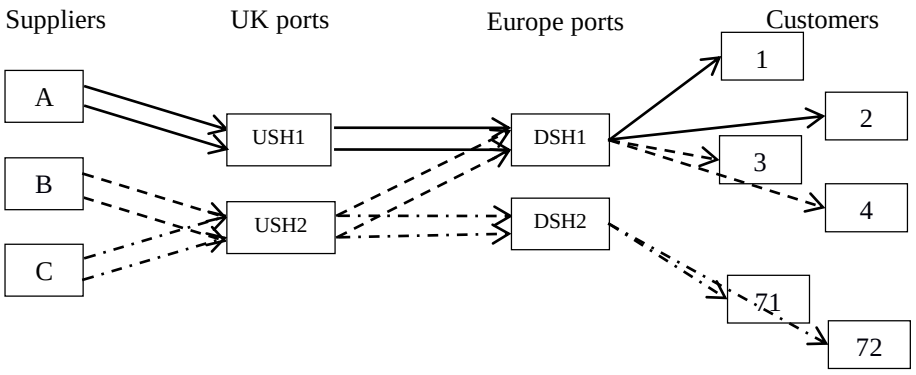

Figure 4.5 – Le réseau de distribution étudié

Les données collectées sur les demandes des clients varient selon le fournisseur. La période étudiée va de janvier 2014 à juin 2014. Les fichiers de données contiennent les informations suivantes :

— Le nom du fournisseur.

— La date de ramassage et le lieu de ramassage.

— La date de livraison et le lieu de livraison.

— Le nombre de palettes requises par chaque client.

Nous utilisons les données supplémentaires suivantes dans notre modèle :

— Les distances réelles entre chaque site sont fournies, ainsi que les temps de trajet.

— La liste des ports au Royaume-Uni et en Europe continentale est fournie ainsi que les connexions autorisées entre ces ports.

— Nous considérons une capacité de 24 palettes pour tous les véhicules.

— Le coefficient d'émissions par kilomètre pris en compte pour un véhicule vide est de 0.767 kgCO_2/km.

— Le coefficient d'émissions par kilomètre pris en compte pour chaque palette ajoutée dans un véhicule est de 14 gCCO_2/km.

— Le coût par kilomètre par véhicule est de 0.525 €/km.

— Le coût par heure par véhicule est de 21.67 €/h.

— Le coût fixe par véhicule utilisé est de 156.81€.

— Les bateaux utilisés pour aller d'un port à un autre sont des navires rouliers (bateaux dans lesquelles les véhicules entrent directement en roulant grâce à des rampes).

— Le coefficient d'émissions par kilomètre par palette dans un navire roulier est de 14.4 gCO_2/km.

— Pour simplifier, les coûts sont calculés de la même manière sur tout le réseau, que ce soit sur route ou sur mer.

4.3.2 Etude expérimentale

Le but de cette étude de cas est de comparer plusieurs scénarios de collaboration en termes de coûts de transport et d'émissions de CO_2. Le scénario 0 est le cas d'origine décrit dans la figure 4.5. Dans le scénario 1 (figure 4.2), les trois fournisseurs réalisent leurs livraisons indépendamment les uns des autres. Dans le scénario 2 (figure 4.3), les trois fournisseurs collaborent pour réaliser leurs livraisons en même temps. Dans ces deux scénarios, le transfert de marchandises dans les ports n'est pas autorisé. Pour ce faire, nous remplaçons les contraintes 4.8 et 4.9 par les contraintes 4.40 et 4.41.

$$\sum_{j \in O} y_{ji}^{vk} = \sum_{j \in DSH} y_{ij}^{vk} \qquad \forall i \in USH, \forall v \in V, \forall k \in K, \tag{4.40}$$

$$\sum_{j \in USH} y_{ji}^{vk} = \sum_{j \in D} y_{ij}^{vk} \qquad \forall i \in DSH, \forall v \in V, \forall k \in K, \tag{4.41}$$

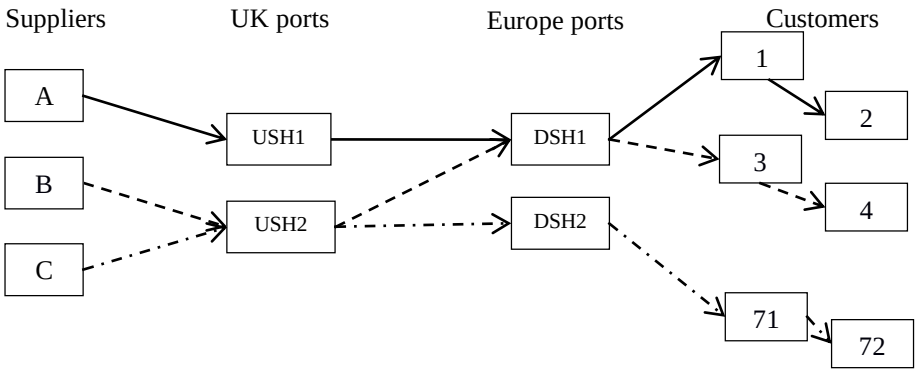

Figure 4.6 – Illustration du scénario 1

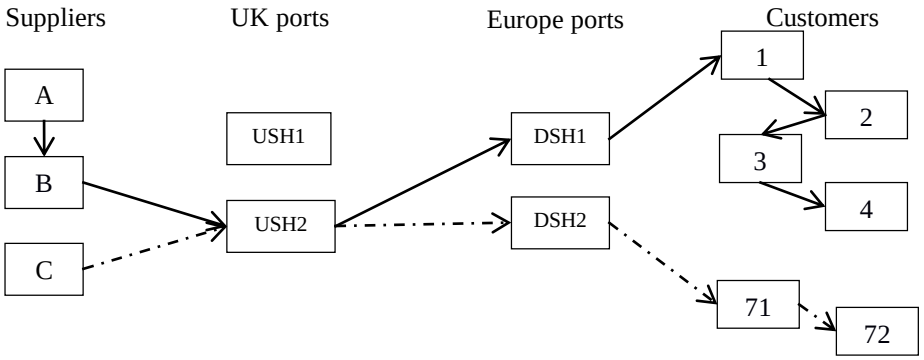

Figure 4.7 – Illustration du scénario 2

Dans le scénario 3 (figure 4.4), les trois fournisseurs collaborent toujours pour réaliser leurs livraisons en commun, mais contrairement aux scénarios précédents, les ports sont utilisés en tant que nœuds de transbordement, ce qui signifie que les marchandises peuvent être transférées d'un véhicule à l'autre.

Dans les figures 4.2, 4.4 et 4.4, les traits pleins, en pointillés et mixtes (points et tirets) représentent trois véhicules différents. Pour résumer, les scénarios 0 et 1 sont des scénarios sans collaboration. Les scénarios 3 et 4 sont des scénarios avec collaboration entre les fournisseurs. Le scénario 3 est le seul scénario où les transferts entre véhicules sont autorisés dans les ports.

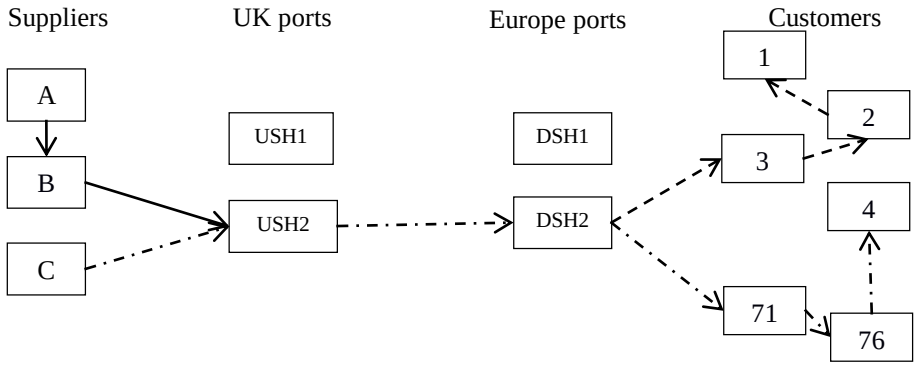

FIGURE 4.8 – Illustration du scénario 3

Les trois scénarios 1, 2 et 3 sont exécutés avec différentes valeurs de α et de β. Pour différencier ces trois différentes variantes, une lettre est accolée au numéro de chaque scénario. Ainsi, dans les scénarios 1a, 2a et 3a, la valeur de α est fixée à 1 et celle de β à 0. Ceci correspond à la minimisation des émissions uniquement. Dans les scénarios 1b, 2b et 3b, la valeur de α et de β à 0.5. Cette variante correspond à un compromis sur la minimisation des émissions et des coûts simultanément. Finalement, dans les scénarios 1c, 2c et 3c, la valeur de α est fixée à 0 et celle de β à 1. Ceci correspond à la minimisation des coûts uniquement. Les différences entre les scénarios sont résumées dans le tableau 4.2.

Nous utilisons le modèle présenté plus haut pour résoudre cette étude de cas. Le modèle est résolu avec le solver CPLEX 12.5 sur un PC disposant d'un processeur core i7 2,7GHz, de 8Go de RAM et tournant sous Windows 8.1. Ce problème est NP-difficile et peut être long à résoudre. Nous avons donc fixé une limite d'une heure pour le résoudre. Le gap entre la meilleure solution trouvée par CPLEX et la borne inférieure varie entre 0% et 4.5%. Le gap moyen est de 1.5%. La simulation est réalisée jour par jour pour la période de temps concernée

	$\alpha = 1, \beta = 0$	$\alpha = 0,5, \beta = 0,5$	$\alpha = 0, \beta = 1$
Sans collaboration Sans transbordement	1a	1b	1c
Avec collaboration Sans transbordement	2a	2b	2c
Avec collaboration Avec transbordement	3a	3b	3c

Tableau 4.2 – Résumé des caractéristiques des scénarios

(entre juin 2014 et novembre 2014). Nous exécutons le modèle une fois pour chaque jour. Durant une journée, l'ensemble des 72 clients n'ont pas à être servis car tous les clients ne font pas de commande en même temps. En moyenne, 10 clients sont servis durant une journée. Après avoir fait tourner le modèle, des tournées respectant les scénarios sont créées afin d'optimiser la fonction objectif pour une journée spécifique. Les résultats de chaque jour sont agrégés sur un mois afin d'obtenir des résultats mois par mois. Les résultats sont toujours des résultats totaux pour les trois fournisseurs pour que nous puissions comparer les scénarios sans collaboration avec les scénarios avec collaboration.

4.3.3 Résultats et discussions

Pour comparer les émissions (respectivement les coûts) de deux scénarios, nous utilisons le gain qui est le pourcentage de différence entre les émissions (respectivement les coûts) de deux scénarios et qui est égal à $100(S1 - S2)/S1$ où $S1$ sont les émissions (respectivement les coûts) du premier scénario à comparer, et $S2$ sont les émissions (respectivement les coûts) du second scénario à comparer. La figure 4.9 présente pour chaque scénario les émissions totales de CO_2 produites par les trois fournisseurs pour chaque mois étudié. Sans collaboration entre les entreprises, le meilleur scénario est le scénario 1a. Lorsque nous comparons le scénario 0 avec le scénario 1, la réduction moyenne des émissions pour les six mois est de 16.4% pour la variante à (minimiser les émissions uniquement), 13.1% pour la variante b (minimiser à la fois les coûts et les émissions), et 7.5% pour la variante c (minimiser les coûts uniquement). Ces résultats sont justifiés car il paraît logique que visiter plusieurs clients avec un

seul véhicule est un bon moyen pour réduire les émissions plutôt que de visiter chaque client avec un véhicule différent (si la capacité du véhicule le permet). Lorsque nous comparons le scénario 1 avec le scénario 2, la réduction moyenne des émissions est de 9% pour la variante a, 10.3% pour la variante b, et 9.6% pour la variante c. Lorsque nous comparons le scénario 1 avec le scénario 3, la réduction moyenne des émissions est de 11.8% pour la variante a, 11.3% pour la variante b, et 11.2% pour la variante c. Nous pouvons remarquer que la collaboration permet d'améliorer l'efficacité de la planification du transport en termes d'émissions de CO_2. Lorsque nous comparons les scénarios 2 et 3, la réduction moyenne des émissions est de 3.1% pour la variante a, 1.1% pour la variante b, et 1.8% pour la variante c. Alors que la collaboration sans transbordement améliore déjà l'efficacité de la planification du transport, cette efficacité peut être davantage améliorée en autorisant le transbordement. Le tableau 4.3 résume les comparaisons des émissions entre les différents scénarios.

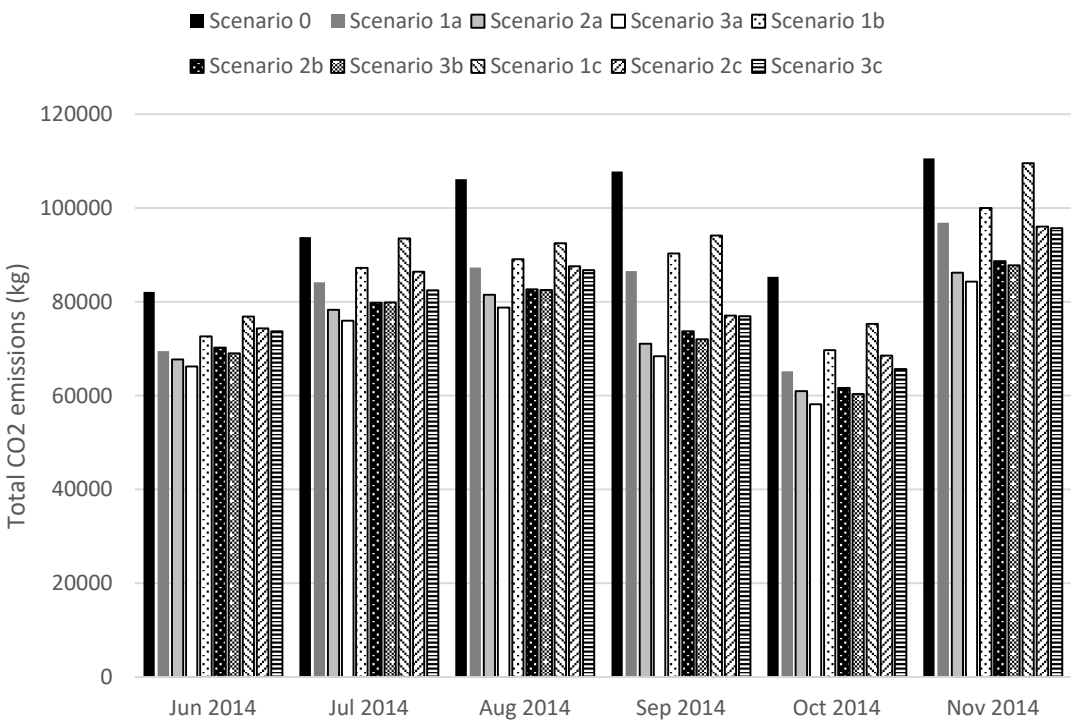

FIGURE 4.9 – Emissions totales de CO_2 par mois pour les trois fournisseurs

Gain (%)	Scen 1a	Scen 2a	Scen 3a	Scen 1b	Scen 2b	Scen 3b	Scen 1c	Scen 2c	Scen 3c
Scen 0	16.4	23.9	26.3	13.1	22.0	22.9	7.5	16.4	17.9
Scen 1a		9.0	11.8	-3.9	6.7	7.8	-10.7	-0.1	1.7
Scen 2a			3.1	-14.2	-2.5	-1.3	-21.6	-9.9	-8.0
Scen 3a				-17.9	-5.8	-4.6	-25.5	-13.5	-11.4
Scen 1b					10.3	11.3	-6.5	3.7	5.5
Scen 2b						1.1	-18.6	-7.3	-5.4
Scen 3b							-20.0	-8.5	-6.6
Scen 1c								9.6	11.2
Scen 2c									1.8

Tableau 4.3 – Comparaison des émissions de CO_2 entre les différents scénarios

La figure 4.10 présente pour chaque scénario les coûts de transport totaux des trois fournisseurs par mois. Le scénario 1c est le meilleur scénario sans collaboration. La réduction moyenne des coûts de transport est de 22.6% entre le scénario 0 et le scénario 1a, de 30.1% entre le scénario 0 et le scénario 1b, et 31.9% entre le scénario 0 et le scénario 1c. La collaboration est aussi une bonne pratique pour réduire les coûts de transport. La réduction moyenne est de 10.6% entre le scénario 1a et le scénario 2a, 9.6% entre le scénario 1b et le scénario 2b, et 8.7% entre le scénario 1c et le scénario 2c. Le transbordement peut également réduire les coûts de transport, en effet la réduction moyenne est de 6.5% entre le scénario 2a et le scénario 3a, 2.6% entre le scénario 2b et le scénario 2c, et 3.6% entre le scénario 2c et le scénario 3c. Le tableau 4.4 résume les comparaisons des coûts de transport entre les différents scénarios.

Ces résultats peuvent être expliqués par le fait que la collaboration améliore le taux d'utilisation des véhicules. En effet, en consolidant les marchandises venant de plusieurs fournisseurs, moins de véhicules sont utilisés pour transporter ces marchandises et la distance totale parcourue par l'ensemble des véhicules est aussi réduite. Comme les coûts de transport et les émissions sont fortement reliés à la distance, en réduisant la distance, les coûts et les émissions sont aussi réduits.

La figure 4.11 représente les coûts totaux en fonction des émissions pour les trois scénarios. Pour tous les scénarios, le point à gauche correspond aux

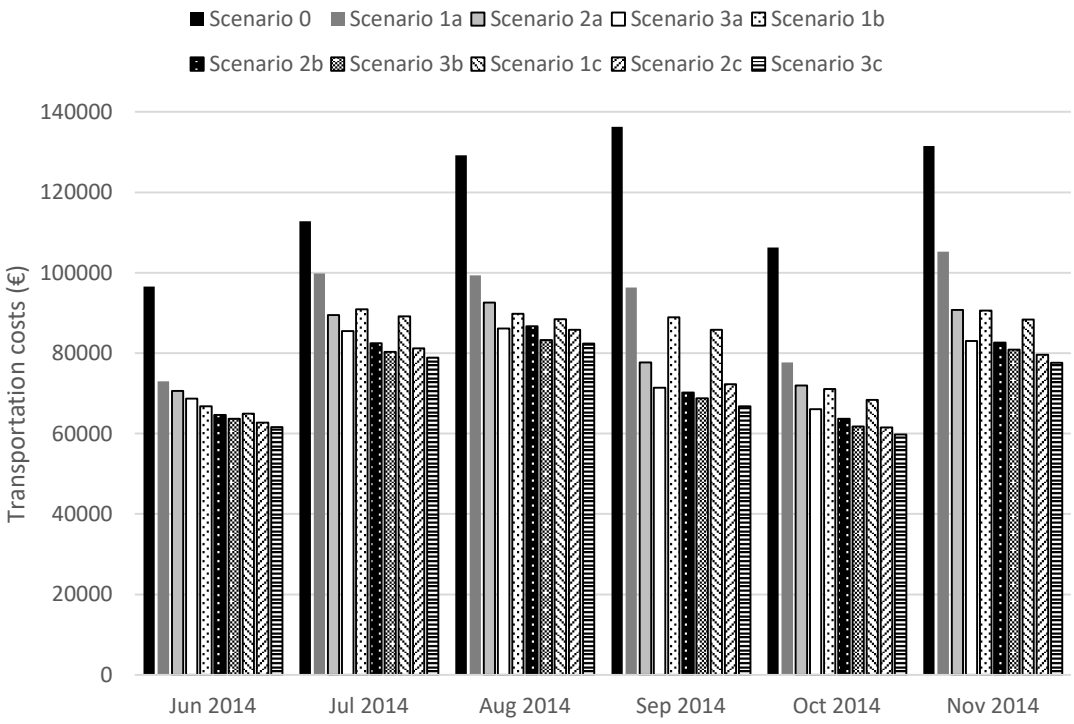

Figure 4.10 – Coûts totaux par mois pour les trois fournisseurs

variantes a des scénarios, le point du milieu correspond aux variantes b des scénarios et le point à droite correspond aux variantes c des scénarios. Dans cette figure, nous pouvons remarquer que le scénario trois est toujours le plus avantageux. Nous pouvons également remarquer que dans les variantes c, les coûts sont légèrement plus petits que dans les variantes a mais les émissions sont légèrement plus élevées. Ceci est dû au mode de calcul des émissions et des coûts. En effet, contrairement aux émissions, les coûts ne se calculent pas uniquement en fonction de la distance mais également en fonction du temps de trajet. De plus, les véhicules ont aussi un coût fixe. Donc lorsque nous minimisons les coûts, moins de véhicules ont tendance à être utilisés, mais la distance totale de tous les véhicules peut être légèrement supérieure, ce qui fait que les émissions sont également légèrement supérieures.

Une fois que ces simulations ont été conduites, des discussions ont eu lieu avec les trois entreprises. En voyant les résultats, les trois équipes opérationnelles

Gain (%)	Scen 1a	Scen 2a	Scen 3a	Scen 1b	Scen 2b	Scen 3b	Scen 1c	Scen 2c	Scen 3c
Scen 0	22.6	30.8	35.3	30.1	36.8	38.4	31.9	37.8	40.1
Scen 1a		10.6	16.4	9.7	18.3	20.4	12.0	19.6	22.5
Scen 2a			6.5	-1.0	8.7	11.0	1.6	10.1	13.4
Scen 3a				-8.1	2.3	4.8	-5.3	3.9	7.3
Scen 1b					9.6	11.9	2.6	11.0	14.3
Scen 2b						2.6	-7.7	1.6	5.2
Scen 3b							-10.6	-1.0	2.7
Scen 1c								8.7	12.0
Scen 2c									3.6

Tableau 4.4 – Comparaison des coûts de transport entre les différents scénarios

des entreprises ont songé à consolider une partie de leurs chargements allant du Royaume-Uni vers l'Europe continentale. Les équipes en charge de l'exécution opérationnelle ont trouvé cette nouvelle façon de considérer la logistique à la fois stimulante et intéressante.

4.4 Conclusion

La collaboration en logistique devient de plus en plus importante pour les entreprises, spécialement lorsqu'elle possède des effets bénéfiques. Dans ce chapitre, nous avons présenté un modèle en nombre entiers mixtes innovant pour un PDVRP dans lequel les marchandises doivent passer exactement deux nœuds de transbordements entre leur origine et leur destination. Nous avons utilisé ce modèle pour étudier les bénéfices de la collaboration sans et avec transbordements. Nous l'avons appliqué à une étude de cas dans le contexte du transport multimodal à courte distance. Grâce à cette étude de cas, nous avons pu tester le modèle à l'aide de données réelles correspondant à une période de six mois. Nous avons montré l'impact positif d'une approche collaborative à la fois en termes de réduction de coûts et en termes de réduction d'émissions de CO_2. De plus, nous pouvons améliorer le potentiel de la collaboration en utilisant le transbordement dans certains nœuds du réseau. Cette analyse expérimentale a été utile pour illustrer auprès des partenaires du projet le rôle de la collaboration

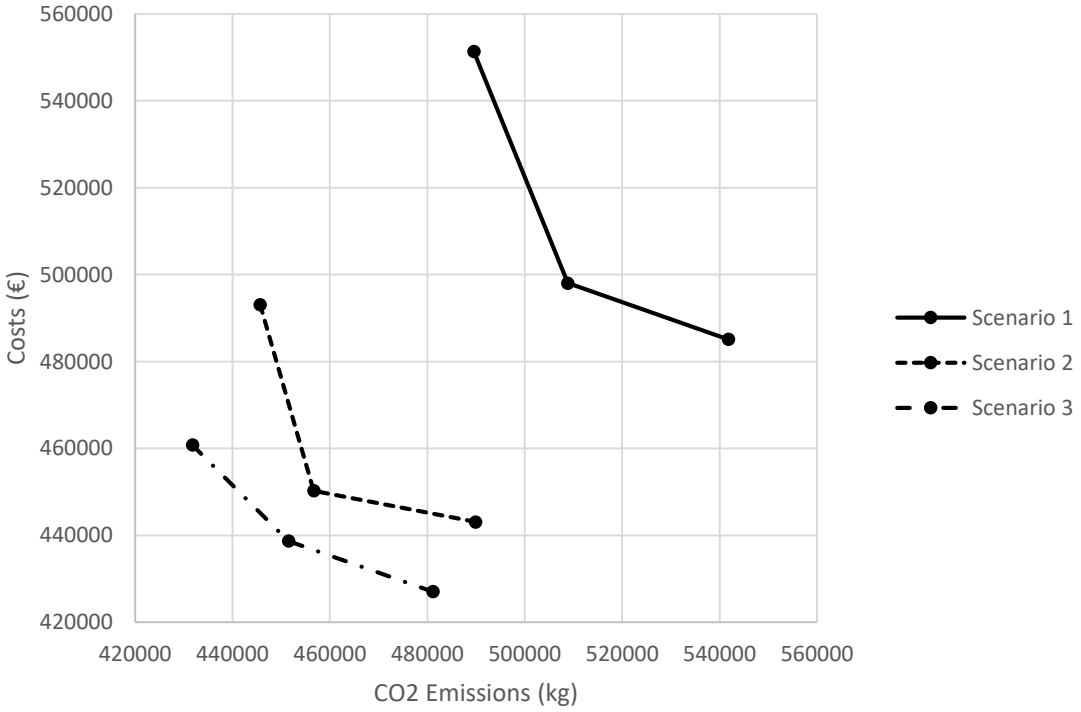

FIGURE 4.11 – Coûts totaux en fonction des émissions de CO_2

et du transbordement comme leviers pour développer une logistique durable.

Conclusion générale

Dans cette thèse, nous avons étudié les possibilités offertes par les problèmes de routage pour améliorer les performances des chaînes logistiques grâce au transport collaboratif, ce qui devient impérieux pour des entreprises soucieuses du développement durable.

Pour ce faire, cette thèse a été découpée en quatre chapitres. Dans le chapitre 1, nous avons étudié les travaux portant sur les problèmes de collectes et livraisons. Les notions générales des problèmes de routage ainsi qu'une classification ont été présentées afin de mieux se situer dans la littérature. Les problèmes étudiés sont les suivants : problèmes de collectes et livraisons sans transbordement, problèmes de collectes et livraisons avec transbordements et problèmes de transport collaboratif. L'étude de la littérature traitant ces problèmes a fait ressortir les points suivants :

— Les problèmes de collectes et livraisons classiques sont largement étudiés, exceptés les PDVRP .

— Les problèmes de collectes et livraisons avec transbordements sont peu étudiés mais le nombre de travaux sur le sujet croît. La faible présence de ces derniers peut s'expliquer par la complexité d'intégrer le principe de transbordement. En effet, ce principe amène à une forte contrainte d'interdépendance entre les routes qui n'existait pas avant dans les modèles précédents.

— Les travaux sur le transport collaboratif sont également très peu présents. Pourtant la collaboration peut être un moyen efficace pour les entreprises d'améliorer la gestion de leur transport sur tous les indicateurs du développement durables.

— Quel que soit le problème étudié dans cette thèse, les aspects environnementaux ne sont quasiment pas étudiés dans la littérature.

Dans le chapitre 2, deux métaheuristiques ont été élaborées pour résoudre le PDPT, un LNS et un GA. Le LNS est déjà utilisé dans la littérature pour résoudre le PDPT et a démontré son efficacité pour ce problème. Les deux méthodes ont d'abord été testées sur des instances de la littérature pour le PDP puis sur des instances de la littérature pour le PDPT. Ce chapitre a permis de mettre en évidence les contributions suivantes :

— La fonction d'insertion des requêtes avec transbordement est améliorée par rapport aux méthodes issues de la littérature. Dans les travaux précédents, soit elle ne permettait pas de trouver forcément la meilleure insertion, soit toutes les insertions possibles étaient testées pour trouver la meilleure. La fonction d'insertion utilisée dans cette thèse permet de trouver la meilleure insertion possible sans tester toutes les possibilités.

— Le GA n'a pas encore été utilisé pour résoudre le PDPT. Cette thèse permet de déterminer si ce type de méthode peut se révéler efficace pour résoudre ce problème.

— Le codage des individus retenus pour le GA n'a pas encore été proposé pour résoudre les problèmes de collectes et livraisons en général. Il permet de toujours obtenir des solutions réalisables, même après un croisement ou une mutation, et pour réduire le temps de décodage, une mémoire tampon en forme d'arbre est utilisée ce qui permet de réduire le temps d'exécution total.

— Les deux métaheuristiques se sont révélées efficaces sur les instances testées. Sur les instances du PDP, la plupart des meilleures solutions connues ont été retrouvées par nos deux méthodes. Quant aux instances du PDPT, les deux méthodes arrivent à obtenir de meilleurs résultats sur certains jeux d'instances. Un seul jeu sur cinq pose problème à nos méthodes. Ceci est dû aux caractéristiques prises en compte qui ne permettent pas à un véhicule de passer deux fois par le même point de transbordement.

Dans le chapitre 3, un modèle pour résoudre le PDVRP a été présenté. Ce modèle a été utilisé pour illustrer les bénéfices du transport collaboratif. Dans

un premier temps, une version sans fenêtre de temps a été proposée. Cette version a d'abord été testée sur des données générées aléatoirement. Ensuite, elle a été appliquée à une étude de cas avec des données réelles provenant d'une entreprise. Dans un second temps, une version avec fenêtres de temps souples a été présentée. Cette version a été testée sur les données réelles de l'étude de cas. Ce chapitre a permis de mettre en évidence les contributions suivantes :

— Une nouvelle formulation pour le PDVRP a été proposée. Notre formulation diffère de celles déjà proposées dans la littérature de par les caractéristiques choisies, notamment la prise en compte de plusieurs types de produits et d'une flotte de véhicules hétérogène. Ces caractéristiques ont été choisies car elles permettent de simuler un réseau de transport collaboratif.

— Une formulation pour le PDVRP avec fenêtres de temps n'a, à notre connaissance, pas encore été proposée. Dans cette thèse les fenêtres de temps sont souples et des pénalités proportionnelles au temps de retard sont ajoutées au coût. Ces pénalités sont plafonnées .

— Une des fonctions objectifs proposée pour ce problème est la minimisation des émissions de CO_2.

— A l'aide de données générées aléatoirement et de données réelles, l'utilité et l'efficacité du PDVRP dans la mise en place d'une collaboration ou dans la planification collaborative du transport sont montrées.

Dans le chapitre 4, un PDVRP particulier a été présenté. En effet en ce qui concerne le problème étudié dans ce chapitre, les marchandises doivent passer par exactement deux points de transbordement entre leur point de collecte et leur point de livraison. Ce problème a été inspiré par une étude de cas réalisée dans le cadre du projet SCALE dans lequel des fournisseurs basés au Royaume-Uni livrent des clients basés en Europe continentale. Les marchandises doivent donc transiter par des ports au Royaume-Uni puis par des ports en Europe continentale. Ce modèle a été appliqué à cette étude de cas dans le contexte du transport à courte distance. Grâce à cette étude de cas, nous avons pu tester le modèle à l'aide de données réelles provenant de trois entreprises correspondant à une période de six mois. Cette expérimentation a été réalisée pour illustrer

l'impact de la collaboration et du transbordement sur les coûts de transport et les émissions de CO_2. Ce chapitre a permis les contributions suivantes :

— Une formulation d'une variante du PDVRP dans lequel les marchandises doivent passer par deux nœuds de transbordements est proposée. Cette formulation permet de résoudre un problème bien précis qui possède des applications réelles de type transport multimodal.

— A l'aide de données réelles, l'utilité et l'efficacité du transbordement dans la mise en place d'une collaboration ou dans la planification du transport collaboratif sont montrées.

L'étude de l'ensemble de ces problèmes peut amener à des perspectives intéressantes, que ce soit au niveau des méthodes de résolutions ou des hypothèses prises en compte, comme par exemple :

— Nous pouvons généraliser les méthodes LNS et GA pour prendre en compte la possibilité pour un véhicule de passer plusieurs fois par un nœud de transbordement.

— Dans le GA, beaucoup de tâches peuvent être parallélisées. Il pourrait donc être intéressant d'étudier l'effet de la programmation parallèle à l'aide de processeurs multi-cœurs ou graphiques sur les temps d'exécution de ces méthodes.

— Il est possible de généraliser la formulation du PDVRP présentée dans le chapitre 3 en prenant en compte d'autres caractéristiques telles que la prise en compte des dépôts pour les véhicules ou encore la possibilité pour un véhicule d'alterner les points de collectes et les points de livraisons.

— Les méthodes LNS et GA doivent être adaptées pour pouvoir les utiliser pour résoudre le PDVRP présenté dans le chapitre 3 et 4.

— La prise en compte du partage des bénéfices lors de la collaboration. En effet, dans cette thèse, ce sont les coûts globaux de tout le réseau qui sont calculés ou optimisés pour justement montrer l'intérêt de la collaboration. Cependant, les entreprises qui collaborent préfèrent connaître leurs propres gains plutôt que les gains globaux. Il serait intéressant de développer des méthodes afin de déterminer les gains de chacun dans le réseau à base de la théorie des jeux par exemple.

Bibliographie

Adams, C.-L. et P. Goldsmith (1999). « Managerial decision-making : Strategic alliances as a governance choice ». In : *International Food and Agribusiness Management Review* 2.2, p. 221–248.

Bahinipati, B. K. et S. Deshmukh (2012). « Vertical collaboration in the semiconductor industry : A decision framework for supply chain relationships ». In : *Computers & Industrial Engineering* 62.2, p. 504 –526. issn : 0360-8352. doi : http://dx.doi.org/10.1016/j.cie.2011.10.017. url : http://www.sciencedirect.com/science/article/pii/S0360835211003238.

Bahinipati, B. K., A. Kanda et S. Deshmukh (2009). « Horizontal collaboration in semiconductor manufacturing industry supply chain : An evaluation of collaboration intensity index ». In : *Computers & Industrial Engineering* 57.3, p. 880 –895. issn : 0360-8352. doi : http://dx.doi.org/10.1016/j.cie.2009.03.003. url : http://www.sciencedirect.com/science/article/pii/S0360835209000886.

Banks, J. (1998). *Handbook of simulation : principles, methodology, advances, applications, and practice*. John Wiley & Sons.

Barratt, M. et A. Oliveira (2001). « Exploring the experiences of collaborative planning initiatives ». In : *International Journal of Physical Distribution & Logistics Management* 31.4, p. 266–289. doi : 10.1108/09600030110394932. eprint : http://dx.doi.org/10.1108/09600030110394932. url : http://dx.doi.org/10.1108/09600030110394932.

Battarra, M., J.-F. Cordeau et M. Iori (2014). « Pickup-and-delivery problems for goods transportation ». In : *Vehicle routing : problems, methods, and applications. MOS/SIAM series on optimization*, p. 161–192.

Benavent, E., M. Landete, E. Mota et G. Tirado (2015). « The multiple vehicle pickup and delivery problem with {LIFO} constraints ». In : *European Journal of Operational Research* 243.3, p. 752 –762. issn : 0377-2217. doi : http://dx.doi.org/10.1016/j.ejor.2014.12.029. url : http://www.sciencedirect.com/science/article/pii/S0377221714010479.

Bent, R. et P. Van Hentenryck (2006). « A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows ». In : *Computers*

& Operations Research 33.4. Part Special Issue : Optimization Days 2003Part Special Issue : Optimization Days 2003, p. 875 –893. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/j.cor.2004.08.001. URL : http://www.sciencedirect.com/science/article/pii/S0305054804001911.

Berbeglia, G., J.-F. Cordeau, I. Gribkovskaia et G. Laporte (2007). « Static pickup and delivery problems : a classification scheme and survey ». In : *TOP* 15.1, p. 1–31. ISSN : 1863-8279. DOI : 10.1007/s11750-007-0009-0. URL : http://dx.doi.org/10.1007/s11750-007-0009-0.

Bettinelli, A., A. Ceselli et G. Righini (2014). « A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows ». In : *Mathematical Programming Computation* 6.2, p. 171–197. ISSN : 1867-2957. DOI : 10.1007/s12532-014-0064-0. URL : http://dx.doi.org/10.1007/s12532-014-0064-0.

Chen, Q., K. Li et Z. Liu (2014). « Model and algorithm for an unpaired pickup and delivery vehicle routing problem with split loads ». In : *Transportation Research Part E : Logistics and Transportation Review* 69, p. 218 –235. ISSN : 1366-5545. DOI : http://dx.doi.org/10.1016/j.tre.2014.06.010. URL : http://www.sciencedirect.com/science/article/pii/S1366554514001057.

Cordeau, J.-F. et G. Laporte (2007). « The dial-a-ride problem : models and algorithms ». In : *Annals of Operations Research* 153.1, p. 29–46. ISSN : 1572-9338. DOI : 10.1007/s10479-007-0170-8. URL : http://dx.doi.org/10.1007/s10479-007-0170-8.

Cordeau, J.-F., G. Laporte, J.-Y. Potvin et M. W. Savelsbergh (2007). « Transportation on demand ». In : *Handbooks in operations research and management science* 14, p. 429–466.

Cordeau, J.-F., G. Laporte et S. Ropke (2008). « Recent Models and Algorithms for One-to-One Pickup and Delivery Problems ». In : *The Vehicle Routing Problem : Latest Advances and New Challenges*. Sous la dir. de B. Golden, S. Raghavan et E. Wasil. Boston, MA : Springer US, p. 327–357. ISBN : 978-0-387-77778-8. DOI : 10.1007/978-0-387-77778-8_15. URL : http://dx.doi.org/10.1007/978-0-387-77778-8_15.

Cortès, C. E., M. Matamala et C. Contardo (2010). « The pickup and delivery problem with transfers : Formulation and a branch-and-cut solution method ». In : *European Journal of Operational Research* 200.3, p. 711 –724. ISSN : 0377-2217. DOI : http://dx.doi.org/10.1016/j.ejor.2009.01.022. URL : http://www.sciencedirect.com/science/article/pii/S0377221709000356.

Créput, J.-C., A. Koukam, J. Kozlak et J. Lukasik (2004). « An Evolutionary Approach to Pickup and Delivery Problem with Time Windows ». In : *Computational Science - ICCS 2004 : 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part III*. Sous la dir. de M. Bubak, G. D. van Albada,

P. M. A. Sloot et J. Dongarra. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 1102–1108. isbn : 978-3-540-24688-6. doi : 10.1007/978-3-540-24688-6_142. url : http://dx.doi.org/10.1007/978-3-540-24688-6_142.

Dai, B. et H. Chen (2012). « Mathematical model and solution approach for carriers' collaborative transportation planning in less than truckload transportation ». In : *International Journal of Advanced Operations Management* 4.1-2. PMID : 45891, p. 62–84. doi : 10.1504/IJAOM.2012.045891. eprint : http://www.inderscienceonline.com/doi/pdf/10.1504/IJAOM.2012.045891. url : http://www.inderscienceonline.com/doi/abs/10.1504/IJAOM.2012.045891.

Dantzig, G. B. et J. H. Ramser (1959). « The Truck Dispatching Problem ». In : *Management Science* 6.1, p. 80–91. issn : 00251909, 15265501. url : http://www.jstor.org/stable/2627477.

Desaulniers, G., J. Desrosiers, A. Erdmann, M. M. Solomon et F. Soumis (2001). « VRP with Pickup and Delivery ». In : *The Vehicle Routing Problem*. Sous la dir. de P. Toth et D. Vigo. Monographs on Discrete Mathematics and Applications. Philadelphia, PA, USA : Society for Industrial et Applied Mathematics. Chap. VRP with Time Windows, p. 225–242. isbn : 0-89871-498-2.

Dias, G. M. et M. d.S. G. Tsuzuki (2010). « System for Shippers Collaboration ». In : *{IFAC} Proceedings Volumes* 43.4. 10th {IFAC} Workshop on Intelligent Manufacturing Systems, p. 240 –245. issn : 1474-6670. doi : http://dx.doi.org/10.3182/20100701-2-PT-4011.00042. url : http://www.sciencedirect.com/science/article/pii/S1474667015301488.

Dondo, R. et J. Cerdá (2013). « A sweep-heuristic based formulation for the vehicle routing problem with cross-docking ». In : *Computers & Chemical Engineering* 48, p. 293 –311. issn : 0098-1354. doi : http://dx.doi.org/10.1016/j.compchemeng.2012.09.016. url : http://www.sciencedirect.com/science/article/pii/S0098135412002943.

Dorigo, M., V. Maniezzo et A. Colorni (1996). « Ant system : optimization by a colony of cooperating agents ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1, p. 29–41. issn : 1083-4419. doi : 10.1109/3477.484436.

Dridi, I. H., R. Kammarti, M. Ksouri et P. Borne (2011). « Multi-objective optimization for the m-PDPTW : aggregation method with use of genetic algorithm and lower bounds ». In : *International Journal of Computers Communications & Control* 6.2, p. 246–257.

Dror, M., D. Fortin et C. Roucairol (1998). *Redistribution of Self-service Electric Cars : A Case of Pickup and Delivery*. Research Report RR-3543. Projet PRAXITELE. INRIA. url : https://hal.inria.fr/inria-00073142.

Dumas, Y., J. Desrosiers et J. Soumis (1991). « The pickup and delivery problem with time windows ». In : *European Journal of Operational Research* 54.1, p. 7–22. ISSN : 0377-2217. DOI : http://dx.doi.org/10.1016/0377-2217(91)90319-Q. URL : http://www.sciencedirect.com/science/article/pii/037722179190319Q.

Ergun, O., G. Kuyzu et M. Savelsbergh (2007a). « Reducing truckload transportation costs through collaboration ». In : *Transportation Science* 41.2, p. 206–221.

Ergun, O., G. Kuyzu et M. Savelsbergh (2007b). « Shipper collaboration ». In : *Computers & Operations Research* 34.6. Part Special Issue : Odysseus 2003 Second International Workshop on Freight Transportation Logistics, p. 1551–1560. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/j.cor.2005.07.026. URL : http://www.sciencedirect.com/science/article/pii/S0305054805002364.

Feo, T. A. et M. G. Resende (1989). « A probabilistic heuristic for a computationally difficult set covering problem ». In : *Operations Research Letters* 8.2, p. 67–71. ISSN : 0167-6377. DOI : http://dx.doi.org/10.1016/0167-6377(89)90002-3. URL : http://www.sciencedirect.com/science/article/pii/0167637789900023.

Flamini, M., M. Nigro et D. Pacciarelli (2011). « Assessing the value of information for retail distribution of perishable goods ». In : *European Transport Research Review* 3.2, p. 103–112. ISSN : 1866-8887. DOI : 10.1007/s12544-011-0051-8. URL : http://dx.doi.org/10.1007/s12544-011-0051-8.

Ghilas, V., E. Demir et T. Van Woensel (2016a). « An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines ». In : *Computers & Operations Research* 72, p. 12–30. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/j.cor.2016.01.018. URL : http://www.sciencedirect.com/science/article/pii/S0305054816300144.

— (2016b). « The pickup and delivery problem with time windows and scheduled lines ». In : *INFOR : Information Systems and Operational Research* 54.2, p. 147–167.

Glover, F. (1986). « Future paths for integer programming and links to artificial intelligence ». In : *Computers & Operations Research* 13.5, p. 533–549. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/0305-0548(86)90048-1. URL : http://www.sciencedirect.com/science/article/pii/0305054886900481.

Goetschalckx, M. et C. Jacobs-Blecha (1989). « The vehicle routing problem with backhauls ». In : *European Journal of Operational Research* 42 (1), p. 39–51. DOI : 10.1016/0377-2217(89)90057-X.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc. isbn : 0201157675.

Gørtz, I. L., V. Nagarajan et R Ravi (2009). « Minimum makespan multi-vehicle dial-a-ride ». In : *European Symposium on Algorithms*. Springer, p. 540–552.

Han, L., B. T. Luong et S. Ukkusuri (2015). « An Algorithm for the One Commodity Pickup and Delivery Traveling Salesman Problem with Restricted Depot ». In : *Networks and Spatial Economics*, p. 1–26. issn : 1572-9427. doi : 10.1007/s11067-015-9297-3. url : http://dx.doi.org/10.1007/s11067-015-9297-3.

Hellström, D. et F. Nilsson (2006). « Combining case study and simulation methods in supply chain management research ». In : *15th Annual IPSERA Conference*.

Hernández, S., S. Peeta et G. Kalafatas (2011). « A less-than-truckload carrier collaboration planning problem under dynamic capacities ». In : *Transportation Research Part E : Logistics and Transportation Review* 47.6, p. 933 – 946. issn : 1366-5545. doi : http://dx.doi.org/10.1016/j.tre.2011.03.001. url : http://www.sciencedirect.com/science/article/pii/S136655451100038X.

Hernández-Pérez, H. et J.-J. Salazar-González (2003). « The One-Commodity Pickup-and-Delivery Travelling Salesman Problem ». In : *Combinatorial Optimization — Eureka, You Shrink! : Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*. Sous la dir. de M. Jünger, G. Reinelt et G. Rinaldi. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 89–104. isbn : 978-3-540-36478-8. doi : 10.1007/3-540-36478-1_10. url : http://dx.doi.org/10.1007/3-540-36478-1_10.

— (2004a). « A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery ». In : *Discrete Applied Mathematics* 145.1. Graph Optimization {IV}, p. 126 –139. issn : 0166-218X. doi : http://dx.doi.org/10.1016/j.dam.2003.09.013. url : http://www.sciencedirect.com/science/article/pii/S0166218X0400071X.

— (2004b). « Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesman Problem ». In : *Transportation Science* 38.2, p. 245–255. doi : 10.1287/trsc.1030.0086. eprint : http://pubsonline.informs.org/doi/pdf/10.1287/trsc.1030.0086. url : http://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0086.

— (2007). « The one-commodity pickup-and-delivery traveling salesman problem : Inequalities and algorithms ». In : *Networks* 50.4, p. 258–272. issn : 1097-0037. doi : 10.1002/net.20209. url : http://dx.doi.org/10.1002/net.20209.

Hernández-Pérez, H. et J.-J. Salazar-González (2014). « The multi-commodity pickup-and-delivery traveling salesman problem ». In : *Networks* 63.1, p. 46–59. issn : 1097-0037. doi : 10.1002/net.21521. url : http://dx.doi.org/10.1002/net.21521.

Hernández-Pérez, H., I. Rodríguez-Martín et J. J. Salazar-González (2009). « A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem ». In : *Computers & Operations Research* 36.5. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X), p. 1639 –1645. issn : 0305-0548. doi : http://dx.doi.org/10.1016/j.cor.2008.03.008. url : http://www.sciencedirect.com/science/article/pii/S0305054808000634.

— (2016). « A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem ». In : *European Journal of Operational Research* 251.1, p. 44 –52. issn : 0377-2217. doi : http://dx.doi.org/10.1016/j.ejor.2015.10.053. url : http://www.sciencedirect.com/science/article/pii/S0377221715009790.

Hickman, J., D. Hassel, R. Joumard, Z. Samaras et S Sorenson (1999). *Methodology for calculating transport emissions and energy consumption*. Rapp. tech.

Holland, J. H. (1975). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press.

Hosny, M. I. et C. L. Mumford (2010). « Solving the One-Commodity Pickup and Delivery Problem Using an Adaptive Hybrid VNS/SA Approach ». In : *Parallel Problem Solving from Nature, PPSN XI : 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part II*. Sous la dir. de R. Schaefer, C. Cotta, J. Kołodziej et G. Rudolph. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 189–198. isbn : 978-3-642-15871-1. doi : 10.1007/978-3-642-15871-1_20. url : http://dx.doi.org/10.1007/978-3-642-15871-1_20.

Jancovici, J. (2007). « Bilan Carbones : Calcul des Facteurs D'émissions et Sources Bibliographiques Utilisées ». In : *ADEME, Paris, France*.

Johnson, E. L. (1989). « Modeling and Strong Linear Programs for Mixed Integer Programming ». In : *Algorithms and Model Formulations in Mathematical Programming*. Sous la dir. de S. W. Wallace. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 1–43. isbn : 978-3-642-83724-1. doi : 10.1007/978-3-642-83724-1_1. url : http://dx.doi.org/10.1007/978-3-642-83724-1_1.

Kalantari, B., A. V. Hill et S. R. Arora (1985). « An algorithm for the traveling salesman problem with pickup and delivery customers ». In : *European Journal of Operational Research* 22.3, p. 377 –386. issn : 0377-2217. doi : http://dx.doi.org/10.1016/0377-2217(85)90257-7. url : http://www.sciencedirect.com/science/article/pii/0377221785902577.

Karlsson, C. (2010). *Researching operations management*. Routledge.

Kerivin, H., M. Lacroix, A. R. Mahjoub et A. Quilliot (2008). « The splittable pickup and delivery problem with reloads ». In : *European Journal of Industrial Engineering* 2.2, p. 112–133.

Kirkpatrick, S., C. D. Gelatt et M. P. Vecchi (1983). « Optimization by Simulated Annealing ». In : *Science* 220.4598, p. 671–680. issn : 0036-8075. doi : 10.1126/science.220.4598.671. eprint : http://science.sciencemag.org/content/220/4598/671.full.pdf. url : http://science.sciencemag.org/content/220/4598/671.

Krause, D. R. (1999). « The antecedents of buying firms' efforts to improve suppliers ». In : *Journal of Operations Management* 17.2, p. 205 –224. issn : 0272-6963. doi : http://dx.doi.org/10.1016/S0272-6963(98)00038-2. url : http://www.sciencedirect.com/science/article/pii/S0272696398000382.

Kulwiec, R. (2004). « Crossdocking as a supply chain strategy ». In : *Target* 20.3, p. 28–35.

Land, A. H. et A. G. Doig (1960). « An Automatic Method of Solving Discrete Programming Problems ». In : *Econometrica* 28.3, p. 497–520. issn : 00129682, 14680262. url : http://www.jstor.org/stable/1910129.

Lau, H. C. et Z. Liang (2001). « Pickup and delivery with time windows : algorithms and test case generation ». In : *Tools with Artificial Intelligence, Proceedings of the 13th International Conference on*, p. 333–340. doi : 10.1109/ICTAI.2001.974481.

Law, A. M., W. D. Kelton et W. D. Kelton (1991). *Simulation modeling and analysis*. T. 2. McGraw-Hill New York.

Lee, Y. H., J. W. Jung et K. M. Lee (2006). « Vehicle routing scheduling for cross-docking in the supply chain ». In : *Computers & Industrial Engineering* 51.2. Special Issue : Logistics and Supply Chain ManagementSelected Papers from The 33rd. ICC&IE, p. 247 –256. issn : 0360-8352. doi : http://dx.doi.org/10.1016/j.cie.2006.02.006. url : http://www.sciencedirect.com/science/article/pii/S036083520600091X.

Li, H. et A. Lim (2003). « A Metaheuristic for the Pickup and Delivery Problem with Time Windows ». In : *International Journal on Artificial Intelligence Tools* 12.02, p. 173–186. doi : 10.1142/S0218213003001186. eprint : http://www.worldscientific.com/doi/pdf/10.1142/S0218213003001186. url : http://www.worldscientific.com/doi/abs/10.1142/S0218213003001186.

Li, Y., H. Chen et C. Prins (2016). « Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests ». In : *European Journal of Operational Research* 252.1, p. 27 –38. issn : 0377-2217. doi : http://dx.doi.org/10.1016/j.ejor.2015.

12.032. URL : http://www.sciencedirect.com/science/article/pii/S0377221715011716.

Liao, C.-J., Y. Lin et S. C. Shih (2010). « Vehicle routing with cross-docking in the supply chain ». In : *Expert Systems with Applications* 37.10, p. 6868–6873. ISSN : 0957-4174. DOI : http://dx.doi.org/10.1016/j.eswa.2010.03.035. URL : http://www.sciencedirect.com/science/article/pii/S0957417410002162.

Liu, R., Z. Jiang, R. Y. Fung, F. Chen et X. Liu (2010). « Two-phase heuristic algorithms for full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration ». In : *Computers & Operations Research* 37.5. Disruption Management, p. 950–959. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/j.cor.2009.08.002. URL : http://www.sciencedirect.com/science/article/pii/S0305054809001968.

Lourenço, H. R., O. C. Martin et T. Stützle (2003). « Iterated Local Search ». In : *Handbook of Metaheuristics*. Sous la dir. de F. Glover et G. A. Kochenberger. Boston, MA : Springer US, p. 320–353. ISBN : 978-0-306-48056-0. DOI : 10.1007/0-306-48056-5_11. URL : http://dx.doi.org/10.1007/0-306-48056-5_11.

Lu, Q. et M. Dessouky (2004). « An exact algorithm for the multiple vehicle pickup and delivery problem ». In : *Transportation Science* 38.4, p. 503–514.

Lu, Q. et M. M. Dessouky (2006). « A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows ». In : *European Journal of Operational Research* 175.2, p. 672–687. ISSN : 0377-2217. DOI : http://dx.doi.org/10.1016/j.ejor.2005.05.012. URL : http://www.sciencedirect.com/science/article/pii/S0377221705004698.

Masson, R., F. Lehuédé et O. Péton (2013). « An adaptive large neighborhood search for the pickup and delivery problem with transfers ». In : *Transportation Science* 47.3, p. 344–355.

Masson, R., F. Lehuédé et O. Péton (2014). « The Dial-A-Ride Problem with Transfers ». In : *Computers & Operations Research* 41, p. 12–23. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/j.cor.2013.07.020. URL : http://www.sciencedirect.com/science/article/pii/S0305054813001998.

Matopoulos, A., M. Vlachopoulou, V. Manthou et B. Manos (2007). « A conceptual framework for supply chain collaboration : empirical evidence from the agri-food industry ». In : *Supply Chain Management : An International Journal* 12.3, p. 177–186. DOI : 10.1108/13598540710742491. eprint : http://dx.doi.org/10.1108/13598540710742491. URL : http://dx.doi.org/10.1108/13598540710742491.

McKinnon, A., M. Browne, A. Whiteing et M. Piecyk (2015). *Green logistics : Improving the environmental sustainability of logistics*. Kogan Page Publishers.

Mitrovic-Minic, S. et G. Laporte (2006). « The pickup and delivery problem with time windows and transshipment ». In : *Infor* 44.3, p. 217. url : http://search.proquest.com/openview/d36f7efb2ed071b0eee29b08649a58c6/1?pq-origsite=gscholar.

Mladenovic, N., D. Urosevic, S. Hanafi et A. Ilic (2012). « A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem ». In : *European Journal of Operational Research* 220.1, p. 270–285. issn : 0377-2217. doi : http://dx.doi.org/10.1016/j.ejor.2012.01.036. url : http://www.sciencedirect.com/science/article/pii/S0377221712000719.

Mladenović, N. et P. Hansen (1997). « Variable neighborhood search ». In : *Computers & Operations Research* 24.11, p. 1097 –1100. issn : 0305-0548. doi : http://dx.doi.org/10.1016/S0305-0548(97)00031-2. url : http://www.sciencedirect.com/science/article/pii/S0305054897000312.

Morais, V. W., G. R. Mateus et T. F. Noronha (2014). « Iterated local search heuristics for the Vehicle Routing Problem with Cross-Docking ». In : *Expert Systems with Applications* 41.16, p. 7495 –7506. issn : 0957-4174. doi : http://dx.doi.org/10.1016/j.eswa.2014.06.010. url : http://www.sciencedirect.com/science/article/pii/S0957417414003510.

Moscato, P. (1989). « On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms ». In : *Caltech concurrent computation program, C3P Report*.

Muckstadt, J. A., D. H. Murray, J. A. Rappold et D. E. Collins (2001). « Guidelines for Collaborative Supply Chain System Design and Operation ». In : *Information Systems Frontiers* 3.4, p. 427–453. issn : 1572-9419. doi : 10.1023/A:1012824820895. url : http://dx.doi.org/10.1023/A:1012824820895.

Musa, R., J.-P. Arnaout et H. Jung (2010). « Ant colony optimization algorithm to solve for the transportation problem of cross-docking network ». In : *Computers & Industrial Engineering* 59.1, p. 85 –92. issn : 0360-8352. doi : http://dx.doi.org/10.1016/j.cie.2010.03.002. url : http://www.sciencedirect.com/science/article/pii/S0360835210000598.

Nagata, Y. et S. Kobayashi (2010). « A Memetic Algorithm for the Pickup and Delivery Problem with Time Windows Using Selective Route Exchange Crossover ». In : *Parallel Problem Solving from Nature, PPSN XI : 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*. Sous la dir. de R. Schaefer, C. Cotta, J. Kołodziej et G. Rudolph. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 536–545. isbn : 978-3-642-15844-5. doi : 10.1007/978-3-642-15844-5_54. url : http://dx.doi.org/10.1007/978-3-642-15844-5_54.

Nanry, W. P. et J. W. Barnes (2000). « Solving the pickup and delivery problem with time windows using reactive tabu search ». In : *Transportation Research Part B : Methodological* 34.2, p. 107 –121. issn : 0191-2615. doi : http://dx.doi.org/10.1016/S0191-2615(99)00016-8. url : http://www.sciencedirect.com/science/article/pii/S0191261599000168.

Newing, R. (2008). « Finding better ways to deliver the goods ». In : *Financial Times*.

Nowak, M., Ö. Ergun et C. C. White III (2008). « Pickup and delivery with split loads ». In : *Transportation Science* 42.1, p. 32–43.

Oertel, P. (2000). « Routing with reloads ». In : *Ph.D. thesis, Universität zu Köln*.

Padberg, M. et G. Rinaldi (1987). « Optimization of a 532-city symmetric traveling salesman problem by branch and cut ». In : *Operations Research Letters* 6.1, p. 1 –7. issn : 0167-6377. doi : http://dx.doi.org/10.1016/0167-6377(87)90002-2. url : http://www.sciencedirect.com/science/article/pii/0167637787900022.

Pan, S., E. Ballot et F. Fontane (2013). « The reduction of greenhouse gas emissions from freight transport by pooling supply chains ». In : *International Journal of Production Economics* 143.1, p. 86 –94. issn : 0925-5273. doi : http://dx.doi.org/10.1016/j.ijpe.2010.10.023. url : http://www.sciencedirect.com/science/article/pii/S0925527310004160.

Pankratz, G. (2005). « A Grouping Genetic Algorithm for the Pickup and Delivery Problem with Time Windows ». In : *OR Spectrum* 27.1, p. 21–41. issn : 1436-6304. doi : 10.1007/s00291-004-0173-7. url : http://dx.doi.org/10.1007/s00291-004-0173-7.

Papadimitriou, C. H. et K. Steiglitz (1982). *Combinatorial optimization : algorithms and complexity*. Courier Corporation.

Parragh, S. N., K. F. Doerner et R. F. Hartl (2008). « A survey on pickup and delivery problems Part II : Transportation between pickup and delivery locations ». English. In : *Journal für Betriebswirtschaft* 58.2, p. 81–117. issn : 0344-9327. doi : 10.1007/s11301-008-0036-4. url : http://dx.doi.org/10.1007/s11301-008-0036-4.

Petersen, H. L. et S. Ropke (2011). « The Pickup and Delivery Problem with Cross-Docking Opportunity ». In : *Computational Logistics : Second International Conference, ICCL 2011, Hamburg, Germany, September 19-22, 2011. Proceedings*. Sous la dir. de J. W. Böse, H. Hu, C. Jahn, X. Shi, R. Stahlbock et S. Voss. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 101–113. isbn : 978-3-642-24264-9. doi : 10.1007/978-3-642-24264-9_8. url : http://dx.doi.org/10.1007/978-3-642-24264-9_8.

Pisinger, D. et S. Ropke (2010). « Large Neighborhood Search ». In : *Handbook of Metaheuristics*. Sous la dir. de M. Gendreau et J.-Y. Potvin. Boston, MA :

Springer US, p. 399–419. ISBN : 978-1-4419-1665-5. DOI : 10.1007/978-1-4419-1665-5_13. URL : http://dx.doi.org/10.1007/978-1-4419-1665-5_13.

Qu, Y. et J. F. Bard (2012). « A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment ». In : *Computers & Operations Research* 39.10, p. 2439 –2456. ISSN : 0305-0548. DOI : http://dx.doi.org/10.1016/j.cor.2011.11.016. URL : http://www.sciencedirect.com/science/article/pii/S0305054811003418.

Rais, A., F. Alvelos et M. Carvalho (2014). « New mixed integer-programming model for the pickup-and-delivery problem with transshipment ». In : *European Journal of Operational Research* 235.3, p. 530 –539. ISSN : 0377-2217. DOI : http://dx.doi.org/10.1016/j.ejor.2013.10.038. URL : http://www.sciencedirect.com/science/article/pii/S0377221713008618.

Ritzinger, U., J. Puchinger et R. F. Hartl (2016). « Dynamic programming based metaheuristics for the dial-a-ride problem ». In : *Annals of Operations Research* 236.2, p. 341–358. ISSN : 1572-9338. DOI : 10.1007/s10479-014-1605-7. URL : http://dx.doi.org/10.1007/s10479-014-1605-7.

Ropke, S. et J.-F. Cordeau (2009). « Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows ». In : *Transportation Science* 43.3, p. 267–286. DOI : 10.1287/trsc.1090.0272. eprint : http://dx.doi.org/10.1287/trsc.1090.0272. URL : http://dx.doi.org/10.1287/trsc.1090.0272.

Ropke, S. et D. Pisinger (2006). « An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows ». In : *Transportation science* 40.4, p. 455–472.

Ropke, S., J.-F. Cordeau et G. Laporte (2007). « Models and Branch-and-cut Algorithms for Pickup and Delivery Problems with Time Windows ». In : *Netw.* 49.4, p. 258–272. ISSN : 0028-3045. DOI : 10.1002/net.v49:4. URL : http://dx.doi.org/10.1002/net.v49:4.

Salazar-González, J.-J. et B. Santos-Hernández (2015). « The split-demand one-commodity pickup-and-delivery travelling salesman problem ». In : *Transportation Research Part B : Methodological* 75, p. 58 –73. ISSN : 0191-2615. DOI : http://dx.doi.org/10.1016/j.trb.2015.02.014. URL : http://www.sciencedirect.com/science/article/pii/S0191261515000429.

Santos, F. A., G. R. Mateus et A. S. da Cunha (2011a). « A branch-and-price algorithm for a vehicle routing problem with cross-docking ». In : *Electronic Notes in Discrete Mathematics* 37, p. 249–254.

— (2011b). « A Novel Column Generation Algorithm for the Vehicle Routing Problem with Cross-Docking ». In : *Network Optimization : 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011. Proceedings*.

Sous la dir. de J. Pahl, T. Reiners et S. Voss. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 412–425. isbn : 978-3-642-21527-8. doi : 10.1007/978-3-642-21527-8_47. url : http://dx.doi.org/10.1007/978-3-642-21527-8_47.

Santos, F. A., G. R. Mateus et A. S. da Cunha (2013). « The Pickup and Delivery Problem with Cross-Docking ». In : *Computers & Operations Research* 40.4, p. 1085 –1093. issn : 0305-0548. doi : http://dx.doi.org/10.1016/j.cor.2012.11.021. url : http://www.sciencedirect.com/science/article/pii/S0305054812002651.

Sarkar, A. et P. K. Mohapatra (2008). « Maximum utilization of vehicle capacity : A case of {MRO} items ». In : *Computers & Industrial Engineering* 54.2, p. 185 –201. issn : 0360-8352. doi : http://dx.doi.org/10.1016/j.cie.2007.07.003. url : http://www.sciencedirect.com/science/article/pii/S0360835207001659.

Savelsbergh, M. W. P. et M. Sol (1995). « The General Pickup and Delivery Problem ». In : *Transportation Science* 29.1, p. 17–29. doi : 10.1287/trsc.29.1.17. eprint : http://dx.doi.org/10.1287/trsc.29.1.17. url : http://dx.doi.org/10.1287/trsc.29.1.17.

Shang, J. S. et C. K. Cuff (1996). « Multicriteria pickup and delivery problem with transfer opportunity ». In : *Computers & Industrial Engineering* 30.4, p. 631 –645. issn : 0360-8352. doi : http://dx.doi.org/10.1016/0360-8352(95)00181-6. url : http://www.sciencedirect.com/science/article/pii/0360835295001816.

Shaw, P. (1998). « Using constraint programming and local search methods to solve vehicle routing problems ». In : *Principles and Practice of Constraint Programming—CP98*. Springer, p. 417–431.

Shi, X., F. Zhao et Y. Gong (2009). « Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem ». In : *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*. T. 1, p. 175–179. doi : 10.1109/ICICISYS.2009.5357913.

Solomon, M. M. (1987). « Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints ». In : *Operations Research* 35.2, p. 254–265. issn : 0030364X, 15265463. url : http://www.jstor.org/stable/170697.

Spekman, R. E., J. W. Kamauff Jr et N. Myhr (1998). « An empirical investigation into supply chain management : a perspective on partnerships ». In : *Supply Chain Management : An International Journal* 3.2, p. 53–67. doi : 10.1108/13598549810215379. eprint : http://dx.doi.org/10.1108/13598549810215379. url : http://dx.doi.org/10.1108/13598549810215379.

Sundarakani, B., R. De Souza, M. Goh, S. M. Wagner et S. Manikandan (2010). « Modeling carbon footprints across the supply chain ». In : *International Journal of Production Economics* 128.1. Integrating the Global Supply Chain, p. 43 –50. issn : 0925-5273. doi : http://dx.doi.org/10.1016/j.ijpe.2010.01.018. url : http://www.sciencedirect.com/science/article/pii/S0925527310000289.

Talbi, E.-G. (2009). *Metaheuristics : from design to implementation*. T. 74. Hoboken, New Jersey : John Wiley & Sons.

Tchapnga Takoudjou, R., J.-C. Deschamps et R. Dupas (2012). « A HYBRID MULTISTART HEURISTIC FOR THE PICKUP AND DELIVERY PROBLEM WITH AND WITHOUT TRANSSHIPMENT ». In : *9th International Conference on Modeling, Optimization & SIMulation*. Bordeaux, France. url : https://hal.archives-ouvertes.fr/hal-00728665.

Thangiah, S. R., A. Fergany et S. Awan (2007). « Real-time split-delivery pickup and delivery time window problems with transfers ». In : *Central European Journal of Operations Research* 15.4, p. 329–349. issn : 1613-9178. doi : 10.1007/s10100-007-0035-x. url : http://dx.doi.org/10.1007/s10100-007-0035-x.

Ubeda, S., F. J. Arcelus et J. Faulin (2011). « Green logistics at Eroski : A case study ». In : *International Journal of Production Economics* 131.1. Innsbruck 2008, p. 44 –51. issn : 0925-5273. doi : http://dx.doi.org/10.1016/j.ijpe.2010.04.041. url : http://www.sciencedirect.com/science/article/pii/S092552731000174X.

Van de Klundert, J. et B. Otten (2011). « Improving LTL truck load utilization on line ». In : *European Journal of Operational Research* 210.2, p. 336 –343. issn : 0377-2217. doi : http://dx.doi.org/10.1016/j.ejor.2010.10.014. url : http://www.sciencedirect.com/science/article/pii/S0377221710006557.

Wang, X. et H. Kopfer (2014). « Collaborative transportation planning of less-than-truckload freight ». In : *OR Spectrum* 36.2, p. 357–380. issn : 1436-6304. doi : 10.1007/s00291-013-0331-x. url : http://dx.doi.org/10.1007/s00291-013-0331-x.

Wen, M., J. Larsen, J. Clausen, J.-F. Cordeau et G. Laporte (2009). « Vehicle routing with cross-docking ». In : *Journal of the Operational Research Society* 60.12, p. 1708–1718. issn : 1476-9360. doi : 10.1057/jors.2008.108. url : http://dx.doi.org/10.1057/jors.2008.108.

Yin, R. K. (2013). *Case study research : Design and methods*. Sage publications.

Zhao, F., S. Li, J. Sun et D. Mei (2009). « Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem ». In : *Computers & Industrial Engineering* 56.4, p. 1642 –1648. issn : 0360-8352. doi : http://dx.doi.

org/10.1016/j.cie.2008.10.014. URL : http://www.sciencedirect.com/science/article/pii/S0360835208002751.

Acronymes

CTP Collaborative Transport Planning. 43, 46–48, 104

DARP Dial-A-Ride Problem. 18, 29, 30, 32

GA algorithme génétique ou Genetic Algorithm. 6, 7, 28, 29, 31, 48, 51, 52, 55–57, 59, 63, 70–72, 75–77, 80–83, 85–91, 158, 160

GRASP Greedy Randomized Adaptive Search Procedure. 28, 41, 54

ILS Iterative Local Search. 28, 55

LCP Lane Covering Problem. 46, 47

LNS recherche à large voisinage ou Large Neighbourhood Search. 6, 7, 31, 41–43, 47, 48, 51, 52, 54–56, 59, 63, 65, 76–78, 82, 83, 85–91, 158, 160

MILP Mixed-Integer Linear Programming. 48

PDP Pickup and Delivery Problem. 18, 20, 21, 29–32, 34, 36, 37, 40, 41, 43, 46–48, 51, 56, 57, 64, 82, 83, 86, 91, 137, 158

PDP-SL Pickup and Delivery Problem with Scheduled Lines. 43, 56

PDPCD Pickup and Delivery Problem with Cross-Docking. 42, 56

PDPT Pickup and Delivery Problem with Transhipments. 7, 32–35, 37, 40–43, 47, 48, 51, 52, 55–57, 63, 65, 70, 73, 82, 83, 86, 88–91, 158

PDTSP Pickup and Delivery Traveling Salesman Problem. 16, 17, 27–29, 57

PDVRP Pickup and Delivery Vehicle Routing Problem. 6–8, 17, 20, 27, 29, 48, 49, 57, 103, 104, 123, 130, 131, 133, 134, 155, 157–160

SCALE Step Change in Agri-food Logistics Ecosystem. 1, 5, 7, 49, 104, 133, 139, 159

TSP problème du voyageur de commerce ou Traveling Salesman Problem. 12

VND Variable Neighborhood Descent. 28, 41

VNS Variable Neighborhood Search. 28, 54

VRP problème de tournées de véhicules ou Vehicle Routing Problem. 12, 14, 15, 31, 63, 83, 137

VRPB Vehicle Routing Problem with Backhauls. 14–16, 22

VRPCD Vehicle Routing Problem with Cross-Docking. 42, 43

VRPPD Vehicle Routing Problem with Pickups and Deliveries. 14, 15, 18, 22, 37, 48

Table des matières

Les problèmes de collectes et livraisons avec collaboration et transbordements : modélisations et méthodes approchées

Résumé

La logistique collaborative est récemment devenue un élément important pour beaucoup d'entreprises afin d'améliorer l'efficacité de leur chaîne logistique. Dans cette thèse, nous étudions les possibilités offertes par les problèmes de collectes et livraisons pour améliorer les performances des chaînes logistiques grâce au transport collaboratif. La thèse est inscrite dans un projet européen nommé SCALE (Step Change in Agri-food Logistics Ecosystem). Dans un premier temps, deux métaheuristiques sont proposées et étudiées pour résoudre le problème de collectes et livraisons avec transbordements. Celles-ci sont comparées aux travaux de la littérature et permettent d'améliorer les résultats sur certaines instances. Dans un deuxième temps, un modèle pour un problème de collectes et livraisons (PDVRP) est proposé. Celui-ci est utilisé pour étudier les bénéfices de la collaboration sur le transport. Il est appliqué sur des données générées aléatoirement et sur des données réelles issues du projet SCALE. Enfin troisièmement, un modèle pour un PDVRP particulier est présenté. Dans ce modèle, les marchandises doivent passer par exactement deux points de transbordement entre les points de collecte et les points de livraison. Ce problème est inspiré d'une seconde étude de cas réalisée dans le cadre du projet SCALE. Ceci permet de mettre en évidence l'intérêt de la collaboration et du transbordement dans le domaine du transport de marchandises.

Mots clés : problèmes de collectes et livraisons ; transport collaboratif ; transbordements ; recherche à voisinage large ; algorithme génétique

---

Abstract

Collaborative logistics have become recently an important element for many companies to improve their supply chains efficiency. In this thesis, we study pickup and delivery problems to improve supply chains efficiency thanks to collaborative transportation. The thesis was part of the European project SCALE (Step Change in Agri-food Logistics Ecosystem). Firstly, two metaheuristics are proposed and studied to solve the Pickup and Delivery Problem with Transshipments. These metaheuristics are compared with literature works and the results of several instances are improved. Secondly, a mathematical model for a pickup and delivery problem (PDVRP) is proposed. This model is used to study the benefits of collaboration on transportation. It is applied on random data and on a case study from SCALE with real data. Finally, a model for a particular PDVRP is presented. In this model, the shipments have to cross exactly two transshipments nodes between their pickup and delivery points. This problem is inspired by a second case study made during the project SCALE. This allows to highlight the importance of collaboration and transshipment in the field of goods transportations.

Keywords: pickup and delivery problems; collaborative transportation; transshipments; large neighbourhood search; genetic algorithm

---

LGI2A

Laboratoire de Génie Informatique et d'Automatique de l'Artois - EA 3926
Faculté des Sciences Appliquées Technoparc Futura 62400 - BÉTHUNE Cedex
France