Outils pour des Problèmes Industriels de Tournées de Véhicules avec Transbordement

Sylvain Fournier

Université Joseph Fourier

Thèse

Pour obtenir le diplôme de
Doctorat en Mathématiques Appliquées et Informatique

Préparée au
laboratoire G-SCOP (Grenoble - Sciences pour la Conception, l'Optimisation et la Production)

En partenariat avec
l'entreprise ILOG S.A.

Dans le cadre de
l'Ecole Doctorale de Mathématiques, Sciences et Techniques de l'Information, Informatique (EDMSTII) de Grenoble

Outils pour des Problèmes Industriels de Tournées de Véhicules avec Transbordement

Tools for Real-Life Vehicle Routing Problems with Transshipment

par

Sylvain Marcel Robert Fournier

soutenue le 16 octobre 2008 devant le jury composé de :

Nadia Brauner-Vettier	Université Joseph Fourier	Directrice
Van-Dat Cung	Grenoble - INP	Président
Bruno De Backer	ILOG S.A.	Directeur
Dominique Feillet	Université d'Avignon	Rapporteur
Gerd Finke	Université Joseph Fourier	Directeur
Ammar Oulamara	Ecole des Mines de Nancy	Examinateur
Marino Widmer	Université de Fribourg	Rapporteur

Remerciements

Je voudrais tout d'abord m'excuser auprès des personnes que je n'aurais pas citées dans ces remerciements. Je le ferai sans doute dans ma prochaine thèse ! Je ne remercierai pas non plus les objets, concepts ou états d'esprit, même si certains d'entre-eux m'ont apporté une grande aide. J'espère qu'ils ne m'en tiendront pas rigueur.

Je tiens tout d'abord à remercier évidemment tous les membres du jury pour m'avoir permis de soutenir cette thèse et m'avoir donné le grade de docteur, et en particulier les rapporteurs Dominique Feillet et Marino Widmer pour m'avoir également donné des commentaires des plus judicieux. Merci à Van-Dat Cung et à Ammar Oulamara d'avoir accepté aussi rapidement de compléter le jury.

Je remercie également mes directeurs de thèse Nadia Brauner et Gerd Finke qui ont toujours été très disponibles quoi qu'il arrive, et malgré la distance entre ILOG et G-SCOP. Merci à Bruno De Backer, mon chef à ILOG. Il a toujours tenu à me faire faire un réel travail de recherche et à ne pas m'impliquer dans le stress des livraisons régulières aux clients.

Un grand merci également à tous ceux qui ont pu m'aider d'une manière ou d'une autre pendant la thèse. Je pense notamment au groupe de travail que nous avions formé avec certains membres du labo dès que je venais sur Grenoble (et qui ont fait plusieurs fois eux-mêmes l'aller-retour jusqu'à Paris) : en plus des personnes déjà citées, je salue Nicolas Teypaz et Christophe Rapine. J'ai eu des discussions très intéressantes avec chacun d'entre-eux qui m'ont permis d'affiner mes formulations mathématiques et ma rédaction de la thèse. Les deux personnes dernièrement citées m'ont également donné du fil à retordre en me faisant visiter les hautes cîmes alpines du haut de nos deux-roues à pédales.

J'en profite pour remercier aussi Vincent Furnon, côté ILOG cette fois, avec qui j'ai beaucoup travaillé notamment à la fin de la thèse, et à qui je dois une bonne compréhension du problème industriel et du logiciel ILOG TPO. Plus généralement, toute l'équipe TPO a énormément facilité mon travail en m'intégrant parfaitement à l'équipe en tant que « Padawan »depuis le début. Merci aussi aux gens de CPLEX comme Emilie Danna ou Ed Klotz, qui ont toujours répondu très vite et très à propos à mes questions.

J'en viens aux remerciements un peu plus tristes pour des gens qui nous

ont quittés depuis que j'ai commencé la thèse en 2004. J'ai une pensée pour Lloyd Clarke avec qui j'ai eu quelques conversations téléphoniques passionnantes concernant ma modélisation et l'utilisation du logiciel ILOG CPLEX. Son décès accidentel m'a beaucoup touché, d'autant que moi aussi je suis un fervent praticant des déplacements à vélo. Une pensée également pour Françoise Lalleman, ma professeur de mathématiques pendant les quatre années de collège qui m'a appris la rigueur mathématique et sans qui je n'en serais certainement pas arrivé là. Mon grand regret est de ne pas avoir pu la revoir pour lui exprimer ma gratitude. En plus des deux personnes précédentes, je dédie ma thèse à mon grand-père maternel Robert Payen, un vrai « Ćh'tí »(avec beaucoup d'accent, et qui m'aurait certainement félicité d'un « Ché bin cha, hein ti père ? »), et de qui je tiens notamment le troisième prénom de mon état civil.

Je remercie les personnes qui, quand j'en ai eu besoin, ont pris le temps de relire et de commenter mes écrits, en particulier le mémoire de la thèse. Merci à tous ceux qui m'ont aidé à élaborer la présentation du travail de thèse pour la soutenance, et à ceux qui m'ont fait le plaisir d'y assister.

Pour conclure, merci à ceux qui auront lu ces remerciements jusqu'au bout. Je vous invite à lire l'intégralité de la thèse avec autant de détermination !

Résumé

Les transporteurs routiers de grande envergure ont de plus en plus besoin d'outils leur permettant de gérer leurs tournées car ce qu'ils pouvaient faire à la main avec peu de véhicules devient insurmontable avec une flotte conséquente. Ces entreprises ont en général un certain nombre de requêtes à satisfaire. Chacun de ces ordres de transport comprend un ramassage d'un produit donné à un site de stockage ou de production, et sa livraison à un autre site. Certains sites, appelés centres de transbordement, permettent l'échange de produits entre véhicules, et peuvent ainsi raccourcir les tournées des véhicules de façon significative, induisant pour l'entreprise des économies non négligeables.

L'entreprise ILOG propose un logiciel, Transport PowerOps (TPO), fournissant à ces entreprises à la fois un moteur de calcul de tournées optimisées, et une interface de visualisation de ces tournées, tout en prenant en compte la notion de tranbordement. Le moteur est basé sur la programmation par contraintes, et une heuristique de première solution est appelée avant une amélioration par recherche locale. Cette approche est satisfaisante mais elle ne considère pas le problème de façon assez globale, l'obligeant à prendre certaines décisions qu'elle juge bonnes (car elles le sont localement) mais qui peuvent s'avérer très douteuses au regard du problème général.

Le but est donc de faire collaborer avec cette méthode déjà implémentée dans ILOG TPO, une approche plus globale permettant de guider certaines décisions que la méthode actuelle a des difficultés à prendre. Dans cette thèse, le choix a été fait de considérer des approches par programmation linéaire, car un modèle mathématique possède une vue plus générale du problème que la recherche locale utilisée par ILOG TPO.

Deux modèles ont été d'abord proposés pour résoudre les petites instances. Le premier est un programme en variables mixtes (*Mixed Integer Program* ou MIP) dédié au Problème de Ramassage et Livraison (*Pickup and Delivery Problem* ou PDP) avec de multiples extensions telles que le transbordement. Le second est un modèle écrit sur les mêmes bases de variables et de contraintes que le premier, mais pour un problème plus général, prenant en compte notamment la séparation de produits. Il s'agit du Problème de Tournées de Véhicules avec Ramassage et Livraison avec les mêmes extensions que précédemment. La principale difficulté à laquelle ces modèles

doivent faire face est la linéarisation des contraintes disjonctives liées à une condition sur une variable binaire. Nous avons choisi une linéarisation par des coupes de type grand-M, et avons calculé ces constantes M de façon à ce qu'elles ne soient pas excessivement élevées. De plus, les coûts sont difficiles à modéliser car ce sont des coûts industriels et, en particulier, non linéaires. A cet effet, de nouvelles variables et contraintes ont dû être ajoutées aux formulations. Pour rendre ces modèles plus rapides à résoudre, plusieurs améliorations ont été conçues, comme des heuristiques de réduction de la taille du problème, un algorithme de plans coupants et des coupes dédiées au problème. Grâce à ces améliorations, nous pouvons obtenir la solution optimale en un temps raisonnable pour la plupart des instances à moins de 12 ordres de transport. De plus, malgré les approximations nécessaires à la modélisation du problème, les solutions obtenues sont proches de celles d'ILOG TPO. Cependant, cette approche est valable uniquement sur les plus petites instances, car dès que leur taille grandit, la résolution par un solveur du MIP devient très longue malgré les améliorations proposées.

Sur les plus grandes instances, nous utilisons un autre modèle de type MIP ainsi qu'une heuristique à deux phases utilisant le moteur de recherche locale d'ILOG TPO. Ce troisième modèle est différent des autres car il s'agit d'un programme linéaire en nombres entiers (alors que les autres sont mixtes), et utilise les aspects de flot du réseau de transport. Il est simplifié de façon à le rendre rapide à résoudre. En particulier, il lui manque tous les aspects de temps du problème initial, comme les fenêtres de temps. Il se base sur un modèle de flot sur les véhicules et un modèle de partitionnement sur les chemins des ordres de transport, ces deux modèles étant liés par les contraintes de capacité. Pour chaque ordre de transport, les chemins considérés sont des chemins de hubs ne pouvant pas contenir d'autres sites, en partie de façon à réduire le nombre de chemins possibles pour chaque ordre de transport, et ainsi réduire également la taille du modèle créé. Pour que les véhicules aient la possibilité d'effectuer plusieurs arrêts sur leur route, nous proposons une heuristique d'agrégation des sites. Cela a également l'avantage de réduire davantage la taille du problème. Ce modèle étant résolu, nous lançons une exécution d'ILOG TPO avec les chemins d'ordres de transport figés, puis à partir de la solution finale de cette exécution, nous relançons ILOG TPO sans ces indications sur les chemins des ordres. En effet, nous n'avons aucune certitude qu'absolument toutes les indications sont bonnes, car notre modèle est très simplifié et ne prend pas en compte notamment le temps. La deuxième phase permet donc de corriger les petites imperfections de la première phase, où les chemins des ordres étaient figés.

Les tests ont d'abord porté sur une comparaison entre le moteur de base d'ILOG TPO et l'heuristique à deux phases. Cette heuristique à deux phases donne des résultats prometteurs, car les solutions trouvés sont en moyennne assez proches de celles trouvées par ILOG TPO seul, mais le temps de calcul est diminué d'un facteur assez conséquent. Ensuite, pour valider la perti-

nence des chemins donnés par le modèle MIP, nous avons comparé l'heuristique avec la même heuristique où les indications sur les chemins des ordres sont générées aléatoirement. En considérant la moyenne des résultats sur 20 exécutions d'indications aléatoires, nous constatons que le modèle MIP donne une bien meilleure solution intermédiaire (entre les deux phases), et une meilleure solution finale, et le temps de calcul est également plus bas. Ceci prouve que notre approche est intéressante et que, moyennant de petites modifications ou des paramétrages adaptés, elle pourrait être d'une grande utilité si elle était intégrée à ILOG TPO. De plus, cette coopération entre une approche par recherche locale (ILOG TPO) et une approche plus globale (le MIP) est également intéressant d'un point de vue théorique.

Contents

List of Figures

List of Tables

List of abbreviations

For readability reasons, the meaning for an abbreviation or an acronym is only given at its first appearance in a chapter. The abbreviations in the list are classified in alphabetical order of the first main term.

Abbreviation	**Meaning**
3PL	Third-Party Logistics provider
B&B	Branch-and-Bound
B&C	Branch-and-Cut
(AD/DT /ZS) C	(Additional Distance / Direct Transportation / Zone-Skipping) Cost
CP	Constraint Programming
GA	Genetic Algorithm
GUI	Graphical User Interface
ILOG TPO	ILOG Transport PowerOps
IP	Integer Program(ming)
LP	Linear Program(ming)
MI(L)P	Mixed-Integer (Linear) Program(ming)
PDP (TW)(T)	Pickup and Delivery Problem (with Time Windows) (with Transshipment)
ZSH	Zone-Skipping Hub
(T/(G)L)S	(Tabu / (Guided) Local) Search
(L/F)TL	(Less than / Full) TruckLoad
TSP	Traveling Salesman Problem
(C)(M) VRP (PD) (TW)(T)	(Capacitated) (Multicommodity) Vehicle Routing Problem (with Pickups and Deliveries) (with Time Windows) (with Transshipment)

List of notations

Object	**Set name**	**Object name**
Site	I	i or j
Site (not the depot)	$I^* = I \setminus \{0\}$	i or j
Site of visit v	-	$i(v)$
Hub	H	h
First / Last site in path p	-	$i^+(p)$ / $i^-(p)$
Arc inside set S	$\alpha(S)$	(ij)
Arc from S to $I \setminus S$	$\delta^+(S)$	(ij)
Arc from $I \setminus S$ to S	$\delta^-(S)$	(ij)
Arc	$\alpha = \alpha(I)$	(ij)
Vehicle	K	k or l
Vehicle path	$P(k)$	p
Fleet	F	f
Vehicle in fleet f	$K(f)$	$k(f)$
Shipment	S	s
Shipment or alternate shipment	S'	s or s'
Complete shipment path	$Q(s)$	p
Shipment path between hubs	$P(s)$	p
Visit	V	v
Visit at site i	$V(i)$	$v(i)$
Visit alternative	$\mathcal{A}$	A
Pickup / Delivery visit for shipment s	-	$p(s)$ / $d(s)$
Product	P	p

For every set Z, the quantity $|Z|$ will denote the number of elements in Z.

Introduction

Globalization and the developpement of means of communication are such that goods travel more and more distance before arriving at their destination. This also holds for people, as the various means of transportation are complementary and make any destination easily available. In particular, the use of road infrastructure to transport goods is becoming more and more important since it is flexible. Moreover, the routing costs are affordable for most of the companies working in the area of transportation and logistics. Many of these companies need tools to help them optimize commodity flows, which is necessary to decrease costs. Indeed, in 2001 in France, the rate of fuel and salaries and charges represented over 50% in the total cost of freight transportation (source: Ministère des Transports, de l'Equipement, du Tourisme et de la Mer). The recent repeated raises of the price of fuel probably made this figure even higher. Hence, we have to incorporate all the features of the problem to propose a good compromise, as a complete study of the problem is useless because of its complexity.

Intermodal transportation (that won't be tackled in this thesis), pooling and the fact that the handling of goods usually requires special structures are the main reasons why transshipment has been a growing activity for several decades. Transshipment, or cross-docking, allows the exchange of commodities or passengers between vehicles at intermediate sites called hubs. This is very important in today's transportation services since it allows a better organization and consolidation of loads which usually implies substential money savings for transportation companies.

Besides costs, the environmental issues raised by the traffic increase, especially in road transportation, are now seriously taken into account and it has become a global concern not to waste fossil fuels and to limit the spillage of exhaust fumes. Nowadays, global warming is one of the scientists' main worries.

ILOG Transport PowerOps (ILOG TPO) is an optimization software package that provides routing solutions for big companies that can own hundreds of sites and vehicles. To obtain solutions which satisfy customer companies, ILOG TPO uses both Operations Research and Constraint Programming technologies. ILOG TPO first computes a simple first solution thanks to a greedy algorithm, and improves it using local search methods.

Transshipment is one of the hardest features to deal with, as the decisions on whether to transship each shipment, and to which hub, are of an exponential complexity.

The objective in this PhD in collaboration with ILOG is to use linear programming tools to guide the decisions for the first solution heuristic and the neighbourhood operators. Indeed, the mathematical model has more insight and a more general point of view of the problem than local search.

In chapter 1, we introduce the real-world problem and explain the context in which our study takes place. In addition to cross-docking, shipments may be delivered to an alternate destination site called ZSH (zone-skipping hub) instead of their regular destination site. Usually, each ZSH is shared by several shipments and it is similar to a hub since it may gather several shipments for them to be routed in the same vehicle. The main difference is that the shipment doesn't need to be routed through the last part of the route (between the ZSH and the regular delivery site). On the other hand, a zone-skipping cost for using a ZSH has to be added to the total routing costs because the shipments are usually routed from the ZSH to their final destinations by subcontractors. Other costs are vehicle-dependent. They are complex and depend mainly on the first and last sites of the vehicle route.

In chapter 2, an overview and a classification of the works on the various aspects of the routing problem with transshipment are presented. The problem we tackle has a large vehicle routing aspect, and is particularly close to the Pickup and Delivery Problem (PDP), since several vehicles have to perform a set of shipments, each of them having its own pickup and delivery sites, and a quantity of products to be routed. Of course, we must consider PDP extensions, such as time constraints and transshipment, to get a better estimation of our problem. In this chapter, we also underline the similarities with other well-known problems such as hub location problems and network flows.

Chapter 3 describes two arc-based Mixed-Integer Programs (MIPs) that are written to solve the problem on the smallest instances. The formulations are dedicated to problems close to the PDP, and they have the same constraints core, especially time constraints and vehicle flow constraints. Both models are based on a classical vehicle routing problem formulation using a 3-index binary variable stating whether a given vehicle goes through a given arc. Time dimension is depicted by continuous variables, which introduces disjunctive constraints that are linearly modeled with big-M constraints. This makes the formulations difficult to solve on large-scale instances. To overcome this, some improvements, such as a cutting-plane algorithm, are proposed.

In chapter 4, a cooperation between another MIP and ILOG TPO is proposed to improve the performance of ILOG TPO, particularly on large instances. The aim is to provide ILOG TPO for a tool that gives shipment path guidelines in order to let ILOG TPO focus on other decisions. We

introduce a simple model that gathers a network flow on vehicles and indicators on shipment paths. No time aspect is modeled here since the focus is on the network plan and not on scheduling. Then, we propose a two-phase heuristic for the cooperation between the MIP and the regular ILOG TPO local search engine. The shipment path guidelines given by the MIP can be used in various ways, by constraining more or less the basic model of ILOG TPO.

Finally, chapter 5 presents the results obtained by the cooperation scheme described in chapter 4 and validates its efficiency on various instances. In particular, the two-phase heuristic is compared with the ILOG TPO original engine on both the computation times and the final solution values. Then, the two-phase heuristic is also compared with the same heuristic whose guidelines are randomly generated, instead of being given by solving the MIP.

Introduction

La mondialisation et le développement des moyens de communication sont tels que les biens parcourent de plus en plus de distance avant d'arriver à destination. C'est également le cas pour les personnes, puisque les différents moyens de transport sont complémentaires et rendent n'importe quelle destination facilement atteignable. En particulier, l'utilisation des infrastructures routières pour transporter des marchandises est de plus en plus importante de par sa flexibilité. De plus, les coûts de transport sont abordables par la plupart d'entreprises impliquées dans le monde de transport et de logistique. Nombre d'entre elles ont besoin d'outils pour les aider à optimiser les flux de produits, condition indispensable pour la réduction de leurs coûts de fonctionnement. En effet, en 2001 en France, la part du carburant et des salaires et charges représentaient plus de 50% du coût total de transport de marchandises (source : Ministère des Transports, de l'Equipement, du Tourisme et de la Mer). Les hausses récentes et répétées du prix de l'essence ont probablement rendu cette part d'autant plus grande. Il est donc primordial d'intégrer toutes les caractéristiques de la problématique afin de proposer un bon compromis, car une étude complète du problème est inenvisageable de par sa complexité.

Le transport multimodal (non traité dans cette thèse), le groupement des marchandises ainsi que le fait que la manipulation des biens ne se fait pas sans des structures adéquates sont les principales raisons pour lesquelles le transbordement est depuis plusieurs décennies une activité en expansion. Le transbordement permet l'échange de produits ou de passagers entre les véhicules sur des sites intermédaires appelés centres de transbordement (ou hubs). C'est d'une importance capitale dans les services de transport actuels car cela permet une meilleure organisation et une consolidation des chargements, ce qui se traduit généralement par des économies substentielles que réalisent les entreprises spécialisées dans le transport.

En plus des problèmes de coûts, les problèmes écologiques soulevés par l'augmentation du trafic, et particulièrement de la circulation routière, sont maintenant pris sérieusement en compte et c'est devenu un intérêt public de ne pas gâcher les combustibles fossiles et de limiter les rejets de gaz d'échappement. De nos jours, le réchauffement de la planète est une des préoccupations principales des scientifiques.

ILOG Transport PowerOps (ILOG TPO) est un logiciel d'optimisation qui fournit des solutions applicables à des entreprises pouvant posséder jusqu'à des dizaines de milliers de sites et des milliers de véhicules. Pour obtenir des solutions satisfaisantes pour ces clients, ILOG TPO utilise à la fois la Recherche Opérationnelle et la programmation par contraintes (PPC). Ce logiciel calcule d'abord une première solution simple au moyen d'un algorithme glouton, puis l'améliore par des méthodes de recherche locale. Le transbordement est une des caractéristiques les plus dures à traiter, car les décisions sur le transbordement de chaque expédition (ou ordre de transport), ainsi que le choix des hubs pour chaque expédition, est d'une complexité exponentielle.

L'objectif dans ce doctorat CIFRE en collaboration avec ILOG est d'intégrer des outils de programmation linéaire pour guider les décisions des heuristiques de première solution et des opérateurs de voisinage. En effet, le modèle mathématique a plus de recul et une vue du problème plus générale que la recherche locale.

Dans le chapitre 1, nous introduisons le problème industriel et expliquons le contexte dans lequel notre étude prend place. En plus du transbordement, les ordres de transport peuvent être livrés à un site auxiliaire appelé plateforme régionale (*Zone-Skipping Hub* ou ZSH) au lieu de leur site de livraison attribué. Généralement, chaque ZSH est commune à plusieurs ordres de transports et est similaire à un hub puisqu'elle peut regrouper plusieurs ordres pour qu'ils soient pris par le même véhicule. La principale différence est que l'on n'a pas besoin de déterminer la fin du chemin de l'ordre (entre la ZSH et le site de livraison normal). En revanche, un coût supplémentaire d'utilisation de ZSH doit être ajouté au coût total car les ordres sont en principe amenés de la ZSH aux destinations finale par des sous-traitants. D'autres coûts sont attribués à chaque véhicule. Ils sont complexes et dépendent principalement des premier et dernier sites de la tournée du véhicule.

Dans le chapitre 2, nous présentons un aperçu et une classification des travaux sur les différents aspects du problème de tournées avec transbordement. Le problème abordé a un aspect tournées de véhicules important, et en particulier, il est proche du Problème de Ramassage et Livraison (ou PDP), car plusieurs véhicules doivent effectuer un certain nombre d'ordres de transport qui ont chacun leur propres sites de ramassage et livraison, et une quantité de produits qui doit être acheminée. Bien sûr, nous devons prendre en compte des extensions du PDP, comme les contraintes de temps et le transbordement, pour se rapprocher un peu plus de notre problème. Dans ce chapitre, nous soulignons également les similitudes avec d'autres problèmes connus comme les problèmes de localisation de hubs et les problèmes de flot.

Le chapitre 3 décrit deux modèles mixtes en nombres entiers (MIPs) à base d'arcs écrits pour résoudre les plus petites instances. Les formulations sont dédiés à des problèmes proches du PDP, et ils ont la même base de contraintes, et plus particulièrement les contraintes de temps et celles concernant le flot de véhicules. Les deux modèles s'appuient sur une formu-

lation classique de problème de tournées de véhicules et utilisent une variable binaire à 3 indices qui détermine si un véhicule donné utilise un arc donné. La dimension de temps est représentée par des variables continues, ce qui introduit des contraintes disjonctives qui sont linéarisées avec des contraintes grand-M. Cela rend les formulations difficiles à résoudre sur les instances de grande taille. Pour contourner ce problème, quelques améliorations, comme un algorithme de plans coupants, sont proposées.

Dans le chapitre 4, une collaboration entre un autre MIP et ILOG TPO est proposée pour améliorer les performances d'ILOG TPO, en particulier sur les grosses instances. Le but est de fournir à ILOG TPO un outil qui donne des indications sur les chemins des expéditions pour permettre à ILOG TPO de se focaliser sur d'autres décisions. Nous introduisons un modèle simple qui regroupe un flot sur les véhicules et des indicateurs sur les chemins des ordres de transport. L'aspect de temps n'est pas modélisé car l'accent est mis ici sur le réseau et non sur l'ordonnancement. Ensuite, nous proposons une heuristique à deux phases pour cette collaboration entre le MIP et le moteur de recherche locale d'ILOG TPO. Les indications sur les chemins des ordres de transport donnés par le MIP peuvent être utilisées de diverses manières, en contraignant plus ou moins le modèle de données d'ILOG TPO.

Enfin, le chapitre 5 présente les résultats obtenus par cette collaboration et valide son efficacité sur différentes instances. En particulier, l'heuristique à deux phases est comparée au moteur original d'ILOG TPO, et à la fois le temps de calcul et la valeur de la solution finale sont testés. Ensuite, l'heuristique à deux phases est comparée avec la même heuristique dont les indications sont générées aléatoirement, au lieu d'être données par la résolution du MIP.

Chapter 1

A routing problem with transshipment

RÉSUMÉ DU CHAPITRE

Le chapitre 1 décrit le problème auquel nous sommes confrontés dans le cadre de cette thèse. De nombreux métiers sont impliqués dans le domaine de la chaîne d'approvisionnement. Nous nous intéressons à l'acheminement des marchandises et aux tournées des véhicules permettant cet acheminement. Les entreprises qui ont besoin d'effectuer cette tâche font souvent appel à des entreprises auxiliaires appelées prestataires logistiques de troisième partie (*Third-Party Logistics provider* ou 3PL). Ces sous-traitants s'occupent de la gestion des entrepôts et du transport des marchandises, et peuvent avoir des contrats avec de multiples entreprises pour mutualiser les chargements. Chaque ordre de transport consiste en un ramassage d'un bien à un site donné, et sa livraison à un autre site.
Pour mieux organiser le chargement de chaque véhicule en cours de tournée, le 3PL possède généralement des centres de transbordement (également appelés hubs) dans lesquels le transbordement (échange de produits entre différents véhicules) est autorisé. Cette possibilité revêt une importance capitale pour l'établissement de tournées optimisées, et de plus en plus d'entreprises l'exploitent afin de minimiser leurs coûts de transport. Les 3PL utilisent certains logiciels pour les aider à choisir de telles tournées. ILOG TPO est un de ces logiciels et est constitué d'une interface graphique conviviale et d'un moteur d'optimisation prenant en compte le transbordement. ILOG TPO permet également aux ordres de transport d'être ramassés ou livrés à des plateformes régionales (*Zone-Skipping Hub* ou ZSH) qui facilitent, de façon similaire aux hubs, la mutualisation des ordres dans les véhicules. La portion de trajet manquante n'a pas à être effectuée mais, en contrepartie, un coût supplémentaire (appelé *Zone-Skipping Cost* ou ZSC) doit être payé. Ce coût s'ajoute aux nombreux coûts introduits par ILOG TPO pour chaque véhicule, qui dépendent principalement des zones de départ et d'arrivée du véhicule.

Supply Chain Management is the study and the optimization of costs in the process of moving goods or services for a supplier to a customer. This domain has acquired a growing interest since the middle of the nineteenth century, and it is now carefully split into distinct areas, each of which having its own features and being usually tackled independently. In particular, goods transportation problems have a lot of practical applications. The last decades have been marked by the rise of traffic in all means of transportation, which unavoidably introduces undesirable outcomes. Safety must be ensured whatever the circumstances, and this implies to pay attention to the vehicle flow, especially when transporting people or hazardous material, even if these activities will not be dealt with in this thesis. Moreover, environmental issues arise with the growth of traffic, such as the lack of fossil fuels and all the problems due to vehicle exhaust fumes, e.g. the greenhouse effect. As a consequence, the need to find appropriate vehicle routes with respect to some criterion is not necessarily a company's own concern but may be a shared interest. Our specific study, however, focuses on a particular company that has to minimize complex routing costs to ship some goods from several origin sites to several destination sites.

The kind of problems that we have to deal with in this context is described in this chapter. The general framework is detailed in section 1.1. In section 1.2, we describe transshipment, which is the fact that vehicles can swap their loads with other vehicles at some sites called hubs where the required facilities are available. Indeed, machines or qualified workforce may be necessary to unload and reload shipments from one vehicle to another, and it may take some time. Transshipment can be used in the transportation of goods, but also as a intermodal transition for passengers, in public transport. Several practical applications take transshipment into account, amongst which the vehicle routing solver ILOG TPO. Section 1.3 is dedicated to the enumeration of particular features that are integrated by ILOG TPO, especially the complex costs that the companies using ILOG TPO have to pay for their routing schedules. It also describes the possibility for a shipment to be delivered at a regional warehouse instead of its regular delivery site, which introduces shipment alternatives.

1.1 Industrial routing

1.1.1 Supply Chain Management

Supply Chain Management is one of the research areas that has the most implications in today's business. **Supply Chain** is the process by which goods arrive at their destination (a customer, for example) after being

produced, shipped possibly by several means of transportation and consolidated if needed. It involves a lot of partners, and each of them needs to perform all the tasks they are supposed to, but minimizing their costs or maximizing their profit. Supply chain is such a considerable area that several service providers have to be included in the whole process, so that the involved companies can take decisions easily. Supply chain management is the study of an efficient planning of the activities and operations that arise in the context of supply chain. Several softwares, such as ILOG Transport PowerOps (routing aspect) and ILOG Plant PowerOps (scheduling aspect), are part of the tools proposed to such companies to help them take decisions for which the consequences can be tremendous.

There are lots of aspects that are of interest for large companies for which any marginal gain of productivity may have a huge impact on their profit. For example, a given commodity is produced in a factory from several raw materials that may come from several suppliers. There may also be several steps in the production of the good in the factory, and numerous machines may be part of this process. The commodity has then to be routed to its customer together with other goods that may have been produced in other factories.

All of these aspects are dependent one another, but are usually tackled separately in a multilevel way because of their complexity.

1.1.2 Logistic issues

In this thesis, only one of these aspects is focused on, which is how the products can be routed between suppliers and customers, in a way that the routing costs are minimized. The decisions that have to be taken inside the factories or warehouses, such as the ones concerning the storage or the production itself, won't be taken into account here.

Several papers [Crainic, 1999, Crainic and Semet, 2005] can be read as reference reviews on freight transportation.

In the routing point of view of logistics, only the flows of products in some vehicles and between sites are considered. A supermarket may be interested in moving periodically a cargo of wine from its department in Bordeaux, where it is produced, to its department in Paris, where there is a great demand. It may be the same for several other products.

This company may either own its vehicle fleet, or contract out the decisions about the transportation of shipments to another company, called a **third-party logistics provider** (3PL). The 3PL case has many advantages, amongst which the fact that it allows the company to focus on its basic activities and not in a domain for which it doesn't have the skills or time, which is choosing their vehicle routes. Furthermore, the 3PL usually has several contracts with various companies, which allow them to consolidate all the requests they must perform and brightly organize their vehicle

fleet in order to minimize their routing costs. The company usually makes every vehicle available anywhere. Hence, the 3PL has no commitment to make them start or end their routes in some site or another.

Warehouses

The 3PL owns several warehouses where the products can be picked-up or delivered, and where the vehicles can park and stay for as long as it is necessary. Usually, these actions are supposed to be performed within a time window, because of some constraints on work and opening times of the sites.

Vehicles

The vehicles of a carrier can be of diverse types. Some are preferably used for short-haul missions. They are usually smaller vehicles, but on the other end, the cost for getting them is much cheaper. Some others are used for longer routes. These are called **pooling vehicles**, and make mostly back-and-forth trips between big regional sites. However, all these vehicles can often be grouped into fleets, because it is rare that the company owns vehicles that are unique.

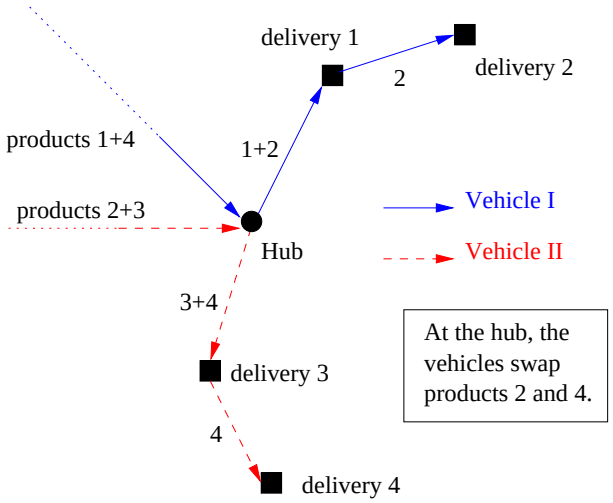
Shipments

A **shipment** is a transportation order that has to be performed by a company. Each product owned by this company has to be transported between two sites during the supply-chain process. Some companies may allow a shipment to be unperformed. This usually results in a penalty that has to be paid, and the amount of this penalty depends on the importance of the ignored shipment. This importance may depend on the shortage or excess of the given product on the delivery site. The type of product (fresh food, for example) may also make the penalty cost higher or lower, depending on the emergency to deliver it on time. Sometimes, there are restrictions in a vehicle cargo because of incompatibilities. It may be forbidden for a vehicle to contain food and chemical products together, and some products, like fresh food, may require a certain type of vehicle.

1.2 Transshipment

In a typical transportation company, there are several types of vehicles (as mentioned in the previous section). Large vehicles may be very useful for carrying big loads over long distances, but are far too expensive for short tasks like tours between customers that are close one from another. Some smaller (and cheaper) vehicles are required for this kind of tasks.

Figure 1.1: Organizing shipments in a hub



Then, two vehicles of different types must be able to swap or at least to share their loads. This cannot be done at a customer site, because of the lack of facilities needed to load and unload the vehicles. Furthermore, a customer will not be pleased to see that his warehouse is used to move products that don't belong to him. There ought to be some sites that are specialized in this handling of loads, and this is precisely the use of transshipment centers (or hubs).

1.2.1 Transshipment centers

The sites where the exchange of products between vehicles (**transshipment**, also known as **cross-docking**) is permitted have several names, amongst which **transshipment centers**, **transportation hubs** and **distribution centers**. A hub is the location where several vehicles have the ability to exchange loads (see figure 1.1 for an example).

Road hubs look like big warehouses with large parking capacities for trucks (see figure 1.2 taken from [Bartholdi and Hackman, 2008]).

Inside the hub, there are also many topics of interest for transportation companies [Li et al., 2004, Bartholdi and Gue, 2002, Chang and Tsui, 1992]. The parking capacity is usually restrictive and the arrival and departure times of each vehicle have to be planned carefully. The routing of commodities inside the hubs and from the incoming vehicle to the leaving vehicle, as well as the storage issue are also deeply studied. In fact, there is seldom a long time of storage in a hub (which may be very costly), otherwise the site is preferably referred to as a warehouse. However, in this thesis, we will only focus on the routing aspect around the hub. Note that in the following, we will consider that the vehicles are trucks, as the ILOG TPO customers

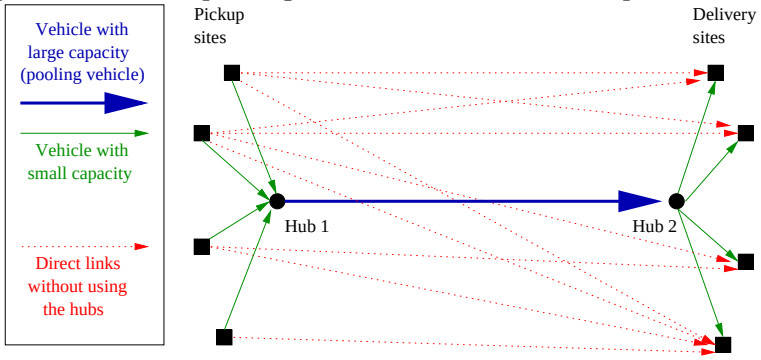
Figure 1.2: U-shaped hub in Portland, USA

are mainly road transportation companies. Moreover, we consider that no storage is possible in any hub. This is a constraint given by ILOG TPO customers due to the fact that the products that must be routed are mainly fresh products.

1.2.2 Truckload

The transportation of small freight is called **less than truckload** (LTL). Larger freight is usually handled by semi-trailers and is referred to as **full truckload** (FTL). In FTL transportation, the vehicle is filled by the shipper in one pickup site, and the shipments are delivered to a unique delivery site. On the other hand, in LTL transportation, the carrier collects freight from various shippers. The advantage of using an LTL carrier is that the transportation of one shipment may be charged only a fraction of the cost of hiring an entire vehicle, whereas the drawback is that the shipment will probably have to be unloaded and loaded several times at various hubs. Indeed, the carrier may be interested in moving a shipment to another vehicle if its costs are decreased by this operation, which is irrelevant for FTL carriers. If the shipper has many shipments to transport to the same area, it is better for him to choose a FTL carrier, and at arrival at the area, change the shipments to an LTL carrier. The vehicles would then have to reorganize their loads in a hub in order to separate the shipments and drive each of them to their destination. In the general case, a carrier usually uses LTL freight before and after the hubs, but FTL freight between the hubs. That's the main reason why the carriers often own shuttles that are supposed to

Figure 1.3: Use of pooling vehicles between transshipment centres



travel between the hubs.

1.2.3 Transshipment uses

There are plenty of real-life situations where transshipment is necessary. Hubs are often used for multiple-pieced products that need to be consolidated or deconsolidated. Indeed, in a hub, vehicles can reorganize their loads but the products they are carrying can also be treated and assembled. For example, several computer components may be transported to a hub, be assembled into a computer, which should then be delivered. On the contrary, some products may need to be disassembled before being delivered to their final destination.

Most of the companies that have the opportunity to take advantage of transshipment have several types of vehicles in order to handle efficiently every part of a shipment route. Although a shipment may be picked up and delivered by the same vehicle, it is often cheaper to assign it to a vehicle for its pickup, another one for its delivery and possibly other vehicles in its route, as it may be crossdocked successfully in different vehicles. In a **hub-and-spoke** network, several shipments are routed through several hubs, usually two. Between the hubs, pooling vehicles are supposed to carry a lot of shipments, and are available only on this kind of trip (see figure 1.3). They have a big capacity and are usually faster than other types of vehicles. Pooling vehicles are used to carry a huge quantity of goods from a region to another.

However, in the case there are few pickup sites and many delivery sites around the hub, the hub-and-spoke becomes an **inbound pooling** situation, such as the distribution of goods in a region after their production in a factory located in another region. Conversely, **outbound pooling** takes place with many pickup sites and few delivery sites. Pooling vehicles are also used for inbound and outbound pooling.

In some cases, transshipment is used between several shippers in a col-

laborative planning. A carrier can be shared by these shippers in order to reduce the costs for each shipper, or even to improve service. For example, a State may decide to make a vehicle fleet available to several shippers at low cost in order to reduce the greenhouse gaz emissions.

1.3 ILOG Transport PowerOps

ILOG Transport PowerOps (ILOG TPO) is a software package for solving routing problems faced by transportation companies. To be as close to the customers' problems as possible, ILOG TPO has a very complete model. In addition to the basic features of the routing problem and transshipment described in the previous sections, the structure of ILOG TPO (especially the cost structure) is quite complex and needs to be detailed in this section. The ILOG TPO environment is explained in the ILOG TPO 3.1 documentation [ILOG TPO, 2007].

The background for all the experiments carried out in this thesis is a road transportation company, although ILOG TPO permits any kinds of means of transportation in its modeling. Therefore, every later reference to vehicles will stand for trucks, and travels will sometimes be used for drives.

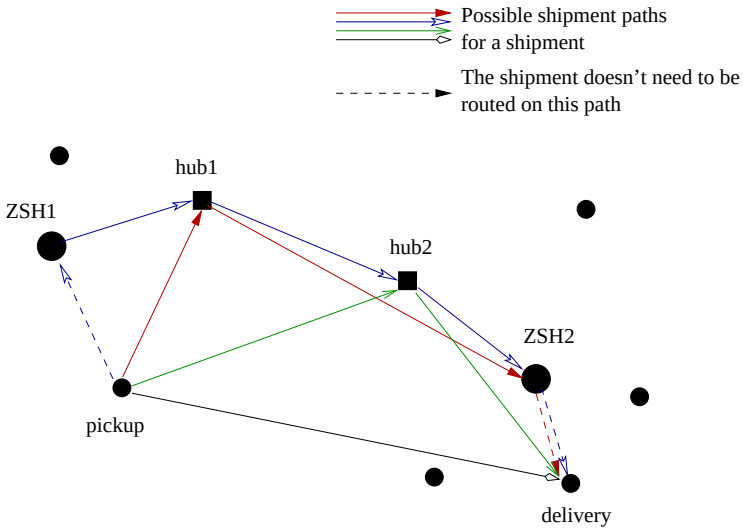
1.3.1 Vehicles

First of all, vehicles are part of a **heterogeneous fleet**, usually separated into several sub-fleets. Indeed, companies that need such tools as TPO to optimize their routing process seldom have unique vehicles. TPO enables the modeling of **shuttles** by allowing each vehicle to have its own departure and arrival sites, as well as time windows for departure and arrival. Shuttles are usually far cheaper or even free of charge, but on the other hand they are limited in the number of stops in their route. Any shuttle has usually a unique possible **lane** to travel on (a lane is a simple trip between two sites). For a given lane, one shuttle is scheduled at each day and has a unique possible departure time. On the other hand, the shuttles are the cheapest vehicles of the complete fleet. Vehicles that are not considered as shuttles may start or end their route at any time and anywhere. Note that shuttles are not exactly pooling trucks since they are not bigger than other vehicles and they travel on rather short distances (they link nearby warehouses owned by the 3PL).

1.3.2 Shipments

A regular shipment is made up of two visits: a pickup visit and a delivery visit. Both visits for each shipment have their own features, such as a time window, a site and a duration. A visit type (pickup or delivery) is essential

Figure 1.4: Shipment paths and alternatives



when considering the visit sequence in a vehicle, as there may be a cost impact (see subsection 1.3.4).

ILOG TPO shipments are also made of two important notions : shipment paths and shipment alternatives.

A shipment can be performed through several paths between hubs. Any shipment can be delivered directly from the origin site to the destination site, meaning that no transshipment will take place during its trip. Some shipments (but not necessary all of them) may also be delivered via a hub or an ordered set of hubs. Any sequence of such hubs is called a **shipment path** and any shipment has a finite set of shipment paths it can go through (as said before, this set contains at least the direct path). A shipment is direct if it is shipped through its direct path (i.e. it is not transshipped).

Several sites, called **zone-skipping hubs** (or ZSH), may be used as regional warehouses if it is too expensive for the vehicles to deliver a shipment to its destination located in the same region. In a sense, the ZSH can be used as a kind of hub, except that the final travel from the ZSH to the destination isn't supposed to be performed by any vehicle, but has an additional cost for the carrier to perform the task. As a result, some shipments (once again, not necessarily all of them) have the possibility to be delivered, either to the destination, or to the nearby ZSH. This choice is called a **shipment alternative**. Of course, all the combinations between shipment paths and shipment alternatives are allowed. An order may be shipped directly (with no intermediate hub) to the ZSH instead of the destination, and may be transshipped at several hubs before getting delivered to its destination.

Figure 1.4 presents several possible shipment paths for a shipment from its pickup site to its delivery site. Alternatives are depicted by the fact

that some paths start at ZSH1 or end at ZSH2, and the broken arrows corresponding to these paths won't have to be covered by a vehicle. The shipment we consider has two paths between hubs (direct or through hub 2), but it has two shipment alternatives:

- pickup → ZSH2, with only one path between hubs (through hub 1),
- ZSH1 → ZSH2, with only one path between hubs (through hubs 1 and 2).

If we merge the notions of paths between hubs and shipment alternatives, this shipment then has 4 distinct shipment paths.

Note that a shipment may go through a hub without being transshipped at this hub (i.e. it stays in the same vehicle). For ILOG TPO, the hub is not considered to be in its shipment path, as this hub is just a transit site in this case. As a result, some shipments may be seen as direct although they go through several hubs.

1.3.3 Additional features

ILOG TPO has many features that enables users to model their problems precisely.

Due to the working schedules at each site, the drivers that arrive at a site at a time when it is not possible to perform an action (e.g. absence of teams, site closed for maintenance) have to wait for the site to be opened again. All the time instants when the site can be visited are grouped into time windows. Vehicles may also be allowed to pickup or deliver at a non-suitable time, but the carrier company has then to pay a penalty corresponding to the drawback brought by this action. This penalty has to be carefully taken into account as one of the costs described below (subsection 1.3.4) when the vehicle routes are chosen.

As logistics involve not only vehicles or sites, but also human resources, some constraints on their working conditions are necessary for a legal working environment. Drivers have to rest after some driving duration and after some working time has passed. Drivers need to stop working for some time (to let them sleep, eat or just rest) before getting back to their assigned tasks. This non-working situation can be simultaneous with a waiting period before the opening of a site, for example. This is referred to as **breaks** in the ILOG TPO documentation.

ILOG TPO also has the possibility to model various kinds of incompatibilities. A visit may be incompatible with a vehicle, meaning that the driver does not have the ability or the equipement available to pickup or deliver a product, and neither has the site where the visit occurs. The vehicle may even be unable to carry a given product, e.g. in refrigerated vehicles.

The so-called LIFO (Last In - First Out) constraints ensure that if a shipment (let's say B) is picked up after another (A), it will be delivered

before, as it is supposed to be in front of A in the vehicle, making it difficult to access A before B. The shipments are supposed to be piled up along the route, and only the head of the pile can be accessed at any time.

ILOG TPO provides resource constraints restrictive for the vehicles at some sites. There may be a limited number of docks where the vehicles can load and unload the shipments, and there can't be more vehicles inside a site than the number of docks, at any time. If there are too many vehicles, the last arrived vehicles have to wait outside the facility.

There are several dimensions on which the capacity constraints of each vehicle can be applied. Usually, trucks are limited in weight and in volume: their cargo can't exceed a given weight or a given number of pallets.

Some vehicles have a limited number of stops. Such vehicles cannot visit more sites than a given number. For example, shuttles are limited to one stop, in order to enforce them to stay on their lane without visiting other sites. Other vehicles are usually limited to 3 or 4 stops.

1.3.4 Costs

In ILOG TPO, the vehicles are bought or loaned based on a contract, which means that some of the routing costs are not directly charged for the shipper, such as petrol bought at stations. However, it is indirectly charged through other costs included in the contract. The ILOG TPO costs consist of several subcosts that contribute to the complexity of the formulation available in the software.

Routing costs

The routing cost is the main subcost, and can be itself decomposed into costs that have various origins. The routing costs in ILOG TPO are defined on each vehicle and depend on the outgoing and incoming zones of this vehicle. Such **lane-based costs** may be fixed costs to which other costs are added in the case the vehicle travels more distance than a maximum distance allowed between the two zones. The main routing cost is a lane-based cost called **direct transportation cost** (or DTC). As a lane-based cost, it depends on the starting and ending sites of the vehicle route, but also on the maximum quantity transported along the route. As a consequence, if possible, it is better to perform a sequence of pickups and deliveries so that there are never too many shipments together in the vehicle, than to perform a sequence of pickups followed by a sequence of deliveries. However, we will notice in chapter 5 that in the usual customer instances, the pickups take often place at the same site or at nearby sites, whereas the vehicles have to travel long distances to deliver the shipments. Practically, this cost is a growing piecewise-linear function of the maximal quantity carried by the vehicle. Some examples are given on figure 1.5.

Figure 1.5: Direct transportation costs (DTC)

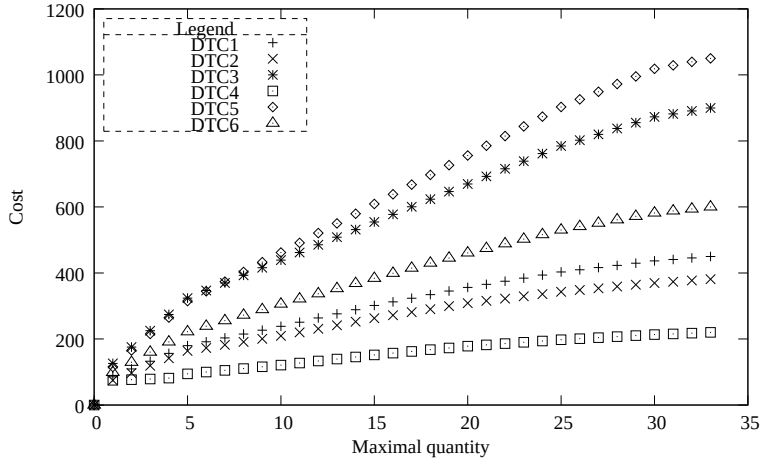
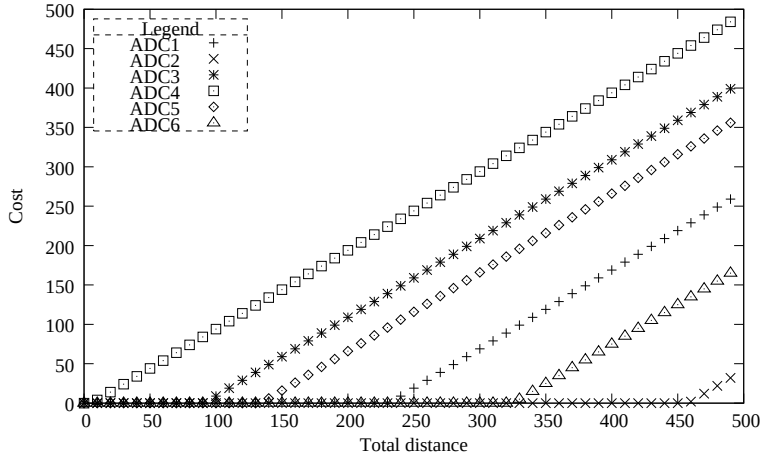


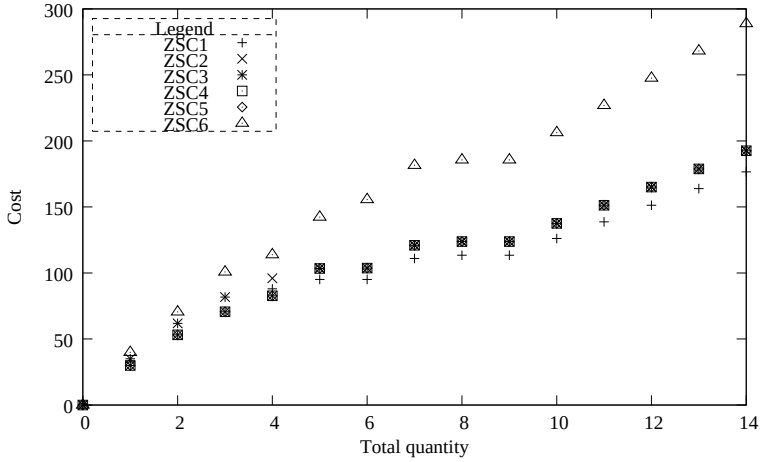
Figure 1.6: Additional distance costs (ADC)



The DTC is coupled with a cost that prevents the vehicles from travelling too much, as the DTC doesn't depend on the total distance travelled. This cost is an **additional distance cost** (or ADC). This is a lane-based cost which also depends on the total distance travelled by the vehicle, but once again it is not an obvious function, as this cost is zero under a certain distance, and becomes linear from that point. Examples of ADC functions for 6 different lanes in the same instance are given on figure 1.6.

A noticeable exception concerns shuttles, as they usually belong to the 3PL. They are only supposed to travel between two fixed and nearby sites, and they have a very low cost due to their use. Consequently, if possible, it is always preferable in terms of cost to use a shuttle than a standard vehicle that can be available anywhere.

Figure 1.7: Zone-skipping costs (ZSC)



Hub costs

The use of transshipment in a hub is charged with a cost depending on the hub required for this transshipment. In addition, there are facilities in the hubs to load and unload vehicles quickly. These facilities are hard to position when a vehicle arrives at a hub, and are distinct for loading and for unloading material. As a result, some time and some cost of handling are charged to any vehicle wishing to load some products after unloading some other ones, and vice-versa.

Shipment-related costs

Of course, any unperformed shipment will be penalized by a significant cost, usually such that it prevents the carrier from unperforming any shipment. Attached with shipments, lateness and earliness penalties are also often used, as a shipper may not be satisfied with a shipment arriving not on time at its destination.

Finally, the cost of using a ZSH is called a **zone-skipping cost** (or ZSC) by ILOG TPO. Indeed, the carrier is charged a cost for not performing the whole shipment and for "skipping" the end (or the beginning) of the route, after (or before) the ZSH. The zone-skipping cost depend on the total quantity of products left in a given ZSH. It is a growing piecewise-linear function, but from a certain level, it becomes linear. Figure 1.7 depicts 6 examples of such zone-skipping costs.

Synthesis of the costs

To sum up, the total cost considered by the 3PL using ILOG TPO can be written as follows:

$$C^{\text{total}} = C^{\text{penalty}} + C^{\text{hub}} + C^{\text{DTC}} + C^{\text{ADC}} + C^{\text{ZSC}}$$

Conclusion

The transportation problem we face is part of the Supply Chain framework and contains many features that are typical of the 3PL who want to optimize their vehicle routes. Transshipment is a key component that allows the goods to be transported into distinct vehicles along their trip, and it leaves more chances for the decision taker to reduce the total distance travelled by the vehicles. The multiple contract-based costs are charged on each vehicle and depend mainly on the starting and ending sites of a vehicle route. In particular, a shipment can be delivered through a regional platform but an additional cost has to be paid for this shipment alternative.

Chapter 2

Vehicle routing, location and flow: description and methods

RÉSUMÉ DU CHAPITRE

Dans ce chapitre, nous décrivons les différents aspects du problème en les rapprochant des travaux déjà effectués.
Une large section est consacrée au problème de tournées de véhicules, qui est une forte composante du problème. Ce problème très étudié depuis un demi-siècle a de nombreuses extensions, en particulier le problème de ramassage et livraison (*Pickup and Delivery Problem* ou PDP) qui est plus proche de notre problème. Nous pouvons également nous intéresser aux fenêtres de temps dans ces problèmes, ainsi qu'à des possibilités comme le partage des chargements (*Split Loads*). Tous ces problèmes peuvent être résolus de multiples façons. Les approches exactes basées sur les formulations mathématiques et éventuellement sur le Branch&Bound sont privilégiées pour les instances de taille raisonnable. Les très grandes instances sont généralement résolues avec des méthodes approchées, comme la recherche locales ou les métaheuristiques. Ces méthodes approchées se basent principalement sur des voisinages classiques des tournées de véhicules, comprenant des modifications intra-tournées (à l'intérieur d'une même tournée) et inter-tournées (entre deux tournées différentes). Les métaheuristiques ont l'avantage, sur les grandes instances, d'atteindre une solution raisonnable en un temps limité. De plus, elles permettent l'introduction de multiples caractéristiques industrielles propres au problème.
Les deux autres aspects abordés sont les problèmes de flots et de localisation de hubs. Les problèmes de flots de coût minimum sont également des problématiques classiques de la Recherche Opérationnelle. Il s'agit d'acheminer des produits dans un réseau dont les arcs possèdent un coût et éventuellement des capacités, de façon à minimiser le coût total. Lorsqu'un seul produit est considéré, il existe des algorithmes simples permettant de résoudre efficacement ce problème. Dans notre cas, nous introduisons plusieurs produits, ce qui complexifie le problème.

Malgré cela, des méthodes de résolution efficaces ont été proposées pour résoudre les problèmes de multiflot de grande taille, dont certaines utilisent notamment des métaheuristiques. La localisation de hubs est un problème où on se propose d'ouvrir des hubs de façon à pouvoir acheminer les produits de façon efficace sur le réseau. En général, à chaque couple origine-destination sera associé deux hubs : l'un proche de l'origine, l'autre proche de la destination. Plusieurs formulations existent pour ce problème. L'une en particulier est quadratique, et mieux adaptée aux approches utilisant des métaheuristiques.

Although the complexity of the problem discussed in the previous chapter makes it difficult to solve, it has some separate aspects for which there have been a significant research interest for several decades. The routing problem described in chapter 1 is in fact a conjunction of a Pickup and Delivery Problem (section 2.1), a Network Flow on the commodities between the hubs (section 2.2) and a Location Problem (section 2.3) to determine which hub is likely to be useful, and for which shipment. These problems have been widely studied, and we also give here some solving methods that are often used.

2.1 Vehicle routing problems

The main feature of the problem that we tackled in the context of this PhD is the routing aspect. Some commodities have to be picked up at some sites and delivered at other sites by a set of vehicles, minimizing the total cost.

For over half a century, vehicle routing problems have been studied by researchers [Dantzig and Ramser, 1959] because of the variety of real problems they contribute to solve. Despite the wide variety of vehicle routing problems and the effort of researchers [Savelsbergh, 1988] to classify such problems, there is no clear classification of vehicle routing problems. That is why they are usually referred to using their acronym. In subsection 2.1.1, we briefly introduce the Vehicle Routing Problem (VRP), and in subsection 2.1.2, we present the well-known Pickup and Delivery Problem, which is very close to our problem. Subsection 2.1.3 is a review of its famous extensions, whereas subsection 2.1.4 deals with some related routing problems. Finally, some classical solving methods are outlined in subsection 2.1.5.

2.1.1 The Vehicle Routing Problem

The problem described in chapter 1 has a deep vehicle routing aspect, as vehicle routes have to be found at a minimum cost in order to perform requests of customers at several sites. Let's call I the set of vertices (or

sites), and the vehicle set is K. If the problem is defined with a depot (a special site where all the vehicles start and end their route) usually referred to as site 0, I^* stands for $I \setminus \{0\}$. In the following, the word "site" will be preferably used instead of "node" or "vertex" in a graph. The reason for this is that the actions involving goods are performed in "sites", and this chapter is supposed to clarify the aspects in which the real-life problem of chapter 1 is close to literature theoretical problems.

The basic **Vehicle Routing Problem** (or VRP) is a generalization of one of the most popular problems in Operations Research known as the **Traveling Salesman Problem** (or TSP). In the TSP, a route visiting all the vertices of an undirected graph (also called a hamiltonian cycle) has to be found, minimizing the total distance traveled. The VRP generalizes the TSP as several routes are allowed, and all the cycles must go through a common site, the depot. Being a generalization of the TSP, which is an $\mathcal{NP}$-hard problem [Garey and Johnson, 1979], the VRP is $\mathcal{NP}$-hard as well.

There are numerous simple VRP extensions that are given substantial interest in research. In the Capacitated VRP (CVRP), a capacity constraint prevents the vehicles traveling on the routes from serving too many sites. In addition, there might be a non-unitary demand on each site corresponding to the quantity of products that have to be delivered.

Another extension, the directed VRP, is such that the graph is directed. It is often the case when for example the distances between the sites are asymetric (if d_{ij} is the distance between sites i and j, $\exists i, j \in I, d_{ij} \neq d_{ji}$). In many cases, the distances are also supposed to satisfy the **triangle inequality** ($\forall i, j, l \in I, d_{il} + d_{lj} \geq d_{ij}$). Note that if the triangle inequality holds, the basic (uncapacitated) VRP has necessarily an optimal solution with only one route, and hence this VRP can be solved as a TSP. The reason for this is that two routes, let's say $(0i_1 \dots i_n 0)$ and $(0j_1 \dots j_m 0)$, can always be merged into a single route $(0i_1 \dots i_n j_1 \dots j_m 0)$ without increasing the total travelled distance.

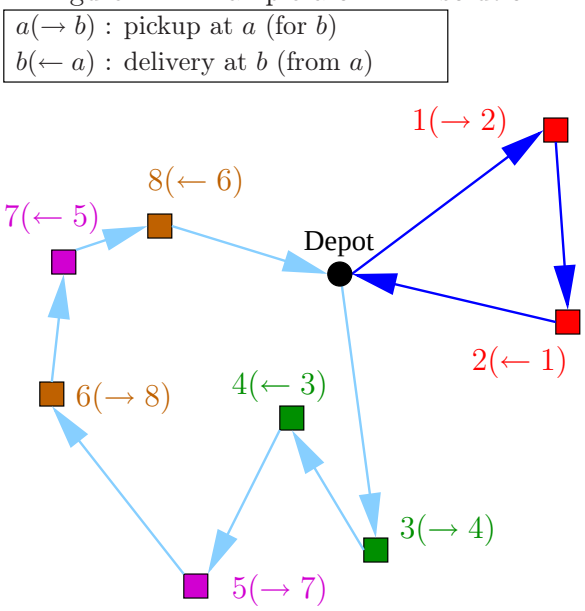
In the following, when referring to the VRP, we suppose that all the previous extensions (capacity constraints, directed graph, triangle inequality) hold.

2.1.2 The Pickup and Delivery Problem

The problem consists of shipments to be picked up and delivered at some places. This is the basic framework of a well-known problem in vehicle routing : the **Pickup and Delivery Problem** (PDP). The PDP generalizes the VRP described in the previous subsection. This makes the PDP an $\mathcal{NP}$-hard problem as well.

The VRP can be seen as a PDP in which the vehicles start at the depot with all the pickups for their route, and each visit to a site of the route can be associated with a delivery. On the other hand, in the PDP, the pickups

Figure 2.1: Example a of PDP solution



take place on other sites than the depot. Hence, there are both pickup and delivery sites. Figure 2.1 is an example of a PDP solution, where one of the vehicles travels for a short route, picking up a product at site 1 before delivering it at site 2.

A survey on the Pickup and Delivery Problem with various extensions and particular cases can be found in [Savelsbergh and Sol, 1995].

Sites

The pickup and delivery sites are often represented as vertices in a directed graph. The graph can also be undirected, in the symmetric case, which means that for every couple of sites (i, j), the distances d_{ij} and d_{ji} are equal, and the same holds for travel times.

Vehicles

Usually, a homogeneous fleet of vehicles is considered, and each vehicle has a capacity that the total amount of shipments it is carying can't exceed. Every vehicle must start and end its route at the depot. Either a fixed number of vehicles is available at the depot, or this number is unlimited but it is included in the objective function as a criterion that has to be minimized.

Shipments

A set of transportation requests (or shipments) has to be performed by the vehicles. Each request consists of two steps:

- a pickup of a given quantity of products in one of the sites,
- a delivery of these products to a final site.

In the standard PDP statement, it is assumed that both the pickup and the delivery of a request must be served by the same vehicle. This can be referred to as a pairing constraint. The precedence constraint between the pickup and the delivery visit induces that the graph of the PDP solution has to be directed.

Objective

Depending on how the problem is solved, the optimization criterion may be of various types. However, in the basic form of the PDP, the objective is usually to minimize the number of vehicles used, or to minimize the total distance travelled by the vehicles. The first one is usually included in a multicriterion analysis [Czarnas et al., 2004, Ombuki et al., 2006, Dell'Amico et al., 1993], but may be minimized alone, especially:

- on big instances as it is a simple comparison tool between various heuristics,
- in problems of type Dial-A-Ride for the transportation of elderly or handicapped people (or taxi companies), as the vehicles and the drivers are the most expensive.

More abstract criteria might be considered, such as the visual attractiveness of the routing plan [Poot et al., 1999], for which one wants to minimize the number of crossings within routes or between two routes, or the average distance between two visits in a route.

2.1.3 Extensions

The PDP has some famous extensions that make it even more realistic and useful. [Golden et al., 2002] describes some of the real-life applications that require the PDP to be extended with numerous features. Some of these extensions were described in subsection 1.3.3.

Time features

The basic PDP is formulated without any time aspect, even though a time dimension is implicit with the precedences on the pickup and delivery visits. However, the time dimension is often explicitly introduced to make the

modeling more realistic [Desaulniers et al., 2002, Thangiah, 1995]. The time dimension may be used to include some additional constraints to the modeling, but also to add a time component in the objective function. Indeed, it seems natural to try to minimize the total duration of the trips. Note that the travel time between two sites is not necessarily proportional to the distance between them, as the vehicle speed may depend on factors such as the road type (freeway / township road) or on traffic.

One of the most famous features of these time aspects are the **time windows** (TW). The **PDP with Time Windows** (PDPTW) is one of the most studied of VRPs and PDPs [Dumas et al., 1991, Mitrović-Minić, 1998]. We present this with respect to the PDP, but it is exactly the same feature in the VRP.

By definition, a time window is a time period in which an action has to take place. The vehicles still have to visit the pickup site before the delivery site for each shipment to carry on, but each visit must be performed within the site time window. For example, a site may be opened between 2 and 4, and any vehicle visiting this site can neither do it before 2, nor after 4.

In this problem, a cost is added to the objective if a request is allowed to be performed out of the time window. In this case, the carrier has to pay a penalty, and this has to be taken into account when minimizing the costs [Jung and Haghani, 2000].

Another feature can be introduced as well, especially when the objective is to minimize the total travel time: the **service times** at some sites. These are useful to depict the time spent by vehicles on sites, which is mainly the handling of products (pickups or deliveries). Every time a vehicle goes through a site, this amount of time is added to the total route duration for this vehicle. This route duration may be included in the cost function. It may as well, in some papers, be bound to be less than a given value. This models a constraint on the driver's working duration. Usually, service times are not modelled as they can be included in transit times between sites.

More complex time constraints are sometimes introduced in models that try to be as realistic as possible. These constraints are often closely related to legislation: the drivers may be forced to have a pause after 10 hours on the road, or after 15 working hours [Xu et al., 2003]. These pauses are referred to in ILOG TPO as breaks (see subsection 1.3.3).

The time dimension may be introduced as a multi-period system (e.g. [Michel, 2006]). In this case, time is not considered as a continuous dimension, but is discretised into several periods. If the time period is a quarter of an hour, a vehicle won't travel during one hour, but will leave at period p and arrive at period $p + 4$. The main advantage of this is to simplify the modeling, and reduce the problem size. On the other hand, modeling a problem this way is sometimes impossible, or leads to a significant lack of precision.

Time sometimes allows each vehicle to have several distinct routes, as

long as the time constraints aren't violated.

Depots

Usually, the PDP is presented only with the 1-depot option: every vehicle starts and ends its route at a special site called the depot. However, more depots can be considered [Dell'Amico et al., 1993, Irnich, 2000]. Each vehicle would then have to start and end its route at one of the depots that are available. A usual example for this kind of problems is the transportation of people, such as the **Dial-A-Ride Problem** (DARP) [Cordeau and Laporte, 2003] and the **Handicapped person Transportation Problems** (HTP) [Toth and Vigo, 1997]. In these problems, it is better for the carrier to have several depots, because the route duration is here closely related with the customer satisfaction, and hence it is very important not to have too long routes. As a result, the depots must be uniformly distributed over the area, so that any request isn't too long to perform.

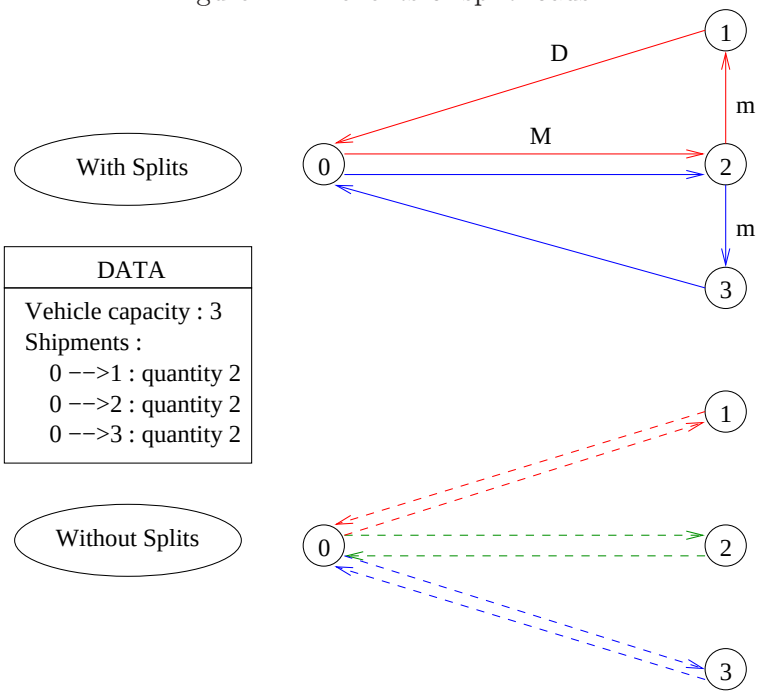
In some other problems, it is assumed that there is no real structure to park and store vehicles. In the PDP with no depot, each vehicle has its own starting and ending locations (both are not necessarily the same site). In some other cases, the vehicles have no predetermined starting and ending location, which means that it is possible to let them start or end their route anywhere. For example, when a shipper contracts a carrier out to pickup and deliver its products, he won't have to take the availability of the vehicle into account, as the carrier may have several contracts with other shippers that allow him to relocate appropriately the vehicle fleet. Therefore, every vehicle of the fleet will be expected to be available anywhere, and may be left anywhere as well.

Vehicle fleet

Some carriers have few vehicles, and their fleet is dedicated to a certain kind of jobs (for example, LTL carriers). In this case, their needs are restraint and it is sufficient for them to have a homogeneous vehicle fleet. But most of the carriers (especially the biggest ones) have vehicles that are of various types. This heterogeneous fleet makes the problem more complex, as some vehicles may be used for a certain kind of task, and some others for other tasks (see section 1.1). In addition, their costs are different and using a vehicle instead of another can have a huge impact on the total routing cost. However, this heterogeneity can often be seen as a "piecewise homogeneity" because vehicles are unlikely to be all different one to another and can be grouped into "subfleets". The heterogeneity of the vehicle fleet is often stated in real-life contexts [Al-Khayyal and Hwang, 2007].

Another aspect can be underlined about the vehicle fleet, which is its size. The carrier may own a given number of vehicles, and all of them are

Figure 2.2: Benefits of split loads



supposed to be used. This is usually referred to as fixed size fleet. On the other hand, this fleet size can be either fixed or infinite, but the number of vehicles used may appear in some way in the objective function (in a fixed cost per vehicle used, for example).

At last, some research is carried out on single (or single-vehicle) VRPs or PDPs, which stands for the fact that the fleet is made of only one vehicle [Van Der Bruggen et al., 1993]. Note that the basic single VRP is the well-known TSP.

The PDP with Split loads

Most of the shippers do not wish their shipments to be split into several parts and transported in various vehicles. However, it is in some cases tolerated, because it may lead to a significant gain in costs. Indeed, figure 2.2 illustrates an example where splitting loads decreases the total distance traveled by the vehicle.

Without splitting, the vehicle, that has a capacity of 3, can only load one shipment at a time. As a result, it has to go back and forth three times, for a total distance of $dist_1 = 4D + 2M$.

In the case splitting loads is allowed, the second shipment can be split into 2 loads, and the vehicle makes only two tours. The total distance is here $dist_2 = 2M + 2D + 2m$, which is less than $dist_1$ as soon as $D > m$.

This remark can be generalized [Nowak et al., 2007].

Splitting loads also enables each site to be visited several times, which is not permitted (and useless anyway) in the basic PDP where splitting loads is not allowed.

Additional constraints

More and more research effort is placed on real life problems due to the growing possibilities of calculation and the huge needs of big transportation companies. The PDP has then to be adapted to meet the new requirements in realism and completeness of the modeling. The basic PDP considers routing the products ignoring what is happening inside the vehicles or inside the sites. Both of the following features have a look at the load of a vehicle and how it is organised.

In lots of PDPs, the products are restricted to certain kinds of vehicles, as well as they may not be put together with other products. Indeed, a frozen product is supposed to be transported in a vehicle that has freezing compartments, and for obvious security reasons, two dangerous chemical products may not be carried together in the same vehicle. These **incompatibility constraints** are becoming a major real-life PDP feature and they are introduced in numerous papers (e.g. [Poot et al., 1999]).

One can also be concerned about the way the products are placed inside a vehicle. How can a pallet be delivered quickly if it is in the back of the vehicle, and several other pallets have to be moved in order to handle this one? A simple way to solve this problem is to introduce new constraints usually called LIFO (Last In - First Out) constraints. Recall that these constraints enforce the delivery of product p_1 before product p_2 if p_1 was picked up after p_2. The example given on figure 2.1 isn't admissible with respect to this constraint, since the product coming from node 5 ($5 \rightarrow 7$) is pickup up and delivered before the product coming from node 6 ($6 \rightarrow 8$). However, there is no convention in naming this additional feature, as in some papers it is called **LIFO constraints** [Carrabs et al., 2007], whereas in some others it is referred to as nested precedence constraints [Xu et al., 2003]. It may even be generalized to two-dimensionnal packing constraints inside each vehicle [Iori et al., 2007], generalizing the capacity constraint as well.

2.1.4 Related routing problems

For a better modeling of our problem, we need to stress some problems related to the PDP. Here, we will only focus on the ones that are of interest for our research, but there may be others that are worth studying.

The Vehicle Routing Problem with Pickups and Deliveries

As a reminder, in the VRP, the vehicles have to deliver the demand at every site, supposing all the pickups are performed at the depot. This may be called a "one-to-many" problem, as there is one pickup site, and many delivery sites. In the **Vehicle Routing Problem with Pickups and Deliveries** (VRPPD) however, the depot isn't the only pickup site. There may be several pickup sites and several delivery sites (which makes it a "many-to-many" problem), generalizing the basic VRP. As a consequence, this problem is $\mathcal{NP}$-hard as well.

The **Vehicle Routing Problem with Backhauls** (VRPB) is an extension of the VRPPD in which the vehicles are supposed to be filled at the depot at the beginning of each vehicle route. However, in the VRPB, there are two separate aspects: "one-to-many" and "many-to-one". A vehicle can't deliver a product if at least one pickup has been made, which ensures the LIFO constraints aren't violated. The vehicles then have to deliver their whole load on the first sites of their routes ("one-to-many" aspect), before they backhaul picking up the products to deliver back to the depot ("many-to-one" aspect).

The Multicommodity Vehicle Routing Problem

The Multicommodity Vehicle Routing Problem with Pickups and Deliveries (MVRPPD) generalizes both the PDP and the VRPPD. In the PDP, each shipment has only one origin, and only one destination. In the VRPPD, only one product is considered, and some sites have a demand for this product, and some other ones have an offer. For the MVRPPD it is the same, but there are several products. The aim is then to find vehicle routes such that each demand is satisfied, minimizing either the number of vehicles used, or the distance or travel time. Of course, it is only possible if, for each product, the sum of the overall offers is greater than the sum of the demands. See table 2.1 for a better understanding of the differences between some of the vehicle routing problems. There is no need in introducing a simple multicommodity vehicle routing problem (without pickups and deliveries), as the quantities of products could be added in every site to make it a 1-product problem. However, it would be justified with additional constraints, such as compatibility.

Note that the MVRPPD has only few references in the literature (unlike the VRPPD [Desaulniers et al., 2002, Nagy and Salhi, 2005]), in spite of the multiple practical applications. However, this problem is very similar to a problem introduced by [Savelsbergh and Sol, 1995], the **General Pickup and Delivery Problem** (GPDP). What distinguishes the MVRPPD from the GDPD is that in the MVRPPD, a product may be, in the same route, picked up and delivered several times, in any order (provided the quantity of

Table 2.1: Comparison of some vehicle routing problems

	PDP	VRP	VRPPD	MVRPPD
Number of products	p	1	1	p
For each product :				
Number of sites with offer	1	1	n_1	n_1
Number of sites with demand	1	n_2	n_2	n_2

this product in the vehicle never falls below 0, of course). On the other hand, in a GPDP route, all the deliveries for a given product must be performed after all the pickups.

Vehicle routing with transshipment

In various papers on vehicle routing, **transshipment** is taken into account in the problem modeling. This transshipment aspect is often coupled with other realistic features, such as a heterogeneous vehicle fleet, time windows or LIFO constraints. There are several ways the authors stress the fact that two vehicles might swap part of their cargo in a hub.

[Mues and Pickl, 2005] tackle a Volkswagen truck dispatching and they mainly focus on a 1-hub network on a Pickup-and-Delivery Problem. Two papers [Armacost et al., 2002, Armacost et al., 2004] deal with an aircraft package delivery service by UPS, on a 1-hub network as well, and where all the pickups occur before visiting the hub, and all the deliveries after the hub visit. It is the same basic framework for [Irnich, 2000] with only one hub and a partition of the pickups and the deliveries before and after the hub. For [Lapierre et al., 2004], with uncapacitated vehicles and several hubs, the density of the packages (implying both weight and volume) imply some complex costs and decisions about delivering directly or through a hub are necessary.

All these references show that for the moment, there is no conventional way of including transshipment into a Vehicle Routing Problem, and the way the problem is modeled and tackled depends on the authors' needs.

2.1.5 Solving the vehicle routing problems

It is well-known that the VRP is a $\mathcal{NP}$-Hard problem, like many of the famous routing problems (such as the PDP), as a generalization of the well-known Traveling Salesman Problem (TSP). Such as every problem belonging to $\mathcal{NP}$, there is a great interest in finding the best approach which could provide the best solution in reasonable time. Many researchers have worked to find algorithms and methods for Vehicle Routing Problems, either for

exact solutions or for approximate solutions. A wide review of methods for the PDPTW is given by [Mitrović-Minić, 1998].

Formulations

As for many other problems, researchers frequently formulate a vehicle routing problem as a linear program or a mixed-integer program. It is a useful way to expose the problem faced, and mathematical models are very often the first step of a solving procedure.

Most of the routing problems have an exact linear modeling that is either obvious, or easily computable.

The simplest and most natural modeling is based on a vehicle flow for the basic Vehicle Routing Problem. It is called an **arc-based model**, as the only variables for this model are x_{ij}, with (ij) any arc of the network. They are integer and indicate if a vehicle travels through arc (i, j). Recall that I is the set of all sites, and $I^* = I \setminus \{O\}$ includes all the sites, except the depot. The set of arcs inside set S is denoted by $\alpha(S)$, whereas the arcs from set S to $I \setminus S$ (resp. from $I \setminus S$ to S) belong to $\delta^+(S)$ (resp. $\delta^-(S)$). In addition, let $\alpha = \alpha(I)$ be the set of all arcs in the network. Note that the graph of sites is not necessarily complete. For any set of sites S, $r(S)$ denotes the minimum number of vehicles needed to serve S. If we call c_{ij} the travel cost between sites i and j, the integer programming model can be written as follows:

$$\min \sum_{(ij)\in\alpha} c_{ij}x_{ij} \tag{2.1.1}$$

subject to :

$$\forall i \in I, \qquad \sum_{(ji)\in\alpha} x_{ji} = \sum_{\substack{j\in I \\ (ij)\in\alpha}} x_{ij} \tag{2.1.2}$$

$$\forall i \in I^*, \qquad \sum_{(ij)\in\alpha} x_{ij} = 1 \tag{2.1.3}$$

$$\sum_{(0j)\in\delta^-(I^*)} x_{0j} \leq |K| \tag{2.1.4}$$

$$\forall S \subseteq I^*, S \neq \emptyset, \qquad \sum_{(ij)\in\delta^-(S)} x_{ij} \geq r(S) \tag{2.1.5}$$

$$\forall (ij) \in \alpha, \qquad x_{ij} \in \{0, 1\} \tag{2.1.6}$$

Constraints (2.1.2) are flow constraints that ensure that there will be the same number of vehicles arriving in and leaving site i. Constraints (2.1.3) enforce each non-depot site i to be served by exactly one vehicle. On the other hand, at most $|K|$ vehicles are used and start their route at the depot, which is imposed by constraint (2.1.4). Constraints (2.1.5) are useful to

eliminate vehicle subtours (which are tours not containing the depot), as well as to express the capacity restrictions. Recall that $r(S)$ is the minimum number of vehicles needed to serve set S. This number can be replaced by a lower bound, which can be calculated in several ways.

There are some well-known arc-based models for vehicle routing problems with several additional features, such as the time aspects. In some of them (particularly in problems including time windows), the time dimension is depicted by a new index on variables, meaning that it is separated into short periods. For example, a variable x_{ijt} may indicate if arc (ij) is taken by a vehicle during time period t. One of the main problems of this modeling is that either the time periods are short and numerous (and many variables have to be introduced), or they are large and few, but it may be a too weak relaxation of the basic problem.

Another way to model the time aspects is to introduce an index k for vehicles (x_{ijk} would then indicate if arc (ij) is taken by vehicle k) and new continuous variables depicting the date when an action is performed [Desaulniers et al., 2002]. T_{ik} may be the variable representing the time when site i is served by vehicle k. Sometimes, boolean variables y_{ik} depict the fact that a site i is served by vehicle k [Golden et al., 1977a]. This kind of modeling is chosen especially when in addition, vehicles have to be individualized (when considering a heterogeneous vehicle fleet, for example).

As such models generally contain lots of constraints (there may be several ones for each arc and each vehicle), other models, based on a smaller number of constraints but a greater number of variables, have been introduced by [Balinski and Quandt, 1964] and are still often used. These models don't refer to arcs explicitely. Instead, some variables are indexed on vehicle paths (if the vehicles have to start and end their route at a depot, the paths can be called routes instead). For any vehicle k, if the set of possible paths is denoted by $P(k)$, then the routing cost for vehicle k driving through path $p \in P(k)$ is c_{kp}. In addition, an indicator belong(i, k, p) states whether site i belongs to path p for vehicle k. For any site i, demand(i) is the amount of products that has to be picked-up (or delivered) at i. For any vehicle k, cap(k) denotes its capacity. Boolean variable θ_{kp} will be equal to 1 if vehicle k drives through path p. The model induced by these variables is a **set-partitioning formulation** that usually has the following constraint basis:

$$\min \sum_{k \in K} \sum_{p \in P(k)} c_{kp} \theta_{kp} \tag{2.1.7}$$

subject to :

$$\forall i \in I^*, \qquad \sum_{k \in K} \sum_{p \in P(k)} \text{belong}(i,k,p)\theta_{kp} = 1 \tag{2.1.8}$$

$$\forall k \in K, \qquad \sum_{p \in P(k)} \theta_{kp} \sum_{i \in I^*} \text{belong}(i,k,p)\,\text{demand}(i) \leq \text{cap}(k) \tag{2.1.9}$$

Constraints (2.1.8) ensure that every site is served by exactly one vehicle. Note however that the capacity constraints (2.1.9) are usually removed from this formulation as they are included instead in the subproblem in column generation procedures. We could obtain a **set-covering formulation** by replacing equality (2.1.8) by a '$\geq$' inequality. This formulation has the same optimal solution as the set-partitioning formulation detailed above [Bramel and Simchi-Levi, 2002]. The main advantage for the set-partitioning modeling is that it can be easily strenghtened with various kinds of constraints (e.g. time constraints) by just constraining the set of possible paths for each vehicle.

Other kinds of models have been introduced, but there is not yet as many references as the two presented above. [Baldacci et al., 2004] proposed a two-commodity flow model based on the two-commodity flow model introduced for the TSP by [Finke et al., 1984], where the commodities are the combined load and the residual capacity in a vehicle.

Exact methods

Both the arc-based and the path-based models introduced previously are hard to solve when they are applied to big instances (where $|N|$ and $|K|$ are big). The path-based model is even far too large to be loaded in a solver, because of the exponential number of variables (there are about $|I|!$ variables if the graph is complete).

As a consequence, researchers have been working on efficient methods either to reduce their size, or to use them as a part of the optimization process. A comprehensive survey of recent improvements in exact methods for the VRP can be found in [Baldacci et al., 2007].

The **Branch-and-Cut** (B&C) technique, which is designed to solve such models, is clearly detailed in [Mitchell, 2002] and we will present an overview here. At the root node of the search tree, the **linear programming relaxation** (or LP relaxation) is solved to optimality. Then either all the variables have integer values, in which case the optimal solution of the integer program (or IP) has been found, or branching is applied on a variable x with a fractional value v. In both branches corresponding respectively to $x \leq \lfloor v \rfloor$ and to $x \geq \lceil v \rceil$, the same process is applied until the optimal IP solution is found. This is the basis of a **Branch-and-Bound** (B&B) procedure. However, the LP relaxation may contain far too many constraints

to be solved efficiently. The arc-based models introduced above are in this case, for instances of large size (hundreds of sites and vehicles). There may also be a need to add valid inequalities to the formulation in order to provide better lower bounds during the solving. The better these lower bounds, the faster the optimal solution is obtained as more branches are pruned during the search. In the case cuts are added to the model, several constraints are dropped off the formulation, and added only when they become violated at some point in the search tree. The separation procedure is a part of this cutting-plane algorithm that decides which of the constraints in the pool are violated and should be inserted in the next LP. If no other constraint can be added or the optimal solution to the basic LP is found, then the B&B process continues with the new lower bound given by the cutting-plane algorithm.

The cutting-plane process may also be inserted during the Branch-and-Bound part of the algorithm and not only at the root of the search tree. In addition, alternative branching strategies may be used.
[Clochard and Naddef, 1993] proposed to branch, not on a variable, but on an inequality, which was then implemented for the VRP by [Augerat, 1995] and [Augerat et al., 1995].

This Branch-and-Cut algorithm can be applied to any of the models presented above. As the arc-based model has an exponential number of subtour constraints of type 2.1.5, they are usually relaxed and inserted in the pool of valid inequalities in the cutting-plane algorithm. They may also contribute to provide the formulation for more cuts [Baldacci et al., 2007]. Based on previous works on the TSP, Naddef and Rinaldi also propose valid inequalities for the VRP based on the arc formulation, the **comb inequalities** [Naddef and Rinaldi, 2002].

On the other hand, the exponential number of variables prevents any solver from loading the path-based model, even for instances of medium size. To overcome this kind of issues, **column generation** is a well-known technique where only a small number of the numerous variables (or the **columns**) are inserted in the IP at a time. At each iteration, the columns that have a negative reduced cost are chosen via a subproblem that is, in most of the cases, linked with the dual problem of a constraint subset of the main IP, and is rather easy to solve (but stays $\mathcal{NP}$-hard). The new variables are then added in the formulation and the process goes on, until no column of negative reduced cost have been found, meaning that the current solution is optimal. The whole exact algorithm is referred to as **Branch-and-Price** in many papers. An overview of a column generation method for the VRP is given by [Chabrier, 2006]. The subproblem is an elementary shortest path problem but the elementary-path constraint is often relaxed because of the high complexity it induces for this problem. Note that the valid inequalities for the set-partitioning formulation can be derived from those for the two-index arc model, by using the following correspondance :

$$\forall (ij) \in \alpha, \quad x_{ij} = \sum_{k \in K} \sum_{\substack{p \in P(k) \\ p \ni (ij)}} \theta_{kp}$$

As told before, the fact that the definition of the vehicle paths can be constrained by various additional features (e.g., time windows, route maximal length) makes the path-based model adapted to any problem having many of such constraints [Choi and Tcha, 2007]. It can be applied to the PDP [Sol and Savelsbergh, 1994] as well.

The last exact approach that will be shortly presented here is **dynamic programming**. It is an algorithm for a problem that is decomposable into overlapping subproblems, and where at each step the best decision must be taken using the recursivity of the subproblems. Every dynamic programming-based algorithm can be written as a search for a shortest path in a graph [Martelli, 1976], which makes this problem a very good example to solve with a dynamic program. Vehicle routing problems can also be solved with dynamic programming, especially the PDP or its extensions for which it is the first known exact algorithm [Psaraftis, 1980]. In the real-life applications, the dynamic programming approach is used mainly as a comparison tool or a lower bound provider, as it is exponential in the computing time.

In [Xu et al., 2003], the dynamic programming approach is combined with a column generation algorithm embedded in an approximate approach. It is applied to the linear relaxation of the column generation problems, in order to get lower bounds for the main problem. This dynamic programming algorithm is based on the Dijkstra algorithm for the shortest path problem, that will be described in section 2.2.

Approximation methods

Due to the fact that the PDP (and the main extensions) are $\mathcal{NP}$-complete, the exact methods are only effective for relatively small problems (not over 100 sites). For larger instances, they can be unefficient, as they would require too much time to provide a good solutions. That's why there is also a tremendous research effort on approximation methods, for which it may be difficult to have an idea of the quality of the solution obtained, but that are much faster. The last example of the previous subsection shows that approximate and exact methods can collaborate in order to get faster to a good solution.

Heuristics

As the VRP and the PDP are tackled in many papers, plenty of methods to solve them and their extensions have been introduced since the 1970s. As the aim of this chapter is to present an overview, we will focus on the most famous ones here. A wide range of heuristics for the VRP is given in [Laporte and Semet, 2002].

The classical heuristics are usually classified into two groups:

- constructive heuristics, that build the solution from scratch,
- improvement methods, that try to improve a feasible solution by exchanging parts of routes or sites with others.

The most famous **constructive procedure** is the **savings algorithm** proposed by [Clarke and Wright, 1964] for the basic VRP. The idea is to start from a solution with $|I| - 1$ minimal routes, each of them starting at the depot and going directly to a site and back to the depot. Note that this solution may not be feasible if the maximum number of vehicles allowed is under $|I| - 1$. Then, iteratively, the current solution is greedily improved by merging two routes, based on the notion of savings. If the first route is $(0...i^- i 0)$ and the second one $(0 j j^+ ...0)$ then the resulting route will be $(0...i^- i j j_+ ...0)$. The choice of the two routes depend on the benefit induced by the replacement of the routes by the merged one. Considering all the couples of routes of the current solution, the couple that will be chosen for the merging is the one with a maximum benefit. This benefit (the savings) can be calculated simply, as it is only the benefit of replacing arcs $(i0)$ and $(0j)$ by arc (ij). Therefore, the saving to be considered is $s_{ij} = c_{i0} + c_{Oj} - c_{ij}$. A list of non-decreasing savings is computed, and updated as soon as two routes are merged. At each step, the routes that are chosen for the merging are the ones corresponding to the first saving of the list. The Clarke and Wright algorithm is simple and efficient, and it is still one of the favourite heuristics for vehicle routing researchers. It has been improved, in particular by adding a parameter λ in $s_{ij} = c_{i0} + c_{Oj} - \lambda c_{ij}$, or by optimizing the way of sorting the savings list.

Amongst the constructive heuristics, there is also a great interest in the **decomposition methods**. These methods divide the problem into several phases. For the basic VRP or PDP, "cluster first - route second" procedure is a two phase-heuristic for which the requests are first grouped into clusters each of which will be served by one vehicle exactly, and then the routing is decided from these clusters. One of the simplest cluster first - route second method is the sweep algorithm [Golden et al., 1977b], where a ray is rotated around the depot. The sites are sequentially added to the current cluster until it is full (i.e. exceeds the vehicle capacity or violates additional constraints such as the maximal route length), in which case a new cluster is started, and each vehicle route corresponding to a cluster is obtained by solving a TSP on the cluster. Other methods are used to form the clusters, amongst which the **Generalized Assignment Problem** (GAP) [Fisher and Jaikumar, 1981] or a location heuristic to determine the cluster seeds [Bramel and Simchi-Levi, 1995], in the case the number of vehicles is fixed or can be computed in a preprocessing heuristic.

"Route first - cluster second" algorithms, where a TSP tour is first formed on the whole graph and is then decomposed into several tours (by solving a shortest path problem), has also been treated but it seems that the results are not as competitive as the previous methods.

In real-life routing problems, decomposition approaches are often chosen because their main advantage is to simplify the problem by creating two or more aspects that are supposed to be easier to solve. In routing problems with transshipment, the papers seldom deal with the whole problem, but only a part of the problem. [Doerner et al., 2000] tackle the Full Truckload (FTL) transportation problem that arises between the hubs. They assume that in a PDP with transshipment, the shipments are transshipped at an outbound and an inbound hubs in their path between their pickup and their delivery. The shipments resulting from the consolidation of the initial shipments at the outbound hubs are then of FTL type and have to be routed to the inbound hubs. FTL's main consequence here is that at any time, there may be only one shipment in a vehicle. [Grünert et al., 1999] decompose the problem of a letter-mail delivery into on the one hand a Direct Flight Problem, which assigns every origin-destination pair (or shipment) to a flight and that can be modelled as a fixed-charge network flow problem with side constraints, and on the other hand a Road Route Planning Problem similar to a Road Network Design Problem.

In some papers, usually those dealing with realistic situations, a simplification of the problem is considered, mainly due to a particularity of the instances that are supposed to be solved. Researchers often introduce only one hub for transshipment, as most of transportation companies are reluctant to open several hubs at a time. Moreover, it is sometimes supposed that all the pickups take place before all the deliveries, with a visit to the hub in between. [Armacost et al., 2002, Armacost et al., 2004] propose a simple formulation for an aircraft network design with such features, as well as a formulation based on extreme routes in order to get better bounds when solving the linear relaxation. In [Irnich, 2000], a set-covering formulation is introduced and the number of routes is reduced by computing a time window for the hub, depending on the time window of each request.

At last, some simple construction heuristics are known to provide acceptable solutions very quickly, such as some **sequencial insertions**. In such methods, only one route is generated at a time, and a new route is started only when the previous one can't be added any additional request. For example, a greedy method called "nearest (customer) addition" adds to the current route the nearest customer to one of the customers already in the route. Such algorithms usually don't provide high-quality solutions, unless they are coupled with improvement heuristics, as there may be no real interest in spending a lot of time in finding a good first solution when the improvement method is efficient enough.

Every **improvement algorithm** starts from a solution that it seeks to

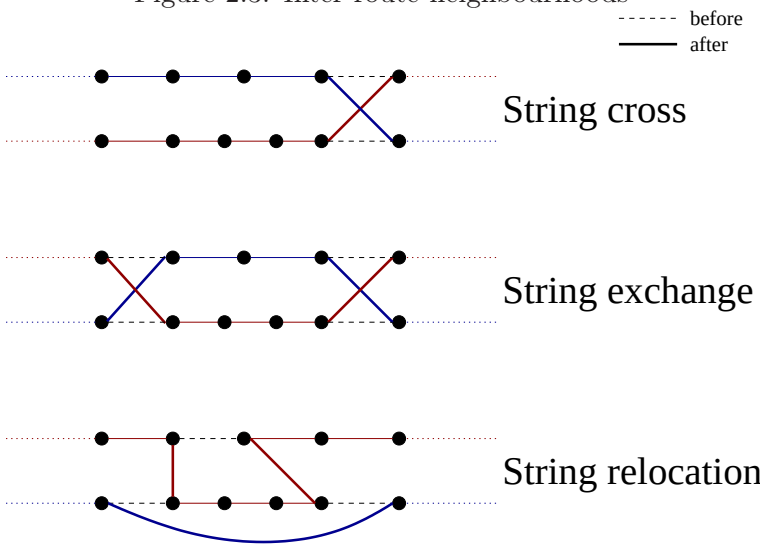
improve after a sequence of moves. Such an improvement algorithm is based on local search, and choses at each move the best candidate of the neighbourhood of the current solution. The **neighbourhood** of a solution is a set of closely related solutions. For example, if a solution can be defined by a sequence of bits, one of its neighbour could be obtained by inversing the value of one of its bits. The solution would then have a neighbourhood of size the length of the sequence of bits. In some algorithms, the whole neighbourhood is explored and the best neighbour is chosen as the next solution, whereas in others, the first neighbour improving the current solution is kept. In any case, the search is stopped as soon as it has found a solution with no improving neighbour. The search has then reached a **local minimum**.

The general definition of a neighbourhood has a considerable impact on the performances of the algorithm. In every paper dealing with an improvement algorithm, the chosen neighbourhood is described carefully, as it has to be complete (any solution can be obtained from any other solution through a sequence of moves), but not too large, which would require too much computing time at each move. On the other hand, a too small neighbourhood would make the point of view more local and the solution search would be more likely to be quickly trapped into a local optimum.

In vehicle routing, the neighbourhoods are usually exchanges of sites or arcs inter or intra-route. Almost all the **intra-route neighbourhoods** for the VRP are particular cases of Lin's λ-Opt for the TSP [Lin, 1965], where λ edges (or arcs in the directed version) are removed from the solution and the remaining parts of the route are reconnected in every possible way. As this general neighbourhood may be too large for big values of λ (the number of possible reconnections is exponential), λ is restricted to small values, up to 3, or even 4 but with some conditions on the reconnections. The **Or-Opt** neighbourhood has been introduced by [Or, 1976]. It consists of moving a sequence of 3, of 2 and finally of 1 sites at another place in the route. More sophisticated intra-route neighbourhoods exist, but the most used ones have been described above, as they are simple to implement and give good results in reasonable time.

The particular neighbourhoods for the VRP with respect to the TSP are the **inter-route neighbourhoods**. Edges or arcs are removed from several routes and reinserted in other routes. As for intra-route moves, there are generic neighbourhoods of which a lot of the well-known inter-route neighbourhoods are particular cases. [Thompson and Psaraftis, 1993] introduce the b-cyclic k-transfer, where at most k requests are chosen and transferred between b routes in a cyclic way, i.e. requests from route r_1 are transferred to route r_2, those from route r_2 to route r_3, ..., those from route r_b to route r_1. As well as Lin's λ-Opt, these neighbourhoods are too large for big values for k and b. Therefore, it is often assumed that $b = 2$, and various famous particular cases are derived from this assumption [Van Breedam, 1994] (see figure 2.3):

Figure 2.3: Inter-route neighbourhoods



- **string exchange**: strings from 2 distinct routes (possibly of different lengths) are exchanged,
- **string cross**: 2 routes are crossed by exchanging the endpoint for one of their edges,
- **string relocation**: one string is moved from one route to another,
- **string mix**: best move between string exchange and string relocation.

All these moves described above can be used for local improvement heuristics. But recent methods have used such neighbourhoods as well : metaheuristics.

Metaheuristics

As heuristics, some **metaheuristics** improve a solution through local search. As a result, the notion of neighbourhoods described previously still holds. The main feature differentiating metaheuristics from standard heuristics is that the non-improving moves are allowed, and sometimes infeasible solutions are part of the search process. Metaheuristics are designed so as to prevent the search from being trapped in a local optimum. Most of the metaheuristics have been inspired from a physical process or a natural rule.

The most simple metaheuristic is probably **tabu search**, and it is one of the favourite one amongst VRP and PDP researchers, mostly due to it flexibility and the possibility to customize it through several parameters. Tabu search has the same basis as the local improvement heuristic described above, but it allows a solution to be worse than the previous one. To avoid looping the search on the same solutions, a tabu list is updated at each

move, and is consulted as soon as a solution is candidate for a move. This list contains either the last solutions, or the last moves, which decreases the chance of looping and allows a wider exploration of the search space. Any solution forbidden by the tabu list will be considered as "tabu" and rejected. The solution that is considered as tabu may be better than the best solution found so far. In some cases, an aspiration criterion is introduced to allow this solution to be used, even if it is tabu. [Kelly and Xu, 1999] solve a basic VRP with a two-phase heuristic using a tabu search strategy. In the first phase, a big number of routes are generated either with several settings of a weighted savings heuristic followed by a 3-Opt improvement, or with a tabu search heuristic using a complex neighbourhood where a solution may exceed the capacity constraint, but is charged a penalty. In the second phase called Integration, some routes are selected thanks to a tabu search method on a set-partitioning formulation based on the routes generated in the first phase.

Simulated annealing imitates annealing in metallurgy, allowing the quality of a material to be improved. A material is heated to allow its atoms to move from their initial position, and slowly cooled back down to make them find another position, possibly better than the previous one. In simulated annealing, a solution is randomly chosen with a probability depending on the difference between its objective value and the one of the previous solution, and on the temperature, a parameter that decreases slowly during the search and such that the randomness is more important when the temperature is high (at the beginning of the algorithm). [Tavakkoli-Moghaddam et al., 2007] solve a split service VRP with Simulated annealing and 1-Opt and 2-Opt operators adapted to split loads. In the modified 1-Opt neighbourhood, for example, a site that is served by one vehicle is not relocated in another vehicle but it is served by one more vehicle, splitting its service in the two vehicles.

A metaheuristic that simulates the development of a population through selections, crossovers and mutations is called a **genetic algorithm** (or GA). Unlike the previous ones, this metaheuristic does not improve a solution through local search. In a GA, each solution consists of a set of several decisions (or genes), called a chromosome, and each gene is involved in the evolution of the population of solutions. The first population is made of randomly generated solutions, and at each step of the algorithm, only the most promising individuals are kept for evolution, depending on an evaluation criterion called fitness. Then, crossover (or recombination) and mutation operators have to be defined to allow the population to get to the following generation. [Thangiah and Nygard, 1992] use a two-phase algorithm for the Multicommodity Transshipment Problem. A GA is used for the first phase to generate the trunk routes, i.e. between the hubs, and another GA coupled with a clustering heuristic determines the feeder routes, i.e. "around" the hubs. Whereas [Grünert et al., 1999] solve their Direct Flight Problem using a hybrid Tabu Search and Branch and Bound strategy as the neigh-

bourhood evaluation is based on a decision tree, their Road Network Design Problem is solved with an evolutionary algorithm (i.e. a GA) that assigns a departure time and a vehicle type to each route going through a hub. A set S_1 of such hub trips is recombinated with another S_2 by considering subsets of hub trips of both sets that have the same origin and the same destination, and choosing randomly, for each such subset, the one corresponding to S_1 or to S_2.

Ant colony algorithms exploit the ant collaboration system to generate a good solution. Initially, the ants wander randomly in the graph and after reaching their goal, they return back to their colony. Each ant lays down a given quantity of pheromone while traveling, and tend to be attracted by the pheromones dropped by other ants previously. The pheromone evaporates gradually, meaning that the paths containing the most pheromone will be the shortest paths. But like in most of the metaheuristics, a random parameter is kept to prevent the ants from always traveling through the same path. In their time-constrained FTL transportation problem, [Doerner et al., 2000] use an ant colony algorithm embedding a dedicated construction heuristic, and based on a set-partitioning model.

Social behaviours are also simulated through **particular swarm algorithms**. As for ant colony optimization, the collaboration of plenty of individuals (or particles) tend to build a good solution. The collaboration here is based on the motion of each individual, depending on its best position so far, the current best position in its neighbourhood and the best position ever for all the particles. Hence, a velocity vector is necessary to compute each particle's next position. An example of such an algorithm for the VRPTW is given by [Zhu et al., 2006].

Other metaheuristics exist, but they have not yet yield to significant results for the VRP and the PDP compared with other approaches.

It seems that metaheuristics are very powerful when they are applied to real-life problems, as the customer instances coming from these problems are usually of large size, and the metaheuristics provide a way to get to a good solution in reasonable time. Usually, the problems are even so huge that they are decomposed and each part of the decomposition is solved by a metaheuristic. As shown in some of the examples above, it is usually the case for PDP with many additional features, such as transshipment.

2.2 Network flow

If the vehicle routing aspect of the problem we face is neglected, the commodities have to be routed in a **network flow** between the hubs. Indeed, the hubs allow the goods to be routed through a path that doesn't depend on the vehicle routes.

Network flow is one of the most famous problems in Operations Research

Figure 2.4: Example of flow

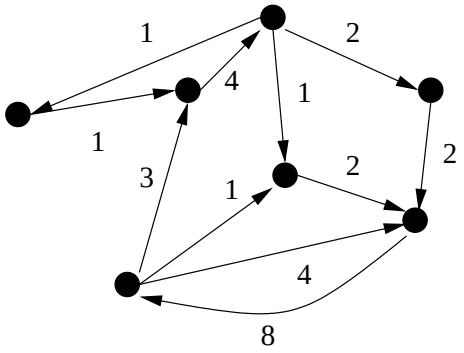
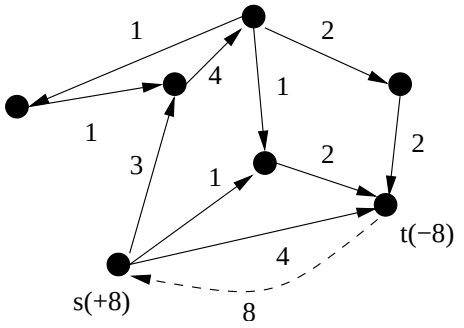


Figure 2.5: Flow between a source and a sink

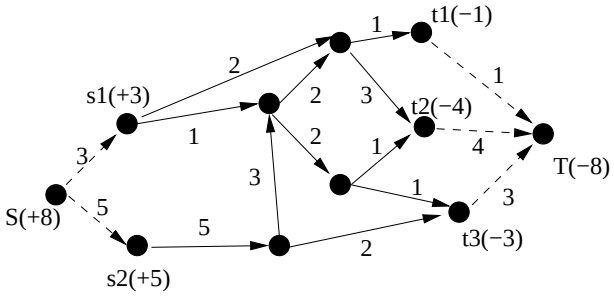


and has been applied in a wide variety of domains (such as transportation) [Ahuja et al., 1993]. There are various types of network flow problems, but in this thesis, only the minimum cost flow problem is of interest. In a directed graph with costs on arcs (and possibly capacities), a minimum cost flow has to be found. Each site has a demand or an offer in the commodity. A flow is a valuation on the arcs such that the total quantity of products entering any vertice has to be equal to the total quantity leaving this vertice (see figure 2.4). It is known in Physics as the Kirchhoff law. By definition, a flow is a valuated cycle on the graph.

By extension, a flow may also be non-cyclic and defined between a source site (with an offer) and a sink site (with a demand equal to the offer on the source). It is equivalent to the cyclic case as it suffices to introduce an additional arc between the sink and the source, on which the valuation is constrained to be equal to the offer of the source and with a null cost. For example, on figure 2.5, if 8 units of flow have to be transported between s and t, then it can be changed into a cyclic flow by adding the dotted arc with value 8. Note that to constrain an arc to have a given valuation v, the usual way is to add capacity constraints and impose the lower bound to be v as well as the upper bound.

In some flow problems, there are several sources and several sinks. To

Figure 2.6: Flow between several sources and several sinks



come back to a flow problem with a unique source and sink, let's just add a pseudo source S and a pseudo sink T, and arcs between S and each initial source and between each initial sink and T, such that the flow on the arc is equal to the offer (resp. demand) of the initial source (resp. sink), and the cost is 0. Figure 2.6 is an example of such a correspondance, with new source S having an offer of 8 and new sink T with a demand of the same amount.

If we suppose that the number of vehicles is fixed on each arc, then we adress a capacitated minimum cost flow problem. If the vehicles routes are determined only in a postprocessing, then the problem is uncapacitated. If several commodities have to be transported in the graph, then a **multicommodity flow** problem has to be tackled. Schrijver presents a theoretical survey of the multicommodity flow in [Schrijver, 2003].

2.2.1 Formulations

Let c_{ij} denote the cost per flow unit on arc (ij), $b(i)$ the offer or demand on site i ($b(i) > 0$ in case it is an offer, $b(i) < 0$ if it is a demand). l_{ij} and u_{ij} are the lower bound and upper bound of the flow on arc (ij). ϕ_{ij} is the flow variable indicating how much flow is going through arc (ij). A capacitated minimum cost flow problem can be modelled as follows :

$$\min \sum_{(ij)\in\alpha} c_{ij}\phi_{ij} \tag{2.2.1}$$

Subject to :

$$\forall i \in I, \qquad \sum_{(ij)\in\alpha} \phi_{ij} - \sum_{(ji)\in\alpha} \phi_{ji} = b(i) \tag{2.2.2}$$

$$\forall (ij) \in \alpha, \qquad l_{ij} \leq \phi_{ij} \leq u_{ij} \tag{2.2.3}$$

This is the **node-arc formulation**. (2.2.1) is the objective function indicating that the total cost has to be minimized. Constraint (2.2.2) is the

Kirchhoff law that enforce the flow conservation before and after a site and constraint (2.2.3) is the capacity constraint

When dealing with multicommodity network flows, another index k has to be introduced to depict the commodity. In the following, the set of commodities will be referred to as C. In the capacity constraints (2.2.3) as well as in the objective (2.2.1), the references to ϕ_{ij} become $\sum_k \phi_{ijk}$.

Another model, the **arc-path formulation**, uses all the possible elementary paths from a commodity source to a sink. Note that here, it is necessary to assume the unicity of the source and the sink, for each commodity. Let $s(k)$ and $t(k)$ be the source and the sink of commodity k, respectively. $d(k)$ is the total amount that has to be routed from $s(k)$ to $t(k)$. Moreover, the lower bound on flow is supposed to be 0 here since otherwise, there could be cycles in every optimal solution and the decomposition into elementary paths would be impossible.

$$\min \sum_{k \in C} \sum_{(ij) \in \alpha} c_{ij} \sum_{\substack{p \in P(k) \\ p \ni (ij)}} \theta_{kp} \tag{2.2.4}$$

Subject to :

$$\forall k \in C, \qquad \sum_{p \in P(k)} \theta_{kp} = d(k) \tag{2.2.5}$$

$$\forall (ij) \in \alpha, \qquad \sum_{k \in C} \sum_{\substack{p \in P(k) \\ p \ni (ij)}} \theta_{kp} \leq u_{ij} \tag{2.2.6}$$

Side constraints and nonlinear costs are more and more introduced to be able to tackle real-life problems such as telecommunication applications. [Mahey and De Souza, 2007] use complex piecewise convex costs to model the discounts obtained by expanding the capacity on arcs. On the other hand, [Croxton et al., 2007] deal with nonconvex piecewise linear costs that arise in many transportation or telecommunication problems. Time requirements on paths are handled in [Holmberg and Yuan, 2003] by using a column generation approach.

2.2.2 Solving network flows

Basic network flow is a problem known for its very powerful dedicated algorithms. One of the most famous is the Ford and Fulkerson algorithm [Ford Jr. and Fulkerson, 1962]. It is based on a residual graph, in which the direction of the arcs depend on the flow on the arc of the initial graph, where the minimum-cost augmenting path has to be found and can be exploited to improve the current solution. Note that as there can be negative-cost arcs in the residual graph, the algorithm used to find a shortest path in the residual graph is Bellman-Ford [Bellman, 1958].

On the other hand, the multicommodity flow problem is known to be $\mathcal{NP}$-hard, even with few commodities. The exact solving methods for the minimum cost multicommodity flow problem are mostly based on linear programming or on the simplex algorithm. A wide review of such methods can be found in [Assad, 1978] and [Kennington, 1978]. More recently, [Bonnans et al., 2000] experiment a column generation scheme on the arc-path formulation and compares it with a standard interior point method. [Larsson and Yuan, 2004] solve large scale multicommodity flow problems with a Lagrangean relaxation coupled with a penalty approach.

Metaheuristics can also be chosen to solve problems either with a large size or with a high complexity level such as nonlinear costs.

[De Souza et al., 2008] embed a cycle canceling procedure in a tabu search framework for a multicommodity network flow problem with separable piecewise convex costs. In [Walkowiak, 2004], the author applies an ant colony algorithm and studies the influence of the algorithm's settings on the efficiency of the algorithm. [Tu et al., 2005] model a water distribution problem into a multicommodity flow over an undirected network. A genetic algorithm provides directions for all arcs, and a population fitness is evaluated thanks to a generalized reduced gradient procedure.

2.3 Location problems

In addition to the routing aspect, we have to determine the hubs that will be necessary for transshipment. As every hub has a fixed opening cost, this subproblem can obviously be related to a location problem.

Location is a wide area having many practical applications. The most famous problem from this domain is the warehouse location problem, in which some warehouses can be opened at predefined locations, but at a certain cost, and each warehouse can supply a maximum number of stores, and for each store to be served, there is an additional cost, usually depending on the distance between the warehouse and the store.

In our case, we have to tackle a **hub location problem**. In this problem, given an amount of flow that must be transported between some pairs of nodes, a set of hubs must be opened to reduce the total transportation cost. The goods travel in a hub-and-spoke network, meaning that they need to go through an outbound hub next to their origin, and an inbound hub next to their destination before being delivered. The hubs can often be chosen amongst all the sites of the network. A comprehensive review of hub location problems is given by [Alamur and Kara, 2008].

Note that the hub location problem only implies shipping goods. The vehicle routes are not decided here.

2.3.1 Formulations

O'Kelly first proposed a two-index quadratic formulation for the basic single allocation hub location problem [O'Kelly, 1992]. In this hub location problem, each site is allocated to only one hub. I is the set of sites and H is the set of potential hubs ($H \subseteq I$). Let d_{ij} be the amount of flow that has to be transported from site i to site j and c_{ij} stand for the total cost per unit of flow from site i to site j. c_{ij} is usually decomposed to take into account each of the three parts of the path of the commodity going from i to j ($i \to k \to l \to j$):

$$c_{ij} = \chi c_{ik} + \alpha c_{kl} + \delta c_{lj}$$

where k and l denote the outbound and inbound hub respectively. Note that k and l are not ambiguous as i has only one associated hub, as well as j. χ, α and δ are the coefficients for collection, transfer and distribution.

In this model, the decision variables are x_{ik}, stating whether site i is allocated to hub k. Note that hub k is opened (at a cost f_k) if and only if $x_{kk} = 1$ (hub k is allocated to itself).

$$\min \sum_{i\in I}\sum_{j\in I} d_{ij} \sum_{k\in H} \chi c_{ik} x_{ik} + \sum_{i\in I}\sum_{j\in I} d_{ij} \sum_{l\in H} \delta c_{lj} x_{jl} + \sum_{i\in I}\sum_{k\in H} x_{ik} \sum_{j\in I}\sum_{l\in H} x_{jl} \alpha c_{kl} d_{ij} + \sum_{k\in H} x_{kk} f_k \quad (2.3.1)$$

Subject to:

$$\forall i \in I, \qquad \sum_{k\in H} x_{ik} = 1 \quad (2.3.2)$$

$$\forall i \in I, k \in H, \qquad x_{kk} \geq x_{ik} \quad (2.3.3)$$

$$\forall i \in I, k \in H, \qquad x_{ik} \in \{0,1\} \quad (2.3.4)$$

Constraints (2.3.2) ensure that every site is covered by exactly one hub, constraints (2.3.3) enforce a hub to be opened if a site is allocated to it, and constraints (2.3.4) are the boolean constraints on variables x_{ik}. Note that, in the case we know the number p of hubs to be opened, constraint (2.3.3) can be replaced with the following constraint:

$$\forall k \in H, \quad (|I| - p + 1) x_{kk} \geq \sum_{i\in I} x_{ik} \quad (2.3.5)$$

In this case (called p-hub median problem), we also have to include the following constraint into the formulation:

$$\sum_{k\in H} x_{kk} = p \quad (2.3.6)$$

A multiple allocation version of this problem exists, and in this case, a site can be allocated to several hubs at the same time. [Campbell, 1994] introduces a 4-index Mixed Integer Linear Programming model to solve this problem. Here, the variables y_{iklj} are continuous and positive and indicate the quantity of flow travelling from i to j through hubs k and l. Boolean variables x_{kk} stating if hub k is opened are also needed. The MILP model is then the following:

$$\min \sum_{i,j \in I} \sum_{k,l \in H} c_{iklj} y_{iklj} + \sum_{k \in H} f_k x_{kk} \tag{2.3.7}$$

Subject to:

$$\forall i, j \in I, \quad \sum_{k,l \in H} y_{iklj} = d_{ij} \tag{2.3.8}$$

$$\forall i, j \in I, \forall k \in H, \quad \sum_{l \in H} y_{iklj} \leq M_{ij} x_{kk} \tag{2.3.9}$$

$$\forall i, j \in I, \forall l \in H, \quad \sum_{k \in H} y_{iklj} \leq M_{ij} x_{ll} \tag{2.3.10}$$

$$\forall i, j \in I, \forall k, l \in H, \quad y_{iklj} \geq 0 \tag{2.3.11}$$

$$\forall i \in I, \quad z_i \in \{0, 1\} \tag{2.3.12}$$

Constraints (2.3.8) ensure that all the flow will be tranfered from i to j. Constraints (2.3.9) and (2.3.10) make sure that if a commodity goes through a hub, this hub is necessarily opened, provided $M_{ij} \geq d_{ij}$. These constraints are known to be **big-M** constraints, which are linear constraints replacing non-linear ones by introducing a new (possibly big) constant M. This usually results in a slowdown of the solving of the LP relaxation, especially for really big values of M. In our case, $M_{ij} = d_{ij}$ is reasonably small.

In this model, there is no constraint about the single allocation of a site to a hub. However, if necessary, it is possible to introduce a new variable x_{ik}, such as in the quadratic model, and a constraint of the type (2.3.2).

As in the previous model, the cost per unit of flow can be rewritten: $c_{iklj} = \chi c_{ik} + \alpha c_{kl} + \delta c_{lj}$. Usually, $\chi = \delta = 1$ whereas $\alpha < 1$, in order to facilitate consolidation and the use of inter-hub routes. Some researchers believe that this cost modelling is not realistic enough and that the discount of using inter-hub routes should increase with the flow. The cost function should then be concave increasing, rather than linear. [Racunica and Wynter, 2000] introduce nonlinear discount functions on interhub routes and between hubs and standard sites. They also use a new variable, called $\hat{x}_{iill}$, to allow the possibility for a request to be transported directly from its origin to its destination, without being transshipped in a hub.

Note that even if the number of hubs to open is fixed, the problem is $\mathcal{NP}$-hard, as the allocation part of the problem is already $\mathcal{NP}$-hard [Kara, 1999].

For the multiple allocation problem, [Campbell, 1994] also proposed a model with boolean variables x_{iklj} standing for the fraction of flow from i to j that is routed via hubs k and l. In this formulation, the integrality constraints on variables x_{iklj} are not needed as the whole flow between i and j can always be routed through the least-cost hub pair.

2.3.2 Solving hub location problems

In many cases, metaheuristic approaches are chosen to solve this problem. The reason for this is that the size of the 4-index model is too big to solve large instances, and the quadratic model is then chosen. One of the main advantages of metaheuristics is that they usually don't need the cost function to be linear.

[Topcuoglu et al., 2005] use the quadratic formulation to solve the single allocation hub location problem with a genetic algorithm. The initial population is not generated at random, but the sites with a big amount of traffic are selected as hubs with higher probability. [Cunha and Silva, 2007] consider a variable discount factor α depending on the amount of freight between hubs, and they solve the problem using a genetic algorithm combined with simulated annealing.

Some researchers still apply methods based on Branch-and-Bound to solve such problems. [Cánovas et al., 2007] propose a dual-ascent technique embedded in a Branch-and-Bound solving. On a capacitated version of the problem (with capacities on hubs) and where direct connections are allowed, [Aykin, 1994] presents a Branch-and-Bound algorithm with Lagrangean relaxation to reduce the computing time by providing lower bounds.

Conclusion

Many solving techniques have been proposed for each of the three well-known problems introduced in this chapter. The use of the result and the size of the instances on which the problem is raised decides whether to choose an exact method or an approximate one. Nonetheless, these two kinds of methods can be mixed in multilevel methods, especially for large instances for which a decomposition approach is necessary.

Chapter 3

The arc-based formulations

RÉSUMÉ DU CHAPITRE

Le problème décrit dans le chapitre 1 est complexe, et il n'est pas aisé de le modéliser mathématiquement sans écarter une de ses caractéristiques essentielles. Dans ce chapitre, nous proposons deux formulations permettant de résoudre le problème de façon exacte, ainsi que quelques améliorations permettant de rendre la résolution par Branch&Bound plus performante.
La première est une formulation dédiée au problème traité : le problème de ramassage et livraison avec transbordement et fenêtres de temps. En revanche, la deuxième est consacrée au problème de tournées de véhicules multi-produits avec ramassages et livraisons, et elle permet de traiter des problèmes où le partage des chargements est autorisé. De plus, sa taille est plus petite pour les grandes instances que celle de la première formulation. Toutes deux ont une base commune de variables et de contraintes. Elles utilisent des variables à trois indices classiques dans les problèmes de tournées de véhicules où il est nécessaire d'individualiser les véhicules. D'autres variables se retrouvent dans des études classiques, comme les variables de temps, mais elles sont ici plus générales de façon à pouvoir traiter le problème industriel avec toutes ses caractéristiques. Le principal inconvénient de la linéarisation de ces modèles est la présence de contraintes grand-M.
Pour améliorer les performances de la résolution des modèles, plusieurs procédures sont proposées. Un prétraitement visant à réduire le nombre de sites et le nombre de véhicules peut être utilisé. Un algorithme automatique de relaxation et plans coupants permet de relâcher les contraintes grand-M qui sont nombreuses et contribuent à la complexité de la formulation. Pour renforcer la formulation, nous proposons également des coupes dédiées au problème. Enfin, des priorités sont appliquées sur l'ordre de branchement des variables intervenant dans le modèle.
Les résultats sur l'ensemble des 19 plus petites instances montrent que la plupart d'entre-elles peuvent être résolues jusqu'à l'optimalité, et que l'utilisation des améliorations introduites précédemment réduit le temps de calcul.

Considering the problem described in chapter 1, each of the formulations enumerated in chapter 2 holds for a part of the problem. In this chapter, we propose to gather all the aspects into a single model. Two formulations are proposed, and both of them can solve problems that are close one to another. Each of them has its own real-life features from the original problem described in chapter 1, as some of them are too complicated to be linearized simply. Note that in both formulations, no storage is possible in any hub, meaning that the transshipment between vehicles must be done if all the vehicles involved with the transshipment are present in the hub at the same time. This is a constraint given by ILOG TPO customers, but it also makes the formulation lighter.

The first mathematical model is described in section 3.1 and is a formulation for the Pickup and Delivery Problem with Transshipment and is the closest to the problem detailed in chapter 1. It contains many additional real-life features, such as shipment alternatives. However, some of the real-life constraints described in section 1.3, such as LIFO constraints or breaks, are not taken into account here for a lighter modelling. Moreover, we will only consider one capacity dimension, which is the number of pallets.

This formulation has been linearized so that it can be solved easily and rather quickly, at least on small instances. The constraints are made of time constraints, flow constraints and some PDP boolean constraints that link the PDP boolean variables together.

Section 3.2 presents a formulation of the problem viewed as a Multicommodity Vehicle Routing Problem with Pickups and Deliveries and Transshipment (using the reduction proposed in subsection 3.2.1 as a preprocessing). This formulation can be used on a problem were split loads are allowed, but its main objective is to decrease the problem size, as the reduction proposed in subsection 3.2.1 generates few products on instances with many shipments that have the same pickup site. The main benefit of using this model instead of the previous one is the possibility of splitting loads. One of its main drawbacks is that the modelling is weaker: in particular, the time constraints related to each shipment are removed from the formulation.

Some improvements are suggested in section 3.3 in order to cope with the negative effects of the structure of the formulations, especially the presence of big-M constraints used to linearize the models.

Then, section 3.4 gathers the results obtained on small instances provided by an ILOG TPO customer.

Note that this chapter has been inspired from a paper written for the ROADEF French-speaking conference [Fournier, 2007]. However, the formulations have been modified and completed since the time this paper was written.

Table 3.1: Special features included in each formulation

Features \Models	PDP	MVRP	Path-based
Vehicle breaks	✘	✘	✘
LIFO constraints	✘	✘	✘
Transshipment	✓	✓	✓
Shipment alternatives	✓	✘	✓
Time constraints	✓	✓	✘
Time constraints for visits	✓	✘	✘
Maximal number of stops	✓	✓	✘
Split loads	✘	✓	✘
Vehicle-product incompatibilities	✓	✓	✘
Product-product incompatibilities	✓	✘	✘

3.1 The PDP model

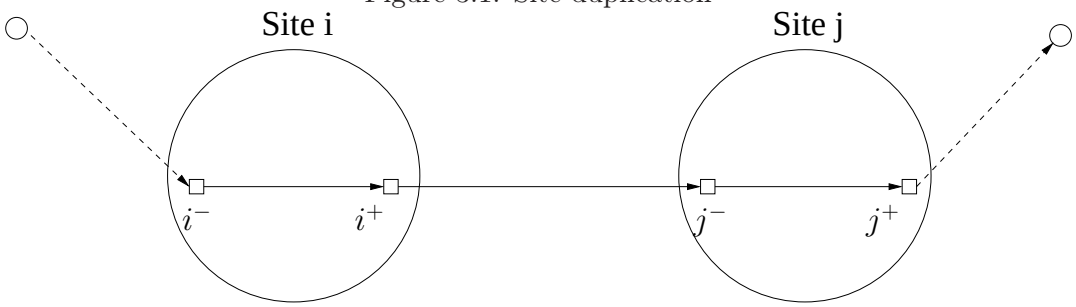
Like a lot of other problems, the PDP can be modeled as a Mixed-Integer Program (or MIP). The fact that here there are many additional constraints only enlarges the basic representation. As a base, a classical 3-index boolean variable (indicating whether a vehicle travels between two sites or not) is introduced. This is the basic framework for many mathematical arc-based models in the literature, as discussed in the previous chapter. However, the number of variables is sometimes reduced by removing the vehicle-dedicated index, and considering the number of vehicles between two sites (see subsection 2.1.5). Here, we strive to model the problem as accurately as possible. However, some features, such as vehicle breaks and LIFO constraints, are not modelled here. Table 3.1 underlines which of the real-life features are taken into account by each formulation presented in this thesis.

3.1.1 Modeling choices

As explained in subsection 1.3.2, some shipments can be either delivered to their final destination or unloaded in a nearby Zone-Skipping Hub (ZSH) in order to group the delivery of other shipments in the same area. For this shipment alternative to be included in the formulation, we need to introduce new shipment and visit objects. Then, there is a disjunction between the "real" shipment from the pickup and the delivery site, and the pseudo-shipment from the pickup site and the ZSH. The initial shipment set S is replaced by a set of shipment S', such that $S \subseteq S'$.

A generalization is possible with this kind of modeling, with over 1 alternate shipment per shipment (supposing more than one ZSH is available for a delivery, and possibly for the pickup as well). In the instances studied in

Figure 3.1: Site duplication



chapter 5, however, there can't be over 1 alternate shipment.

3.1.2 Variables

The model contains boolean and continuous variables. To make the model more efficient, the upper bounds on continuous variables must be minimized.

Writing conventions

A '$-$' superscript means that the variable is associated with the arrival of a vehicle on a site, whereas a '$+$' superscript is linked to a vehicle departure. This stands in a way for a duplication of the nodes in the initial graph. Indeed, for each site i of the network, we will introduce i^- and i^+ as the arrival and departure pseudo-sites (respectively) of site i (see figure 3.1). This has been done mainly for time reasons, in the special case where a vehicle route is closed (has the same departure site and arrival site). Indeed, in this case, the arrival time in every site of the route is less than the departure time, except for one site: the first and last site of the route. As any site is a potential first and last site for a route, the distinction between the arrival and departure of a site is necessary.

The '$\pm$' superscript is used to decrease the number of constraints. Each constraint where this character appears can be decomposed into two constraints:

- the one where every occurence of '$\pm$' are replaced by '$+$',
- the other where every occurence of '$\pm$' are replaced by '$-$'.

For example, $x^{\pm} = y^{\pm} + z^{+} + w$ would be used for:

$$\begin{cases} x^{+} = y^{+} + z^{+} + w \\ x^{-} = y^{-} + z^{+} + w \end{cases}$$

Let's suppose a shipment s is made of two visits: the pickup visit $p(s)$ and the delivery one $d(s)$. The site where visit v takes place is denoted by

Table 3.2: Data in the mathematical models

Type	Data	Meaning
Time	servTime(i)	Fixed service time on site i
	servTime(v)	Service time for pickup or delivery visit v
	travTime(i, j, k)	Travel time for vehicle k between sites i and j
	maxTravel(k)	Max. travel time allowed for vehicle k
	minTimeDep(k)	Min. departure date allowed for vehicle k
	maxTimeDep(k)	Max. departure date allowed for vehicle k
	[twLB(i), twUB(i)]	Time window on site i
	[twLB(v), twUB(v)]	Time window for visit v
Load	qty(s)	Quantity of products in shipment s
	qty(v)	Quantity of products related to visit v
	cap(k)	Capacity of vehicle k
	offer(i, p)	Quantity of product p available at site i
Site	extrem$^+$(k)	Departure site for vehicle k (if any)
	extrem$^-$(k)	Arrival site for vehicle k (if any)
Integer	maxSplit(i, p)	Max. number of splits for product p on site i
	maxNbStops(k)	Max. number of stops for vehicle k
	size(f)	Number of vehicles in fleet f

$i(v)$. Concerning subscripts, i and j are sites (including hubs), h a hub, k and l vehicles, s a shipment and v a visit (shipment pickup or delivery). They are always used in this order: (site, vehicle, shipment or visit).

Recall that a visit alternative A is a set of visits of the same type (pickup or delivery) amongst which at least one visit has to be performed. The set of all visit alternatives will be called $\mathcal{A}$. Note that a shipment alternative may be changed into a visit alternative on their delivery visits.

The last convention is that the continuous variables are referred to with uppercase letters (such as T^+_{ik}), whereas the integer variables are written with lowercase letters (such as x_{ijk}).

All the data parameter names are given in table 3.2.

Time variables

Every time variable is a real number (and bound to 0 if vehicle k doesn't go through site i), and are the following:

- the date $T^{\pm}_{ik}$ of arrival (or departure) of vehicle k at site i,
- the dates T^+_k and T^-_k of begin and end of use of vehicle k,

- the date T_v of beginning of visit v,
- the waiting times W_{ik}^- and W_{ik}^+ of vehicle k, at site i and leaving site i, respectively.

All of these variables are supposed to be continuous and positive. Note that the variables of type $T_{ik}^{\pm}$ generalize the well-known variables T_i that are introduced in various papers such as [Desaulniers et al., 2002].

Their upper bounds can be computed from the data, especially the site and visit time windows.

For any visit v, the date T_v will never be above $\text{twUB}(v)$. Let i_v be the site where the visit takes place. Then T_v won't be more than $\text{twUB}(i_v)$, either. So its upper bound will be $UB(T_v) = \min(\text{twUB}(v), \text{twUB}(i_v))$.

Let k be a vehicle for which site i is the last route site. The dates $T_{ik}^{\pm}$ have upper bounds of $UB(T_{ik}^{\pm}) = \text{twUB}(i)$. For any other case, that is either vehicle k has no ending site for its route in the data, or i isn't this ending site, the upper bound can be slightly improved by introducing the visit time windows. $T_{ik}^{\pm}$ is less than the time window upper bound of at least one visit taking place in site i. The reason for this is that in any optimal solution, vehicle k will perform at least one visit in any site i of its route (except maybe the last one), as going to site i without doing anything can be improved by skipping site i of vehicle k route (recall that the triangular inequality holds). As a result, if we call $v(i)$ a visit taking place in site i and $V(i)$ the set of all such visits, $T_{ik}^{\pm}$ will be less than $\max_{v(i)\in V(i)}(\text{twUB}(v(i)))$, and its upper bound will be $UB(T_{ik}^{\pm}) = \min(\text{twUB}(i), \max_{v(i)\in V(i)}(\text{twUB}(v(i))))$

Eventually, as for any site i and any vehicle k, $T_{ik}^{\pm} \leq T_k^-$, $UB(T_k^{\pm}) = \max_i(\min(\text{twUB}(i), \max_{v(i)\in V(i)}(\text{twUB}(v(i)))))$.

Boolean flow variables

Variables $y_{ik}^{\pm}$ stand for the arrival (or departure) of vehicle k at site i. Unlike the usual models in the literature, this variable uses the duplication described above. Variables x_{ijk} are usually used in VRP models (see subsection 2.1.5) and are bound to 1 if arc (ij) $(i \neq j)$ is used by vehicle k. Other variables indicate:

- if site i is the ending point (resp. starting point) of the tour of vehicle k: r_{ik}^- (resp. r_{ik}^+),
- the use of vehicle k (resp. of hub h): u_k (resp. z_h). Note that z_h is similar to a variable x_{hh} introduced in hub location problems (see subsection 2.3.1).

This kind of modeling induces a limitation: a vehicle can't go to the same site several times, except if nodes are duplicated.

Table 3.3: Variables of the arc-based MIPs

Model	Type	Variable	Meaning
PDP&MVRP	Time	$T_{ik}^{\pm}$	Date of arrival/departure of vehicle k on site i
		$T_{k}^{\pm}$	Date of begin/end of use of vehicle k
		$T_{v}^{\pm}$	Date of beginning of visit v
		W_{ik}^{-}	Waiting time of vehicle k at site i
		W_{ik}^{+}	Waiting time of vehicle k leaving site i
	Boolean	$y_{ik}^{\pm}$	Arrival/departure of vehicle k on site i
		x_{ijk}	Site i is just before site j on the route of vehicle k
		$r_{ik}^{\pm}$	Site i is the first/last site of the route of vehicle k
		u_k	Vehicle k is used
		z_h	Hub h is used
	Load	Q_k	Maximal quantity of products inside vehicle k during its route
PDP	Boolean	c_{iks}	Vehicle k loads shipment s on site i
		d_{iks}	Vehicle k unloads shipment s on site i
		$a_{iks}^{\pm}$	Vehicle k leaves/arrives in site i with shipment s
		b_{vk}	Visit v is performed by vehicle k
		b_{v}	Visit v is performed
		f_{vw}	Visit v is performed before visit w, on the same site
MVRP	Load	$L_{ikp}^{\pm}$	Quantity of products p in vehicle k leaving/arriving at site i
		$L_{ikp}^{=}$	Quantity of products p loaded by vehicle k in site i
	Boolean	q_{ikp}	Vehicle k loads or unloads product p in site i

Boolean PDP variables

Variables c_{iks} (resp. d_{iks}) indicate if vehicle k loads (resp. unloads) shipment s on site i, and $a^{\pm}_{iks}$ if vehicle k has shipment s arriving (resp. leaving) site i.

We also introduce variables that are linked with visits. Variables b_v and b_{vk} indicate whether visit v is performed (resp. is performed by vehicle k). Finally, variables f_{vw} states that visit v is performed before visit w, if this is relevant.

Every variable is supposed to be positive.

Table 3.3 is a complete summary of all variables introduced in this chapter.

3.1.3 Constraints

About big-M constraints

Some of the constraints introduced in this section are basically non-linear constraints, but can be modeled linearly.

Let's suppose that a given inequality holds if some boolean variables are equal to specific values. For example, if a vehicle k arrives in site i ($y^-_{ik} = 1$) and leaves this site ($y^+_{ik} = 1$), then its arrival date at site i will be lower than its departure date from this site ($T^-_{ik} \leq T^+_{ik}$).

For a generalization, let's call CST the following constraint:

$$\text{if } (\mathcal{C}) \text{ then } \mathcal{S}$$

with $\mathcal{C}$ a linear condition involving some boolean variables (x_i), and $\mathcal{S}$ a linear statement, involving some variables (not necessarily boolean) (y_j), that must be true if $\mathcal{C}$ holds.

This kind of constraints can be called a conditional constraint, because statement $\mathcal{S}$ depends on the value of some boolean variables. Note that as CST is equivalent to (not $\mathcal{C}$ or $\mathcal{S}$), it is also a disjunctive constraint. We assume that the set $\mathcal{C}$ of conditions on boolean variables (x_i) can always be written in the form: $\lambda_0 + \sum \lambda_i x_i = 0$ where $\forall i, \lambda_i$ are constant and quantity $\lambda_0 + \sum \lambda_i x_i$ is non-negative. Indeed, here is the simple corresponding between boolean constraints and numerical constraints:

- (x is true) $\rightarrow$ ($1 - x = 0$)
- (x is false) $\rightarrow$ ($x = 0$)
- (x_1 and x_2 are true) $\rightarrow$ ($2 - x_1 - x_2 = 0$)

In the case where $\mathcal{C}$ is (x_1 or x_2 are true), the constraint CST can be simply written as a conjunction of two constraints:

$$\begin{cases} \text{if } (x_1 \text{ is true}) & \text{then} & \mathcal{S} \\ \text{if } (x_2 \text{ is true}) & \text{then} & \mathcal{S} \end{cases}$$

Let's now suppose the statement $\mathcal{S}$ is an equality, which is $\mu_0 + \sum \mu_j y_j = 0$, where μ_j are fixed coefficients. Once again, the constraint can be separated into two constraints:

$$\begin{cases} \text{if } (\mathcal{C}) & \text{then} & \mu_0 + \sum \mu_j y_j \geq 0 \\ \text{if } (\mathcal{C}) & \text{then} & -\mu_0 + \sum (-\mu_j) y_j \geq 0 \end{cases}$$

As a result, in the general case, the conditional constraint CST can be written as follows:

$$\text{if } \left(\lambda_0 + \sum \lambda_i x_i = 0\right) \text{ then } \left(\mu_0 + \sum \mu_j y_j \geq 0\right)$$

where $\forall i, x_i$ and y_i are variables, whereas λ_i and μ_j are fixed coefficients, and quantity $\lambda_0 + \sum \lambda_i x_i$ is non-negative.

There are several ways of modeling such a constraint linearly. One of the simplest and most classical way is to introduce a big constant, usually called **big-M**. This kind of constraints was first described by [Williams, 1978]. The constraint CST will then be modeled as follows:

$$\mu_0 + \sum \mu_j y_j + M \cdot \left(\lambda_0 + \sum \lambda_i x_i\right) \geq 0$$

Constant M must be chosen big enough to make the constraint true whichever values for variables y_i, in the case the condition on variables x_i doesn't hold (that is: $\lambda_0 + \sum \lambda_i x_i \geq 1$). However, if constant M is too big, the optimization will require more processing time as the constraint will have huge coefficients on some variables, which is known to be a drawback on integer programs. Consequently, such a constant M must be calculated carefully and minimized for a better modeling.

The big-M computations will be precised in the following subsections each time it will be necessary.

Time constraints

The time dimension introduces a lot of constraints in the realistic model, and especially conditional constraints, as several times a condition has to hold in order to make the constraint valid.

For convenience matters, we introduce for each site i and vehicle k a variable standing for the total service time of vehicle k on site i:

$$servTime_{ik} = \text{servTime}(i) + \sum_{v(i) \in V(i)} \text{servTime}(v(i)) \cdot b_{v(i)k} \qquad (3.1.1)$$

where $v(i)$ is a visit taking place at site i, and $V(i)$ is the set of all such visits. $servTime_{ik}$ has to be handled as a variable because this time depends on the visits performed at site i.

$(\forall i, j \in I, \forall h \in H, \forall k, l \in K, \forall v, v' \in V)$

- Vehicle k travels from i to j:

$$T_{jk}^- = T_{ik}^+ + \text{travTime}(i,j,k) + W_{ik}^+ \qquad \text{if } x_{ijk} = 1 \qquad (3.1.2)$$

It has been modelled through the following linear constraints:

$$\begin{aligned} T_{ik}^+ + \text{travTime}(i,j,k) + W_{ik}^+ &\leq T_{jk}^- + M_{\text{time}}^{(3.1.2\text{a})} \cdot (1 - x_{ijk}) \\ T_{ik}^+ + \text{travTime}(i,j,k) + W_{ik}^+ &\geq T_{jk}^- - M_{\text{time}}^{(3.1.2\text{b})} \cdot (1 - x_{ijk}) \end{aligned}$$

- Link between the arrival and the departure date of vehicle k at site i:

$$T_{ik}^- + servTime_{ik} + W_{ik}^- = T_{ik}^+ \qquad \text{if } y_{ik}^- = 1 \text{ and } r_{ik}^- = 0 \qquad (3.1.3)$$

The condition given here is equivalent to ($y_{ik}^- = y_{ik}^+ = 1$ and $r_{ik}^- = r_{ik}^+ = 0$), as inferred from boolean constraints (3.1.19) and (3.1.20) described below. In other words, the left-hand part is true if vehicle k enters and leaves site i, in this order. As previously, this conditional constraint is modelled with linear big-M constraints:

$$\begin{aligned} T_{ik}^- + servTime_{ik} + W_{ik}^- &\leq T_{ik}^+ + M_{\text{time}}^{(3.1.3\text{a})} \cdot (1 - y_{ik}^- + r_{ik}^-) \\ T_{ik}^- + servTime_{ik} + W_{ik}^- &\geq T_{ik}^+ - M_{\text{time}}^{(3.1.3\text{b})} \cdot (1 - y_{ik}^- + r_{ik}^-) \end{aligned}$$

We also have to consider the case where the route of vehicle k has i as a departure site and/or arrival site, as in this case T_{ik}^+ and T_{ik}^- can't be connected:

$$T_{ik}^- + servTime_{ik} + W_{ik}^- = T_k^- \qquad \text{if } r_{ik}^- = 1 \qquad (3.1.4)$$

$$T_k^+ + servTime_{ik} + W_{ik}^- = T_{ik}^+ \qquad \text{if } r_{ik}^+ = 1 \qquad (3.1.5)$$

It is the same kind of modelling here, with 2 big-M constraints per conditional equality:

$$\begin{aligned} T_{ik}^- + servTime_{ik} + W_{ik}^- &\leq T_k^- + M_{\text{time}}^{(3.1.4\text{a})} \cdot (1 - r_{ik}^-) \\ T_{ik}^- + servTime_{ik} + W_{ik}^- &\geq T_k^- - M_{\text{time}}^{(3.1.4\text{b})} \cdot (1 - r_{ik}^-) \\ T_k^+ + servTime_{ik} + W_{ik}^- &\leq T_{ik}^+ + M_{\text{time}}^{(3.1.5\text{a})} \cdot (1 - r_{ik}^+) \\ T_k^+ + servTime_{ik} + W_{ik}^- &\geq T_{ik}^+ - M_{\text{time}}^{(3.1.5\text{b})} \cdot (1 - r_{ik}^+) \end{aligned}$$

- Simultaneity of the presence of vehicles k and l in hub h in the case they handle the same shipment s (no storage in the hub):

$$T_{hk}^- + servTime_{hk} \leq T_{hl}^+ \qquad \text{if } d_{hks} + c_{hks} = d_{hls} + c_{hls} = 1 \qquad (3.1.6)$$

modelled with the constraint:

$$T_{hk}^- + servTime_{hk} \leq T_{hl}^+ + M_{\text{time}}^{(3.1.6)} \cdot (2 - d_{hks} - c_{hks} - d_{hls} - c_{hls})$$

Note that constraint 3.1.25 defined below ensures that $d_{hks} = c_{hks} = 1$ is impossible (and the same holds for vehicle l).

- Vehicle k route duration:

$$0 \leq T_k^- - T_k^+ \leq \text{maxTravel}(k) \tag{3.1.7}$$

- Vehicle k departure time:

$$\text{minTimeDep}(k) \leq T_k^+ \leq \text{maxTimeDep}(k) \tag{3.1.8}$$

- Time window in site i:

$$\text{twLB}(i) \cdot y_{ik}^{\pm} \leq T_{ik}^{\pm} \leq \text{twUB}(i) \cdot y_{ik}^{\pm} \tag{3.1.9}$$

PDP Time constraints

In this subsection, only the time constraints involving PDP objects (such as visits) are described.

- Time window for visit v:

$$\text{twLB}(v) \leq T_v \leq \text{twUB}(v) \qquad \text{if } b_v = 1 \tag{3.1.10}$$

This time, the linearization is simpler as multiplying both the right hand side and the left hand side of the double inequality by variable b_v gives correct inequalities, even in the case where $b_v = 0$:

$$\text{twLB}(v) \cdot b_v \leq T_v \leq \text{twUB}(v) \cdot b_v$$

- Link between two visits v and v' on the same site (such that $i(v) = i(v')$):

$$T_{v'} \geq T_v + \text{servTime}(v) \qquad \text{if } f_{vv'} = 1 \tag{3.1.11}$$

modelled with the constraint:

$$T_{v'} + M_{\text{time}}^{(3.1.11)} \cdot (1 - f_{vv'}) \geq T_v + \text{servTime}(v)$$

- Link between the date when visit v is performed on vehicle k and the date of arrival at site $i(v)$:

$$T_v \geq T_{i(v)k}^- \qquad \text{if } b_{vk} = 1 \text{ and } (\exists s \in S', v = d(s) \text{ or } r_{i(v)k}^+ = 0) \tag{3.1.12}$$

The condition here has to be further detailed. Of course, for this constraint to hold, visit v has to be performed by vehicle k, hence the $b_{vk} = 1$ term. Then, the constraint only holds if visit v is performed after vehicle k has arrived in $i(v)$, which is garanteed if the route of vehicle k doesn't start in $i(v)$, or if v is a delivery visit, as no delivery can be performed at the first site of a route. Similarly, the link between

the date visit v is performed and the date vehicle k departs from site i is:

$$T_v + \text{servTime}(v) \leq T^+_{i(v)k} \quad \text{if } b_{vk} = 1 \text{ and } (\exists s \in S', v = p(s) \text{ or } r^-_{i(v)k}) \quad (3.1.13)$$

In other words, in the case v is a pickup visit, both these constraints can be modelled as follows:

$$\begin{aligned} T_v &\geq T^-_{i(v)k} - M^{(3.1.12)}_{\text{time}} \cdot (1 - b_{vk} + r^+_{i(v)k}) \\ T_v + \text{servTime}(v) &\leq T^+_{i(v)k} + M^{(3.1.13)}_{\text{time}} \cdot (1 - b_{vk}) \end{aligned}$$

whereas if v is a delivery visit, these constraints are:

$$\begin{aligned} T_v &\geq T^-_{i(v)k} - M^{(3.1.12)}_{\text{time}} \cdot (1 - b_{vk}) \\ T_v + \text{servTime}(v) &\leq T^+_{i(v)k} + M^{(3.1.13)}_{\text{time}} \cdot (1 - b_{vk} + r^-_{i(v)k} = 0) \end{aligned}$$

- Link between the date visit v is performed and the dates vehicle k starts and ends its route:

$$T^+_k \leq T_v \leq T^-_k - \text{servTime}(v) \qquad \text{if } b_{vk} = 1 \qquad (3.1.14)$$

modelled through the following constraints:

$$\begin{aligned} T^+_k &\leq T_v + M^{(3.1.14\text{a})}_{\text{time}} \cdot (1 - b_{vk}) \\ T_v + \text{servTime}(v) &\leq T^-_k + M^{(3.1.14\text{b})}_{\text{time}} \cdot (1 - b_{vk}) \end{aligned}$$

Each M_{time} computation is detailed in subsection 3.1.4.

Flow constraints

In this set of constraints, there are still conditional constraints, as given values for some boolean variables may imply a valuation on other boolean variables. However, such constraints containing only boolean variables are easy to linearize, so in most of the cases, the final constraint is given with few explanations.

$(\forall i \in I, \forall h \in H, \forall k \in K)$

- Constraint on a tour extremity, except if $\text{extrem}^{\pm}(k)$ (the potential extremity) isn't given in the data:

$$r^{\pm}_{ik} = 0 \qquad \text{if } i \neq \text{extrem}^{\pm}(k) \qquad (3.1.15)$$

- Link between variables $y^{\pm}_{ik}$ and x_{ijk}. A vehicle k can only come to (or leave) i by one way:

$$\sum_{j \in I} x_{jik} = y^{-}_{ik} \tag{3.1.16}$$

$$\sum_{j \in I} x_{ijk} = y^{+}_{ik} \tag{3.1.17}$$

  These constraints generalize constraints of type (2.1.2) and (2.1.3) from the classical VRP arc-based model described in subsection 2.1.5.

- If vehicle k leaves site i, then either it came to i, or i is the starting site of its tour. Same kind of constraint if k comes to i:

$$y^{+}_{ik} \leq y^{-}_{ik} + r^{+}_{ik} \tag{3.1.18}$$

$$y^{-}_{ik} \leq y^{+}_{ik} + r^{-}_{ik} \tag{3.1.19}$$

- If vehicle k comes to and leaves site i, it means that i is either both the starting and the ending site of the tour, or none of them (k is just passing through i):

$$y^{-}_{ik} = y^{+}_{ik} = 1 \implies r^{+}_{ik} = r^{-}_{ik} \tag{3.1.20}$$

  A way to linearize this constraint is to introduce a conjunction of two constraints, where if the condition $y^{-}_{ik} = y^{+}_{ik} = 1$ holds, the first one will ensure $r^{+}_{ik} \leq r^{-}_{ik}$ and the other one will state the reverse inequality.

$$r^{+}_{ik} - r^{-}_{ik} + y^{-}_{ik} + y^{+}_{ik} \leq 2$$

$$r^{-}_{ik} - r^{+}_{ik} + y^{-}_{ik} + y^{+}_{ik} \leq 2$$

- If vehicle k isn't used, then it doesn't come to (resp. doesn't leave) site i. Furthermore, if k doesn't come to (resp. doesn't leave) site i, i can't be the starting (resp. ending) site of its tour:

$$r^{\pm}_{ik} \leq y^{\pm}_{ik} \leq u_k \tag{3.1.21}$$

- If vehicle k comes to (leaves) hub h then h is used:

$$y^{\pm}_{hk} \leq z_h \tag{3.1.22}$$

- To be used, vehicle k must have both a starting and an ending site for its tour, but can't have over one of each:

$$u_k = \sum_{j \in I} r^{\pm}_{jk} \tag{3.1.23}$$

- Vehicle k is not allowed to stop more than maxNbStops(k) times, if this number is given in the data:

$$\sum_{i \in I} y^{-}_{ik} \leq \text{maxNbStops}(k) \tag{3.1.24}$$

PDP boolean constraints

$(\forall i, j \in I, \forall h \in H, \forall k \in K, \forall s \in S', \forall A \in \mathcal{A})$

- If vehicle k unloads shipment s on site i, it can neither load it nor have it leaving i:

$$c_{iks} + d_{iks} \leq u_k \tag{3.1.25}$$
$$a^+_{iks} + d_{iks} \leq 1 \tag{3.1.26}$$

- If vehicle k has shipment s leaving site i, then it must either have had it coming to i or have loaded it on i (but not both). If i is the starting site of k's tour, then it is the second option:

$$a^+_{iks} \leq a^-_{iks} + c_{iks} \leq 1 \tag{3.1.27}$$
$$a^+_{iks} + r^+_{ik} \leq 1 + c_{iks} \tag{3.1.28}$$

- Vehicle k must go through site i if it loads or unloads shipment s on this site:

$$a^\pm_{iks} \leq y^\pm_{ik} \tag{3.1.29}$$

- Vehicle k contains shipment s after loading, or before unloading it:

$$c_{iks} \leq a^+_{iks} \tag{3.1.30}$$
$$d_{iks} \leq a^-_{iks} \tag{3.1.31}$$

- If vehicle k travels from i to j, every shipment it carries arriving in j were also in k leaving i:

$$x_{ijk} + a^-_{jks} - a^+_{iks} \leq 1 \tag{3.1.32}$$

- Vehicles are forced to go to the delivery and the pickup sites if a shipment is performed. Let s be a shipment, of pickup visit $p(s)$ and delivery visit $d(s)$. In addition, let $i(v)$ denote the site where visit v is performed. If $i(p(s))$ is not a hub, then:

$$b_{p(s)} = \sum_{k \in K} c_{i(p(s))ks} \tag{3.1.33}$$

and in any other non-hub site i:

$$c_{iks} = 0 \tag{3.1.34}$$

The same kind of constraints holds for $d(s)$ and $i(d(s))$: If $i(d(s))$ is not a hub:

$$b_{d(s)} = \sum_{k \in K} d_{i(d(s))ks} \tag{3.1.35}$$

and in any other non-hub site i:

$$d_{iks} = 0 \tag{3.1.36}$$

- Given the same notations, we can write the flow conservation in site $i(p(s))$ or $i(d(s))$ if they are hubs:

$$\sum_{k\in K} d_{i(p(s))ks} + b_{p(s)} = \sum_{k\in K} c_{i(p(s))ks} \tag{3.1.37}$$

$$\sum_{k\in K} c_{i(d(s))ks} + b_{d(s)} = \sum_{k\in K} d_{i(d(s))ks} \tag{3.1.38}$$

For any regular hub h (that is not linked with shipment s in being its pickup or its delivery site), we can simply write:

$$\sum_{k\in K} c_{hks} = \sum_{k\in K} d_{hks} \tag{3.1.39}$$

- Capacity constraint:

$$\sum_{s\in S'} \text{qty}(s) \cdot a^{\pm}_{iks} \leq \text{cap}(k) \cdot y^{\pm}_{ik} \tag{3.1.40}$$

Note that the right-hand side term $y^{\pm}_{ik}$ isn't essential, as constraints (3.1.29) already ensure that variables $a^{\pm}_{iks}$ are set to zero if $y^{\pm}_{ik} = 0$. However, this gives better results in the linear relaxation, as a fractional value of $y^{\pm}_{ik}$ restreints a little more this constraint.

- Incompatibility constraints:

$$a^{\pm}_{iks} + a^{\pm}_{iks'} \leq 1 \qquad \text{if} \quad s \text{ and } s' \text{ incompatible} \tag{3.1.41}$$

$$a^{\pm}_{iks} = 0 \qquad \text{if} \quad s \text{ and } k \text{ incompatible} \tag{3.1.42}$$

- Exactly one of the alternate visits has to be performed:

$$\sum_{v\in A} b_v = 1 \tag{3.1.43}$$

If a visit v is not included in any alternative, or $v \notin A, \forall A \in \mathcal{A}$, it has to be performed. This can be ensured by introducing a visit alternative $A \in \mathcal{A}$ only containing visit v, and applying constraint (3.1.43) on this alternative.

- Two visits performed on the same site i have to be ordered:

$$\forall v, v' \in V(i), b_v + b_{v'} \leq 1 + f_{vv'} + f_{v'v} \tag{3.1.44}$$

Note that this constraint doesn't prevent the situation where $f_{vv'} = f_{v'v} = 1$. This is impossible due to both time constraints of type (3.1.11) on visits v and v', provided at least one of the visit service times is greater than zero.

- Link between a shipment and a vehicle:

$$b_{p(s)} \leq c_{i(p(s))ks} \tag{3.1.45}$$
$$b_{d(s)} \leq d_{i(d(s))ks} \tag{3.1.46}$$

  Recall that $p(s)$ and $d(s)$ are respectively the pickup and the delivery visits for shipment s, and $i(p(s))$ is the site where visit $p(s)$ takes place. Note, in constraint (3.1.45) for example, that it can't be an equality, as a vehicle may load a shipment in another site than its pickup site (in a hub, for instance).

- By definition, the link between variables b_v and b_{vk} is the following:

$$b_v = \sum_{k \in K} b_{vk} \tag{3.1.47}$$

- Both the pickup and delivery visits for shipment s are performed if one of them is performed:

$$b_{p(s)} = b_{d(s)} \tag{3.1.48}$$

Instead of inserting in the formulation a constraint where a variable equals a constant (0 in general), we will remove the variable from the formulation and replace it by its value in any other constraint in which it is used.

3.1.4 Big-Ms computation

The value of each M_{time} must be as small as possible but big enough to make the constraint true for each value of the other variables, in the case the factor it multiplies is non-zero.

Here are the computed values:

- In (3.1.2), we want to have the inequalities:

$$T^+_{ik} + \text{travTime}(i,j,k) + W^+_{ik} \leq T^-_{jk} + M^{(3.1.2\text{a})}_{\text{time}}$$
$$T^+_{ik} + \text{travTime}(i,j,k) + W^+_{ik} \geq T^-_{jk} - M^{(3.1.2\text{b})}_{\text{time}}$$

  always true if $x_{ijk} = 0$. $T^+_{ik} + W^+_{ik}$ is at most the arrival date at any other site than i and j (as $x_{ijk} = 0$). Consequently, $T^+_{ik} + W^+_{ik} \leq \max_{i' \in I \setminus \{i,j\}} \text{twUB}(i')$. As $T^-_{jk} \geq 0$, we can choose:

$$M^{(3.1.2\text{a})}_{\text{time}} = \text{travTime}(i,j,k) + \max_{i' \in I \setminus \{i,j\}} \text{twUB}(i')$$

  The same kind of arguments imply the following possible equality:

$$M^{(3.1.2\text{b})}_{\text{time}} = \text{twUB}(j) - \text{travTime}(i,j,k)$$

- In (3.1.3), we want $M_{\text{time}}^{(3.1.3a)}$ such as $T_{ik}^- + servTime_{ik} + W_{ik}^- \leq T_{ik}^+ + M_{\text{time}}^{(2)}$ as soon as $r_{ik}^- = 0$ or $y_{ik}^- = 1$. Here, $T_{ik}^- + servTime_{ik} + W_{ik}^- \leq \text{twUB}(i)$ as it is a date when vehicle k is still at site i. Consequently,

$$M_{\text{time}}^{(3.1.3a)} = \text{twUB}(i)$$

As we must have $M_{\text{time}}^{(3.1.3b)} \geq T_{ik}^+$,

$$M_{\text{time}}^{(3.1.3b)} = \text{twUB}(i)$$

seems to be a good choice as well. We use the same kind of justification to compute all the other big-M constants:

$$\begin{aligned}
M_{\text{time}}^{(3.1.4a)} &= \text{twUB}(i) \\
M_{\text{time}}^{(3.1.4b)} &= \max_{i' \in I \setminus \{i\}} \text{twUB}(i') \\
M_{\text{time}}^{(3.1.5a)} &= \text{twUB}(i) \\
M_{\text{time}}^{(3.1.5b)} &= \text{twUB}(i) \\
M_{\text{time}}^{(3.1.6)} &= \text{twUB}(h) \\
M_{\text{time}}^{(3.1.11)} &= \text{twUB}(v) + \text{servTime}(v) \\
M_{\text{time}}^{(3.1.12)} &= \text{twUB}(i(v)) \\
M_{\text{time}}^{(3.1.13)} &= \text{twUB}(v) + \text{servTime}(v) \\
M_{\text{time}}^{(3.1.14a)} &= \max_{i' \in I} \text{twUB}(i') \\
M_{\text{time}}^{(3.1.14b)} &= \text{twUB}(v) + \text{servTime}(v)
\end{aligned}$$

Of course, every M_{time} could have been computed more accurately by using more of the data. However, we think that it is useless to complicate the calculations as the gain of reducing each M_{time} a little would have a negligeable impact on the performances of the MIP solving.

3.1.5 Objective

Our optimization criterion is to minimize the sum of all the routing costs:

$$\min \text{obj} = \text{C}^{\text{fixed}} + \text{C}^{\text{DTC}} + \text{C}^{\text{ADC}} + \text{C}^{\text{ZSC}}$$

We suppose here that no shipment can be unperformed, and every shipment has to be performed within the time windows for its pickup and delivery visits. Therefore, no penalty cost is considered, but the model is more constrained than the basic problem described in section 1.3. Our main concern is to keep the global cost linear, in order to be able to solve the model as a MIP. Each of the costs detailed in this subsection is described in subsection 1.3.4 and is modelled as follows:

- $\mathrm{C}^{\text{fixed}}$ is the sum of the fixed costs of using a vehicle or a hub.

$$\mathrm{C}^{\text{fixed}} = \sum_{k \in K} \mathrm{C}^{\text{vehicle}}(k) \cdot u_k + \sum_{h \in H} \mathrm{C}^{\text{hub}}(h) \cdot z_h$$

  $\mathrm{C}^{\text{vehicle}}$ and C^{hub} are the unit costs of using a vehicle, and of using a hub, respectively.

- C^{DTC} is the **direct transportation cost** (DTC). Let $\mathrm{C}^{\text{DTC}}_k$ be the DTC for vehicle k, with $\mathrm{C}^{\text{DTC}} = \sum_{k \in K} \mathrm{C}^{\text{DTC}}_k$. It depends both on the first and last sites of the vehicle route, and on the maximal quantity transported during the trip. It is necessary to introduce a new (positive) variable Q_k to keep track of the maximal quantity over the route of vehicle k, and new constraints for each site i:

$$Q_k \geq \sum_{s \in S'} \text{qty}(s) \cdot a^{\pm}_{iks} \tag{3.1.49}$$

  The $|I| \cdot |K|$ capacity constraints 3.1.40 can then be replaced by the $|K|$ following constraints:

$$Q_k \leq \text{cap}(k) \tag{3.1.50}$$

  Then, new constraints have to be introduced to express the cost:

$$\mathrm{C}^{\text{DTC}}_k \geq Q_k \cdot \mathrm{C}^{\text{DTC}}(i,j) \quad \text{if } r^+_{ik} = r^-_{jk} = 1 \tag{3.1.51}$$

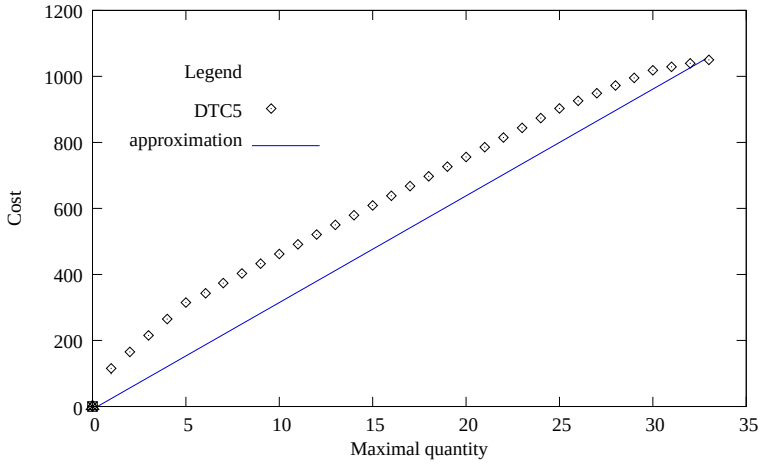
  $Q_k \cdot \mathrm{C}^{\text{DTC}}(i,j)$ is a linear approximation of the DTC of vehicle k, providing i and j are respectively the departure and the arrival site for vehicle k (Q_k being a variable and $\mathrm{C}^{\text{DTC}}(i,j)$ a constant). As shown on figure 3.2, the DTC can be under-approximated with a linear function having a value of 0 at quantity 0 and the same value as the real DTC at quantity $\text{cap}(k)$ (usually 33 or 34 pallets). This value, divided by $\text{cap}(k)$, is the value we choose for $\mathrm{C}^{\text{DTC}}(i,j)$. The main benefit in an under-approximation of the cost is that the objective value of the linear approximation will always be a valid lower bound for the initial problem. As before, a $M^{(3.1.51)}_{\text{time}}$ constant has to be introduced to linearize this new constraint:

$$\mathrm{C}^{\text{DTC}}_k \geq Q_k \cdot \mathrm{C}^{\text{DTC}}(i,j) - M^{(3.1.51)}_{\text{time}} \cdot (2 - r^+_{ik} - r^-_{jk})$$

  $M^{(3.1.51)}_{\text{time}} = \text{cap}(k) \cdot \max_{i',j' \in I} \mathrm{C}^{\text{DTC}}(i',j')$ seems to be a good choice.

  Of course, it is possible to provide a more accurate approximation of the DTC. The DTC function could be approximated by a continuous and piecewise-linear function. It would however introduce far more variables in the model and complexify the formulation, whereas the model is anyway not an utterly exact representation of the problem. On the other hand, the simple linear approximation as proposed combines accuracy and simpleness.

Figure 3.2: DTC linear approximation



- $\mathrm{C}^{\mathrm{ADC}}$ is the **additional distance cost** (ADC). Recall that the ADC depends on the first and last sites of a vehicle route, but also on the total distance travelled. If this distance is less than a given value, ADC is null. Otherwise, it is linear, starting from 0 at this breakpoint (see figure 1.6). As before, let's decompose $\mathrm{C}^{\mathrm{ADC}}$ into $\sum_{k\in K} \mathrm{C}^{\mathrm{ADC}}_k$. For each vehicle k:

$$\mathrm{C}^{\mathrm{ADC}}_k = \sum_{i,j\in I} \mathrm{C}^{\mathrm{ADC}}(i,j)\cdot D_{ijk}$$

$\mathrm{C}^{\mathrm{ADC}}(i,j)$ is the unit cost per kilometer travelled over the "free limit" $\mathrm{limit}(i,j)$, with i as the starting site of the route, and j as the ending site. D_{ijk} is the distance travelled over the limit. Of course, $D_{ijk}=0$ if i or j are not the starting or ending sites or the route of vehicle k. It is sufficient to impose the non-negativity constraint $D_{ijk}\geq 0$, as being a cost, it will be set to 0 in any optimal solution, unless the left-hand part of the following constraint holds:

$$D_{ijk} \geq \sum_{(i'j')\in\alpha} x_{i'j'k}\cdot \mathrm{distance}(i',j') - \mathrm{limit}(i,j) \qquad \text{if } r^+_{ik}=r^-_{jk}=1 \tag{3.1.52}$$

With the usual linear transformation, it becomes:

$$D_{ijk}+M^{(3.1.52)}_{\mathrm{time}}\cdot(2-r^+_{ik}-r^-_{jk}) \geq \sum_{(i'j')\in\alpha} x_{i'j'k}\cdot\mathrm{distance}(i',j')-\mathrm{limit}(i,j)$$

$M^{(3.1.52)}_{\mathrm{time}}$ has to be greater than the total distance travelled by vehicle k. Let $\mathrm{maxDistance}(k)$ be the maximal distance vehicle k can travel. It is given by $\mathrm{maxTravel}(k)$ (which is the maximal travel time indicated in the data) and $\mathrm{maxSpeed}(k)=\max_{(i'j')\in\alpha}\frac{\mathrm{distance}(i',j')}{\mathrm{travTime}(i',j',k)}$:

$$\mathrm{maxDistance}(k)=\mathrm{maxTravel}(k)\cdot\mathrm{maxSpeed}(k)$$

We can choose $M_{\text{time}}^{(3.1.52)} = \text{maxDistance}(k)$.

- C^{ZSC} is the **zone-skipping cost** on the use of Zone-Skipping Hubs (ZSH). Recall that a shipment may be delivered directly, or to a nearby ZSH which allows the vehicles to consolidate their loads, like in a hub. On the other hand, an additional cost has to be paid for this possibility, and this is the zone-skipping cost or ZSC. This cost depends on the total quantity of products handled by the vehicle at the ZSH. As before, we decide to linearize simply the ZSCs (see figure 1.7) by considering that every additional load in the ZSH will increase proportionally the total cost, and the linearization is the same under-estimation as for the DTC. For any vehicle k and any site i used as a ZSH, the zone-skipping cost can then be written as follows:

$$C_{ik}^{ZSC} = C^{ZSC}(i) \cdot \sum_{v(i) \in V(i)} \text{qty}(v(i)) \cdot b_{vk} \tag{3.1.53}$$

As there may be "regular" visits on a ZSH (i.e. visits not using the site as a ZSH because the visit is a regular delivery), the sum of the $v(i)$ include only visits that use site i as a ZSH. $C^{ZSC}(i)$ is the unit ZSC for ZSH i, and it is approximated as explained above.

3.1.6 Size of the model

Number of variables

All the variables are listed in table 3.3. There are:

- $4|I| \cdot |K| + 2|K| + |V| = O(|I| \cdot |K|)$ time variables,
- $|I|^2 \cdot |K| + 4|I| \cdot |K| + |K| + |H| = O(|I|^2 \cdot |K|)$ boolean flow variables,
- $4|I| \cdot |K| \cdot |S'| + |V| + |V| \cdot |K| + |V|^2 = O(|I| \cdot |K| \cdot |S'|)$ boolean PDP variables (note that $|V| = 2|S'|$, as each shipment is made of two visits).

Overall, the formulation contains $O(|I| \cdot |K| \cdot (|I| + |S'|))$ variables, and this number of variables is big even for small instances. Note that the number of hubs $|H|$ has no impact on the total number of variables and can be neglected. Table 3.4 gives some examples of the number of variables for several values of the size of the data.

old-415-1530-1312-1 is one of the biggest instances that is in the pool of benchmarks solved in chapter 5. For this instance size, there are over 780 million variables in the formulation. Of course, in this case, the model is not even loadable in a MIP solver. Therefore, just considering the number of variables, this formulation can only be used on small instances. Even old-44-84-89 is likely to be very difficult to solve, just regarding the number of boolean variables.

Table 3.4: Number of variables for some data values

Instance name	hubUse	Jeu_0600_6	Jeu_0700	Jeu_00133280
sites	$\|I\| = 4$	$\|I\| = 11$	$\|I\| = 35$	$\|I\| = 415$
vehicles	$\|K\| = 2$	$\|K\| = 18$	$\|K\| = 52$	$\|K\| = 329$
shipments	$\|S'\| = 4$	$\|S'\| = 13$	$\|S'\| = 89$	$\|S'\| = 1312$
Time vars	44	874	7,562	549,422
Boolean flow vars	66	2988	71,032	57,208,494
Boolean PDP vars	216	11,466	689,038	724,286,976
Total	326	15,308	767,632	782,044,892

Number of constraints

Constraints of type (3.1.32) are the ones that contribute most to the complexity of the model, as there are $|I|^2 \cdot |K| \cdot |S'|$ such constraints. There are also about 20 constraints of complexity $|I| \cdot |K|$ and 10 constraints of complexity $|I| \cdot |K| \cdot |S'|$. Overall, more than 76 billion constraints for old-415-1530-1312-1 and over 7.7 million for old-44-84-89 make it impossible to solve large instances, only considering the size of the model.

3.2 The MVRPPD model

Whereas the previous formulation deals with a lot of binary variables, this model is a flow formulation over the products of a Multicommodity Vehicle Routing Problem with Pickups and Deliveries (MVRPPD). We will show in subsection 3.2.1 that any PDP can be written as a MVRPPD by using a shipment-to-product transformation. Therefore, this model is adapted to the PDP as well. Moreover, using a product flow over the network instead of boolean variables permits split loads as described in subsection 2.1.3, which can generate better solutions, as the problem has then fewer constraints. Nevertheless, other features, such as shipment alternatives (see subsection 1.3.2) are more difficult to formulate and have been ignored here. Time constraints are also less accurate, as no reference on the (un)loading time of products is given any more. Recall that table 3.1 sums up the ability of each formulation to model some of the real-life features of the problem.

This model has been written with the same basis as the previous one. Hence, numerous variables and constraints are similar. As a result, only the differences between both models are explained below.

3.2.1 Reduction PDP-to-MVRPPD

First, the way a PDP can be reduced to a MVRPPD is described below. This reduction was published at the ROADEF French-speaking conference [Fournier, 2007], but has been completed for this thesis.

Recall that the PDP is a problem where several shipments have to be routed from their pickup site to their delivery site. This problem has been described in detail in subsection 2.1.2. In the MVRPPD, several products have to be routed from their origin sites (where there are offers) to their destination sites (where there are demands). The MVRPPD is part of subsection 2.1.4. The differences between both problems have been summed up in table 2.1.

Simple reduction

To model a MVRPPD from a PDP, it is sufficient to associate a product with each shipment of the PDP. This product must only be available at the pickup site of the shipment, and have a demand only at the delivery site of the shipment. This makes the number of products exactly equal to the number of shipments in the initial problem.

Nonetheless, it is possible to introduce fewer products, which is useful to reduce the size of the problem.

Product aggregation

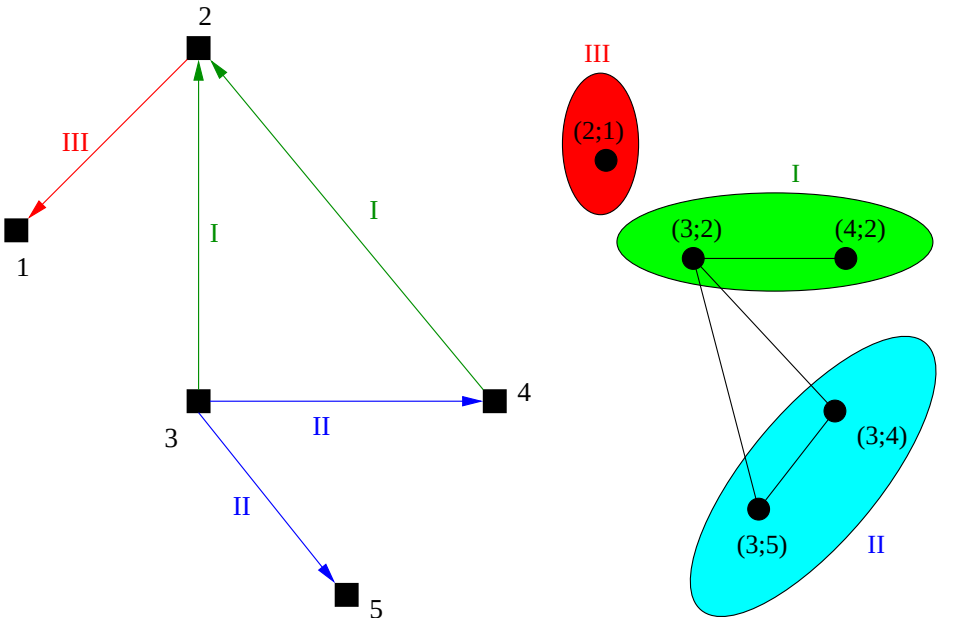
After this correspondence between shipments and products, we can group the products together, in such a way that there is no ambiguity on the origin or destination of a product. To minimize the number of products we need to reformulate the given PDP: a minimum number of arc subsets have to be created, such that in each subset, all the arcs have the same origin or the same destination.

An example of such a configuration is given on figure 3.3. On subfigure 3.3(a), if the arcs depict the shipments that have to be performed, they can be aggregated as follows:

- Product I: shipments (3;2) and (4;2)
- Product II: shipments (3;4) and (3;5)
- Product III: shipment (2;1)

Let's suppose we want to minimize the number of products obtained with the reduction PDP-to-MVRP. In the graph where the arcs are the shipments, we have to look for a minimal partition into subsets of edges, such that in each subset, all the arcs have either the same origin, or the same destination.

Figure 3.3: Example of a shipment aggregation



(a) Aggregated shipments

(b) Partition of the line graph into cliques

Colouring a line graph

We introduce an undirected graph from the basic graph of shipments. This graph is a particular **line graph**: its vertices are the arcs (the shipments) of the initial directed graph, and there is an edge between two vertices if and only if the corresponding arcs of the graph of shipments have one of their extremities in common (same origin or destination). For example, for subfigure 3.3(a), the corresponding undirected graph is given by subfigure 3.3(b).

Recall that a **complete graph** (or clique) is such that there is an edge between each couple of nodes. A clique of size n is usually referred to as K_n. We can first show by induction that any clique of the line graph is such that either all the shipments of the basic graph corresponding to the nodes in the clique have a common origin or they all have a common destination. In other words, the shipments corresponding to a clique in the line graph are connected alltogether by the same node (the origin or the destination).

Let n be the size of the clique. The proof is the following:

- It is obviously true if $n = 2$ (there are only two arcs).
- We assume that the property is true for a given integer $n \geq 2$. Let's consider a clique of size $n+1$ in the line graph. Let i be one of its nodes,

and s the corresponding shipment. If we consider the remaining nodes in the clique, they form a clique of size n, so all the corresponding shipments are connected by their origin or by their destination (using the induction assumption). If all the shipments have both the same origin and the same destination, then obviously adding i to the clique makes the property still true. In the other case, without loss of generality, let's suppose that the shipments of the clique of size n have a common origin, and at least two of them do not share the same destination. Let s_1 and s_2 be two shipments corresponding to these remaining nodes, such that they have distinct destinations. s can't have a common destination with both s_1 and s_2, by transitivity. So at least one of s_1 or s_2 has the same origin as s. As a result, by transitivity, s has a common origin with all the shipments corresponding to the clique, which makes the property true for any clique of size $n+1$.

- Hence, the property is true for any $n \geq 2$. □

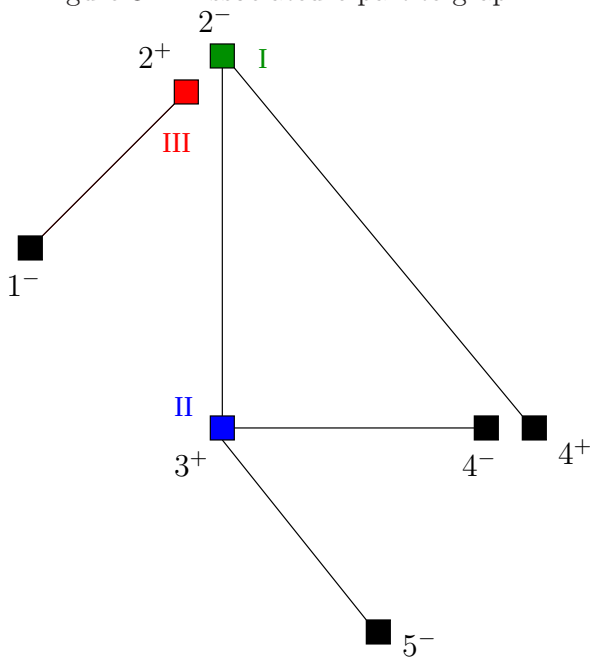
Consequently, minimizing the number of product reduces to searching the minimal partition of the line graph into cliques. This problem is famous and it is usually tackled by considering the complementary problem. Recall that for any undirected graph G, the **complementary graph** $C(G)$ is defined by the same set of nodes as in G but its edges are the ones that don't exist in G. In other words, for any couple of nodes in G, the edge between these nodes either exists in G or in $C(G)$, but not in both. Searching a minimal partition of the line graph into cliques is then equivalent to searching a minimal partition of the complementary graph of the line graph into stable sets (recall that a set is stable if for any couple of nodes in this set, there is no edge linking them). This last problem is the well-known **graph colouring problem**, which is $\mathcal{NP}$-complete [Karp, 2003], except for some particular classes of graphs.

Minimal vertex cover in a bipartite graph

Let's start again from the directed graph where each shipment is represented by an arc. As previously, we change this graph into an undirected graph. This time, any arc of the basic graph is changed into an edge in the undirected graph. A node i is duplicated into i^- and i^+ if there are incoming and outgoing arcs from this node:

- node i^- is the incoming node where all edges corresponding to the basic incoming arcs are connected,
- node i^+ is the outgoing node where all edges corresponding to the basic outgoing arcs are connected,
- there is no edge between i^- and i^+.

Figure 3.4: Associated bipartite graph



If there are only incoming arcs (resp. only outgoing arcs), then we just rename i to i^- (resp. i^+).

The undirected graph associated with the graph of subfigure 3.3(a) is given in figure 3.4.

The undirected graph is obviously **bipartite** (with all the nodes of type i^- on the one side, and of type i^+ on the other side), since all the edges of the graph are of type i^+j^-, by definition.

The problem of minimizing the number of products then reduces to finding a **minimal vertex cover** in this bipartite graph. The nodes marked with I, II and III in figure 3.4 form a minimum vertex cover of this example of undirected graph. To prove this property, we can show that given the two graphs (the basic directed one and the associated undirected bipartite graph), any solution of size M of the first problem can be written as a solution of size M of the second problem, and vice-versa.

First, we consider a partition of size M of the directed graph into arc subsets such that, in every subset, all the arcs either have the same origin or the same destination. We build a set Z of nodes in the associated undirected graph by keeping, for each subset of arcs of the basic graph, the node connected to each arc of the subset. Hence, if an arc subset of the basic graph is such that all its arcs have the same destination (resp. origin) i, then the node of the associated undirected graph that we keep into Z will be i^- (resp. i^+). If they have both the same origin i and the same destination j (it may

occur for arc subsets containing only one arc, for example), then either i^+ or j^- is picked in Z, but not both. As exactly one node of Z is associated with each arc subset of the basic graph, Z is obviously of size M. We just have to show that Z is a vertex cover of the associated graph. Suppose that there is an edge i^+j^- such that $i^+ \notin Z$ and $j^- \notin Z$. Then, by construction of Z, arc ij from the basic graph cannot be in any arc subset. Therefore it is not a partition of the set of arcs, which is a contradiction with the first assumption. Consequently, Z is a vertex cover of size M. If (I; II; III) is the partition of the basic graph on subfigure 3.3(a), then an associated vertex cover for the undirected graph could be given by (I; II; III) on figure 3.4. Note that node 1^- could have been picked instead of 2^+ for III.

Reversly, we suppose a vertex cover of size M has been found in the undirected graph. Every node in the vertex cover is connected to a set of edges corresponding to arcs of the basic directed graph. For each node in the vertex cover, we associate this arc subset. There may be arcs with several possible subsets. Such an arc is included in any of these subsets, arbitrarily chosen. As we build exactly one arc subset per node in the vertex cover, the total number of arc subsets is M. Let ij be an arc of the directed graph. It is associated with edge i^+j^- in the undirected graph. Either i^+ or j^- is in the vertex cover. Without loss of generality, suppose it is i^+. By construction of the arc subset associated with i^+, arc ij is necessarily in this arc subset, or maybe another. As a result, the arc subsets are a partition of the set of arcs in the directed graph. If (I; II; III) is the vertex cover of the undirected graph on figure 3.4, then an associated arc partition for the basic graph could be given by (I; II; III) on subfigure 3.3(a). Note that arc (3,2) could have been included in subset II instead of I. □

As the construction of the undirected graph from the basic directed graph is polynomial, the minimization of arc subsets reduces to a minimal vertex cover in the associated bipartite graph. Finding a minimal vertex cover in a bipartite graph is known to be polynomial, as it can be reduced to finding a maximal matching in the same bipartite graph (by the theorem of König). As a consequence, our problem of minimization of the number of products is **polynomial**.

What is more, it shows that the complementary graph of the line graph defined above belongs to the particular classes of graphs for which the colouring problem is polynomial. In fact, this line graph is restricted due to its definition. For example, the line graph can not contain any complete bipartite graph of type $K_{1,3}$ (with no additional edge), where $K_{n,m}$ is a bipartite graph where all the nodes of the first side (of size n) of the bipartition are connected by an edge to all the nodes of the second side (of size m). Then $K_{1,3}$ is defined by four nodes i, j_1, j_2 and j_3, and by three edges (ij_1), (ij_2) and (ij_3). The shipment corresponding to i has the same origin or the same destination as the one corresponding to j_1, and to j_2 and j_3 as well. Consequently, there is at least one edge between j_1, j_2 and j_3 (for two of

them at least, the corresponding shipments have the same origin or the same destination).

Greedy algorithm

The minimization of the number of products must be almost instantaneous because it is only a small part of solving the PDP problem. Even if we showed previously that the problem can be solved optimally in polyniomal time, we chose to apply a simple greedy algorithm on the line graph to group the products:

- Choose a vertex (for example in lexicographic order). It is associated to product I.
- Choose the following vertex. If it is connected to the previous one, let's associate it with product I, otherwise, with the new product II.
- For a vertex i, if it is connected to every other vertex of product p, let's associate it with p. If such a product can't be found then i is associated with a new product $p+1$.

To be able to group together two products, they have nevertheless to be similar: they must have the same incompatibilities, and they must of course be compatible.

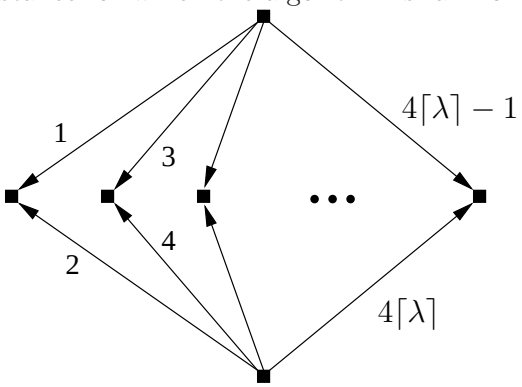
Even if this fast algorithm is effective practically, the quality of the reduction may not be good for some particular instances. In fact, this algorithm does not even have any performance ratio. If $n(\mathcal{I})$ and $n^*(\mathcal{I})$ are respectively the number of products given by the greedy algorithm for an instance $\mathcal{I}$ and the minimal number of products to be introduced for this instance, we can prove that:

$$\forall \lambda > 1, \exists \mathcal{I} \text{ instance such that } n(\mathcal{I}) > \lambda \cdot n^*(\mathcal{I})$$

Figure 3.5 depicts an example of such an instance $\mathcal{I}$ (and a given order on shipment arcs, depicted by the arc valuations). Note that the associated line graph is made of two complete graphs of size $2\lceil\lambda\rceil$ such that, in addition, each node is connected with its counterpart in the other complete graph.

We must show that for this instance, $n(\mathcal{I}) > \lambda \cdot n^*(\mathcal{I})$. Applying the previous algorithm on this instance aggregates shipments 1 and 2, shipments 3 and 4, and so on until shipments $4\lceil\lambda\rceil - 1$ and $4\lceil\lambda\rceil$. In other words, the shipments are aggregated with respect to their destinations. The number of products that are introduced is then $n(\mathcal{I}) = 2\lceil\lambda\rceil$. But obviously, the optimal solution is to aggregate the shipments with respect to their origin: the "odd" shipments on the one side, and the "even" shipments on the other side. Therefore, the minimal number of products to be introduced for this instance is $n^*(\mathcal{I}) = 2$.

Figure 3.5: Instance for which the algorithm is far from optimality



Fortunately, such instances are very particular cases. For example, note that very few numberings on this instance provide such a bad result.

Both of the reductions presented here are a proof that the MVRPPD is a generalization of the PDP, and therefore is $\mathcal{NP}$-hard.

3.2.2 Variables

The MVRPPD model will now be described in detail. We have the same conventions as before, but there is no shipment any more. In this section, a product is referred to with the letter p, and the set of all products is P. The superscript '=' means "loaded at the site". Any load variable with this superscript is negative if and only if the products are unloaded.

The time variables and the VRP boolean variables (enumerated in subsection 3.1.2) are kept here, unlike the PDP boolean variables. Instead of them, there are load variables $L^{\pm}_{ikp}$, which stand for the quantity of product p carried by a vehicle k arriving in and leaving a site i, and another boolean variable q_{ikp} indicating if an action (loading or unloading) is performed by vehicle k at site i. In addition, $L^{=}_{ikp}$ is the quantity of products p loaded by vehicle k at site i. It is the only variable that can have negative values, occuring when vehicle k unloads product p at site i.

3.2.3 Constraints

In the data, qty(s) is generalized to every site by introducing a quantity of product offer(i, p) that is available on site i. This quantity is negative in case of a demand. We introduce a value maxSplit(i, p) which is the maximum number of times a product p can be picked up or delivered at any site i. To use this formulation as a regular PDP solver, all these maxSplit(i, p) have to be fixed in order to ensure that the MVRP solution will be writable as

a PDP. In case the shipment-to-product reduction that was used was the one of subsection 3.2.1 (only one product per shipment), it suffices to choose $\text{maxSplit}(i,p) = 1 \quad \forall i \in I, \forall p \in P$).

In the case of the subsection 3.2.1, let p a product made of n shipments. By construction, either p has only one origin site, or one destination site. Without loss of generality, let's suppose that p has one origin site i' and n destination sites. Then $\text{maxSplit}(i',p) = n$ is sufficient to get a PDP final solution. Note that it is not necessary to impose constraints on the destination sites, as the maximum number of splits can't be over 1 for any destination site, otherwise we would have $\text{maxSplit}(i',p) > n$.

General time constraints and VRP boolean constraints are also part of the MVRP model. The only time constraint that has to be rewritten is constraint (3.1.6), as it refers to a shipment. Instead, we simply refer to a product p by introducing the following constraint:

$$T^-_{hk} + servTime_{hk} \leq T^+_{hl} \qquad \text{if } q_{hkp} = q_{hlp} = 1 \tag{3.2.1}$$

modelled with the constraint:

$$T^-_{hk} + servTime_{hk} \leq T^+_{hl} + M^{(3.1.6)}_{\text{time}} \cdot (2 - q_{hkp} - q_{hlp})$$

On the other hand, all the PDP-related constraints are replaced by the following load constraints:
$(\forall i,j \in I, \forall h \in H, \forall k \in K, \forall p \in P)$,

- In the four following constraints, let's suppose that i is not a hub:

$$|L^{=}_{ik'p}| \leq |\,\text{offer}(i,p)| \tag{3.2.2}$$
$$q_{ikp} = 0 \implies L^{=}_{ik'p} = 0 \tag{3.2.3}$$
$$\text{offer}(i,p) \cdot L^{=}_{ikp} \geq 0 \tag{3.2.4}$$
$$q_{ikp} = 0 \quad \text{if offer}(i,p) = 0 \tag{3.2.5}$$

  Constraints (3.2.2) and (3.2.3) can be merged into the following constraint:

$$-q_{ikp} \cdot |\,\text{offer}(i,p)| \leq L^{=}_{ik'p} \leq q_{ikp} \cdot |\,\text{offer}(i,p)|$$

  Constraint (3.2.4) states that $\text{offer}(i,p)$ and $L^{=}_{ikp}$ must be of the same sign, and constraint (3.2.5) that the vehicles cannot load or unload products on non-hub sites where there is neither an offer, nor a demand.

- Link between the load of a vehicle departing from site i and its load arriving at site i:

$$L^+_{ikp} \leq \begin{cases} L^{=}_{ikp} & \text{if } r^+_{ik} = 1 \\ L^{=}_{ikp} + L^-_{ikp} & \text{else} \end{cases} \tag{3.2.6}$$

As $L^-_{ikp} \geq 0$, it can be modelled through the following constraints:

$$\begin{aligned} L^+_{ikp} &\leq L^=_{ikp} + L^-_{ikp} \\ L^+_{ikp} &\leq L^=_{ikp} + M_{\text{load}} \cdot (1 - R^+_{ik}) \end{aligned}$$

- The vehicles can't load more than the offer (at pickup sites), and they have to unload at least the demand (at delivery sites) :

$$\sum_{k' \in K} L^=_{ik'p} \leq \text{offer}(i,p) \qquad \text{if offer}(i,p) \neq 0 \tag{3.2.7}$$

- Product flow conservation in a hub h:

$$\sum_{k' \in K} L^=_{hk'p} = 0 \qquad \text{if offer}(h,p) = 0 \tag{3.2.8}$$

- Limitation on the number of split pickups or deliveries:

$$\sum_{k' \in K} q_{ik'p} \leq \text{maxSplit}(i,p) \tag{3.2.9}$$

- Continuity of the product flow between sites i and j:

$$L^-_{jkp} \leq L^+_{ikp} \qquad \text{if } x_{ijk} = 1 \tag{3.2.10}$$

  In the MIP model, the constraint is written as follows:

$$L^-_{jkp} \leq L^+_{ikp} + M_{\text{load}} \cdot (1 - x_{ijk})$$

  Note that if constraint 3.2.10 was an equality, its linearization would require 2 constraints. But an inequality is sufficient, as in an optimal solution, there would be no use in having strictly less products arriving at j than leaving i.

- Capacity constraint at site i:

$$\sum_{p' \in P} L^\pm_{ikp'} \leq \text{cap}(k) \cdot y^\pm_{ik} \tag{3.2.11}$$

- At last, the incompatibility constraints have to be written using product-related variables:

$$q_{ikp} = 0 \qquad \text{if } p \text{ and } k \text{ incompatible} \tag{3.2.12}$$

  Note that unlike in the PDP model, there is no incompatibility between products in this formulation, as it would be hard to model in a simple way.

3.2.4 Big-Ms computation

Here, the big-M value is easier to find as in subsection 3.1.4, because the loads are limited by the vehicle capacity. Therefore, for each conditional constraint where M_{load} is needed, it is sufficient to choose $M_{\text{load}} = \text{cap}(k)$.

However, a slightly more precise constant could be chosen, especially if there is little demand in this product over the network, using the following inequality:

$$\forall i \in I, \forall k \in K, \forall p \in P, \quad L^{\pm}_{ikp} \leq \sum_{\substack{i' \in I \\ \text{offer}(i',p)<0}} |\,\text{offer}(i', p)|$$

Consequently, a better value is:

$$M_{\text{load}} = \min \left(\text{cap}(k), \sum_{\substack{i' \in I \\ \text{offer}(i',p)<0}} |\,\text{offer}(i', p)| \right)$$

3.2.5 Objective

The cost function is the same as the one in part 3.1.5. For the calculation of C^{DTC}, this time the quantity Q_k is given by ($\forall i \in I, \forall k \in K$):

$$Q_k \geq \sum_{p' \in P} L^{\pm}_{ikp'} \tag{3.2.13}$$

As specified in the introduction, there are no alternative in this model, so the C^{ZSC} component is null here.

3.2.6 Size of the model

The time and boolean flow variables are the same as for the PDP formulation, so the number of variables of both kinds are the same as in subsection 3.1.6. There are no boolean PDP variables any more but the $4|I|.|K|.|P|$ VRP load variables enumerated in subsection 3.2.2. Then, this model has $O(|I|.|K|.(|I| + |P|))$ variables, which is similar to the total number of variables of the PDP formulation, even if, using the reduction based on a shipment aggregation and given in subsection 3.2.1, we have $|P| \leq |S|$.

The total number of constraints is closely related to the number of constraints of type (3.2.10). The model is of size $O(|I|^2.|K|.|P|)$, which again is rather similar to the size of the PDP formulation.

Practically, it is interesting to see that in the instances given for the example in table 3.4, most of the shipments can be grouped into sets in which all the shipments' pickup site is the same. Consequently, the shipment aggregation described in subsection 3.2.1 is particularly efficient. In hubUse,

2 products replace the 4 shipments. In old-23-80-13-1, 3 products can be used (instead of the 13 shipments), and in old-44-84-89, 5 products (instead of 89 shipments) allow the model not to be as huge as the PDP formulation. However, the model is still big, with for example 115,000 variables and over 540,000 constraints for old-44-84-89.

3.3 Improvements

In the previous sections, the formulations seem to describe the problem accurately, but their size prevents any solver from loading the biggest instances. In this section, we provide some solutions to this problem, as well as some solving tips, as solving big MIP instances is time-consuming.

3.3.1 Models merged

The MVRPPD model described in section 3.2 is very similar to the PDP model of section 3.1, with a lot of variables and constraints in common. Therefore, it is easy to make a big model out of these two models, by considering that both product flows and shipments have to be routed in the network.

All the variables and constraints enumerated in sections 3.1 and 3.2 are part of the merged model. The only constraint that links both basic models is the capacity constraint. Constraints (3.1.40) of the PDP and (3.2.11) of the MVRPPD are merged into a new capacity constraint:

$$\sum_{p' \in P} L^{\pm}_{ikp'} + \sum_{s \in S'} \text{qty}(s) * a^{\pm}_{iks} \leq \text{cap}(k) * y^{\pm}_{ik} \tag{3.3.1}$$

This new capacity constraint models the fact that both the products and the shipments are transported by the vehicles. Moreover, both basic models have their own constraint ensuring the simultaneous presence of the vehicles in a hub if they swap some products or shipments. Both of these constraints (3.1.6) and (3.2.1) are in the merged model.

The main advantage of merging the models is that in the case of a PDP-to-MVRP transformation as described in section 3.2.1, the transformation is allowed to be partial. It might be interesting to change into products only the shipments that have a common extremity with at least one other shipment, so as to be able to use the shipment aggregation of subsection 3.2.1 and reduce the model size. For the other "isolated" shipments, there is then a choice: either change them into products as well and use the MVRP model, or keep them as shipments and use the mixed model. This choice can depend on the performance of both models, which will be studied in section 3.4.

Furthermore, shipments that are likely to lose some of their properties if they are transformed, such as the shipments concerned with alternatives

at a ZSH, can be kept as shipments as well, instead of being changed into products.

In the following, we call S' the set of remaining shipments (after the PDP-to-MVRP transformation) and P the set of products created by this transformation.

3.3.2 Preprocessing

To reduce the size of the formulations, the most natural way is to ignore the objects (sites, arcs, vehicles) that will not appear in any optimal solution.

Indeed, the instances we solve are customer instances. ILOG TPO customers have a set of sites and vehicles that are used for several instances with possibly various numbers of shipments. As a result, in any data file, some sites or vehicles might be useless.

Reducing the number of sites

Let i be a site such that (with the same notations as previously):

- i is not a hub,
- $\forall k \in K, i \neq \text{extrem}^{\pm}(k)$,
- $\forall s \in S', i \neq i(p(s))$ and $i \neq i(d(s))$,
- $\forall p \in P, \text{offer}(i, p) = 0$.

i can't be included in an optimal solution, as no action can be performed at i. In any solution containing i as one of the site of the route of a vehicle k, as the triangular inequality holds, it is better for vehicle k to remove site i from its route and take a shortcut. Then, any site i as described above can be removed from the data without ignoring any optimal solution.

Reducing the number of vehicles

It is obvious that in many instances, the number of vehicles available for the routing is far too high. In some of them, there may be 4 times as many vehicles as shipments, although a shipment fills very rarely a vehicle's capacity by itself. This usually introduces many useless variables in the model, as a lot of the vehicles are not used. Therefore, it is necessary to have an automatic tool able to reduce the number of vehicles depending on the number of shipments and their load size.

We propose an heuristic procedure based on the total quantity of goods (including products and shipments) that have to be routed. In each vehicle fleet (in which all vehicles are identical), only a subset is kept, such that all the requests (shipments or product demands) could be performed by the vehicles of the subset.

Let:

$$Q = \sum_{s \in S'} \text{qty}(s) + \sum_{p \in P} \sum_{\substack{i \in I \\ \text{offer}(i,p) < 0}} |\,\text{offer}(i,p)|$$

be the total quantity of goods that have to be transported over the network, and cap the capacity of any vehicle in the fleet. In each fleet, we choose to keep only $\lceil \frac{Q}{\text{cap}} \rceil$ vehicles (where $\lceil x \rceil$ is the lowest integer greater than or equal to x).

It is an heuristic in the sense that this elimination procedure may cut off some (possibly good) solutions from the search space but practically, it is very unlikely that with this elimination, good solutions are removed. If we further reduce the number of vehicles in each fleet, however, we may get a smaller formulation but worse solutions when solving it.

In some small instances, there are many fleets although a few of them are likely to ship the transportation requests. For example, any shuttle that travels between two non-hub sites for which there is no demand and no offer is ignored in the formulation, as no action can be performed by this shuttle. Formally, let a site i be such that:

- i is not a hub,
- $\forall p \in P, \text{offer}(i,p) = 0$,
- $\forall v \in V, i \neq i(v)$.

Any shuttle k having i as its departure site ($i = \text{extrem}^+(k)$) or its arrival site ($i = \text{extrem}^-(k)$) will be removed from the formulation. In addition, site i can be removed as well. In any solution containing i as one of the site of the route of a vehicle k, as the triangular inequality holds, it is better for vehicle k to remove site i from its route and take a shortcut. Then, any site i as described above can be removed from the data without moving any optimal solution apart.

3.3.3 Cutting-plane algorithm

In subsections 3.1.6 and 3.2.6, we noticed that a lot of constraints are part of the model even for middle-size instances. To overcome this issue, we can apply a well-known technique, called **cutting-plane algorithm**. It is clearly described in Naddef and Rinaldi's paper [Naddef and Rinaldi, 2002].

The main idea is to relax many of the constraints, possibly the ones that make the formulation difficult to solve, or the ones that are less likely to be violated in any solution. These removed constraints are nevertheless kept in a pool, which is consulted whenever a solution is found. If any of the constraints in the pool is violated by the current solution, it is added to the model, which is solved again.

In a MIP Branch-and-Bound solving, the cutting-plane algorithm made be applied several times on the linear relaxation, as discussed in subsection 2.1.5. In our case, we choose to relax all the big-M constraints, namely constraints of type:

- (3.1.2), (3.1.3), (3.1.4), (3.1.5) and (3.1.6) of time constraints (subsection 3.1.3),
- (3.1.12), (3.1.13) and (3.1.14) of PDP time constraints (subsection 3.1.3),
- (3.2.6) and (3.2.10) of MVRP load constraints (subsection 3.2.3).

This choice can be explained by the complexity of such big-M constraints in a MIP model, due to the loose link between the boolean and the continuous variables. Moreover, they are numerous, especially in the MVRP model: $O(|I|^2.|K|)$ in the PDP model, and $O(|I|^2.|K|.|P|)$ in the MVRP model. Relaxing big-M constraints results in a decrease of the problem size for instance old-44-84-89 of 2.22% for a pure PDP model, and of over 92% for a pure MVRP model.

This method improves considerably the performances of the MIP solver, here ILOG CPLEX 11. Note that to detect a violated constraint amongst the pool, we need a separation algorithm which excludes the current solution from the polyedra by finding a constraint to add in the formulation. As the API already exists, we chose to let ILOG CPLEX automatically decide which constraint to add, and we do not interfere in the solving process.

3.3.4 Dedicated cuts

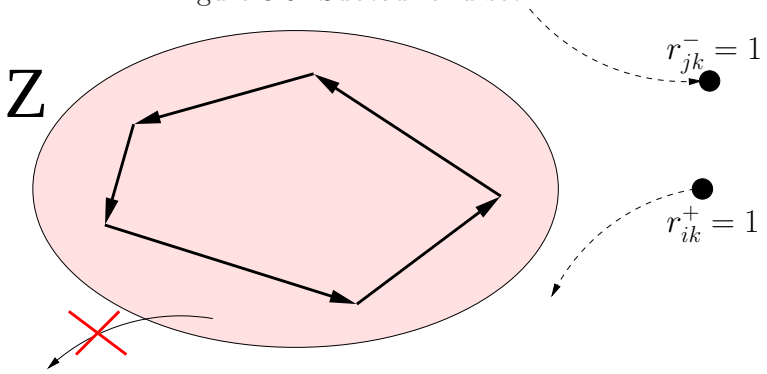
On the other hand, we propose some dedicated cuts in order to strengthen the formulation, especially when some of the time constraints are relaxed.

Subtour elimination cuts

Indeed, only the time constraints ensure that the solution does not contain any subtour. The route of a vehicle k is closed if and only if its departure site is the same as its arrival site. Indeed, along the route, the time continuity constraints (3.1.2) and (3.1.3) hold, except at the last site i of the route, where there is no link between T^-_{ik} and T^+_{ik}. This allows the last site to be the first site as well. The subtours are forbidden by the fact that only one route last site is allowed for each vehicle (boolean constraints (3.1.23)), and by the growing values of T^-_{ik} and T^+_{ik}, for all i in the route. Without some time constraints (especially (3.1.2) and (3.1.3)), no constraint can ensure that the subtours are forbidden.

Nonetheless, we propose some cuts that remove some subtours from the formulation. First, we can easily remove the subtours of length 2 by noticing

Figure 3.6: Subtour and set Z



that if a vehicle goes back and forth between two sites, one of the two sites is necessarily the arrival site. Mathematically:

$$\forall (ij) \in \alpha, \forall k \in K, x_{ijk} + x_{jik} \leq 1 + r_{ik}^- + r_{jk}^- \tag{3.3.2}$$

Note that if $x_{ijk} = x_{jik} = 1$, from the previous constraint we get $r_{ik}^- = 1$ or $r_{jk}^- = 1$. Let's say, without loss of generality, that $r_{ik}^- = 1$.

From $x_{ijk} = 1$ and $x_{jik} = 1$ we can also conclude respectively that $y_{ik}^+ = 1$ (using constraint (3.1.17)) and that $y_{ik}^- = 1$ (using constraint (3.1.16)). Therefore, from constraint (3.1.20), we get $r_{ik}^+ = 1$. Consequently, i is also the departure site of the route of vehicle k.

This cut can be generalized (for $n \geq 2$): $\forall i_1, i_2, \cdots, i_n \in I$,

$$\sum_{p=1}^{n-1} (x_{i_p i_{p+1} k}) + x_{i_n i_1 k} \leq n - 1 + \sum_{p=1}^{n} r_{i_p k}^- \tag{3.3.3}$$

Note that these constraints can't be easily generalized into classical subtour elimination constraints (such as constraints (2.1.5)) because of variables $r_{ik}^{\pm}$. Moreover, there are many of these as the n sites can be chosen arbitrarily. As a result, we should either use only a small part of these constraints (for example only the particular case (3.3.2)), or include them in the pool of cutting-planes in the technique described above (subsection 3.3.3).

We can also notice that, for any vehicle k, a subtour is the smallest set Z such that some internal variables x_{ijk} are non-zero, although no flow is going out of Z and the departure and arrival sites of the route of k are outside Z (see figure 3.6).

Let's denote:

- $\delta^+(Z) = \{(ij) \in \alpha | i \in Z \text{ and } j \notin Z\}$
- $\delta^-(Z) = \{(ij) \in \alpha | i \notin Z \text{ and } j \in Z\}$

- $\alpha(Z) = \{(ij) \in \alpha | i \in Z \text{ and } j \in Z\}$

To separate the subtours from a solution, we could therefore use the following constraint as a cut:

$$\left.\begin{array}{c} \forall i \in Z, r_{ik}^{\pm} = 0 \\ \forall (ij) \in \delta^{+}(Z), x_{ijk} = 0 \end{array}\right\} \Longrightarrow \forall (ij) \in \alpha(Z), x_{ijk} = 0 \tag{3.3.4}$$

Nevertheless, it is not easy to linearize this constraints, as many variables are involved. It is possible for example to introduce a new big-M constraint:

$$\sum_{(ij)\in\alpha(Z)} x_{ijk} \leq M \cdot \left(\sum_{(ij)\in\delta^{+}(Z)} x_{ijk} + \sum_{i\in Z} (r_{ik}^{+} + r_{ik}^{-}) \right)$$

As $\sum_{(ij)\in\alpha(Z)} x_{ijk}$ can be at most $|Z|$ in the case of a tour over all the sites in Z, we choose $M = |Z|$. Be aware that such cuts are more general than cuts (3.3.2) presented above, as here there is no indication on which one of the arcs in Z are travelled by the vehicle. Of course, it is useless to use both (3.3.2) and (3.3.4) as cutting planes.

Vehicle cuts

The following constraints are not included in the formulation although they are part of the problem description. The reason for this is that, considering the cost function, they are not necessary as they are upper bounds on expressions of variables that have to be minimized. In other words, they can't be violated in any optimal solution. What's more, removing them decreases the model size.

In the basic formulation, a vehicle may go through a site without performing an action. However, this produces bad-looking solutions if the solving engine is stopped before the optimal solution. To overcome this issue, we can use the following constraints as cuts:

$$\begin{array}{lll} \forall i \in I, \forall k \in K, y_{ik}^{\pm} = 1 & \Longrightarrow & (r_{ik}^{\pm} = 1 \\ & \text{or} & (\exists s \in S, c_{iks} = 1 \text{ or } d_{iks} = 1) \\ & \text{or} & \exists p \in P, q_{ikp} = 1) \end{array} \tag{3.3.5}$$

It has been modelled linearly as follows:

$$y_{ik}^{\pm} \leq r_{ik}^{\pm} + \sum_{s\in S} (c_{iks} + d_{iks}) + \sum_{p\in P} q_{ikp}$$

Breaking symmetry

To reduce even more the polyedra, we can add some symmetry-breaking constraints. In this problem, the vehicles are individualized, but in fact they are clones that have the same features, in each fleet. For each fleet of vehicles, we then impose an arbitrary order (e.g. lexicographical) for them to be used. Let $K(f)$ be the set of all vehicles in fleet f. By a constraint, we enforce vehicle k_1 to be used before vehicle k_2, if they are in the same fleet f:

$$\forall f \in F, \forall k_1, k_2 \in K(f), u_{k_2} \leq u_{k_1} \tag{3.3.6}$$

The main advantage of these simple constraints is that there are only one per vehicle at most, and they don't dramatically increase the size of the model.

3.3.5 Branching priorities

In a regular Branch-and-Bound scheme (see subsection 2.1.5), the variables on which the branching operates are chosen depending on their value and its closeness to an integer value. However, some of the variables don't give valuable global information to the solver when they are fixed. For instance, fixing a variable like x_{ijk} is a very local action and has a limited impact on other variables. On the other hand, setting $u_k = 0$ (meaning that vehicle k won't be used) has a lot of implications on other variables. All the variables indexed on vehicle k are then set to 0. The search tree after branching on variable u_k is then supposed to be smaller than after branching on x_{ijk}. The strategy derived from this assertion is to branch first on such "global" variables in order to fix some other variables.

In our case, global variables are easy to determine, as they are variables with few indexes, or that can be written as a sum of other variables. We chose to introduce three levels of priorities over the boolean variables:

- Level 1: u_k, z_h, b_v,
- Level 2: $y_{ik}^{\pm}$,
- Level 3: all other boolean variables.

Note that we can also relax the integrity constraint of any boolean variable that can be written as the sum of other boolean variables. Variables u_k, b_v and $y_{ik}^{\pm}$ (by respectively constraints (3.1.21), (3.1.47) and (3.1.16)-(3.1.17)) meet this condition. This would imply that no branching has to be made on such variables, as they would be considered by the optimization engine as a continuous variable. This leaves us with a choice: either maintaining the integrity constraints inside the model and branching on these variables first in order to quickly fix other variables values, or relaxing the integrity constraints and branching on fewer variables in the search tree.

3.4 Results

In this section, we enumerate the solution values and computation time on the smallest ILOG TPO instances for various sets of parameters. We also try to validate the quality of the model by comparing its solution with the one found with local search by ILOG TPO. The instances have up to 12 shipments to be routed, the number of shipments being the third number in the instance name. For each of the 19 instances, the PDP MIP model is first runned for a basic parameter configuration and this run is used as a reference to the other runs. This basic setting is the solving of the basic MIP model, with all the basic constraints (including big-M constraints) and without any cutting plane or cut. Each other run corresponds to a modified setting, and it is compared to the reference run by calculating the ratio between its solution value and the final solution value of the basic run (the same stands for the computation time). Note that the time limit is set to 3 hours, so the final solution is not necessarily optimal. We use the MIP solver of ILOG CPLEX 11.0 on an AMD Opteron 2.4 GHz.

Tables 3.5 and 3.6 gather the results for the 19 instances. The first row stands for the settings that are applied to the MIP solving:

- "Lazy": instead of being inserted in the model at the beginning (as in the basic run), big-M constraints are relaxed and used as cutting planes, as described in subsection 3.3.3. ILOG CPLEX "lazy constraints" allow us to have such cutting planes during the Branch-and-Bound solving.

- "Lazy+Cuts": as before, big-M constraints are cutting planes. In addition, the cuts detailed in subsection 3.3.4 are inserted in the model.

- "IloIfThen": the conditional constraints are not written in the model as big-M constraints. An ILOG CPLEX constraint of type IloIfThen is used instead.

- In table 3.6: the same instance is changed into a MVRPPD model using the PDP-to-MVRPPD reduction described in subsection 3.2.1 and solved as a MVRPPD model (see section 3.2). Note that since the shipment alternatives are not taken into account in the MVRPPD formulation, we suppose that for each alternative, every shipment has to be performed (instead of only one of them in the standard PDP). As a result, the solution cost for MVRPPD is supposed to be bigger than the basic one, as the problem is more constrained. However, this is not necessarily the case, as the zone-skipping costs are not modelled in the MVRPPD formulation. Moreover, some other constraints, such as the visit time windows, are relaxed in the MVRPPD. Consequently, we decided to compare the solving time between the PDP and the MVRP formulation, unlike the solution value.

Table 3.5: Solution ratios for various settings on the arc-based formulation

Setting	Lazy		Lazy+Cuts		IloIfThen	
Dimension	value	time	value	time	value	time
old-19-18-12-1	1	1055.1804	1	440.5677	1	1.7323
old-19-18-12-2	1	662.5282	1	134.108	1	0.8384
old-20-46-12-1	1	0.7626	1	0.8808	1.3048	1.0389
old-20-46-12-3	0.9556	1	0.9264	1	1.0698	1
old-20-46-12-4	0.9919	1	0.9878	0.8464	1.1529	1
old-20-46-12-5	1.1258	1	0.9878	1	1.567	1
old-20-46-12-6	1.0811	1	0.9108	0.5393	1.0012	1
old-20-32-12	1.0811	1	0.9108	0.5393	1.0012	1
old-19-42-8-1	1	1	1	1	1	5.2941
old-19-42-8-2	1	1.8832	1	2.773	1	5.133
old-19-36-4-1	1	1.3557	1	0.3454	1	6.7234
old-19-36-4-2	1	1.5755	1	0.3275	1	3.7615
old-19-36-4-3	1	0.6704	1	0.2376	1	6.4602
old-18-8-4-1	1	1.1276	1	1.2058	1	2.5802
old-19-8-4	1	0.4737	1	0.6579	1	2
old-18-8-4-2	1	0.9395	1	0.7803	1	1.8631
new-29-52-8	1	3.3308	1.1472	8.8628	1	3.5224
old-21-51-6	1	7.7895	1	1.6053	1	2.1842

First, we must point out that when the ratio on the solution values is exactly 1, it means that the solutions of both runs are optimal (but the proof of optimality is not necessarily reached in one of them), whereas a ratio of exactly 1 in the computing time means that both runs have reached the time limit of 3 hours and consequently we have a proof of optimality for neither of the solutions (the gap between the solution value and the lower bound is still strictly positive). Reversly, when the ratio on computing time is not exactly 1, then at least one of the runs has not reached the time limit and therefore has found the optimal solution. Note also that the results on instances old-20-46-12-1 and old-20-46-12-2 are exactly the same. It is because this is quite the same instance, but the only differences involve features that are not taken into account in the model. Therefore, both of them are formulated the same way. The same holds for instances old-20-46-12-6 and old-20-32-12.

We can notice that besides a few instances (especially the first two ones), the computing time to the optimal solution is lower than the basic run in the "Lazy" case and even lower in the "Lazy+Cuts" case. This shows that the improvements described in section 3.3 lead to good results and that they must be applied to get the optimal solution faster. In the special cases of

Table 3.6: Solving time ratios between the MVRP and PDP models

Instance	Time ratio
old-19-18-12-1	2081.8374
old-19-18-12-2	1033.9642
old-20-46-12-1	1.0389
old-20-46-12-3	1
old-20-46-12-4	1
old-20-46-12-5	1
old-20-46-12-6	1
old-20-32-12	1
old-19-42-8-1	3538.4696
old-19-42-8-2	30.4566
old-19-36-4-1	157.8744
old-19-36-4-2	102.4815
old-19-36-4-3	118.603
old-18-8-4-1	1.4609
old-19-8-4	9.2368
old-18-8-4-2	1.1178
new-29-52-8	24.7093
old-21-51-6	2.4737

the first two instances, the basic MIP is solved in a few seconds whereas the modified solving takes several minutes. These instances are particular in the sense that every shipment is of type full truckload (FTL), as the vehicle capacity is too small for the vehicles to carry over one shipment at a time. As a result, the only decisions to be taken are related to the matching of vehicles to shipments and departure and arrival times. In fact, they are even very easy to solve by hand. When big-M constraints are not part of the model, the ILOG CPLEX presolve is less efficient and it cannot reduce the polyedron as much as when all the constraints are in the formulation. Furthermore, the cuts at the beginning of the ILOG CPLEX Branch-and-Bound are far more efficient in the basic case, as the problem is more constrained. In the other case, the cutting planes (that contain a great quantity of time constraints) are usually violated and are added to the formulation very often, which slows the solving down.

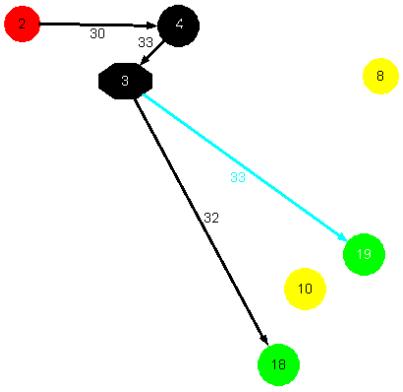
Another clear statement that we can get from the table is that the ILOG CPLEX IloIfThen method isn't as efficient as our big-M constraints. In almost all the cases, either the solution found by IloIfThen is worse than the basic one, or the same solution is found in more time. This shows that our constants M were carefully chosen and that they are not too big. This is also an example which shows that a clever customized heuristic is often better than automatical tools that are "blind" of the problem data, in some way.

At last, the MVRPPD formulation is usually solved in more time than the basic version. This is because there are not enough shipments to get a real advantage in the shipment aggregation described in subsection 3.2.1. Both models are approximately of the same size but the MVRPPD formulation contains far more big-M constraints. On the other hand, the MVRPPD model is able to provide a solution for largest instances, where the PDP formulation is too big to get to its first solution in reasonable time.

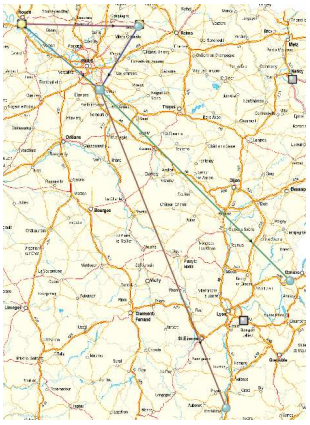
Let's now compare one of the solutions found by the MIP with the solution found by ILOG TPO on the same instance old-20-46-12-1. On figure 3.7, both the vehicle routes for the MIP optimal solution and for the ILOG TPO final solution are depicted. On subfigure 3.7(a), the arcs are indexed by the total quantity transported by the vehicle corresponding to the arc.

At the first glance, we can see that the vehicle routes of both solvers are similar. The regular delivery sites 18 and 19 are chosen instead of ZSHs 10 or 8, probably because the zone-skipping costs are too high. Another reason may be that anyway, two vehicles are required to deliver the shipments in that area (the quantities carried by the vehicles are 33 and 32), and both of them are almost filled, so they don't need to gather their loads and deliver to the ZSH. There is one difference, however. In the ILOG TPO solution, a shuttle travels between site 2 and hub 3, whereas the MIP model doesn't need any. Indeed, the costs on shuttles are not the same as on the other vehicles. The fact that a shuttle has lower costs than a regular vehicle is not modelled in the MIP formulation because the shuttle costs are complex.

Figure 3.7: Vehicle routes for solutions of instance old-20-46-12-1



(a) PDP MIP model



(b) ILOG TPO

Hence, in the MIP model, shuttles have the same costs as other vehicles, which can explain the only difference in the vehicle routes. The main issue is that the computing time for the MIP model on this instance (with the big-M constraints as cutting planes and the cuts included in the model) is about 2 hours and a half whereas ILOG TPO's computation time on this instance is 2 minutes. However, on smaller instances such as old-19-8-4, the MIP takes less than half a second when ILOG TPO get its final solution after over 3 seconds.

Conclusion

Two models were proposed for the routing problem described in chapter 1. They are adapted to solve small instances to optimality, and can be a great help for ILOG TPO on these instances, especially if the extensions such as the cutting-plane algorithm are used. In particular, the optimality of solutions given by ILOG TPO can be proven by solving one of these models. Bigger instances could be solved as well by separating the set of shipments into several subsets, so that the problem size becomes small enough, and solve the MIP model on each of the subsets of shipments.

Chapter 4

Cooperation algorithm with ILOG TPO

RÉSUMÉ DU CHAPITRE

Pour résoudre les instances de grande taille, il s'agit de faire coopérer le moteur de résolution d'ILOG TPO avec un algorithme permettant de déterminer, pour chaque ordre de transport, dans quelle suite de hubs il serait judicieux de le transborder. Ce genre de décisions est difficile à prendre pour ILOG TPO car le nombre de tels chemins de hubs est exponentielle pour chaque ordre de transport. De plus, il s'agit de décisions plutôt globales, alors que le moteur d'ILOG TPO est basé sur des techniques de recherche locale.

Pour fournir de telles indications à ILOG TPO, nous proposons un nouveau modèle mathématique à base de chemins, cette fois très simplifié par rapport au problème initial. Cette formulation n'est basée que sur la structure du réseau et néglige certains aspects comme les considérations temporelles (fenêtres de temps, etc.). Elle représente les véhicules comme un flot, et un certain nombre de variables booléennes de chemin sont associées à chaque ordre de transport. Ce sont ces variables qui sont utilisées pour déterminer les chemins de hubs prometteurs pour chaque ordre de transport.

Nous proposons trois façons de communiquer ces indications à ILOG TPO. Il s'agit de contraindre de façon plus ou moins importante le modèle d'ILOG TPO et le forcer, lors de la résolution, à utiliser les chemins de hubs indiqués par le modèle mathématique. Ceci est intégré dans une heuristique à deux phases. D'abord le modèle d'ILOG TPO contraint par ces indications est résolu avec les méthodes de résolution d'ILOG TPO. Les indications sur les chemins des ordres de transport sont ensuite relâchées, et le modèle d'ILOG TPO est résolu une nouvelle fois, avec comme première solution la solution finale de la phase précédente. Cette phase sert à corriger les imperfections sur les chemins des ordres qui découlent du caractère approché de la formulation mathématique à base de chemins.

ILOG TPO hardly manages to decide whether each shipment should be transshipped at a hub or if it should be delivered directly. Section 4.1 describes some of the ILOG TPO standard solving strategies. They are based on local search, and some of their decisions may be questionable globally, as local search improves a solution step by step without considering the whole problem. The smallest instances can be solved by the models defined in the previous chapter, as they are a close representation of the routing problem and they can reach optimality in reasonable time. However, their large size doesn't allow the solving of the largest instances of ILOG TPO benchmarks, that can have up to 1500 shipments. The idea is then, for these instances, to help the ILOG TPO solving by guiding some decisions, such as the ones concerning the shipment paths, with a global MIP model that keeps the network structure, but that is simpler and smaller than the previous formulations. Section 4.2 describes a model based on a network representation of the vehicles and where the decision variables involving shipments are the possible paths through which each shipment can be routed. At last, in section 4.3, the cooperation between the ILOG TPO solving engine and the MIP model is exposed. The MIP model is solved as a preprocessing and the MIP optimal solution provides guidelines for shipment paths. These guidelines are then given to ILOG TPO before the solving.

4.1 ILOG TPO solving methods

ILOG TPO tackles very complex problems as described in section 1.3. Some of the customer instances it has to solve are large (hundreds of shipments), and can't be solved to optimality, or at least, there can't be a systematic proof that a final solution is optimal. The best solving strategies for this kind of problems is local search with no doubt, as it may include several problem features, and yet finds a good solution in reasonable time. Anyway, the customers using ILOG TPO would not expect an optimal solution, but at least a better solution than their own handmade plan and that they can't improve easily by themselves.

This justifies the use of **local search** (LS) in the ILOG TPO solving engine. The core of the strategies used by ILOG TPO is described by De Backer et al. [De Backer et al., 2000]. In fact, it mixes Local Search and **constraint programming** (CP). CP is a well-known solving technique where every variable has a domain that is reduced during the search thanks to constraints triggering. Propagation is called as soon as the domain of a variable x is reduced, and every other variable linked with x by a constraint can have its domain modified as well by applying this constraint on the new domain of x.

The optimization model used by ILOG TPO is based on visits rather than sites. Almost all the basic decisions and local search moves imply one

or more visits. In particular, the integer variable R_i indicates the visit that is performed just after visit i on the same vehicle. This enables the use of several constraints, mostly conditional, to model load accumulation or time precedences. To model side constraints such as incompatibilities, a vehicle tag τ_i, standing for the vehicle that serves visit i, is also needed.

Any solution has two representations:

- the active one that is used for the local search of the next solution,
- the passive one, on which the checking procedure that validates the feasibility of the solution, with respect to all the constraints, is carried out.

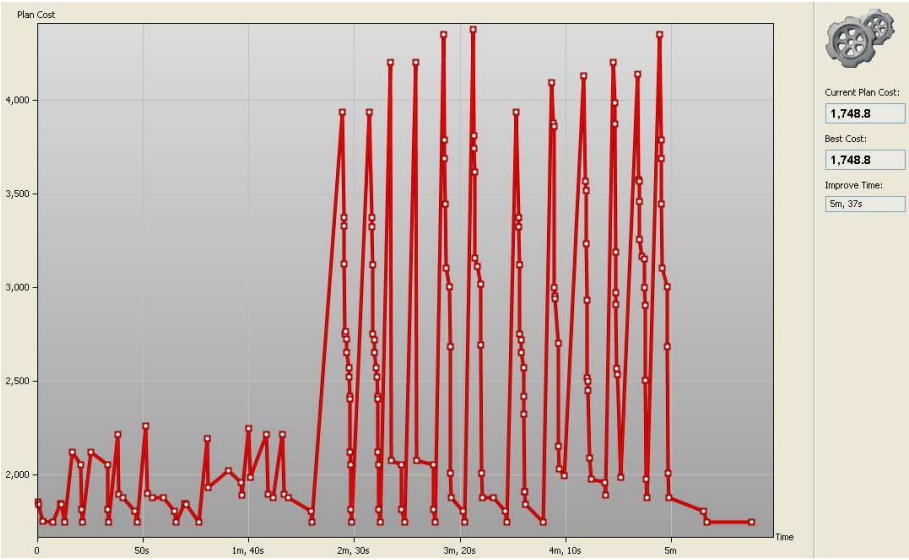
As checking all the constaints might be time-consuming, some methods are proposed to avoid the complete checking, such as a reduction of the domain of cumulative variables (time or load), or restricting the checking to modified routes only.

The local search mainly uses five of the neighbourhoods described in subsection 2.1.5: 2-Opt, Or-Opt, Relocate, Exchange and Cross. In addition, some complex neighbourhoods are implemented to take into account the advanced features of the problem such as transshipment or shipment alternatives. In particular, a neighbourhood switches the transshipment status of a shipment (direct or indirect), whereas another one unperforms shipment s and performs an alternate shipment s' instead.

All these LS operators can be used in a pure LS method, or in a more sophisticated one based on a metaheuristic in order to be able to escape from local minima. In particular, a basic tabu search (TS) is available. This TS defines two tabu lists (one for arcs added and another for arcs removed) that are inspected at each move, and if the move contains more than a given threshold (that depends on the kind of move) of added and removed arcs already in the tabu list, it is rejected. It is also possible to apply another metaheuristic: **guided local search** (GLS). GLS is a local search procedure that adds a penalty factor in the objective function as soon as it reaches a local optimum. This penalty factor depends on search experience and is kept until the end of the optimization. The main advantage of GLS compared with TS is that it is easy to implement and a few tests suffice to determine if this metaheuristic is appropriate for a problem. On the other hand, TS has many parameters to be customized and a slight modification in one of them may result in a deep change in the performances of the algorithm. In other words, the parameter customization of a TS algorithm is almost an optimization problem in itself!

Practically, the instances we face are usually solved using first-improvement moves, for which the first improving neighbour of the current solution is kept for the next local search step. The main advantage of this choice is to avoid searching over all the neighbourhood at each move, even if each

Figure 4.1: Example of ILOG TPO local search trace



improvement is then expected to be lower than a best-improvement move (for which the best neighbour is kept).

Figure 4.1 depicts a typical search process by ILOG TPO on a reasonnable size instance. The current solution value is displayed depending on computing time. As we can see, the search is a sequence of descents and when a **local minimum** is reached, ILOG TPO has a diversification strategy by a modification of the costs based on the previous search experience. This diversification leads to a new solution with a high cost, and a new local search descent begins from this point.

This algorithm is used in our tests (see next chapter), especially on small and middle-size instances, since for large instances, the first local search descent isn't usually finished at the end of the time limit of 3 hours.

4.2 The path-based model

Before describing the path-based formulation, let us define precisely some important notions.

Recall that in an ILOG TPO point of view, a **shipment path** is a (possibly empty) sequence of hubs in which a given shipment is transshipped. Here, a shipment path is made of the hubs the shipment goes through in its path, without any knowledge whether it is transshipped or not. A shipment path also contains information on the pickup and delivery sites, in the case the shipment has alternate shipments (i.e. in simple terms, several possible pickup and delivery sites). In addition, let a **complete shipment path** be an extension of shipment paths on all sites: a shipment path is a path

between hubs whereas a complete shipment path may include any kind of sites. A complete shipment path is then the sequence of sites, from the pickup site to the delivery site, through which the shipment goes before being delivered. For example, if a shipment goes through sites 1,5,2,4,7,8,3 in this order, where 1 is its pickup site, 3 is its delivery site and 2 and 8 are hubs, then its complete shipment path is $(1, 5, 2, 4, 7, 8, 3)$ and its shipment path is $(1, 2, 8, 3)$, regardless of the vehicle(s) in which it travels. Note that both paths can be seen as sequences of sites, as well as sequences of arcs, providing the graph of sites is complete.

4.2.1 Motivation

ILOG TPO optimization engine is based on Local Search (LS) and has hardly a global view of the problem. Hence, the decisions are locally improving but might be questionable from a global point of view. Using LS, the current solution might not have a sequence of neighbour solutions to get to a very good solution, or if it has one, the sequence may contain very bad solutions that would never be chosen by the optimization engine, whichever the heuristic or metaheuristic is. Furthermore, the sequence might be very long, which would be a serious drawback in terms of computing time. Therefore, we need to take good decisions from a global point of view at the very start of the solving process, so that these decisions impact both the quality of the final solution found and the computing time to find it. For example, we may provide the solver with a bad first solution in terms of cost, but with a structure such that few moves are required to get to a good solution. The aim of this is to obtain a better final solution, or at least a similar final solution in less time.

Transshipment is one of the features that cause questionable decisions. As a cross-dock action involves several vehicles, it is not easy to introduce a simple neighbourhood that is likely to improve the solution by taking advantage of transshipment. For instance, for a shipment s, let a move be the modification between "s direct" (meaning not transshipped) and "s through hub h". If, at that time, there is no other shipment transshipped at hub h, this move will result in a raise of the costs, as there is no benefit from transshipment if less than two shipments are involved. Therefore, any such move will be rejected in a pure LS strategy, whereas it could be an improving move to modify the status of two direct shipments instead of only one.

The basic ILOG TPO first solution is built with a greedy procedure that uses local search, and may take wrong transshipment decisions, which can be the source of the suboptimality of the final solution. A solution to the issue presented above is to find some way to provide every shipment with paths between hubs that are expected to be of a good quality. Of course, such a procedure has to be efficient in a short time, as it is only a small part of the solving. The shipment paths given by this tool could then be frozen

and the optimization would only be focused on other decisions, and removing some of the defined neighbourhoods (those corresponding to shipment paths) obviously accelerates the local search.

A simple way to get the right paths between hubs is to model the problem with variables indexed on shipment paths. This formulation, unlike the arc-based formulation proposed in chapter 3, has to be concise but it has to model the problem as closely as possible, keeping in mind that the only decision that has to come out of the solving of this model is the choice of the shipment path for each shipment.

We chose to neglect the time aspects, for several reasons. First, it seems that to take a decision on the shipment paths, only the network structure is necessary, and time requirements such as time windows wouldn't modify the benefit of using a given shipment path, except in some particular cases. Then, time is convenient to ignore as the vehicles no longer need to be individualized as in the models presented previously. By this way, we can model the vehicles as a flow that only specifies the capacity on each arc travelled by shipments. Recall that table 3.1 lists some of the real-life features of the problem and makes clear whether each of the three formulations can model them.

Path-based models are often provided for vehicle routing problems, especially when the vehicle fleet is heterogeneous (see subsection 2.1.5 for more details). But most of the time, vehicle paths (or tours) are introduced. In our case, the vehicles are modelled as a flow, and the paths aspect is on the shipments.

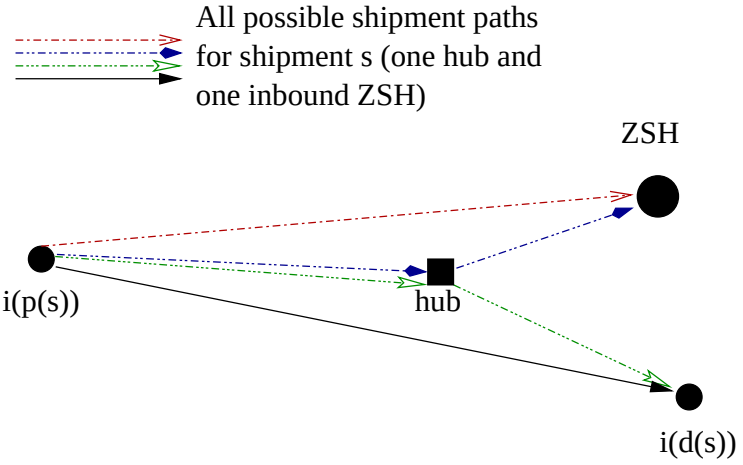
4.2.2 Variables

The variables can be separated into two groups.

The **vehicle flow variables** are the variables that indicate how many vehicles of each fleet travel in the network. n_{ijf} depicts the number of vehicles of fleet f that travel from site i to site j, whereas n^+_{if} (resp. n^-_{if}) is the number of vehicles of fleet f for which the route starts (resp. ends) at site i.

Let $Q(s)$ be the set of all complete shipment paths for shipment s. The **complete shipment path variables** are denoted by θ_{sp}, where shipment s travels through complete path $p \in Q(s)$ if and only if $\theta_{sp} = 1$. Recall that S is the set of shipments, whereas S' also contains the alternate shipments, for instance to model the possiblity for a shipment to be delivered via a ZSH. For example, a shipment $s \in S$, that can be delivered to a nearby ZSH instead of being shipped directly to its destination, is associated with two shipments in S': itself and a shipment s' for which the delivery visit takes place at the ZSH. In this section however, we consider only the shipments $s \in S$, and the shipment alternatives are modelled as additional paths for a shipment s (these paths may have distinct origins and destinations, of

Figure 4.2: Alternative modelled as additional shipment paths



course). Figure 4.2 depicts a basic example with 4 sites where a shipment s has a shipment alternative, and both of the alternate shipments have two shipment paths (direct or through the hub). This results in 4 possible shipment paths for shipment s. Recall that $i(p(s))$ and $i(d(s))$ are its pickup and delivery sites, respectively. However, for the time being, the shipment paths that are considered in $Q(s)$ are complete shipment paths containing any kind of sites. For any shipment, there may be a lot of such complete shipment paths.

In the following, the size of a fleet f is the number of vehicles available in f and is referred to by size(f). Any data notation that has been used previously for vehicles (such as cap(k)) can be kept for fleets (i.e. cap(f)) because in each fleet, all the vehicles are identical.

4.2.3 Constraints

The variables described above are subject to the following constraints:

$$\forall i \in I, \forall f \in F, \quad \sum_{j \in I} n_{ijf} - \sum_{j \in I} n_{jif} = n^+_{if} - n^-_{if} \tag{4.2.1}$$

$$\forall i \in I, \forall f \in F, \quad n^{\pm}_{if} = 0 \quad \text{if } i \neq \text{extrem}^{\pm}(f) \tag{4.2.2}$$

$$\forall f \in F, \quad \sum_{i \in I} n^+_{if} \leq \text{size}(f) \tag{4.2.3}$$

$$\forall s, \in S, \quad \sum_{p \in Q(s)} \theta_{sp} = 1 \tag{4.2.4}$$

$$\forall (ij) \in A, \quad \sum_{s \in S} \left(\text{qty}(s) \sum_{\substack{p \in Q(s) \\ p \ni (ij)}} \theta_{sp} \right) \leq \sum_{f \in F} \text{cap}(f) n_{ijf} \tag{4.2.5}$$

$$\forall i, j \in I, \forall f \in F, \quad n_{ijf}, n^+_{if}, n^-_{if} \in \mathbb{N} \tag{4.2.6}$$

$$\forall s \in S, \forall p \in Q(s), \quad \theta_{sp} \in \{0, 1\} \tag{4.2.7}$$

Constraint (4.2.1) is the flow conservation constraint on each site i and for each vehicle fleet f. This constraint generalizes the Kirchhoff constraint (2.2.2) of the classical network flow formulation described in subsection 2.2.1. Constraint (4.2.2) only holds if the sites $\text{extrem}^{\pm}(f)$ are given in the data, and ensures that no vehicle will start or end its route at a wrong site. Constraint (4.2.3) ensures that in each fleet, the number of vehicle used doesn't exceed the number of vehicles available. Constraint (4.2.4) enforces each shipment to be transported through exactly one shipment path, and it is similar to constraint (2.1.8) that belongs to the path-based formulation of the VRP described in subsection 2.1.5. Here however, we consider shipment paths, and not vehicle paths. Finally, constraint (4.2.5) is the capacity constraint, as well as constraint (2.1.9) from the same VRP path-based formulation.

4.2.4 Objective

The objective is, as described in the previous models (subsection 3.1.5) to minimize costs that can be composed into 4 terms: the fixed cost, the zone-skipping cost (ZSC), the direct transportation cost (DTC) and the additional distance cost (ADC).

$$\min \text{obj} = \text{C}^{\text{fixed}} + \text{C}^{\text{DTC}} + \text{C}^{\text{ADC}} + \text{C}^{\text{ZSC}}$$

The modelling of costs, especially the DTC and the ADC, is complex in this formulation, because they should depend on the first and last sites of the vehicle routes, although no real vehicle route is visible here. Of course,

we could determine each of the vehicle routes a posteriori, but here the point is to have access to them during the search, which is impossible unless many other variables and constraints are introduced, but this is inconceivable as it would break the flow structure for the vehicles. Moreover, there is obviously no way of getting the maximal quantity over a route (denoted by Q_k in subsection 3.1.5), needed for the DTC. Therefore, the idea is to approximate the costs as well as possible, but above all so that a solution that dominates another in terms of the "real" costs will also be better when comparing the model costs. We give here a rough approximation of these costs, as a lot of problem features have already been approximated or even removed (time for example). Moreover, a more sophisticated approximation would not necessarily have a significant impact on the final solution.

- The fixed cost is simply:

$$\mathrm{C}^{\mathrm{fixed}} = \sum_{f \in F} \left(\mathrm{C}^{\mathrm{vehicle}}(f) \sum_{i \in I} n^{+}_{if} \right)$$

- Recall that a zone-skipping cost is a cost that has to be paid for using a special site called ZSH instead of a regular pickup or delivery site. It depends on the total quantity of products that go through a given ZSH. If we decide to linearize it as for the previous models, the zone-skipping cost is also rather easy to write in this model, as shipment alternatives are merged with the notion of shipment paths here:

$$\mathrm{C}^{\mathrm{ZSC}} = \sum_{s \in S} \left(\mathrm{qty}(s) \sum_{p \in Q(s)} \theta_{sp}\, \mathrm{C}^{\mathrm{ZSC}}(p) \right)$$

  $\mathrm{C}^{\mathrm{ZSC}}(p)$ is the ZSH unit cost for path p. In fact, it depends only on the departure and arrival sites (let's say $i^{+}(p)$ and $i^{-}(p)$) of path p. Recall that there may be a ZSH alternative on the delivery visit, but also on the pickup visit. As a result, we have the following correspondance: $\mathrm{C}^{\mathrm{ZSC}}(p) = \mathrm{C}^{\mathrm{ZSC}}(i^{+}(p)) + \mathrm{C}^{\mathrm{ZSC}}(i^{-}(p))$, where $\mathrm{C}^{\mathrm{ZSC}}(i)$ has been introduced for any site i in subsection 3.1.5. If neither of these sites is a ZSH, then $\mathrm{C}^{\mathrm{ZSC}}(p) = 0$.

- The ADC is the additional distance cost and it is only dependent on vehicle routes (and especially on their first and last sites), that however are not explicit in this model.

$$\mathrm{C}^{\mathrm{ADC}} = \sum_{(ij) \in A} \left(\mathrm{C}^{\mathrm{ADC}}(i,j) \cdot \mathrm{distance}^{\mathrm{ADC}}(i,j) \cdot \sum_{f \in F} n_{ijf} \right)$$

where $\text{distance}^{\text{ADC}}(i,j) = \max\{\text{distance}(i,j) - \text{limit}(i,j), 0\}$ is the additional distance travelled between sites i and j. Recall that $\text{limit}(i,j)$ is the distance threshold below which the cost is zero and above which it raises linearly. Here, we handle the routes arc by arc, using the fact that as they are limited in their number of stops, the vehicle routes are made of few arcs. This approximation is good for shuttles in particular, as they are bound to a single lane. Note that this segmentation of the cost over each arc of the route was inspired from the cost function (2.3.1) of the hub location quadratic model decribed in subsection 2.3.1.

- The DTC is the direct transportation cost and should depend on the maximal quantity transported over a vehicle route. As it is impossible to get such information from any variable combination, an approximation has to be considered here as well. First, we noticed that in most of the instances, the vehicle routes don't really mix pickups and deliveries. Usually, a set of pickups is performed at the beginning of the route, and the shipments are delivered without any other pickup in between. However, there may be a transshipment with the vehicle swapping one or more shipments at a hub. In our approximation, we simplify the calculation by considering once again the route arc by arc. Assuming that the vehicles have a limited number of stops, we associate each arc of a shipment path to a vehicle. As a result, the approximate DTC only depends on shipment paths:

$$\text{C}^{\text{DTC}} = \sum_{s \in S} \left(\text{qty}(s) \cdot \sum_{p \in Q(s)} \text{C}^{\text{DTC}}(p) \cdot \theta_{sp} \right)$$

where $\text{C}^{\text{DTC}}(p) = \sum_{(ij) \in \alpha(p)} \text{C}^{\text{DTC}}(i,j)$ is the unit DTC for shipment path p. In other words, the DTC is computed from all the shipments that go through an arc, as if the vehicle routes where limited to one single arc.

Let's consider the route defined by the visit sequence $P_1P_2D_2P_3D_3D_1$ (P are for pickups and D for deliveries), where all the shipments are unitary in terms of quantity. The real DTC for such a route depends on the maximal quantity over the route, which is 2, and on the first and last sites, that are the sites where respectively visits P_1 and D_1 are performed. On the other hand, the approximate DTC is the sum over the path of the real DTC corresponding to a quantity of 1 between the sites of P_1 and D_2, 2 between the sites of P_2 and D_2, 1 between the sites of D_2 and P_3, etc. Therefore, for this kind of route configuration, the approximate DTC is likely to be rough. However, as discussed above, most of the vehicle routes are of the type $P_1 \cdots P_n D_n \cdots D_1$, with all the pickups at the same site or at least at nearby sites, and the same

property holds for deliveries. In this case, the approximation becomes close as the linearization comes from the fact that all the shipments go approximately from the same site to the same site. In the case a shipment is cross-docked, we know that over its path, the vehicle is not the same before the hub and after the hub. Consequently, separating the DTC in all the arcs of shipment paths seems appropriate.

4.2.5 Size of the model

Let $|P|$ be the maximal size of a set of complete paths for any shipment. The number of variables in the model is $|I|^2.|F| + 2|I|.|F| + |S|.|P|$, whereas there are $|I|^2.|F| + |I|.|F| + |S|$ constraints. Obviously, the number of paths for each shipment is determinative of the size of the formulation.

Let's suppose that the length of a complete path is the number of intermediate sites between the pickup ($i(p(s))$) and the delivery sites ($i(d(s))$). Only the direct path between $i(p(s))$ and $i(d(s))$ has a length of 0, and $|I|-2$ paths have only one intermediate site (we assume that the shipment paths contain any site at most once). In the general case, the paths of length $n \leq |I|-2$ are arrangements of n sites in set $I \setminus \{i(p(s)), i(d(s))\}$. The whole set of paths has a factorial size, and if we don't remove some of the paths, our model is unsolvable for even very small instances.

4.2.6 Observations and precisions

Recall that this simplified formulation is only designed to provide ILOG TPO with a promising path between hubs for each shipment. As discussed in the previous subsection, the model is too large if we consider all the complete paths for every shipment. However, it is possible to easily eliminate some paths that we know would be useless in the formulation.

First, the path length can be upper bounded regarding the limit on the number of stops for each vehicle. A shipment path that does not contain any hub has necessarily a length below this limit. The shipment paths can also be restricted by constraints, especially the ones that are not modelled in this formulation (time, for instance). Two sites i_1 and i_2 such that $\text{twLB}(i_1) > \text{twUB}(i_2)$ can't be in the same shipment path unless i_2 is before i_1.

But despite these limitations, the number of paths is still very large, essentially when the shipment is transshipped several times to another vehicle. Furthermore, another problem arises with complete shipment paths. In the formulation, no constraint prevents a shipment from being swapped between two vehicles at any site, even non-hub. Figure 4.3 is a typical example where one of the two shipments s_1 with path $(i'ii_1)$ or s_2 with path $(i'ii_2)$ necessarily has to be swapped between two vehicles in i. Note that the vehicle flow conservation constraint at site i is not violated as soon as one of the two vehicles departing from i ($i \rightarrow i_1$ or $i \rightarrow i_2$) is arriving from another site

Figure 4.3: Example of vehicle change for a shipment

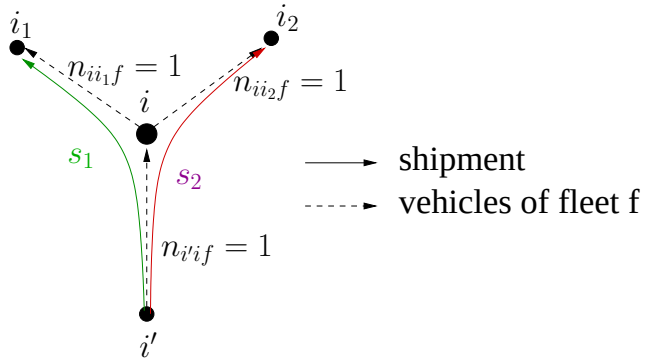
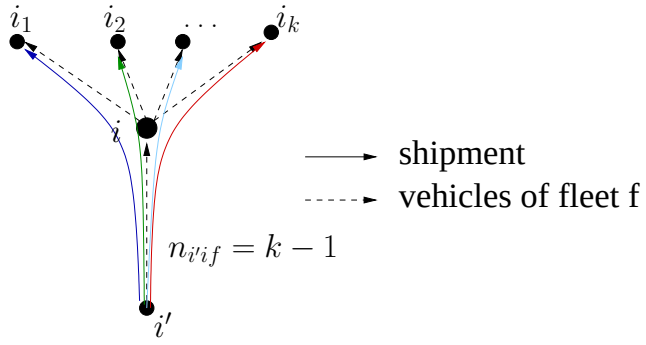


Figure 4.4: Generalization of vehicle change for a shipment



(say $n_{jif} = 1$) or starting its route in i ($n^+_{if} = 1$).

This example raises the issue that nothing in the model prevents a non-hub site from having shipments swapped from a vehicle to another, and adding such a general constraint would be too costly. A generalization of the previous example case is given on figure 4.4: given a set of shipments travelling through arc $i'i$ and then through k distinct arcs, this case occurs whenever the total number of vehicles that carry these shipments on arc $i'i$ is strictly less than k (in particular, if $n_{i'if} = k - 1$).

We have the same kind of configuration by reversing the arcs in figure 4.4.

A simple way to overcome this issue is to consider only the hubs as possible intermediate sites in shipment paths: all the shipment paths would then be paths between hubs. Let $P(s)$ be the set of all possible shipment paths (between hubs) for shipment s. The formulation we consider here can simply be obtained from the previous one by replacing every occurence of set $Q(s)$ in subsections 4.2.3 and 4.2.4 by $P(s)$.

In this model, any site i as described in the previous figures would be a hub, and vehicles swapping shipments in i would not be a problem any more. Furthermore, allowing only hubs to be intermediate sites in shipment paths

dramatically decreases the number of possible paths for every shipment, and hence the size of the problem.

On the other hand, this simplification removes some solutions from the solution set. With this hub restriction, the vehicle routes are also limited. For example, a vehicle can't go to a site i to pickup a shipment, and then to another non-hub site j to pickup another shipment, otherwise the first shipment would have a non-hub site in its path. This reduces dramatically the possiblities of grouping shipments together in a vehicle. However, we chose to use this restriction because of its simplicity and we provide a workaround for the grouping issue.

4.2.7 Site aggregation

To overcome the impossibility of grouping several shipments into the same vehicle, we propose as a preprocessing to group similar sites in the same area in order to be able to pickup or deliver shipments that come from or must go to the same region. Advanced techniques could be used for this matter (such as k-nearest neighbour) but recall that this model is already approximated and there is no need to propose exact or complex improvement algorithms. The sites are then grouped with a greedy heuristic. This also allows us to reduce the number of sites and hence the size of the formulation.

We can notice that in most of the instances, there are sets of sites (usually two sites) that are located at the same place, modelling the fact that two warehouses are close and each of them has specific features (e.g. allocated vehicles). In our formulation, we don't need to keep these sites distinct, as soon as some site-specific data, such as time windows, match. Therefore, such sites located at the same place are merged.

But it is also possible to heuristically aggregate sites that are located roughly in the same area. The idea is to allow shipments that have to be picked up or delivered in the same region to be routed in the same vehicle(s).

Our heuristic is greedy and rather simple. The sites are scanned in a lexicographic order. A set of "master sites" M is maintained during the algorithm. If the current site is near one of the master sites and has the same hub status (hub or non-hub), it is aggregated with this master site. Otherwise, it is added to the set of master sites M, and the same procedure carries on until all sites have been processed. The closeness of two sites can be decided simply by comparing the distance between these sites with a constant parameter (called the **aggregation distance**). If this parameter is too small, few sites are aggregated. On the other hand, if it is too high, some shipments may have their pickup and delivery sites aggregated together, and these shipments then have to be removed from the formulation. Therefore, we have to find a good compromise and the aggregation distance must take into account the network structure.

Before the site aggregation described above, we apply the preprocessing

described in subsection 3.3.2 in order to reduce the number of sites and vehicles.

4.3 Cooperation and integration

In the previous section, we detailed the model that computes the shipment paths that are likely to be of a good quality, regarding the network structure. Each such shipment path for any shipment s is the path $p \in P(s)$ such that $\theta_{sp} = 1$ in the optimal solution. This section will deal with the way ILOG TPO can integrate such guidelines and how they may be helpful for the solving.

4.3.1 Freezing shipment paths

We give shipment paths guidelines to shipments during the solving in ILOG TPO by freezing these shipment paths at the beginning of the local search (LS). The shipments are then forced to travel through the shipment path for which they are provided (if any), and the ILOG TPO local search then just has to focus on other routing decisions, such as the assignment of shipments to vehicles or the scheduling. Freezing decisions on paths has two main advantages:

- ILOG TPO no longer needs to take such decisions for which the engine is not efficient enough in general,
- it reduces the neighbourhood size for any solution during the LS, which usually leads to a significant gain in computation time.

As discussed in the previous section, the notion of shipment alternative (possibility to deliver a shipment through a ZSH) is modelled by additional shipment paths that have the ZSH as a final destination (see figure 4.2). Then, when the path-based formulation provides us with a path p, it gives implicitely the alternate shipment as well. This allows us to use the shipment path guidelines in three different ways:

- Freeze the path between hubs only for the alternate shipment. During the search, if this alternate shipment is chosen, it will be forced to be routed via the path p. Nevertheless, if another alternate shipment is performed, no constraint is imposed on its shipment path. For example, on figure 4.2, it results in removing one of the 4 shipment paths.
- Freeze the path between hubs for all the shipments of the alternative. Given the assumption that all the shipments of an alternative have the same possible shipment paths (recall that a shipment path is just a possibly empty sequence of hubs), we freeze the path over all the alternatives and not only on the alternate shipment that is performed

in the MIP solution. For example, if the alternate shipment is direct in the MIP solution, then all the shipments of its alternative will be forced to be shipped directly in any ILOG TPO solution. This decision of applying the same shipment path to all the shipments in an alternative can be justified by the fact that the ZSH are located in the same region as their regular delivery sites (as well as the pickup sites), and a good shipment path for one of the alternate shipments is likely to be good also for all the other alternate shipments. On figure 4.2, it results in removing two of the 4 shipment paths (either both the direct paths, or both the indirect ones).

- Freeze the path between hubs and the alternate shipment. It is the most constraining way of taking advantage of the shipment paths given by the path-based model. Indeed, we keep only one possibility over all the shipment alternatives and the possible paths between hubs for a given shipment, and there is no choice any more about the transshipment decisions on this shipment. On figure 4.2, 3 shipment paths out of 4 are forbidden.

The three possibilities have their own interest. The last one can be chosen if we are sure of the quality of the shipment path guidelines. Freezing both the shipment path and the alternate shipment results in a drop in the computing time thanks to the neighbourhood reduction. Nonetheless, there is a risk that this constrains too much the shipments and that no solution can be found (because of time constraints, for example), or only bad solutions. The second point is suitable in the case the shipment paths are sure but the choice of the alternate shipments is delicate, for example when the path-based model gives similar costs for every alternate shipments. The first point is the "security" possibility, because the local search can always choose the alternate shipment for which no shipment path has been fixed. However, the processing time is not supposed to be reduced a lot, because this doesn't considerably constrain the ILOG TPO model.

4.3.2 Two-phase algorithm

As the path-based model is an approximated formulation of the problem faced by ILOG TPO, we can't be sure that with the MIP guidelines, we don't remove good solutions from the solution polyedra. At the end of a ILOG TPO run using the guidelines, we can therefore keep searching by relaxing the shipment paths constraints that were added at the beginning. This two-phase procedure can be easily tested as an ILOG TPO external user:

- first run with the guidelines on shipment paths leading to an intermediate solution S,

- second "classical" run (without the guidelines), with S as a first solution.

At the end, we find good solutions in reasonable time, as the first phase is more constrained than a regular ILOG TPO run, and the second phase starts from a good first solution, which is time-saving in local search.

4.3.3 Industrial integration and testing

This work being part of an industrial PhD, we need to detail the way all this work was integrated in the complete software ILOG TPO. ILOG TPO is made of two components: a Graphical User Interface (GUI) coded in Java and a solving engine coded in C++. Of course, in the context of this PhD, there is only a connection with the engine part. However, the GUI is very useful for testing matters as the entire solution information is available and small changes for the current solution are enabled thanks to interactions by hand.

ILOG TPO engine has a data model to store data read from a file. This ILOG TPO data model D_1 is scanned and copied into a simplified MIP data model. Then, the MIP model is solved and the shipment paths are extracted from the solution. A new ILOG TPO data model D_2 is created from the same file as before, and the shipment paths guidelines are given to this data model. D_2 is solved, and its final solution is given as a first solution for the solving of D_1.

In fact, the MIP guidelines are only plugged to ILOG TPO at its inputs and outputs, and is not fully integrated yet. For example, the MIP model has its own data model, which is a simplified version of the ILOG TPO data model. The reason for this is that it allows the guidelines to be easily applied to any other software, and the MIP model doesn't need to be modified when ILOG TPO is updated to a new version. In spite of the external state of the guidelines, it would be easy to fully integrate them into an upcoming customer version of ILOG TPO.

Conclusion

In this chapter, we proposed several algorithms based on the shipment paths guidelines, in order to help ILOG TPO take global decisions a priori. This is included in a two-phase algorithm, with the first phase being a solving of ILOG TPO with these guidelines, and the second phase is just a local search improvement from the last solution of the first phase, on the initial model (the guidelines are relaxed). This is supposed to provide better solutions, or at least comparable solutions in less time, than ILOG TPO alone.

This cooperation between Constraint Programming (CP) and a MIP formulation is also interesting for ILOG, as it allows the perspective of including

another ILOG product into ILOG TPO. For the time being, CP is solved in ILOG TPO using ILOG Dispatcher (software to solve vehicle routing problems), which is derived from ILOG Solver (CP-based solver). With the MIP guidelines, ILOG TPO is likely to integrate ILOG CPLEX in addition in its releases. ILOG TPO is then representative of the efficiency of ILOG softwares in Operations Research.

Chapter 5

Solving real-life routing problems

RÉSUMÉ DU CHAPITRE

L'approche décrite dans le chapitre précédent est testée dans ce chapitre. D'abord, les instances traitées sont décrites. Sauf exception, il s'agit d'instances industrielles qu'ILOG TPO a été amené à résoudre dans les dernières années. Elles sont réparties en quatre groupes, selon leur taille.
Pour les premiers tests, nous choisissons de comparer une exécution de l'heuristique à deux phases décrite au chapitre précédent avec une exécution classique d'ILOG TPO. Les critères de comparaison sont à la fois la valeur de la solution finale et le temps de calcul. Nous montrons ainsi que le principal gain de performance de l'utilisation de l'heuristique à deux phases se trouve dans l'amélioration du temps de calcul. En effet, les solutions trouvées par cette heuristique sont légèrement moins bonnes, mais le temps de calcul est nettement meilleur, surtout lorsque le modèle d'ILOG TPO de la première phase est le plus contraint. Notons que l'amélioration du temps de calcul est plus importante pour les instances de taille moyenne, mais l'écart entre la valeur des solutions trouvées par ILOG TPO et par l'heuristique à deux phases est aussi légèrement plus grand.
Dans un deuxième temps, nous comparons cette heuristique à deux phases où ILOG TPO est guidé par la formulation à base de chemins, avec une moyenne géométrique de 20 exécutions de la même heuristique où, cette fois, les indications sont générées aléatoirement. Ces tests ont pour fonction de valider la qualité des indications données par le modèle mathématique. Nous comparons ici également la solution intermédiaire obtenue entre les deux phases. Nous montrons ainsi que l'heuristique est mieux guidée par la formulation mathématique que par des indications aléatoires. En effet, à la fois la solution finale et le temps de calcul sont meilleurs, ce qui prouve que le modèle à base de chemins donne des indications correctes à ILOG TPO.

The algorithms based on shipment paths guidelines and described in the previous chapter need to be carefully tested. The MIP model defined in section 4.2 is a simplified formulation compared with the core problem. It is only a static representation of the network, without any reference to time aspects or additional constraints such as the maximal vehicle route length. Then, the question is to know if the guidelines given by the MIP are consistent and robust. For that, we propose two testing ways to validate our approach. After a brief description in section 5.1 of the benchmark instances at our disposal and the conventions used in this chapter, we start in section 5.2 by applying the algorithms on small instances for a better understanding. We then compare in section 5.3 the final solution found by TPO with the guidelines each of the algorithms described in the previous chapter with the final solution found by TPO without the guidelines. Both the solving time and the solution values are compared, through ratios calculated for each instance. At last, we seek in section 5.4 to determine if the MIP guidelines are better than randomly generated guidelines by applying the algorithms on several random guidelines for some instances.

5.1 Instances description

The tests that we will present in this section are performed on customer instances that were given by a ILOG TPO third-party logistics provider (3PL), and for matters of confidentiality, their names have been modified.

5.1.1 Generalities

We decided to divide the instances into two categories, depending on the date when they were added to the ILOG TPO benchmark set.

- The oldest instances are called "old-n_1-n_2-n_3-*id*", where $n_1 = |I|$ is the number of sites, $n_2 = |K|$ is the number of vehicles and $n_3 = |S'|$ is the number of shipments and alternate shipments. *id* is a one-digit numerical ID in the case several instances have exactly the same size. Note that n_1, n_2 and n_3 are calculated before any preprocessing such as the ones described in subsections 3.3.2 and 4.2.7.

- The most recent instances are called "new-n_1-n_2-n_3-*id*", with the same conventions as before. These are the instances on which a great work was lately done to improve the ILO TPO engine and they are usually handmade instances from the 3PL in order to point out a problem in the solution of the current version of ILOG TPO. As such, despite their

relatively small size, they are rather hard to solve, as they often have special features.

Instances of exactly the same size (same n_1, n_2 and n_3) have usually been made from the same core instance, but with slightly different features (more constrained time windows, forbidden shipment paths, etc.): the 3PL isn't satisfied when the most constrained version of the benchmark leads to a better solution than the unconstrained one. As a result, solutions of the same size are likely to lead to the same solution, or at least close solutions.

5.1.2 Classification

We chose to classify the set of 119 instances in 4 categories, depending on their number of shipments. Indeed, the number of shipments is the best indicator of the size of the problem, as the number of sites and vehicles can be heuristically estimated from the set of shipments.

- Class "4-12": 19 instances with 12 shipments or less. These instances were used for the arc-based models tests (section 3.4).
- Class "13-43": 41 instances with between 13 and 43 shipments. They are rather small instances, and 23 of the 25 "new" instances are included in this class.
- Class "44-85": 33 instances with between 44 and 85 shipments.
- Class "86-1497": 26 instances with 86 shipments or more. Solving these instances reaches the time limit of 3 hours, and for the biggest ones, no local minimum is reached within the allotted time.

5.1.3 Specificities

The 119 instances that we solve in this chapter are customer instances, and as told before, they contain real-life features that make them hard to solve. In the biggest ones, up to 750 shipments have to be routed in a network with possibilities to go through hubs and to be delivered to a ZSH. In other words, $|S|$ can be up to 750, even if there may be 1497 shipments or alternate shipments in $|S'|$ (recall that S' gathers all the shipments in S and the alternate shipments going through the ZSHs). Numerous constraints stand: maximal duration of any trip, driver breaks, incompatibilities. For more information about these features, see chapter 1.

However, some of the instances are instances modified by the 3PL for testing issues. Sometimes, if instance $\mathcal{I}$ is modified into a more constrained instance $\mathcal{I}'$, the 3PL wants the solution found by ILOG TPO for $\mathcal{I}$ to be better than the one for $\mathcal{I}'$. In other cases, they want the routing plan found on a subset of shipments to be used for the global instance.

Of course, all the instances for which there is no decision to take on shipment paths were removed from the set of benchmarks. In each of the 119 instances, at least one shipment has the possibility to be transshipped in one of the hubs. In fact, the only choice stands on the shipping mode: either direct (no transshipment) or indirect (via a hub). Indeed, in any of the test instances, only one indirect shipment path is proposed for any shipment: only one hub is opened in the whole network. Note also that the hub is always located near the pickup sites and there is no instance with an inbound hub, which classifies the network as outbound pooling.

Recall that in addition of transshipment, every shipment can also have several possible couples pickup and delivery sites referred to as alternatives. In the set of instances, every shipment has at most two alternatives: its regular pickup and delivery sites on the one hand, and for some of them the possibility to be delivered at a ZSH on the other hand. This network can therefore be considered as hub-and-spoke, as the ZSH stands for the inbound hub.

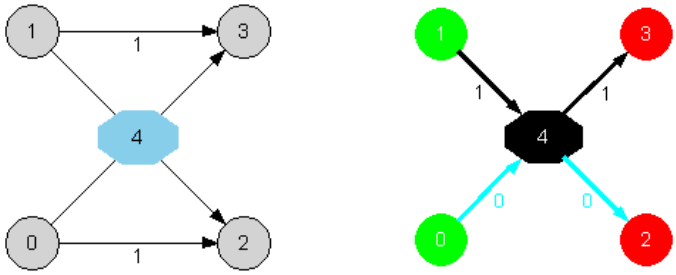
5.2 Small particular cases

In this section, we will illustrate on a couple of examples how the algorithm described in the previous chapter helps ILOG TPO to find better solutions.

The first instance we tackle in this section is a simple handmade instance, designed to show that the MIP model guidelines are useful for ILOG TPO. The instance only contains 5 sites (amongst which 1 hub) denoted by a numerical ID from 0 to 4, 2 vehicles (they can start and end their route anywhere, and have an infinite capacity) and 4 shipments (2 "crossing" shipments $(1 \rightarrow 2)$ and $(0 \rightarrow 3)$, and 2 "straight" shipments $(1 \rightarrow 3)$ and $(0 \rightarrow 2)$). There is no shipment alternative, and every shipment may either be delivered directly to its destination or through the hub. Subfigure 5.1(a) is a representation of the network. The arrows depict the shipments, and the central blue octagonal site (named 4) is the hub. Considering the total distance travelled as the cost function and ignoring any additionnal constraint such as time windows or vehicle capacity, it is quite easy to see that an optimal solution is to transship two of the shipments at the hub. On subfigure 5.1(b) depicting an optimal solution, we can see that both vehicles (denoted by their numerical ID on arcs, and by their colour) meet at the hub in order to swap one shipment each (the crossing shipments). Another optimal solution can be obtained by swapping the second part of the route for both vehicles.

A simple proof of optimality is that similar routes are already optimal only considering these crossing shipments. In this case, a vehicle could travel directly from 1 to 2 with shipment $(1 \rightarrow 2)$ and another one from 0 to 3 with shipment $(0 \rightarrow 3)$, without stopping at the hub. The reason why the plan

Figure 5.1: Shipments and optimal solution for a small instance



(a) Shipments (b) Optimal solution

is optimal in this case can be obtained by enumerating all plausible routes. In particular, a single vehicle route defined by $(1 \rightarrow 0 \rightarrow 2 \rightarrow 3)$, which costs $3a$ (let's say the side length of the square is a), whereas the cost of the solution given above is $2a\sqrt{2}$, which is better. Note that in the case with 4 shipments, the cost of the solution given in subfigure 5.1(b) is $2a\sqrt{2}$ as well because there is no additional cost for the vehicles to swap shipments at the hub.

The fact the both vehicles go through the hub doesn't mean that the shipment path of every shipment contains hub 4, in ILOG TPO sense. For instance, shipment $(1 \rightarrow 3)$ is in vehicle 1 going through hub 4. Nevertheless, it stays in the same vehicle after the hub, and therefore there is no transshipment for it. ILOG TPO considers that this shipment is delivered directly from 1 to 3, even if there is a hub in its vehicle route. In the case ILOG TPO has to build up a solution with each of the 4 shipments enforced to be transshipped at hub 4, any vehicle unloads both shipments it was carrying arriving at hub 4, and then loads back one of them and one that was previously in the other vehicle. Here it is not a real issue as there is no cost for loading and unloading shipments, but in the real-life instances described in section 5.1, it may result in an increase in the total cost. This is a hard thing to determine for the path-based model of the previous chapter, even a posteriori, as the vehicles are not individualized. Fortunately, the solution with too many shipments considered as transshipped at a hub can be improved by simple local search moves in the second phase of the algorithm described in subsection 4.3.2, by just unperforming the "dummy" unloading and loading visits.

This simple instance was tested using ILOG TPO 3.1.3 version. Not designed to tackle so small problems, ILOG TPO returns a solution with two vehicle routes $(0 \rightarrow 1 \rightarrow 2)$ and $(1 \rightarrow 0 \rightarrow 3)$ with both vehicles picking up 2 shipments that have the same delivery site, which of course is not optimal. The cost of this solution is $2a(1 + \sqrt{2})$. On the other hand, when it is guided by the MIP model (which enforces the 4 shipments to be

shipped through hub 4), ILOG TPO finally finds the right solution with the vehicles swapping one of their shipments at hub 4. For this instance, the path-based model improves ILOG TPO solution by a multiplying factor of $\frac{2\sqrt{2}}{2+2\sqrt{2}} = 2 - \sqrt{2} \approx 0.5858$.

However, the most recent ILOG TPO version 3.1.4 manages to find the optimal solution, even without the MIP guidelines. Nonetheless, the MIP guidelines allows the search to reduce the computation time. Alone, ILOG TPO 3.1.4 needs about 17.7 seconds to get to the optimal solution, whereas using the two-phase algorithm described in subsection 4.3.2, it just needs about 6.3 seconds.

As shown with the previous handmade instance, the MIP guidelines can be very helpful to ILOG TPO, either to find a better solution, or to find a similar solution faster.

5.3 Direct comparison

In this section, we compare simply the results of an execution of ILOG TPO as it is without the MIP guidelines (that we will call "TPOalone" in the following) with ILOG TPO launched on a model guided by the shipment paths given by the MIP model ("TPOMIP"). For each class of instances, we will present the geometric mean of the ratio $\frac{Z_{\text{TPOMIP}}}{Z_{\text{TPOalone}}}$ over all instances of each class introduced in section 5.1. Z_{TPOalone} (resp. Z_{TPOMIP}) is the final solution value of a run of ILOG TPO without (resp. with) the guidelines. It means that if the ratio is over 1, the final TPOMIP solution value is greater than the one of TPOalone, and the solution found by TPOMIP is then worse than the one found by TPOalone. We will also present the geometric mean over all instances of the running time ratio between both runs. Both geometric means are calculated for each of the 3 possible uses of the guidelines presented in subsection 4.3.1. The running time for TPOMIP doesn't take into account the time for solving the MIP model: the optimal solution is found at once for almost all instances (using ILOG CPLEX 11.0 on an AMD Opteron 2.4 GHz), and for the biggest ones, a gap of 0.5% between the current solution value and the best lower bound is reached within a few seconds. Anyway, the optimality is not so essential in this case, as the MIP solving only provides ILOG TPO for shipment path guidelines.

The following results were obtained using ILOG TPO 3.1.4 which is the current version.

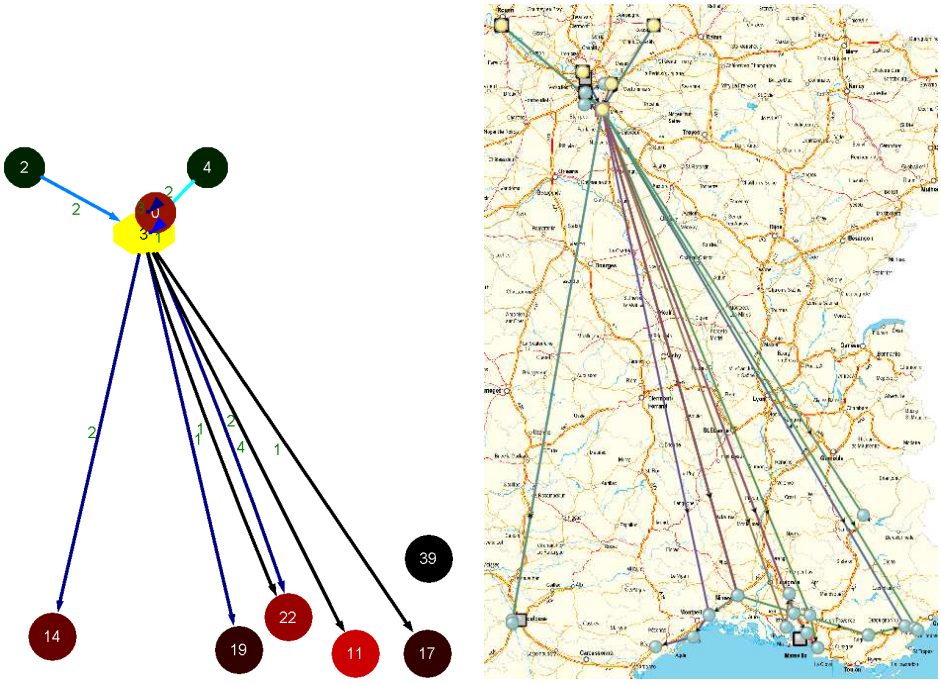
Table 5.1 shows the results obtained when only one shipment path is frozen. Columns "#Worse" and "#Better" enumerate the number of instances for which the result is at least 0.1 percent worse or better for TPOMIP than for TPOalone. We can first notice that globally, TPOMIP does not lead to better solutions, but the difference is only of 1.25 percent in average, whereas it reduces the computing time of 35 percent.

Table 5.1: Solution ratios between TPOMIP and TPOalone with one shipment path frozen

Class	Dimension	Geom. mean	Min	Max	#Worse	#Better
4-12	Value	1.0050	1	1.0774	2	0
	Time	0.7224	0.4545	1.8213	6	13
13-43	Value	1.0206	0.9477	1.2392	18	4
	Time	0.6771	0.2716	1.2907	9	32
44-85	Value	1.0113	0.9895	1.0671	26	5
	Time	0.5443	0.2589	1.0159	1	32
86-1497	Value	1.0035	0.9820	1.0299	9	6
	Time	0.8126	0.3098	1.9218	4	11
Global	Value	1.0125	0.9477	1.2392	55	15
	Time	0.6584	0.2589	1.9218	20	88

The computing time may be high for TPOMIP on the largest instances of class 86-1497, as the time limit may be reached in both phases. Thus, at worst, the ratio between the time spent in TPOMIP and the time spent in TPOalone is almost 2. On the other hand, the geometric mean of the ratio $\frac{Z_{\text{TPOMIP}}}{Z_{\text{TPOalone}}}$ is the better for this class than for any other class, and is close to 1. Moreover, the maximal ratio doesn't exceed 1.03. The problem is that even when a lot of time is spent in TPOMIP, the final solution is not necessarily good. Instance old-188-660-517 reaches the maximal value ratio (almost 1.03) of all 86-1497 instances, although it has a high time ratio (1.9137). In fact, the first phase finds a good solution that is only improved by less than 3 percent in the second phase. This is mainly due to the low improvement at each local search step, because of the first improvement algorithm in ILOG TPO. However, for instance old-42-84-111-1, a good local minimum is found after less than one hour at the first phase, which improves the solution found by the TPOalone execution even before the second phase, and it is three times faster. This is a good example where the cooperation between ILOG TPO and the MIP guidelines produces a very good solution. We can see on figure 5.2 that the vehicle flow given by the solving of the path-based model is quite similar to the one of the final solution found with TPOMIP. On subfigure 5.2(a), the arcs are valuated by the number of vehicles and they are of different colors for distinct fleets, whereas on subfigure 5.2(b), each vehicle has its own route and its own color. A site i is colored in black if no vehicle starts or ends at i, and is gradually green (resp. red) if more vehicles start (resp. end) there, and the yellow octagonal site (3) is a hub. For example, four vehicles of the fleet colored in black end their route in site

Figure 5.2: Comparison of the vehicles flows for instance old-42-84-111-1



(a) MIP flow (b) ILOG TPO flow

11. Note that there are fewer sites in the MIP flow, as the preprocessing described in section 4.2.7 aggregated some sites together. The aggregation distance was set to 80 kilometers, when the maximal distance is usually between 500 and 800 kilometers.

In the class 4-12 containing the smallest instances, the time differences between TPOalone and TPOMIP are not really significant, as for these instances the solving only lasts a few seconds. Nevertheless, TPOMIP is still faster than TPOalone, with similar values for the final solution cost. 44-85 is the class where the best results are obtained, with computing times that are almost twice faster in average for TPOMIP than for TPOalone.

In table 5.2, we compare the solution values and the computation times between TPOalone and TPOMIP with the shipment path frozen for every shipment in the alternative. Globally, the results are slightly less good than the ones in table 5.1. For the smallest instances however, the solution value is better for TPOMIP than for TPOalone, in average. Note that out of the 10 instances that have a lower cost for TPOMIP than for TPOalone, 7 are already better after the first phase. The best time results are still obtained by class 44-85, but the values are less good than before (only one improved instance, whereas 31 of them have a higher cost).

At last, table 5.3 provides the results when both the shipment path and

Table 5.2: Solution ratios between TPOMIP and TPOalone with one shipment path frozen for each alternate shipment

Class	Dimension	Geom. mean	Min	Max	#Worse	#Better
4-12	Value	0.9949	0.9109	1.0774	1	2
	Time	0.8885	0.4938	2.8936	7	12
13-43	Value	1.0249	0.9792	1.1308	19	3
	Time	0.7614	0.2890	1.7643	15	26
44-85	Value	1.0206	0.9901	1.0872	31	1
	Time	0.5738	0.2914	0.8472	0	33
86-1497	Value	1.0141	0.9746	1.1007	11	4
	Time	0.9077	0.5900	1.9222	6	10
Global	Value	1.0168	0.9109	1.1308	62	10
	Time	0.7367	0.2890	2.8936	28	81

the alternate shipment are frozen. The first thing to notice is that the computing time is clearly better in this case than in both other methods. Nonetheless, TPOalone has a solution value 3 percent better in average than TPOMIP. This is due to the fact that the model used in the first phase is more constrained than the core model, so it is solved faster (there are fewer neighbourhoods), but some good solutions may be removed from the solution set, as the MIP model is an approximation. In fact, the cooperation between the MIP model and ILOG TPO is not as good here as in the previous cases. On the other hand, the processing time is tremendously reduced.

ILOG TPO has been lately improved so that the new instances get good results in terms of solution values, regardless of the computation time. This improvement of the ILOG TPO engine was performed on the basic local search heuristics, and ignores elaborate procedures such as the two-phase heuristic described in the previous chapter. This explains the large gap that can be observed between TPOalone and TPOMIP, especially on new instances. In particular, the geometric mean of the solution values in class 13-43 is 1.0593, mainly due to the fact that almost all the new instances are part of this class. The geometric mean obtained for old instances of class 13-43 is only 1.0361 whereas the new instances of class 13-43 reach a mean of 1.0778. This shows that the cooperation between the ILOG TPO engine and the MIP guidelines has a large potential for improvement.

The over-constrained first phase model has another drawback. The first solution heuristic doesn't find any solution for 18 of the 119 instances, not because there is no solution at all, but because the first solution greedy algorithm is not designed for models that are constrained like this one. This issue

Table 5.3: Solution ratios between TPOMIP and TPOalone freezing both the shipment path and the alternate shipment

Class	Dimension	Geom. mean	Min	Max	#Worse	#Better
4-12	Value	0.9998	0.9252	1.0774	1	1
	Time	0.5451	0.2693	2.5191	5	14
13-43	Value	1.0593	0.9792	1.3078	32	1
	Time	0.4032	0.1426	2.0339	4	37
44-85	Value	1.0297	0.9918	1.0810	28	3
	Time	0.2021	0.0795	1.5494	1	32
86-1497	Value	1.0219	0.9746	1.1149	13	2
	Time	0.7697	0.2497	1.9647	9	8
Global	Value	1.0336	0.9252	1.3078	74	7
	Time	0.3815	0.0795	2.5191	19	91

was overcome by setting a high penalty cost to the other alternate shipments than the one that must be performed. However, with this workaround the computation time benefits of constraining the model are lost, and for these instances, the ratio between the running times of TPOMIP and TPOalone is much higher than for any other instance. By the way, the maximum time ratio for each instance class is reached for one of these 18 instances.

5.4 Random guidelines

Comparing the results of TPOMIP with TPOalone is not enough: even if TPOMIP provides a better solution than TPOalone, nothing proves that it is thanks to the guidelines given by the MIP model. These guidelines may be of a bad quality, and the ILOG TPO Local Search engine may be powerful enough to fix this up. The efficiency of the cooperation between the MIP model and ILOG TPO can then be further tested by considering random guidelines and applying the same two-phase procedure as the one described for TPOMIP in subsection 4.3.2. This algorithm using random guidelines can be called TPOrandom. On each instance, we can then compare the results given by TPOMIP with the average on an empirical study of TPOrandom. In TPOrandom, shipment paths and alternate shipments are chosen depending on a uniform distribution.

The problem is that all the instances could not be tested a sufficient number of times to get valid results, especially on instances of type 86-1497. As a consequence, a subset of 62 instances were tested, with 20 TPOrandom

Table 5.4: Solution ratios between TPOMIP and TPOrandom

Class	Dimension	Geom. mean	Min	Max	#Worse	#Better
4-12	Interm	0.9364	0.8222	1.1134	4	11
	Final	0.9912	0.9414	1.0912	2	6
	Time	0.7567	0.5222	1.0784	2	13
13-43	Interm	0.7848	0.0071	1.0864	2	36
	Final	0.9860	0.8549	1.1917	11	24
	Time	0.9036	0.4671	1.6637	12	25
44-85	Interm	0.9361	0.9196	0.9682	0	8
	Final	0.9937	0.9866	0.9997	0	7
	Time	0.7832	0.6267	1.0016	1	7
86-1497	Interm	0.9344	0.9344	0.9344	0	1
	Final	1.0174	1.0174	1.0174	1	0
	Time	0.9911	0.9911	0.9911	0	1
Global	Interm	0.8403	0.0071	1.1134	6	56
	Final	0.9887	0.8549	1.1917	14	37
	Time	0.8511	0.4671	1.6637	15	46

runs for each:

- 15 instances from class 4-12,
- 38 instances from class 13-43,
- 8 instances from class 44-85,
- 1 instance from class 86-1497.

The results are given in table 5.4. In these tests, we used the first method described in subsection 4.3.1, i.e. we freeze only the shipment path for one alternate shipment. This time, the calculations are made from the ratio $\frac{Z_{\text{TPOMIP}}}{Z_{\text{TPOrandom}}}$ of each instance. In fact, $Z_{\text{TPOrandom}}$ is the arithmetic mean over all random runs of the solution value found by TPOrandom. As such, it is not a "real" solution value, it is the approximate value that is expected to be obtained when launching a single random run. However, in most of the cases, the standard deviation is rather high, showing that random runs are not robust at all. In table 5.4, the first line of each instance class ("Interm") stands for the ratio obtained comparing each intermediate solution of TPOMIP and TPOrandom, i.e. the one obtained after freezing the shipment paths and that is used as a first solution in the second phase of the algorithm. "Final" denotes the ratio for the final solution and "Time" the time ratio.

The last three lines show that the quality of the solution found by TPOMIP is better than for TPOrandom, but also that the computing time is lower. In fact, the intermediate solution has usually a far lower cost for TPOMIP than for TPOrandom: only 6 instances are better for TPOrandom in average. For these 6 instances, the TPOMIP final solution isn't better either, which shows that when the TPOMIP intermediate solution isn't good, its final solution isn't likely to have a low cost either. This shows that TPOMIP finds good quality intermediate solutions thanks to the shipment path guidelines, and that it contributes to reaching good final solutions. The time benefit of using TPOMIP instead of TPOrandom can also be explained by the quality of the intermediate solution, as the second phase is likely to last longer with a worse intermediate solution (more moves have to be made to reach a local minimum).

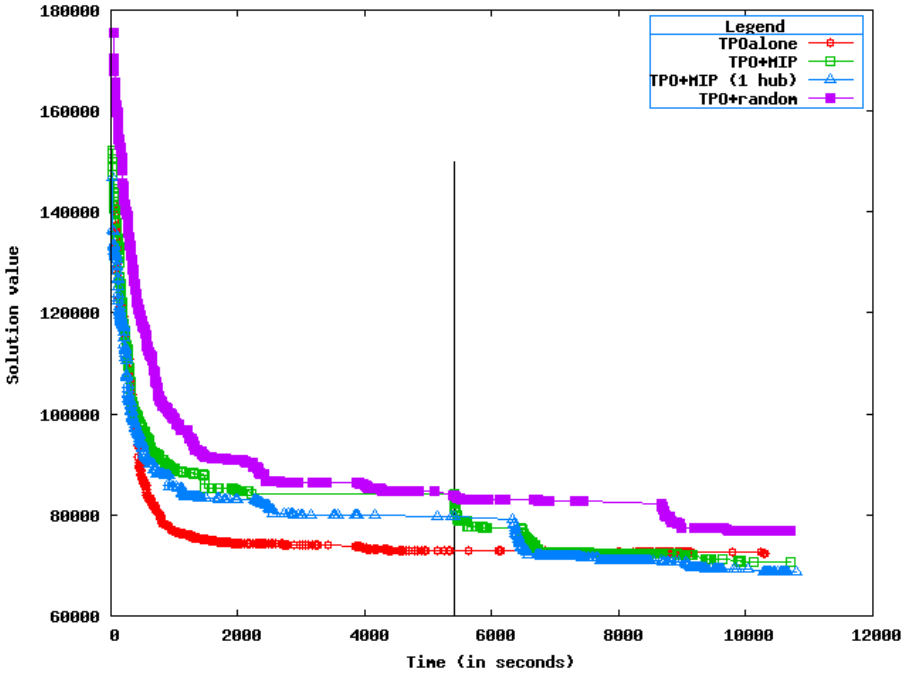
However, some intermediate solutions may be good without leading to an outstanding final solution. The instance from class 86-1497 (namely old-188-660-517) is in this case: after 3 hours spent in the second phase of the algorithm, the intermediate solution is only improved by 2.6%, whereas this rate is over 10.5% in average for TPOrandom. The cost of the TPOMIP intermediate solution is even lower than any of the 20 TPOrandom intermediate solutions. As for any other big instance, no local minimum is reached within the allotted time, and here every local search move in the second phase of TPOMIP only improves the solution by a small step. Note that old-188-660-517 is also the instance of class 86-1497 for which the maximal solution value ratio is reached in the comparison with TPOalone (see table 5.1).

All tested instances of class 44-85 have better intermediate and final solution values for TPOMIP than for TPOrandom, and the computing time is better of over 20% in average. In class 13-43, the minimum ratio on the intermediate solution values is very low. This is because for instance old-23-2-20, the random guidelines sometimes provide an intermediate solution with a very high cost, as one of the shipments is unperformed, which adds a great penalty to the total cost.

In order to get an idea of the performances of the algorithm on any kind of instances, we test it on an instance (called HubsData) that is used by ILOG TPO for customer demonstrations. The main particularity of this instance compared with the instances previously introduced is that it allows several hubs to be used for transshipment. There is both an outbound hub and an inbound hub, meaning that most of the shipments can be cross-docked twice during their trip.

On figure 5.3, we can compare the search descent of ILOG TPO without any guideline (TPOalone) with the ones of the guided ILOG TPO, either with the MIP guidelines (TPO+MIP) or with random guidelines (TPO+random). In the MIP guidelines, we also tried to enforce in the MIP solving that the paths should go through at most one hub (TPO+MIP 1 hub). Note that unlike what was presented on figure 4.1 there is only

Figure 5.3: Example of local search process on instance HubsData



one local search descent, as the local minimum isn't reached by any of the runs. At last, the vertical line stands for the time when the first phase of the algorithm is stopped and the second phase starts (except for TPOalone, for which there is no second phase). As the time limit is set 10800 seconds (or 3 hours), we chose to stop the first phase after 5400 seconds.

ILOG TPO manages to find good solutions at the beginning of the search, but it doesn't improve a lot after some additional time. The advantage of the two-phase heuristic is that there are usually at least two drops of the solution value (one for each phase), as we can see on figure 5.3. Other drops can be justified by the fact that an effective neighbourhood is reached. As TPOalone is less constrained than any other runs, it has the best start and reaches a cost of 80000 quite quickly. We can also notice that the random run has the highest cost at almost any time, which shows that ILOG TPO doesn't manage to repair the potentially bad guidelines that it has to conform during the first phase. What is surprising is that the guidelines that come from the most constrained MIP (where shipment paths with two hubs are forbidden) lead to the best results with a final solution at a cost below 69000.

Conclusion

The last version of ILOG TPO globally gives better solutions with its core engine than when it is guided by the MIP model. However, with the guidelines, the solutions are very close to the ones found without the guidelines, and for some instances, they may even be better. The main point is that the computation time is reduced a lot when using the guidelines because they constrain the model and the final solution is reached faster, despite the two phases of the heuristic. Using any of the freezing algorithms described in section 4.3.1 corresponds to constraining the model more or less. The results show that when the model is more constrained, the solution value is not better but the computation time is far lower.

On the other hand, the MIP guidelines are better than the random guidelines both in terms of solution value and computation time. ILOG TPO guided randomly usually finds a worse intermediate solution between the two phases, which enforces the second phase to spend more time in the local search process until the local minimum. This shows that at least, the shipment paths guidelines given by the MIP model are likely to give some help to ILOG TPO.

The comparison of the solutions of the two-phase heuristic with the solutions found by ILOG TPO alone does not necessarily show that the MIP guidelines are bad. Anyway, this is contradicted by the comparison with the random guidelines. This is just a warning that the cooperation between the MIP model and ILOG TPO may not be the best we can expect. Moreover, there are lots of parameters, on the MIP side as well as ILOG TPO side, that may be more suitable to this cooperation. Overall, the time gain and the comparison with random guidelines show that the results on the MIP shipment paths guidelines are encouraging.

Conclusions and future work

Transportation companies are concerned to find low cost routing plans, especially when they have to serve many sites. As a slight change in a vehicle route may increase (or decrease) dramatically the total cost, they often call upon Operations Research tools. ILOG TPO is one of them, and is designed to deal with hard real-life problems whose main feature is transshipment.

In the context of this PhD, the aim was to provide ILOG TPO with a tool to improve its performance because the software doesn't necessarily take the good cross-docking decisions. For the smallest instances, we proposed two complex Mixed-Integer Programming models in order to be as close to the real-life problem as possible. These MIP models are based on a classical formulation for which some variables are indexed on arcs and vehicles. Even for small instances, the arc-based models have a lot of variables and constraints. Therefore, a few improvements were proposed to either reduce the size of the initial problem, or to help the solving, such as a cutting-plane technique. The results show that the smallest instances are solved to optimality in reasonnable computation time, and the final MIP solutions are similar to the ones found by ILOG TPO.

On larger instances, we decided to provide ILOG TPO with another MIP model that is able to find which paths between hubs are promising, for each shipment. This model is based on paths through which shipments can be routed, and on a vehicle flow aspect. Since we consider in our model that only hubs may be included in shipment paths, we have to apply some preprocessing procedures such as a site aggregation, in order to allow the vehicles to perform some sequences of pickups or deliveries in distinct sites. All the costs are approximated as in the previous models, but this one is simplified a lot and doesn't take into account time aspects, among others. As a result, the computing time is tremendously low even for large instances. The shipment paths found by this path-based model can be frozen in three possible ways for an ILOG TPO solving, and a two-phase procedure is applied using the ILOG TPO solving engine. With this procedure, we can notice that, in average, the solutions found on the set of ILOG TPO customer benchmarks are just slightly worse than the ones found by ILOG TPO without the MIP shipment paths guidelines. On the other hand, the computation time is improved by a great factor. Moreover, the intermediate solution found between

the two phases is usually rather good already. Tests were also launched on comparison with randomly generated guidelines. The intermediate solution and the time computation are clearly worse for the random guidelines than for the MIP guidelines, and the final solution is also worse, to a lesser extent. This validates the whole guiding procedure as well as the quality of the shipment paths provided by the MIP model.

This cooperation between the local search methods used by ILOG TPO and the mathematical programming model is one of the main theoretical benefits of this PhD. The MIP model, with its global view of the network structure, can help ILOG TPO take decisions linked with vehicle routes or the way shipments are routed. An integration of the MIP model is considered for after the PhD as some of the ILOG TPO customers are not satisfied with the high computing time on some instances, that could be reduced by using smartly the path-based model.

Despite all the benefits that we have already got from this work, the PhD leaves us with many perspectives of improvements.

The arc-based formulation still has some flaws in the modelling. Even if this situation is rare in practice, vehicles should be able to go several times through the same site. We could also introduce an inventory management aspect by allowing vehicles to arrive at a hub at distinct dates for the same transshipment. One vehicle could for example come before others and leave at the hub some shipments that would be taken later by other vehicles. The problem we had with the MVRPPD model while testing both arc-based formulations could be resolved by allowing alternatives to be modelled. Then, new boolean variables and constraints would have to be added to the formulation, but then the MVRPPD model could be used as well as the PDP model to solve the ILOG TPO customer instances to optimality, or at least the smallest ones. Furthermore, there has to be some tests on the quality of suboptimal solutions. There may be a great time saving in stopping Branch-and-Bound when the gap between the lower bound and the current best solution value is lower than a given limit, say 1 or 2 percent. The solutions obtained at this point would have to be compared to optimal solutions to determine if this premature stop is acceptable.

The cutting-plane heuristic could also be improved. We designated the big-M constraints as the cutting planes, but there might be a smarter decision. In particular, constraints that are hard for the formulation and that are not often violated are very good cutting-plane candidates. More tests could be performed in order to see if a subset of big-M constraints, or even other constraints, would be a better choice. Moreover, the cutting-plane procedure is an automatic method from ILOG CPLEX, whereas a customized heuristic with our own separation procedure would probably be more suitable.

In spite of the good results of big-M constraints compared with the ILOG CPLEX IloIfThen method, some other disjunction-linearizing procedures could be tested. Several works from Balas, Glover or Jeroslaw, that are

summerized in [Sherali and Shetty, 1980], could be used for this matter.

The solutions obtained by the MIP formulations could be compared with the ones found by ILOG TPO , especially in terms of costs. Despite the fact that the MIP costs are not exactly similar to the industrial costs, the real cost of an optimal MIP solution could be computed (or estimated) after the solving in order to determine whether this MIP solution is better than the one given by ILOG TPO.

Concerning the path-based model and the two-phase cooperation heuristic, the main remaining work is to better customize the settings of both sides (MIP model, ILOG TPO), but also of the cooperation itself, in order to get the best results. For this, a complete testing process has to be organized, launched and carefully examined. For example, we must seek which kinds of decisions are reconsidered by ILOG TPO between the intermediate and the final solutions. Moreover, we have to determine why, for some instances, intermediate and final solutions are not as good as expected, and which of the MIP guidelines or the ILOG TPO local search engine are responsible for this. For that matter, the ILOG TPO Graphical User Interface helped us improve the formulation, and particularly the zone-skipping costs. By the way, there are still some modelling improvements to look into, since as it is now, the formulation is very fast to solve, even on large instances. Better costs, incompatibilities, vehicles maximum number of stops could be introduced. These improvements are not easy to implement because of the fact that the vehicles are not individualized in the path-based model. Moreover, the guidelines have to be generated from the MIP solution in a better way, using for example the fact that a shipment inside a given vehicle while arriving at a hub, and that stays in the same vehicle after the hub, is not supposed to have been transshipped at the hub. As vehicles are not individualized, we could generalize this for vehicle fleets (especially if there is only one vehicle arriving at the hub and leaving the hub) and see if the generated shipment paths give better results. Besides, ILOG CPLEX 11.0 allows the user to get several solutions at the end of the search. It could be interesting to compare some good solutions and to keep for the guidelines only the shipment paths that they all share.

Instead of only focusing on paths between hubs and allowing several vehicle stops by aggregating sites, we could consider all shipment paths, but introduce more constraints to prevent a non-hub intermediate site from being considered as a hub in the vehicle flow. However, such constraints would be complex, even for small path lengths, and could jeopardize the efficiency of the formulation. On the other hand, restrictions on shipment paths could come from constraints such as time window consistency or the maximal number of vehicle stops.

Another major perspective is also to think of a better cooperation between the path-based MIP model and the ILOG TPO local search engine. We chose to keep from the MIP solution only the shipment paths (and the

shipment alternatives), since such decisions are hard to take alone for ILOG TPO. But there may be other benefits from the solution given by the mathematical formulation. In particular, we saw that the vehicle routes obtained with the MIP model are very similar to the ones of the ILOG TPO final solution. Then, the MIP routes could be used to help ILOG TPO find its first solution, or even during the local search. We could also consider a back-and-forth communication between the MIP model and ILOG TPO local search, in particular to get a better approximation of the costs in the MIP. A MIP solution could be exported (when possible) in a ILOG TPO format in order to get its real cost. The cost could then be updated in the Integer Program, which could be solved again. All these cooperation ideas are interesting, but rather hard to implement, as some work has to be performed in the basic ILOG TPO engine as well. Anyway, the ILOG TPO local search engine has to be readapted to the cooperation with the MIP model, as in some cases, the best guidelines don't necessarily drive ILOG TPO to the best solution (with the same shipment paths as in these guidelines).

At last, some work can be done on data given by the 3PL. In the instances we received from them, there is always only one available hub, although there are usually four potential hubs. This may be a preprocessing from their side that eliminates all the hubs that aren't likely to be used in a good solution. But our tools are adapted to networks with several hubs, and they might be interested in a suggestion not to ignore any hub and let our tools decide which one(s) would be the best to open.

Conclusions et perspectives

Les entreprises de transport ont intérêt à trouver des plans de tournées à faible coût, en particulier lorsqu'ils doivent servir beaucoup de sites. Comme la moindre modification dans la tournée d'un véhicule peut augmenter (ou diminuer) considérablement le coût total, elles font souvent appel aux outils de Recherche Opérationnelle. ILOG TPO est l'un d'eux, et est adapté aux problèmes industriels difficiles dont la principale caractéristique est le transbordement.

Dans le contexte de cette thèse, nous voulons fournir à ILOG TPO un outil pour améliorer ses performances car le logiciel ne prend pas nécessairement les bonnes décisions de transbordement. Pour les plus petites instances, nous avons proposé deux programmes mixtes en nombres entiers (MIP) pour se rapprocher le plus possible de la problématique industrielle. Ces modèles MIP sont basés sur une formulation classique où des variables sont indexées sur les arcs et les véhicules. Même pour de petites instances, les modèles à base d'arcs ont beaucoup de variables et de contraintes. Par conséquent, quelques améliorations ont été proposées pour soit réduire la taille du problème initial, soit pour aider la résolution, comme par exemple une technique de plans coupants. Les résultats montrent que les plus petites instances sont résolues à l'optimalité en un temps raisonnable, et les solutions finales du MIP sont similaires à celles trouvées par ILOG TPO.

Sur de plus larges instances, nous avons décidé de fournir à ILOG TPO un autre modèle MIP qui peut déterminer quels chemins de hubs sont prometteurs, pour chaque ordre de transport. Ce modèle s'appuie sur les chemins par lesquels les ordres peuvent être acheminés, et sur un aspect flot de véhicules. Puisque nous supposons dans notre modèle que seuls les hubs peuvent être inclus dans les chemins des ordres de transport, nous devons appliquer des procedures de pré-traitement comme une aggrégation des sites, de façon à permettre aux véhicules d'accomplir des suites de ramassages ou de livraisons sur des sites distincts. Tous les coûts sont approchés comme dans les modèles précédents, mais celui-ci est très simplifié et ne prend pas en compte les aspects de temps, entre autres. En conséquence, le temps de calcul est extrèmement bas même pour de grandes instances. Les chemins des ordres trouvés par ce modèle à base de chemins peuvent être figés de trois différentes manières dans une résolution d'ILOG TPO, et une heuristique

à deux phases est utilisée avec le moteur de résolution d'ILOG TPO. Avec cette procédure, nous remarquons qu'en moyenne, les solutions trouvées sur l'ensemble des jeux de tests industriels d'ILOG TPO sont juste légèrement moins bonnes que celles trouvées par ILOG TPO sans les indications du MIP sur les chemins des ordres de transport. Par contre, le temps de calcul est amélioré d'un facteur considérable. De plus, les solutions intermédiaires trouvées entre les deux phases est souvent déjà bonne. Des tests ont été également lancées pour une comparaison avec des indications générées aléatoirement. La solution intermédiaire et le temps de calcul sont clairement moins bons pour les indications aléatoires que pour les indications du MIP, et la solution finale est aussi moins bonne, dans une moindre mesure. Cela valide la procédure complète de coopération ainsi que la qualité des chemins des ordres de transports fournies par le MIP.

Cette collaboration entre les méthodes de recherche local d'ILOG TPO et le modèle mathématique est un des principaux intérêts théoriques de cette thèse. Le modèle MIP, avec sa vue globale de la structure du réseau, peut aider ILOG TPO à prendre des décisions liées aux tournées des véhicules ou aux moyens par lesquels les ordres de transport sont acheminés. Une intégration du modèle MIP est envisagée pour après la thèse car quelques uns des clients d'ILOG TPO ne sont pas pleinement satisfaits des temps de calcul importants sur certaines instances, qui pourraient être réduits en utilisant le modèle à base de chemins de façon ingénieuse.

Malgré tous les bienfaits que nous pouvons déjà tirer de ce travail, la thèse nous laisse beaucoup de perspectives d'amélioration.

La formulation à base d'arcs a encore quelques défauts dans la modélisation. Même si cette situation arrive rarement en pratique, les véhicules devraient être capables de passer plusieurs fois par le même site. Nous pourrions également introduire un aspect gestion de stocks en permettant aux véhicules d'arriver à un hub à des dates distinctes pour un même transbordement. Un véhicule peut par exemple arriver avant les autres et laisser au centre de transbordement des ordres de transport qui peuvent être chargés par d'autres véhicules plus tard. Le problème que nous avons eu avec le modèle MVRPPD en comparant les deux formulations à base d'arcs pourrait être résolu en modélisant les alternatives d'ordres de transport. Alors de nouvelles variables et contraintes apparaîtraient dans la formulation, mais le modèle MVRPPD pourrait être utilisé comme le modèle PDP pour résoudre des instances industrielles d'ILOG TPO à l'optimalité, ou au moins les plus petites d'entre elles. De plus, la qualité des solutions légèrement sous-optimales doit être testée. Il est probable qu'il y ait un gain de temps coonsidérable en arrêtant le Branch-and-Bound quand l'écart entre la borne inférieure et la valeur de la meilleure solution courante est inférieure à une certaine limite, disons 1 ou 2 pour cent. Les solutions obtenues à ce moment-là devraient être comparées aux solutions optimales pour déterminer si cet arrêt prématuré est acceptable.

L'heuristique de plans coupants pourrait également être améliorée. Nous avons choisi les contraintes grand-M comme plans coupants, mais il peut y avoir une décision plus judicieuse. En particulier, les contraintes qui sont dures pour la formulation et qui ne sont pas violées très souvent sont de très bonnes candidates. Des tests peuvent être approfondis pour vérifier si un sous-ensemble de contraintes grand-M, ou même d'autres constraintes, serait un meilleur choix. De plus, notre procédure de plans coupants est une méthode automatique d'ILOG CPLEX, alors qu'une heuristique dédiée avec notre propre procédure de séparation serait probablement mieux adaptée.

En dépit des bons résultats des contraintes grand-M par rapport à la méthode IloIfThen d'ILOG CPLEX, d'autres procédures de linéarisation de disjonctions pourraient être testées. De nombreux travaux de Balas, Glover ou Jeroslaw, qui sont résumés dans [Sherali and Shetty, 1980], pourraient être utilisés dans cette optique.

A propos du modèle à base de chemins et de l'heuristique de coopération à deux phases, le principal travail restant est de mieux paramétrer les deux côtés (modèle MIP et ILOG TPO), mais aussi la coopération elle-même, pour avoir les meilleurs résultats. Pour cela, un processus de test complet doit être organisé, lancé et examiné attentivement. Par exemple, nous devons chercher quels types de décisions sont remises en cause par ILOG TPO entre les solutions intermédiaire et finale. De plus, nous devons déterminer pourquoi, pour certaines instances, les solutions intermédiaires et finales ne sont pas aussi bonnes que prévu, et qui entre des indications du MIP et la recherche locale d'ILOG TPO en est responsable. Pour cela, l'interface graphique d'ILOG TPO nous a aidé à améliorer la formulation, et plus particulièrement les coûts de ZSH (plate-forme régionale). D'ailleurs, il y a toujours des améliorations de modélisation à examiner, d'autant que dans on état actuel, la formulation est très rapide à résoudre, même sur les grosses instances. Nous pourrions introduire de meilleurs coûts, les incompatibilités, le nombre d'arrêts maximum pour les véhicules. Ces améliorations ne sont guère faciles à implémenter à cause du fait que les véhicules ne sont pas individualisés dans le modèle à base de chemins. De plus, les indications doivent être mieux générées depuis la solution du MIP, en utilisant par exemple le fait qu'un ordre de transport se trouvant dans un véhicule quand celui-ci arrive à un hub, et qui y reste en quittant le hub, n'est pas supposé avoir été transbordé à ce hub. Comme les véhicules ne sont pas individualisés, nous pourrions généraliser cela aux flottes de véhicules (en particulier s'il n'y a qu'un véhicule arrivant au hub et quittant le hub) et voir si les chemins générés donnent de meilleurs résultats. D'autre part, ILOG CPLEX 11.0 permet à l'utilisateur d'obtenir plusieurs solutions à la fin de la recherche. Il pourrait être intéressant de comparer quelques bonnes solutions et de garder comme indications seulement les chemins des ordres de transport communs à toutes ces solutions.

Au lieu de ne se focaliser que sur les chemins de hubs et d'autoriser plu-

sieurs arrêts pour les véhicules en agrégeant les sites, nous pourrions prendre en compte tous les chemins des ordres de transport, mais introduire plus de contraintes pour empêcher tout site intermédiaire non-hub d'être considéré comme un hub dans le flot de véhicules. Cependant, de telles contraintes seraient complexes, même pour de petites longueurs de chemins, et pourraient porter préjudice à l'efficacité de la formulation. En revanche, des restrictions sur les chemins des ordres de transport pourraient venir de contraintes comme la cohérence des fenêtres de temps ou le nombre d'arrêts maximal pour les véhicules.

Une autre perspective essentielle est également de penser à une meilleure coopération entre le modèle MIP à base de chemins et le moteur de recherche locale d'ILOG TPO. Nous avons choisi de ne garder de la solution du MIP que les chemins des ordres de transport (et les alternatives d'ordres de transport) étant donné que de telles décisions ont du mal à être prises par ILOG TPO. Mais on pourrait tirer d'autres avantages de la solution donnée par la formulation mathématique. En particulier, nous avons vu que les tournées de véhicules obtenues avec le modèle MIP étaient très similaires à celles de la solution finale d'ILOG TPO. Alors, les tournées du MIP pourraient être utilisées pour aider ILOG TPO à trouver sa première solution, ou même pendant la recherche locale. Nous pouvons aussi imaginer un aller-retour d'informations entre le modèle MIP et la recherche locale d'ILOG TPO, pour en particulier avoir une meilleure approximation des coûts dans le MIP. Une solution du MIP pourrait également être exportée (quand ceci est possible) dans un format ILOG TPO pour obtenir son coût réel. Toutes ces idées de collaboration sont intéressantes mais plutôt dures à implémenter, car une partie du travail doit se porter sur le moteur de base d'ILOG TPO également. De toute façon, le moteur de recherche locale doit certainement être réadapté à la coopération avec le MIP puisque dans certains cas, les meilleures indications ne conduisent pas nécessairement ILOG TPO à trouver la meilleure solution (avec les mêmes chemins d'ordres de transport que dans les indications).

Enfin, nous pouvons travailler sur les données du 3PL. Dans les instances que nous avons reçues de leur part, il existe toujours un seul hub disponible, bien qu'il y ait généralement quatre potentiels hubs. Cela peut être un prétraitement de leur côté qui élimine tous les hubs qui ne sont aps susceptibles d'être utilisés dans une bonne solution. Mais nos outils sont adaptés aux réseaux avec plusieurs hubs, et les 3PL seraient peut-être intéressés qu'on leur suggère de ne pas ignorer de hub et de laisser nos outils décider desquels il serait judicieux d'ouvrir.

Bibliography

[Ahuja et al., 1993] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall. 48

[Al-Khayyal and Hwang, 2007] Al-Khayyal, F. and Hwang, S.-J. (2007). Inventory Constrained Maritime Routing and Scheduling for Multi-Commodity Liquid Bulk, Part I: Applications and Model. *European Journal of Operational Research*, 176(1):106–130. 32

[Alamur and Kara, 2008] Alamur, S. and Kara, B. Y. (2008). Network Hub Location Problems : The State of the Art. *European Journal of Operational Research*, 190(1):1–21. 51

[Armacost et al., 2002] Armacost, A. P., Barnhart, C., and Ware, K. A. (2002). Composite Variable Formulations for Express Shipment Service Network Design. *Transportation Science*, 36(1). 36, 43

[Armacost et al., 2004] Armacost, A. P., Barnhart, C., Ware, K. A., and Wilson, A. M. (2004). UPS Optimizes Its Air Network. *Interfaces*, 34(1):15–25. 36, 43

[Assad, 1978] Assad, A. A. (1978). Multicommodity Network Flow - A Survey. *Networks*, 8(1):37–91. 51

[Augerat, 1995] Augerat, P. (1995). *Approche Polyèdrale du Problème de Tournées de Véhicules*. PhD thesis, Institut National Polytechnique de Grenoble, France. 40

[Augerat et al., 1995] Augerat, P., Belenguer, J.-M., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G. (1995). Computational Results With a Branch and Bound Code for the Capacitated Vehicle Routing Problem. Technical Report RR 949-M, Université Joseph Fourier, Grenoble, France. 40

[Aykin, 1994] Aykin, T. (1994). Lagrangean Relaxation Based Approaches to Capacitated Hub-and-Spoke Network Design Problem. *European Journal of Operational Research*, 79(3):501–523. 54

[Baldacci et al., 2004] Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A. (2004). An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation. *Operations Research*, 52(5):723–738. 39

[Baldacci et al., 2007] Baldacci, R., Toth, P., and Vigo, D. (2007). Recent Advances in Vehicle Routing Exact Algorithms. *4OR: A Quarterly Journal of Operations Research*, 5(4):269–298. 39, 40

[Balinski and Quandt, 1964] Balinski, M. and Quandt, R. (1964). On an Integer Program for a Delivery Problem. *Operations Research*, 12:300–304. 38

[Bartholdi and Gue, 2002] Bartholdi, J. J. and Gue, K. R. (2002). Reducing Labor Costs in an LTL Crossdocking Terminal. *Operations Research*, 48(6):823–832. 14

[Bartholdi and Hackman, 2008] Bartholdi, J. J. and Hackman, S. T. (2008). *Warehouse & Distribution Science*. 14

[Bellman, 1958] Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90. 50

[Bonnans et al., 2000] Bonnans, J. F., Haddou, M., Lisser, A., and Rébaï, R. (2000). Interior Point Methods with Decomposition for Multicommodity Flow Problems. Rapport de recherche INRIA-RR - 3852, INRIA, Unité de recherche de Rocquencourt, France. 51

[Bramel and Simchi-Levi, 1995] Bramel, J. and Simchi-Levi, D. (1995). A Location Based Heuristic for General Routing Problems. *Operations Research*, 43:649–660. 42

[Bramel and Simchi-Levi, 2002] Bramel, J. and Simchi-Levi, D. (2002). Set-covering-based algorithms for the capacitated VRP. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 85–108. SIAM. 39

[Campbell, 1994] Campbell, J. F. (1994). Integer Programming Formulations of Discrete Hub Location Problems. *European Journal of Operational Research*, 72(2):387–405. 53, 54

[Cánovas et al., 2007] Cánovas, L., García, S., and Marín, A. (2007). Solving the uncapacitated mutliple allocation hub location problem by means of a dual-ascent technique. *European Journal of Operational Research*, 179:990–1007. 54

[Carrabs et al., 2007] Carrabs, F., Cordeau, J.-F., and Laporte, G. (2007). Variable Neighborhood Search for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading. *INFORMS Journal on Computing*, 19(4):618–632. 34

[Chabrier, 2006] Chabrier, A. (2006). Vehicle Routing Problem with Elementary Shortest Path Based Column Generation. *Computers and Operations Research*, 33(10):2972–2990. 40

[Chang and Tsui, 1992] Chang, C.-H. and Tsui, L. Y. (1992). Optimal Solution to a Dock Door Assignment Problem. *Computers and Industrial Engineering*, 23:283–286. 14

[Choi and Tcha, 2007] Choi, E. and Tcha, D.-W. (2007). A Column Generation Approach to the Heterogeneous Fleet Vehicle Routing Problem. *Computers and Operations Research*, 34(7):2080–2095. 41

[Clarke and Wright, 1964] Clarke, G. and Wright, J. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12:568–581. 42

[Clochard and Naddef, 1993] Clochard, J.-M. and Naddef, D. (1993). Use of Path Inequalities for TSP. In Rinaldi, G. and Wolsey, L., editors, *Proceedings of the Third Workshop on Integer Programming and Combinatorial Optimization*, pages 291–312. CORE. 40

[Cordeau and Laporte, 2003] Cordeau, J.-F. and Laporte, G. (2003). The Dial-a-Ride Problem (DARP): Variants, Modeling Issues and Algorithms. *4OR: A Quarterly Journal of Operations Research*, 1(2):89–101. 32

[Crainic, 1999] Crainic, T. G. (1999). Long-Haul Freight Transportation. In Hall, R., editor, *Handbook of Transportation Science*, pages 433–491. Kluwer Academic Publishers. 12

[Crainic and Semet, 2005] Crainic, T. G. and Semet, F. (2005). Recherche Opérationnelle et transport de marchandises. In Paschos, V. T., editor, *Optimisation combinatoire : applications*, pages 47–115. Hermès Lavoisier, Paris. 12

[Croxton et al., 2007] Croxton, K. L., Gendron, B., and Magnanti, T. L. (2007). Variable Disaggregation in Network Flow Problems with Piecewise Linear Costs. *Operations Research*, 55(1):146–157. 50

[Cunha and Silva, 2007] Cunha, C. B. and Silva, M. R. (2007). A Genetic Algorithm for the Problem of Configuring a Hub-and-Spoke Network for a LTL Trucking Company in Brazil. *European Journal of Operational Research*, 179(1):747–758. 54

[Czarnas et al., 2004] Czarnas, P., Czech, Z. J., and Gocyla, P. (2004). Parallel Simulated Annealing for Bricriterion Optimization Problems. In Heidelberg, S. B. ., editor, *Lecture Notes in Computer Science: Parallel Processing and Applied Mathematics*, pages 233–240. 30

[Dantzig and Ramser, 1959] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91. 27

[De Backer et al., 2000] De Backer, B., Furnon, V., Shaw, P., Kilby, P., and Prosser, P. (2000). Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics. *Journal of Heuristics*, 6(4):501–523. 101

[De Souza et al., 2008] De Souza, M. C., Mahey, P., and Gendron, B. (2008). Cycle-Based Algorithms for Multicommodity Network Flow Problems with Separable Piecewise Convex Costs. *Networks*, 51(2):133–141. 51

[Dell'Amico et al., 1993] Dell'Amico, M., Fischetti, M., and Toth, P. (1993). Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem. *Management Science*, 39(1):115–125. 30, 32

[Desaulniers et al., 2002] Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., and Soumis, F. (2002). VRP with Pickup and Delivery. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 225–242. SIAM. 31, 35, 38, 61

[Doerner et al., 2000] Doerner, K., Gronalt, M., Hartl, R. F., and Reimann, M. (2000). Time Constrained Full Truckload Transportation: Optimizing Fleet Size and Vehicle Movements. Working Paper POM 07/2000, Department of Production and Operations Management, University of Vienna, Austria. 43, 47

[Dumas et al., 1991] Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 54(1):7–22. 31

[Finke et al., 1984] Finke, G., Claus, A., and Gunn, E. (1984). A two-commodity network flow approach to the traveling salesman problem. *Congress Numer*, 41:167–178. 39

[Fisher and Jaikumar, 1981] Fisher, M. and Jaikumar, R. (1981). A Generalized Assignment Heuristic for the Vehicle Routing Problem. *Networks*, 11:109–124. 42

[Ford Jr. and Fulkerson, 1962] Ford Jr., L. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press. 50

[Fournier, 2007] Fournier, S. (2007). Modèles Mathématiques pour des Problèmes de Tournées de Véhicules avec Transbordement et Ramassage et Livraison Multi-Produits. In *Livre des Articles de la Conférence Conjointe Francoro V / ROADEF 2007*, pages 117–128. Presses universitaires de Grenoble. 57, 77

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Bell Telephone Laboratories, Incorporated. 28

[Golden et al., 2002] Golden, B. L., Assad, A. A., and Wasil, E. A. (2002). Routing Vehicles in the Real World: Applications. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 245–286. SIAM. 30

[Golden et al., 1977a] Golden, B. L., Magnanti, T. L., and Nguyen, H. (1977a). Implementing Vehicle Routing Algorithms. *Networks*, 7:113–148. 38

[Golden et al., 1977b] Golden, B. L., Magnanti, T. L., and Nguyen, H. Q. (1977b). A Heuristic Algorithm for the Vehicle Dispatch Problem. *Operations Research*, 22:340–349. 42

[Grünert et al., 1999] Grünert, T., Sebastian, H.-J., and Thärigen, M. (1999). The Design of a Letter-Mail Transportation Network by Intelligent Techniques. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, page 6030. IEEE Computer Society. 43, 46

[Holmberg and Yuan, 2003] Holmberg, K. and Yuan, D. (2003). A Multicommodity Network-Flow with Side Constraints on Paths Solved by Column Generation. *INFORMS Journal on Computing*, 15(1):42–57. 50

[ILOG TPO, 2007] ILOG TPO (2007). *ILOG Transport PowerOps 3.1 Manuals*. ILOG S.A. 17

[Iori et al., 2007] Iori, M., Salazar-González, J.-J., and Vigo, D. (2007). An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints. *Transportation Science*, 41(2):253–264. 34

[Irnich, 2000] Irnich, S. (2000). A Multi-Depot Pickup and Delivery Problem with a Single Hub and Heterogeneous Vehicles. *European Journal of Operational Research*, 122(2):310–328. 32, 36, 43

[Jung and Haghani, 2000] Jung, S. and Haghani, A. (2000). Genetic Algorithm for a Pickup and Delivery Problem with Time Windows. *Transportation Research Record*, 1733:1–7. 31

[Kara, 1999] Kara, B. (1999). *Modeling and analysis of issues in hub location problems*. PhD thesis, Bilkent University Industrial Engineering Department, Ankara, Turkey. 53

[Karp, 2003] Karp, R. M. (2003). Reducibility Among Combinatorial Problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computers Computations*, pages 85–103. New York: Plenum. 79

[Kelly and Xu, 1999] Kelly, J. P. and Xu, J. (1999). A Set-Partitioning-Based Heuristic for the Vehicle Routing Problem. *INFORMS Journal on Computing*, 11(2):161–172. 46

[Kennington, 1978] Kennington, J. L. (1978). A Survey of Linear Cost Multicommodity Network Flows. *Operations Research*, 26(2):209–236. 51

[Lapierre et al., 2004] Lapierre, S. D., Ruiz, A. B., and Soriano, P. (2004). Designing distribution networks. *Transportation Science*, 38(2):174–187. 36

[Laporte and Semet, 2002] Laporte, G. and Semet, F. (2002). Classical Heuristics for the Capacitated VRP. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 109–128. SIAM. 41

[Larsson and Yuan, 2004] Larsson, T. and Yuan, D. (2004). An Augmented Lagrangian Algorithm for Large Scale Multicommodity Routing. *Computational Optimization and Applications*, 27(2):187–215. 51

[Li et al., 2004] Li, Y., Lim, A., and Rodrigues, B. (2004). Crossdocking - JIT scheduling with time windows. *Journal of Operational Research Society*, 55(12):1342–1351. 14

[Lin, 1965] Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269. 44

[Mahey and De Souza, 2007] Mahey, P. and De Souza, M. C. (2007). Local Optimality Conditions for Multicommodity Flow Problems with Separable Piecewise Convex Costs. *Operations Research Letters*, 35:221–226. 50

[Martelli, 1976] Martelli, A. (1976). An application of heuristic search methods to edge and contour detection. *Communications of the ACM*, 19(2):73–83. 41

[Michel, 2006] Michel, S. (2006). *Inventory Routing Problem*. PhD thesis, Université Bordeaux 1, France. 31

[Mitchell, 2002] Mitchell, J. E. (2002). Branch-and-Cut Algorithms for Combinatorial Optimization Problems. In Pardalos, P. M. and Resende, M. G. C., editors, *Handbook of Applied Optimization*, pages 65–77. Oxford University Press. 39

[Mitrović-Minić, 1998] Mitrović-Minić, S. (1998). Pickup and Delivery Problem with Time Windows: A Survey. Technical Report CMPT1998-12, Simon Fraser University, Canada. 31, 37

[Mues and Pickl, 2005] Mues, C. and Pickl, S. (2005). Transshipment and Time Windows in Vehicle Routing. In *ISPAN'05: Proceedings of the 8th*

International Symposium on Parallel Architectures, Algorithms and Networks, pages 113–119. IEEE Computer Society. 36

[Naddef and Rinaldi, 2002] Naddef, D. and Rinaldi, G. (2002). Branch-and-Cut Algorithms for the Capacitated VRP. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, pages 53–84. SIAM. 40, 89

[Nagy and Salhi, 2005] Nagy, G. and Salhi, S. (2005). Heuristic Algorithms for Single and Multiple Depot Vehicle Routing Problems with Pickups and Deliveries. *European Journal of Operational Research*, 162(1):126–141. 35

[Nowak et al., 2007] Nowak, M., Ergun, O., and White, C. C. (2007). Pickup and Delivery with Split Loads. pages 53–84. 34

[O'Kelly, 1992] O'Kelly, M. (1992). Hub facility location with fixed costs. *The Journal of the Regional Science Association International*, 71:293–306. 52

[Ombuki et al., 2006] Ombuki, B., Ross, B. J., and Hanshar, F. (2006). Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows. *Applied Intelligence*, 24(1):17–30. 30

[Or, 1976] Or, I. (1976). *Traveling Salesman-Type Combinatorial Optimization Problems and their Relation to the Logistics of Regional Blood Banking*. PhD thesis, Departement of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. 44

[Poot et al., 1999] Poot, A., Kant, G., and Wagelmans, A. P. M. (1999). A Savings Based Method for Real-Life Vehicle Routing Problem. Technical Report EI 9938/A, Erasmus University Rotterdam. 30, 34

[Psaraftis, 1980] Psaraftis, H. N. (1980). A Dynamic Programming Solution to Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, 14(2):130–154. 41

[Racunica and Wynter, 2000] Racunica, I. and Wynter, L. (2000). Optimal Location of Intermodal Freight Hubs. Rapport de recherche INRIA-RR - 4088, INRIA, Unité de recherche de Rocquencourt, France. 53

[Savelsbergh, 1988] Savelsbergh, M. W. P. (1988). *Computer Aided Routing*. PhD thesis, Centrum voor Wiskunde en Infomatica, Amsterdam, Netherlands. 27

[Savelsbergh and Sol, 1995] Savelsbergh, M. W. P. and Sol, M. (1995). The General Pickup and Delivery Problem. *Transportation Science*, 29(1):17–29. 29, 35

[Schrijver, 2003] Schrijver, A. (2003). *Combinatorial Optimization: Polyedra and Efficiency*. Springer. 49

[Shen et al., 1995] Shen, Y., Potvin, J.-Y., Rousseau, J.-M., and Roy, S. (1995). A Computer Assistant for Vehicle Dispatching with Learning Capabilities. *Annals of Operations Research*, 61(1):189–211.

[Sherali and Shetty, 1980] Sherali, H. D. and Shetty, C. (1980). *Optimization with Disjunctive Constraints*, volume 181. Springer. 135, 139

[Sol and Savelsbergh, 1994] Sol, M. and Savelsbergh, M. W. P. (1994). A Branch-and-Price Algorithm for the Pickup and Delivery Problem with Time Windows. Report COC-94-06, Georgia Institute of Technology, Atlanta, Georgia, USA. 41

[Tavakkoli-Moghaddam et al., 2007] Tavakkoli-Moghaddam, R., Safaei, N., Kah, M., and Rabbani, M. (2007). A New Capacitated Vehicle Routing Problem with Split Service for Minimizing Fleet Cost by Simulated Annealing. *Journal of the Franklin Institute*, 344:406–425. 46

[Thangiah, 1995] Thangiah, S. R. (1995). Vehicle Routing with Time Windows Using Genetic Algorithms. In Chambers, L., editor, *Applications Handbook of Genetic Algorithms: New Frontiers*, volume 2, pages 253–277. CRC Press. 31

[Thangiah and Nygard, 1992] Thangiah, S. R. and Nygard, K. E. (1992). MICAH: A Genetic Algorithm System for Multi-Commodity Transshipment Problems. In *Proceedings of the Eighth Conference on Artificial Intelligence for Applications*, pages 240–246, Monterey, CA. 46

[Thompson and Psaraftis, 1993] Thompson, P. M. and Psaraftis, H. N. (1993). Cyclic Transfer Algorithms for Multi-Vehicle Routing and Scheduling Problems. *Operations Research*, 41:935–946. 44

[Topcuoglu et al., 2005] Topcuoglu, H., Corut, F., Ermis, M., and Yilmaz, G. (2005). Solving the Uncapacitated Hub Location Problem using Genetic Algorithms. *Computers and Operations Research*, 32(4):967–984. 54

[Toth and Vigo, 1997] Toth, P. and Vigo, D. (1997). Heuristic Algorithms for the Handicapped Persons Transportation Problem. *Transportation Science*, 31(1):60–71. 32

[Tu et al., 2005] Tu, M.-Y., Tsai, F. T.-C., and Yeh, W. W.-G. (2005). Optimization of Water Distribution and Water Quality by Hybrid Genetic Algorithm. *Journal of water resources planning and management*, 131(6):431–440. 51

[Van Breedam, 1994] Van Breedam, A. (1994). *An Analysis of the Behaviour of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-Related, Customer-Related and Time-Related Constraints*. PhD thesis, University of Antwerp, Belgium. 44

[Van Der Bruggen et al., 1993] Van Der Bruggen, L., Lenstra, J., and Schuur, P. (1993). Variable-Depth Search for the Single-Vehicle Pickup and Delivery Problem with Time Windows. *Transportation Science*, 27(3):298–311. 33

[Walkowiak, 2004] Walkowiak, K. (2004). On Application of Ant Algorithms to Non-Bifurcated Multicommodity Flow Problem. In Heidelberg, S. B. ., editor, *Lecture Notes in Computer Science: Artificial Intelligence and Soft Computing*, pages 922–927. 51

[Williams, 1978] Williams, H. P. (1978). *Model Building in Mathematical Programming*. John Wiley & Sons, New York, first edition. 64

[Xu et al., 2003] Xu, H., Chen, Z.-L., Rajagopal, S., and Arunapuram, S. (2003). Solving a Practical Pickup and Delivery Problem. *Transportation Science*, 37(3):347–364. 31, 34, 41

[Zhu et al., 2006] Zhu, Q., Qian, L., Li, Y., and Zhu, S. (2006). An Improved Particle Swarm Optimization Algorithm for Vehicle Routing Problem with Time Windows. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1386–1390. IEEE. 47

Index