On Some Generalized Routing Problems

Von der Fakultät für Wirtschaftswissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines
Doktors der Wirtschafts- und Sozialwissenschaften
genehmigte Dissertation

vorgelegt von

Dipl.-Kfm. Michael Drexl, M.O.R.

aus Schwabmünchen

Berichter: Univ.-Prof. Dr.rer.nat.habil. Hans-Jürgen Sebastian
Univ.-Prof. Dr.rer.pol.habil. Michael Bastian

Tag der mündlichen Prüfung: 31. Oktober 2007

Contents

List of Symbols and Abbreviations

$=$	equality operator
$\neq$	inequality operator
$\leqq$	less-than-or-equal operator
$\geqq$	greater-than-or-equal operator
$:=$	assignment operator in definitions and algorithms
$\Rightarrow$	implies
$\wedge$	logical and
$\vee$	logical inclusive or
$\exists$	there exists a(n)
$\forall$	for all
$a \in A$; $A \ni a$	a is an element of A
$a \notin A$; $A \not\ni a$	a is not an element of A
$\{a : \ldots\}$	set of all a such that ...
$A \subsetneqq B$	A is a proper subset of B
$A \subseteqq B$	A is a proper or improper subset of B
$A \cup B$	disjoint union of A and B
$A \cup B$	(not necessarily disjoint) union of A and B
$\bigcup_{i=1,\ldots,n} A_i$	(not necessarily disjoint) union of A_i, $i = 1, \ldots, n$
$\bigcup_{i \in I} A_i$	(not necessarily disjoint) union of A_i, where index i is an element of the index set I
$A \cap B$	intersection of A and B
$\bigcap_{i=1,\ldots,n} A_i$	intersection of A_i, $i = 1, \ldots, n$
$\bigcap_{i \in I} A_i$	intersection of A_i, where index i is an element of the index set I
$A \setminus B$	$\{a \in A : a \notin B\}$
$\mathscr{P}(X)$	power set of X
$A \times B$	Cartesian product of A and B
$\lvert A \rvert$	cardinality of A; number of elements in A
$\emptyset$	empty set
$\mathbb{N}$	set of natural numbers
$\mathbb{N}^0$	$\mathbb{N} \cup \{0\}$
$\mathbb{Z}$	set of integer numbers
$\mathbb{Z}_+$	$\{z \in \mathbb{Z} : z > 0\}$
$\mathbb{Z}_-$	$\{z \in \mathbb{Z} : z < 0\}$
$\mathbb{Z}_+^0$	$\{z \in \mathbb{Z} : z \geqq 0\}$

$\mathbb{Z}_-^0$	$\{z \in \mathbb{Z} : z \leqq 0\}$
$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	$\{r \in \mathbb{R} : r > 0\}$
$\mathbb{R}_-$	$\{r \in \mathbb{R} : r < 0\}$
$\mathbb{R}_+^0$	$\{r \in \mathbb{R} : r \geqq 0\}$
$\mathbb{R}_-^0$	$\{r \in \mathbb{R} : r \leqq 0\}$
∞	infinity
M	arbitrarily, 'sufficiently' large positive number
$\sum_{i=1}^{n} x_i$	sum of values of x_i, $i = 1, \ldots, n$
$\sum_{i \in I} x_i$	sum of values of x_i, where index i is an element of the index set I
$\min_{i=1,\ldots,n}\{x_i\}$	minimum of values of x_i, $i = 1, \ldots, n$
$\min_{i \in I}\{x_i\}$	minimum of values of x_i, where index i is an element of the index set I
$\max_{i=1,\ldots,n}\{x_i\}$	maximum of values of x_i, $i = 1, \ldots, n$
$\max_{i \in I}\{x_i\}$	maximum of values of x_i, where index i is an element of the index set I
$\bigvee_{i \in I} c_i$	disjunction of clauses c_i, where index i is an element of the index set I
$f : D \to R$	function f with domain D and range R
$f(x)$; f_x	image of a function $f : D \to R$ for $x \in D$
$E(X)$	expected value of random variable X
$\mathcal{O}$	Landau symbol; order of worst-case time or memory complexity
w.l.o.g.	without loss of generality
ARP	arc routing problem
ATSP	asymmetric travelling salesman problem
BGL	Boost Graph library
CPP	Chinese postman problem
DCPP	directed Chinese postman problem
DRPP	directed rural postman problem
ESPPRC	elementary shortest path problem with resource constraints
ESPPTW	elementary shortest path problem with time windows
GATSP	generalized asymmetric travelling salesman problem
GDRPP	generalized directed rural postman problem
GP	generic programming
GRP	general routing problem
GTSP	generalized travelling salesman problem
IP	integer program(ming)
LB	lower bound
LP	linear program(ming)
LRP	location-routing problem
LTC	lorry-trailer combination
MIP	mixed integer program(ming)
MRPP	mixed rural postman problem
OOP	object-oriented programming
REF	resource extension function

RPP	rural postman problem
RTSP	road travelling salesman problem
SPPRC	shortest path problem with resource constraints
SPPTW	shortest path problem with time windows
STL	C++ Standard Template Library
STSP	symmetric travelling salesman problem
TSP	travelling salesman problem
TSPTW	travelling salesman problem with time windows
TTRP	truck-and-trailer routing problem
UB	upper bound
UCPP	undirected Chinese postman problem
URPP	undirected rural postman problem
VRP	vehicle routing problem
VRPTT	vehicle routing problem with trailers and transshipments
VRPTW	vehicle routing problem with time windows
WCPP	windy Chinese postman problem
WGRP	windy general routing problem
WRPP	windy rural postman problem
WRPPZZ	windy rural postman problem with zigzag service

Zusammenfassung

Die Arbeit behandelt Anwendungen der kombinatorischen Optimierung in Logistik und Transport. Sie betrachtet einige mathematische Optimierungsprobleme aus der Perspektive des Operations Research. Die Arbeit beschreibt die zahlreichen Anwendungen dieser Probleme in der ökonomischen Realität, erläutert, wie die Probleme mathematisch modelliert werden können, schlägt Algorithmen zu ihrer Lösung vor und präsentiert die Ergebnisse umfangreicher Rechenexperimente mit Implementierungen der vorgeschlagenen Algorithmen.

Routingprobleme betrachten Objekte, welche sich auf oder über oder entlang anderer Objekte bewegen können, die also raumzeitlichen Transformationen unterworfen werden können und die selbst in der Lage sind, solche Transformationen an anderen Objekten vorzunehmen, um bestimmte vorgegebene Aufgaben zur Erreichung gewisser Ziele zu erfüllen. Dies kann nicht in beliebiger Art und Weise geschehen, sondern nur unter Beachtung ebenfalls vorgegebener Rahmenbedingungen. Das Problem besteht darin, die hinsichtlich eines oder mehrerer Ziele bestmögliche Art und Weise der Bewegungen bzw. der raumzeitlichen Transformationen unter Beachtung der Rahmenbedingungen zu finden. Routingprobleme stellen mathematische Optimierungsprobleme (Rechenaufgaben) dar, d.h. sie bestehen aus Entscheidungsvariablen, einer (nicht notwendig skalaren) Zielfunktion und Nebenbedingungen. Routingprobleme werden auf bewerteten Graphen (Netzwerken) modelliert bzw. formuliert. Die Lösungen von Routingproblemen bestehen ganz oder teilweise aus Kanten- oder Bogenfolgen in den zugrundeliegenden Netzwerken. Einige bekannte Routingprobleme sind das Kürzeste-Wege-Problem (shortest path problem, SPP), das Problem des Handlungsreisenden (travelling salesman problem, TSP), das Tourenplanungsproblem mit Zeitfenstern (vehicle routing problem with time windows, VRPTW) und das Chinesische Briefträgerproblem (Chinese postman problem, CPP).

In der Arbeit werden die folgenden drei verallgemeinerten Routingprobleme untersucht:

(i) das verallgemeinerte gerichtete Rural-Postman-Problem (generalized directed rural postman problem, GDRPP)
Das GDRPP ist, wie der Name schon sagt, eine Verallgemeinerung des Rural-Postman-Problems auf gerichteten Graphen (DRPP). Das DRPP besteht darin, eine optimale Briefträgertour in einem gerichteten Graphen zu finden, d.h. einen kostenminimalen Zyklus, der jeden Bogen einer gegebenen Teilmenge der Bogenmenge des Digraphen mindestens einmal durchläuft. Im GDRPP sind mehrere Klassen (Gruppen, Cluster) von Bögen gegeben, und die Aufgabe besteht darin, einen kostenminimalen Zyklus zu finden, der mindestens einen Bogen jeder Klasse mindestens ein Mal durchläuft.

(ii) das Tourenplanungsproblem mit Anhängern und Ladungstransfers (vehicle routing problem with trailers and transshipments, VRPTT)
Das VRPTT ist eine Verallgemeinerung des Tourenplanungsproblems mit Zeitfenstern, bei dem die Fahrzeugflotte aus autonomen und nicht-autonomen Fahrzeugen besteht (z.B. aus LKW und Anhängern). Erstere können sich selbständig bewegen, letztere müssen von einem autonomen Fahrzeug begleitet werden, um sich bewegen zu können. Darüber hinaus sind die Fahrzeuge entweder Sammel- oder Unterstützungsfahrzeuge. Sammelfahrzeuge werden benutzt, um Kunden zu besuchen und deren Vorräte abzuholen; Unterstützungsfahrzeuge werden von den Sammelfahrzeugen als mobile Depots benutzt.

(iii) das Tourenplanungsproblem mit Anhängern (truck-and-trailer routing problem, TTRP)
Das TTRP stellt einen Spezialfall des VRPTT dar, bei dem eine feste Zuordnung jedes nicht-autonomen Fahrzeugs zu einem autonomen Fahrzeug existiert. Dies bedeutet, daß keine Unterstützungsfahrzeuge zum Einsatz kommen.

Zwei dieser Probleme, das GDRPP und das VRPTT, sind neu in dem Sinne, daß sie noch nicht in der Literatur beschrieben wurden, d.h., es existieren weder Formulierungen noch exakte oder heuristische Lösungsansätze.

Die Probleme sind in folgender Hinsicht *verallgemeinert*: Routingprobleme wie das TSP, das VRP und das CPP sind in erster Linie *Reihenfolgeprobleme*. Die Schwierigkeit bei diesen Problemen besteht darin, eine (optimale) Reihenfolge von Objekten zu bestimmen. Das GDRPP, das VRPTT und das TTRP enthalten einen zusätzlichen Freiheitsgrad: Es muß eine *Auswahl* getroffen werden hinsichtlich derjenigen Objekte, welche die gestellten Aufgaben ausführen und/oder hinsichtlich derjenigen Objekte, die in eine Reihenfolge zu bringen sind.

Die Arbeit soll einen Beitrag auf dem Gebiet des Operations Research darstellen. Operations Research ist ein interdisziplinärer Wissenszweig an der Schnittstelle von Wirtschaftswissenschaften, Mathematik und Informatik. Daher versucht die Arbeit, einen Beitrag in diesen drei Gebieten zu leisten. Sie besitzt

(i) einen problem- und anwendungsorientierten Aspekt (Wirtschaftswissenschaften)
Der Anwendungsaspekt besteht dabei nicht aus der Lösung echter Probleminstanzen aus der Praxis oder der Durchführung von Fallstudien, sondern die praktische Bedeutung der behandelten Probleme wird herausgestellt und durch Anwendungsbeispiele belegt.

(ii) einen modell- und methodenorientierten Aspekt (Mathematik)
Der mathematische Aspekt besitzt einen modell- und einen methodenorientierten Teil. Der modellorientierte Teil besteht aus Formulierungen der (gemischt-)ganzzahligen Programmierung, welche für die drei betrachteten Probleme entwickelt werden. Der methodenorientierte Teil besteht in der algorithmischen Behandlung der Probleme. Die Probleme werden mit Branch-and-Cut-Verfahren gelöst und ganz oder teilweise auch mit auf dynamischer Programmierung basierenden Markierungsalgorithmen, die auf dem Ressourcenkonzept beruhen. Zusätzlich werden für das GDRPP verschiedene Heuristiken vorgeschlagen und für das TTRP wird ein Branch-and-Price-Algorithmus vorgestellt.

(iii) einen implementationsorientierten Aspekt (Informatik)
Der implementationsorientierte Aspekt besteht aus den durchgeführten Rechenexperimenten und einem Abschnitt, in dem ein allgemein verwendbarer Computer-Code zur Lösung des Kürzeste-Wege-Problems mit Ressourcenbeschränkungen vorgestellt wird. Der Code wurde im Rahmen der Arbeit entwickelt und wird in den Rechenexperimenten benutzt.

Abstract

The paper examines applications of combinatorial optimization in logistics and transport, and considers some mathematical optimization problems from the perspective of Operational Research (OR). It also lists some of the numerous applications of these problems in economic reality, describes how such problems can be mathematically modelled, proposes algorithms for their solution, and presents the results of extensive computational experiments with implementations of the proposed algorithms.

Routing problems consider (primary) objects which are able to move on or over or along other (secondary) objects. That is, the primary objects can be subjected to spatial-temporal transformations, and they are themselves able to perform such transformations on the secondary objects in order to fulfill given tasks to achieve certain objectives. This may not be done in an arbitrary manner, but only in compliance with given general conditions, within a general framework or set-up. The problem consists in finding the best possible manner of movements, respectively, of spatial-temporal transformations of the primary objects, with regard to the given objectives, while respecting the general conditions. Routing problems are mathematical optimization problems (arithmetic problems), i.e., they consist of decision variables, a (not necessarily scalar) objective function, and side constraints. Routing problems are modelled and formulated on weighted graphs (networks). The solutions to routing problems consist completely or partly of undirected or directed walks in the underlying networks. Some well-known routing problems are the shortest path problem (SPP), the travelling salesman problem (TSP), the vehicle routing problem with time windows (VRPTW), and the Chinese postman problem (CPP).

The paper considers the following three generalized routing problems:

(i) the generalized directed rural postman problem (GDRPP) (which generalizes the CPP)
The GDRPP is a straightforward generalization of the directed rural postman problem (DRPP). The DRPP consists in finding an optimal postman tour in a digraph, i.e., a least-cost cycle traversing each arc of a specified subset of the digraph's arcs at least once. The GDRPP is to the DRPP what the generalized travelling salesman problem (GTSP) is to the travelling salesman problem (TSP): In the GDRPP, there are several subsets (groups, classes, clusters) of arcs and the requirement is to find a least-cost cycle traversing at least one arc from each subset at least once.

(ii) the vehicle routing problem with trailers and transshipments (VRPTT) (which generalizes the VRPTW)
In the VRPTT, the vehicle fleet consists of autonomous vehicles able to move on their own, and of non-autonomous vehicles which must be accompanied by an autonomous vehicle to be able to move (e.g., lorries and trailers). Moreover, both autonomous and non-autonomous vehicles are either collection or support vehicles. Collection vehicles are used to pick up the supplies of the customers. Support vehicles are used as mobile depots by the collection vehicles.

(iii) the truck-and-trailer routing problem (TTRP) (also a generalization of the VRPTW and a special case of the VRPTT)
In the TTRP, there is a fixed assignment of each non-autonomous vehicle to an autonomous vehicle. This means that no support vehicles are used.

Two of these problems, the GDRPP and the VRPTT, are new in the sense that they have not yet been described in the literature, i.e., there are neither formulations nor heuristic or exact solution procedures for them.

The problems are *generalized* in the following sense. Routing problems like the TSP, the VRPTW, and the CPP are essentially *sequencing* problems. The difficulty in these problems consists in determining an optimal sequence of objects. The GDRPP, the VRPTT, and the TTRP involve an additional degree of freedom: A *selection* must be made as to which objects to sequence (which primary objects to 'use' and/or which secondary objects to 'visit' or 'service') in order to fulfill the given tasks.

The paper is intended to be a contribution to the field of OR. OR is an interdisciplinary field which lies at the interface of business administration, mathematics, and computer science. Hence, the paper strives to make a three-fold contribution. It covers

(i) the problem- and application-oriented aspect (business administration)
The application aspect does not consist in the solution of real-world problem instances or the presentation of case studies, but the practical relevance of the three problems dealt with is emphasized by the description of actual and potential areas of application and references to pertinent literature.

(ii) the model- and method-oriented aspect (mathematics)
The mathematical aspect has a model-oriented and a method-oriented part. The model-oriented part consists in the (mixed) integer programming formulations developed for the three problems under consideration. The method-oriented part consists in the algorithmic treatment of the three problems. All problems are solved by branch-and-cut, and the problems are also solved partially or completely by dynamic-programming based labelling algorithms using the resource concept. Additionally, several heuristics are developed for the GDRPP, and for the TTRP, a branch-and-price algorithm is proposed.

(iii) the implementation-oriented aspect (computer science)
The computer science aspect is covered by sections on computational experiments and a section where the design and the implementation of a generally usable computer code for solving the shortest path problem with resource constraints are described. The code was developed as part of the paper and is used in the computational experiments.

Chapter 1

Introduction

This paper is concerned with applications of combinatorial optimization in logistics and transport, and considers some mathematical optimization problems from the perspective of Operational Research (OR). It lists some of the numerous applications of these problems in economic reality, describes how such problems can be mathematically modelled, proposes algorithms for their solution, and presents the results of extensive computational experiments with implementations of the proposed algorithms.

1.1 Subject Matter

Routing problems consider (primary) objects which are able to move on or over or along other (secondary) objects. That is, the primary objects can be subjected to spatial-temporal transformations, and they are themselves able to perform such transformations on the secondary objects in order to fulfil given tasks to achieve certain objectives. This may not be done in an arbitrary manner, but only in compliance with given general conditions, within a general framework or set-up. The problems consist in finding the best possible manner of movements, respectively, of spatial-temporal transformations of the primary objects, with regard to the given objectives, while respecting the general conditions. Routing problems are mathematical optimization problems (arithmetic problems), i.e., they consist of decision variables, a (not necessarily scalar) objective function, and side constraints. Routing problems are represented on weighted graphs (networks). The solutions to routing problems consist completely or partly of undirected or directed walks in the underlying networks.

Examples of routing problems abound. The most common routing problem is the shortest path problem (SPP). This is what an internet router faces when sending packets of data (primary objects), received from a host computer, to a client machine via a minimal number of 'hops' (objective) over other internet routers (secondary objects) in order to satisfy the client's request for the information contained in such a packet (task). *The* routing problem per se is the famous travelling salesman problem (TSP), where the salesman is the primary object that is moving in space and time through a road network (roads: secondary objects) in order to visit customers (secondary objects, too) and fulfil their visiting requests (the salesman's task) while not wasting time or fuel en route (the salesman's objective). Another example is the vehicle routing problem with time windows (VRPTW), where several capacity-constrained vehicles (primary objects) located at a depot visit geographically dispersed customers (secondary objects) during the customers' working hours (conditions) via a road network in order to collect supplies (task) while not driving more kilometres than necessary (objective). Yet another example is the Chinese postman problem (CPP), where a postman (primary object) traverses the streets in his district (secondary objects) in order to distribute mail (task) while keeping his overall walking distance at a minimum (objective).

This paper considers the following three generalized routing problems:

(i) the generalized directed rural postman problem (GDRPP) (which generalizes the CPP)

(ii) the vehicle routing problem with trailers and transshipments (VRPTT) (which generalizes the VRPTW)

(iii) the truck-and-trailer routing problem (TTRP) (which also generalizes the VRPTW and is a special case of the VRPTT)

The problems are described in detail in the respective chapters. Two of these problems, the GDRPP and the VRPTT, are new in the sense that they have not yet been described in the literature, i.e., there exist neither formulations nor heuristic or exact solution procedures for them.

What is *generalized* in these problems? Usually, in mathematical optimization, a problem A is said to be a generalization of another problem B if every instance of B constitutes an instance of A. This is the case when A considers all aspects relevant in B (and perhaps additional aspects as well; one can then also say that A is an *extension* of B). For example, the VRPTW is a generalization of the TSP, because every TSP instance can be interpreted as a VRPTW instance with one vehicle, where all customers may be visited at any time and have a supply of zero. In this paper, the word 'generalized' has an extended meaning (the term itself is generalized): Routing problems like the TSP, the VRPTW, and the CPP are essentially *sequencing* problems. The difficulty in these problems consists in determining an optimal sequence of objects/customers/streets. The GDRPP, the VRPTT, and the TTRP involve an additional degree of freedom: A *selection* must be made as to which primary objects to 'use' and/or which secondary objects to sequence (i.e., to 'visit' or 'service') in order to fulfil the given tasks. A well-known example where the term 'generalized' is used in this sense is the generalized travelling salesman problem. In this problem, the salesman has several visitation options for each customer (for example, offices of one and the same customer in different cities), and he must select one out of these options for each customer and must visit the selected options in an optimal sequence.

The selection of the problems treated in this paper is a result of the author's occupational activity in industry and academia. However, this does not mean that the problems are not of very general relevance. On the contrary: As will be demonstrated, these problems encompass important aspects encountered in many diverse practical applications that are as multi-faceted as the problems themselves.

1.2 Contribution

The paper is intended to be a contribution in the field of OR. OR is an interdisciplinary field which lies at the interface of business administration, mathematics, and computer science. Hence, the paper strives to make a three-fold contribution. While covering

(i) the problem- and application-oriented aspect (business administration) by considering problems which are relevant in the real world and which are, hence, interesting for practitioners,

(ii) the model- and method-oriented aspect (mathematics) by showing how the problems can be modelled, formulated, and solved, and

(iii) the implementation-oriented aspect (computer science/programming) by examining and comparing solution algorithms through performing computational experiments,

this paper focuses mainly on the mathematical aspect. The application aspect does not consist in the solution of real-world problem instances or in the presentation of case studies, but the practical relevance of the three problems dealt with is emphasized by the description of actual and potential areas of application and references to pertinent literature. The computer science aspect is covered by the sections on computational experiments and by Section 2.4, where the design and the implementation of a computer code for solving the shortest path problem with resource constraints are described.

The mathematical aspect has a model-oriented and a method-oriented part. The model-oriented part consists in the (mixed) integer programming formulations developed for the three problems under consideration. The method-oriented part consists in the algorithmic treatment of the three problems. All problems are solved by branch-and-cut, and the problems are also solved partially or completely by dynamic-programming-based labelling algorithms using the resource concept. Additionally, several heuristics are developed for the GDRPP, and for the TTRP, a branch-and-price algorithm is proposed.

A large part of the results presented in this paper are already contained in Drexl 2005a, Drexl 2005b (Chapter 3), Drexl 2005c (Chapter 4), and Drexl 2006 (Chapter 5).

1.3 Outline

In Chapter 2, the shortest path problem with resource constraints and its solution by labelling algorithms based on dynamic programming is considered. The resource concept, as specified in this section, constitutes a very powerful framework for the solution of many different kinds of routing problem. The present paper offers pertinent examples. Indeed, a central topic of this paper is the evaluation of the possibilities and limitations of dynamic-programming-based labelling algorithms for the solution of routing problems. Chapter 3 is concerned with the generalized directed rural postman problem (GDRPP); in Chapters 4 and 5, the vehicle routing problem with trailers and transshipments (VRPTT) and the truck-and-trailer routing problem (TTRP) are treated respectively. For each of the three problems being dealt with, different formulations are given, solution algorithms are developed, and computational experiments are presented and analyzed. Chapter 6 outlines the numerous possible lines of research on the GDRPP, the VRPTT, and the TTRP that could not be pursued in this paper.

1.4 Mathematical Prerequisites, Terminology, and Notation

The material covered in this paper, and the exposition thereof, require knowledge of the standard OR modelling and algorithmic techniques used to address problems of logistics and transport. Since many textbooks covering these prerequisites are available, the fundamentals are not repeated once again here. However, the central problem and the central algorithmic technique used throughout the paper, the (elementary) shortest path problem with resource constraints and its solution via a labelling algorithm, is treated in detail, not least because the resource concept used in labelling algorithms is, as yet, not a standard topic in OR textbooks.

The problems particularly relevant for this paper are shortest path problems with and without resource constraints (SPPs), uncapacitated arc routing problems (ARPs), travelling salesman problems (TSPs), and vehicle routing problems (VRPs). Standard references are Ahuja et al. 1993, p. 93 ff. (SPPs), Assad /Golden 1995, Eiselt et al. 1995a, Eiselt et al. 1995b, Dror 2000b (ARPs), Lawler et al. 1985, Gutin/ Punnen 2002 (TSPs), Toth/Vigo 2002 (VRPs).

As for the algorithmic side, familiarity with graph theory and linear and mixed integer programming is necessary. In particular, knowledge of the Dijkstra algorithm for the SPP and of the branch-and-cut and the branch-and-price algorithmic principles is assumed. Recommendable textbooks are Volkmann 1996, West 1996 (graph theory), Dantzig 1963, Chvátal 1983, Murty 1983 (linear programming), Sierksma 1996, Wolsey 1998, Martin 1999 (mixed integer programming). Standard references for branch-and-price are Barnhart et al. 1998, Vanderbeck 2000, Desrosiers/Lübbecke 2005, Lübbecke/Desrosiers 2005. Desaulniers et al. 1998 present a unified model for the solution of time-constrained vehicle routing and scheduling problems by branch-and-price.

Additionally, basic knowledge of complexity theory and data structures and algorithms is useful (Aho et al. 1983, Sedgewick 1992).

Textbooks covering the basic problems as well as the methods relevant for this paper are Domschke 1995, Domschke 1997, Bramel/Simchi-Levi 1997. The excellent books Grünert/Irnich 2005a, Grünert/ Irnich 2005b are particularly recommendable.

The term *formulation* is used to denote a concrete algebraic representation of a model of a problem. To represent the data defining a problem instance, systems of sets and functions representing properties of the elements of the sets are used. Such functions are called *attributes*. Throughout, for a function f, the index notation f_i is used for the value $f(i)$. M is a 'sufficiently large' positive constant.

$G = (V, L)$ denotes a graph with vertex set V and link set L. The relationship between V and L is established via two functions, $ta : L \to V$, the *tail function*, and $he : L \to V$, the *head function*. ta_a is called *tail of* a, and he_a is called *head of* a. A link l with $ta_l = he_l$ is called *loop*. All considered graphs are assumed to be loop-free w.l.o.g. Links l that may be traversed in either direction, i.e., from ta_l to he_l and vice versa, are called *edges*; links l that may be traversed only from ta_l to he_l are called *arcs*. Where there is no danger of confusion, the notations (ta_a, he_a) and $ta_a he_a$ are used to denote an arc a by its tail and head. Graphs containing only edges are called *undirected graphs* and are denoted by $G = (V, E)$. Graphs containing only arcs are called *directed graphs* or *digraphs* and are denoted by $D = (V, A)$. Graphs containing edges as well as arcs are called *mixed graphs*. Mixed graphs are a generalization of undirected and directed graphs. For any graph $G = (V, L)$, $L := E \cup A$, where E (A) is the edge (arc) set of G, i.e., the set of links that can be traversed in both directions, and the set of links that can only be traversed from tail to head respectively. Two edges $e, e' \in E$ are called *parallel* if $\{ta_e, he_e\} = \{ta_{e'}, he_{e'}\}$; two arcs $a, a' \in A$ are called parallel if $ta_a = ta_{a'}$ and $he_a = he_{a'}$. A graph without loops and without parallel links is called *simple*. All graphs considered in this paper are simple digraphs unless otherwise specified. Digraphs with an arc attribute representing *costs of traversal* are also referred to as *networks*. Given a digraph $D = (V, A)$, for each $V' \subsetneqq V$, the *forward* (*backward*) *star of* V' is the set $\{a \in A : ta_a \in V' \not\ni he_a\}$ ($\{a \in A : ta_a \notin V' \ni he_a\}$).

Given a graph $G = (V, L)$, $S := (i_0, l_1, i_1, l_2, i_2, l_3, \ldots, i_{p-1}, l_p, i_p)$ is called *link sequence* if $i_q \in V, q = 0, \ldots, p$, $l_q \in L, q = 1, \ldots, p$, $\{ta_{l_q}, he_{l_q}\} = \{i_{q-1}, i_q\}, q = 1, \ldots, p$. If S is a link sequence, it is called *closed* if $i_0 = i_p$ and *open* otherwise, and it is called *walk* or *non-elementary path* if $l_q \in A$ implies $ta_{l_q} = i_{q-1}, q = 1, \ldots, p$. A walk is called *directed* if all of its links are arcs. A (directed) walk is called (directed) *trail* if no link (arc) appears twice in the walk, i.e., if $q \neq q'$ implies $l_q \neq l_{q'}, q, q' = 1, \ldots, p$. An open (directed) trail is called (directed) (*elementary*) *path* if no vertex appears twice in the trail, i.e., if $q \neq q'$ implies $i_q \neq i_{q'}, q, q' = 0, \ldots, p$; this implies that no link (arc) appears twice. A closed (directed) trail is called (directed) *cycle*. It is assumed throughout that all considered (di)graphs are (*strongly*) *connected*, i.e., that there is a (directed) walk between any pair of vertices. Given a link traversal cost attribute of the form $c : L \to \mathbb{R}$, the *length* or *costs* c_S of a walk S is/are equal to the sum of the traversal costs of the links in S in the direction the links are traversed in S. $S := (i_0, \ldots, i_p)$ is a *shortest* walk if $c_S \leqq c_{S'}$ for all walks S' from i_0 to i_p. Independent of the meaning of c, a closed directed trail S with $c_S \leqq 0$ ($c_S < 0$) is called *non-positive* (*negative*) *cycle* (*with respect to* c) or, if the textual denomination of the attribute is *attr*, *non-positive* (*negative*) *attr cycle*. Note that it is usual in the literature to use the term 'shortest' walk, independent of the actual meaning of the attribute c, i.e., walks with minimal c_S value are referred to as 'shortest' even if c denotes costs or time or something else.

Chapter 2

Resource-Constrained Shortest Paths and Labelling Algorithms

The shortest path problem with resource constraints (SPPRC) seeks a shortest (cheapest, 'best') path in a network with arbitrary arc lengths (costs) from an origin vertex o to a destination vertex d subject to one or more 'resource constraints'. For example, one might seek a path of minimal length from o to d subject to the constraints that

- the total travel time must not exceed some upper bound and/or
- the total amount of some good that has to be collected at the vertices along the path be less than or equal to some capacity limit and/or
- if two vertices i and j are visited on a path, then i must be visited before j.

The SPPRC and its solution by a 'labelling algorithm' based on dynamic programming constitute a central topic of this paper. In actual fact, however, there is no such thing as *the* SPPRC; rather, SPPRCs are a general *framework*: Many different problems can be viewed and modelled as (special types of) resource-constrained shortest path problem(s).

The next section describes the workings of labelling algorithms. To illustrate the presented concepts, Section 2.2 provides a detailed numerical example. Section 2.3 discusses the issue of negative cycles in SPPRCs and their implications for the complexity of the problem. Section 2.4 describes a generic software framework for the solution of SPPRCs, which was written as part of this paper.

2.1 Labelling Algorithms

A labelling algorithm is the solution method of choice for SPPRCs. It works similarly to a labelling algorithm for shortest path problems without resource constraints (SPPs), e.g., the Dijkstra algorithm (cf. Ahuja et al. 1993, p. 93 ff.). The *basic concepts* used in a labelling algorithm for SPPRCs are the following (cf. Irnich/Desaulniers 2005):

- A *resource* is an arbitrarily scaled one-dimensional quantity that is *consumed* when moving along the arcs and whose consumption can be measured or computed at the vertices of a directed walk in a network. Examples are cost, time, load, or an information such as 'Has lorry k already visited customer i?' or 'Is lorry k currently pulling its trailer?'. The *value* of a resource at a vertex is stored in a *resource variable*. An arbitrarily scaled resource is *constrained* if there is at least one vertex in the network where the associated resource variable must not take all possible values. A cardinally scaled resource is constrained if there is at least one vertex in the network with a finite upper or lower bound on the value of the resource. The *resource window* of a nominally scaled resource r at a vertex is the set of allowed values of r at this vertex. The resource window of a cardinally scaled resource r at a vertex i is the interval $[lb_i^r, ub_i^r] \subseteq]-\infty, +\infty[$.

- A *resource extension function* (*REF*) is defined on each arc in a network for each resource considered. An REF for a resource r maps the set of all possible vectors of resource values at the tail of an arc to the set of possible values of r at the head of the arc. More precisely, let $R := (\sigma^1, \ldots, \sigma^{|R|})^T$ be a vector of (values of) resource variables. Then, an REF for a resource r is a function $f^r : A \times \mathbb{R}^{|R|} \to \mathbb{R}$. For simplicity, let $f^r_{ij}(R) := f^r(((i, j), R))$. An REF for a cardinally scaled resource r indicates lower bounds on the consumptions of r along the arcs. When seeking a path from an origin vertex o to a destination vertex d, partial paths from o to a vertex $i \neq d$ are *extended* along all arcs (i, j) emanating from i to create new, extended paths.

- For each o-i-path, there is a corresponding *label l resident* at i that stores the values of all resource variables at i for its path, along with the information on how it was created: the arc (h, i) over which i was reached and (a reference to) the label of the o-h-path whose extension along (h, i) yielded the o-i-path. (This makes it easy to reconstruct the path corresponding to a label.) Initially, there is exactly one label corresponding to the path (o). For nominally scaled resources, the values of the resource variables of the initial label are an input to the labelling algorithm. For cardinally scaled resources, the values of the resource variables of the initial label are w.l.o.g. all set to the lower bounds of their respective resource windows at o. A *label l* is *feasible* if and only if the value of each resource variable in l is within the resource window of its respective resource. If a label is not feasible, it is discarded. An *extension* of a path/label along an arc (i, j) is *feasible* if the resulting label at j is feasible. An *h-j-path* is *feasible* if, for each arc (i, i') in the path, a feasible label at i exists which can be extended along (i, i') to a feasible label at i'.

- To keep the number of labels as small as possible, it is decisive that a *dominance procedure* be performed to eliminate feasible but unnecessary labels. A *label l dominates* a label l' if both reside at the same vertex, if the value of the resource variable of each nominally scaled resource in l is equal to the corresponding value in l', if the value of the resource variable of each cardinally scaled resource in l is 'better' (less or greater, depending on the resource) than or equal to the corresponding value in l', and if the value of the resource variable of at least one cardinally scaled resource in l is strictly 'better' than the corresponding value in l'. A *path p dominates* a path p' if the label corresponding to p dominates the label corresponding to p'. Dominated paths/labels are discarded as well. An undominated path/label is called *Pareto-optimal*. A dominance procedure that always compares one label with one other label performs *pairwise dominance*. Pairwise dominance relationships are transitive. It is sometimes also possible and useful to devise dominance procedures for *non-pairwise* dominance, where it is checked whether a set of labels dominates one other label.

A labelling algorithm for an SPPRC consists of the following five basic steps:

(i) Initialization
The algorithm maintains a set of unprocessed labels. As mentioned above, the algorithm is initialized with a first label resident at o. This label is added to the set of unprocessed labels.

(ii) Label selection
An unprocessed label is selected from the set of unprocessed labels as a candidate for extension.

(iii) Dominance
Only undominated labels should be extended. Hence, before a label is extended, it should be checked whether the label is dominated or not. Different strategies for dominance are possible, and the dominance check can be performed at different points in the algorithm. In any case, it should be ensured that the check is performed as efficiently as possible. In particular, no dominance check should be performed more than once, and no redundant dominance check should be performed at all.

(iv) Label extension
The path corresponding to a label is extended along each arc in the forward star of the end vertex

of the path (the resident vertex of the label), i.e., for each forward star arc, a new label is created at the end vertex of the arc, and the values of the resource variables of the new label are computed by applying the REFs. Each new label is checked for feasibility. If it is feasible, it is added to the set of unprocessed labels.

(v) Solution selection
At the end of the algorithm, if the desired destination vertex could be reached, it must be decided which Pareto-optimal solution(s) (undominated label(s) resident at the destination vertex and the corresponding path(s)) should be returned.

2.2 A Concrete Example: the Shortest Path Problem with Time Windows

The easiest example of an SPPRC is the shortest path problem with time windows (SPPTW). It can be defined on a simple digraph $D = (V, A)$ with a cost function $c : A \to \mathbb{Z}$, an arc traversal time function $\tau^{tr} : A \to \mathbb{Z}_+$, and vertex time windows $tw : V \to \{[a, b] : a, b \in [0, T^{max}], a \leqq b\}$ with $T^{max} \in \mathbb{Z}_+$. $[a_i, b_i]$ is used in this chapter to denote the time window of vertex $i \in V$. Given an origin vertex $o \in V$ and a destination vertex $d \in V$, the task in the SPPTW is to determine a shortest path from o to d that respects the vertex time windows of all vertices in the path. Respecting vertex time windows means that the arrival time at any vertex i is not later than the respective right time window bound b_i. Arrival before the left time window bound a_i ('waiting') is allowed; the arrival time at a vertex i is simply assumed to be always later than or equal to a_i. This problem can be formulated as a mixed-integer programming problem as follows:

(SPPTW):

$$\sum_{(i,j)\in A} c_{ij}x_{ij} \to \min \text{ subject to} \tag{2.1a}$$

$$\sum_{\{h\in V:(h,i)\in A\}} x_{hi} - \sum_{\{j\in V:(i,j)\in A\}} x_{ij} = \begin{cases} -1, & i = o \\ 1, & i = d \\ 0, & \text{otherwise} \end{cases} \tag{2.1b}$$

$$x_{ij} = 1 \Rightarrow t_i + \tau_{ij}^{tr} \leqq t_j \quad \forall\, (i, j) \in A \tag{2.1c}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, (i, j) \in A \tag{2.1d}$$

$$a_i \leqq t_i \leqq b_i \quad \forall\, i \in V \tag{2.1e}$$

(2.1c) are equivalent to the linear constraints

$$t_i + \tau_{ij}^{tr} - t_j \leqq M(1 - x_{ij}) \quad \forall\, (i, j) \in A. \tag{2.1c'}$$

In the SPPTW, two resources have to be considered:

(i) an unconstrained, cardinally scaled resource for cost, and

(ii) a constrained, cardinally scaled resource for time.

For cost, the resource r^{cost} with resource variable σ_i^{cost} (indicating the value of the cost resource at vertex i) and REF $f^{r^{cost}}$ with

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}) = \sigma_i^{cost} + c_{ij} \tag{2.2}$$

can be used.

For time, the resource r^{time} with resource variable σ_i^{time} and REF $f^{r^{time}}$ with

$$f_{ij}^{r^{time}}(\sigma_i^{time}) = \max\{a_i, \sigma_i^{time} + t_{ij}\} \tag{2.3}$$

can be used. The resource window at a vertex i is $[a_i, b_i]$.

Usually, in a labelling algorithm, the cost REF models the objective function of an MIP model, and the REFs for resources other than cost correspond to the constraints.

What about dominance? Applying the definition given in the previous section and denoting by $\sigma^r(l)$ the (value of the) resource variable of resource r for a label l, the dominance relationship for two labels l, l' is as follows. l dominates l' if and only if both reside at the same vertex, $\sigma^{cost}(l) \leqq \sigma^{cost}(l')$, $\sigma^{time}(l) \leqq \sigma^{time}(l')$, and at least one of the two inequalities is strict. To understand why it is important to consider all resources in the dominance check, consider the network in Figure 2.1. It is easy to see that the shortest feasible path from o to d is $(o, a_2, j, a_4, i, a_3, d)$. There are two paths from o to i: $p_1 := (o, a_1, i)$ with costs of 2 and an arrival time at i of 3, and $p_2 := (o, a_2, j, a_4, i)$ with costs of 3 and an arrival time at i of 2. If there are no time windows, path p_2 can be discarded, because p_1 is shorter (has lower costs) and thus its extension to d is shorter (cheaper) than the extension of p_2: p_1 *dominates* p_2. However, with the time windows indicated in the figure, p_1 cannot be extended to d, because the arrival time at d of the extended path is 5, which is later than the right time window bound of d. The extension of p_2 to d, by contrast, is possible: The arrival time of the extended path at d is 4, which lies within d's time window. The only other o-d-path, $p_3 := (o, a_2, j, a_5, d)$ has costs of 7 and an arrival time at d of 3. Thus, if a shortest (i.e., cheapest) path from o to d is sought, discarding path p_2 by judging only by cost leads to a sub-optimal solution. If p_3 were also infeasible, no feasible solution at all would be discovered. The decisive point is that paths p_1 and p_2 (equivalently, their corresponding labels) are *incomparable*. p_1 is 'better' with respect to costs, p_2 is 'better' with respect to time. So, neither does p_1 dominate p_2, because for any extension of p_1, the arrival time at the last vertex of the corresponding extension of p_2 is not later, nor does p_2 dominate p_1, because for any extension of p_2, the costs of the corresponding extension of p_1 are lower. Neither path is 'better' than the other one; both paths are undominated or Pareto-optimal.

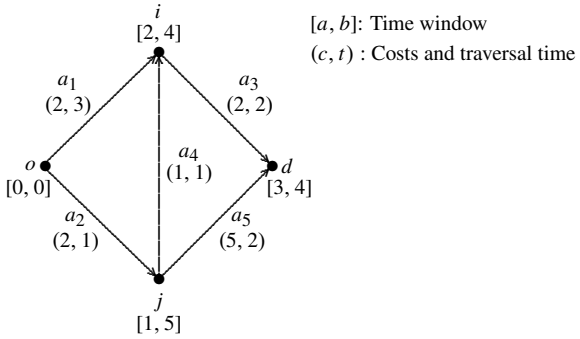


Figure 2.1: Example of an SPPTW

What is still needed is a rule for the selection of a label to be extended. The easiest rule is the FIFO rule: Add the labels to and remove them from the set S of unprocessed labels in the order that they are created. Many other strategies exist.

A labelling algorithm for the SPPTW can now be described as follows:

SPPTW-Labelling:
Given: a digraph $D = (V, A)$ constituting an SPPTW instance; origin and destination vertices $o \in V$ and $d \in V$; an empty set of unprocessed labels S; for each vertex $i \in V$, an empty set of labels resident at i.
Create an initial label l_0 at o with $\sigma^{cost}(l_0) := 0$, $\sigma^{time}(l_0) := a_o$, predecessor arc := $-$, and predecessor label := $-$.

Insert l_0 into the set S of unprocessed labels.
While S is not empty:
    Select and remove a label l^{curr} from S according to the FIFO rule.
    Perform a dominance check for all labels resident at the vertex i where l^{curr} resides.
    Mark all dominated labels as dominated.
    Delete all labels which are both dominated and processed.
    If l^{curr} is not dominated:
        Mark l^{curr} as processed.
        For each arc (i, j) in the forward star of i, the vertex where l^{curr} resides:
            Extend l^{curr} along (i, j) to a new label l^{new} by applying REFs (2.2) and (2.3).
            If l^{new} is not feasible
                Delete l^{new}.
            else
                Insert l^{new} into S.
                Insert l^{new} into the set of labels resident at j.
    else
        Delete l^{curr}.
If d could be reached from o:
    For each label l resident at d:
        Decide whether to return l as a Pareto-optimal solution; if so, recursively construct the corresponding Pareto-optimal o-d-path.

Figure 2.2 shows another example of an SPPTW network and the labels that are generated when computing a shortest path from o to d with the above labelling algorithm.

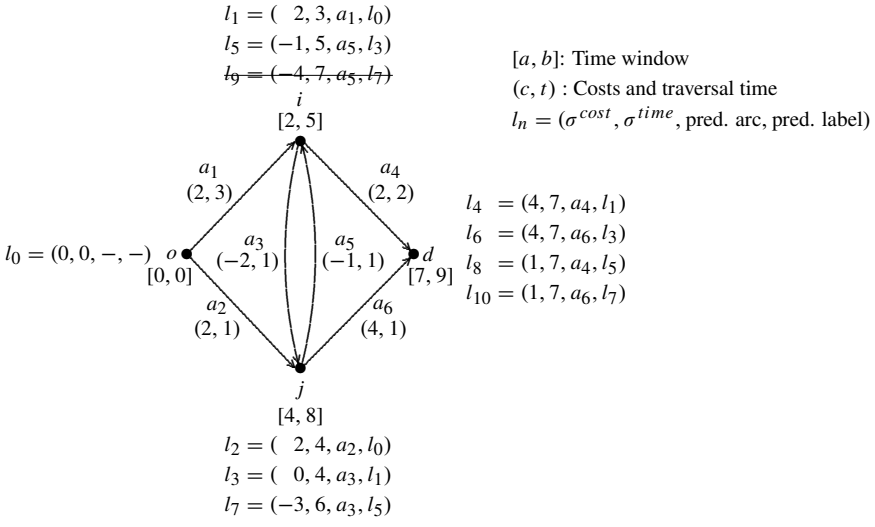


Figure 2.2: Example of an SPPTW with labels

The algorithm is initialized with the label $l_0 := (\sigma^{cost}(l_0), \sigma^{time}(l_0)$, predecessor arc, predecessor label) $:= (0, 0, -, -)$ at vertex o. l_0 is inserted into S. The while loop is entered, and l_0 is selected and removed from S. l_0 is not dominated, so it is marked as processed and extended along the arcs in the forward star of o, first to $l_1 := (2, 3, a_1, l_0)$ at i, and then to $l_2 := (2, 4, a_2, l_0)$ at j. Both new labels are feasible and are added to S and to the sets of labels resident at i and j respectively. The while loop is repeated, and l_1 is selected and removed from S. (The FIFO rule is applied here by selecting the label with the lowest number.) l_1 is undominated, so it is marked as processed and extended along the arcs in the forward star of its resident vertex, i, to labels $l_3 := (0, 4, a_3, l_1)$ at j and $l_4 := (4, 7, a_4, l_1)$ at d (in this order).

Both new labels are feasible and are added to S and to the sets of labels resident at j and d respectively. Next, l_2 is selected and removed from S. The dominance check at vertex j, l_2's resident vertex, yields the result that l_2 is dominated by l_3, which has the same arrival time but a lower cost. Therefore, l_2 is deleted. After that, l_3 is selected and removed from S. It is currently the only label resident at j, and it is not dominated, so it is marked as processed and extended along the arcs in the forward star of j to $l_5 := (-1, 5, a_5, l_3)$ at i and $l_6 := (4, 7, a_6, l_3)$ at d (in this order). Both new labels are feasible and are added to S and to the sets of labels resident at i and d respectively. l_4 is the next label that is selected and removed from S. It is undominated, so it is marked as processed. It is not extended, because there are no arcs emanating from d. The next label to be selected and removed from S is l_5. The dominance check at i yields the result that all labels at i (l_1 and l_5) are undominated. Hence, l_5 is marked as processed and extended to the feasible labels $l_7 := (-3, 6, a_3, l_5)$ at j and $l_8 := (1, 7, a_4, l_5)$ at d (in this order). Again, both new labels are feasible and are added to S and to the sets of labels resident at j and d respectively. l_6 is the next candidate label for extension, but it is dominated by l_8 and is therefore deleted, together with l_4. Now, l_7 is selected and removed from S. The dominance check at j yields no dominance relationship between the two existing labels (l_3 and l_7). l_7 is marked as processed and extended to $l_9 := (-4, 7, a_5, l_7)$ at i and $l_{10} := (1, 7, a_6, l_7)$ at d. l_9 is not feasible and is deleted. l_{10} is feasible and is added to S and to the set of labels resident at d. l_8 is the next label to be selected and removed from S. l_8 is not dominated, so it is marked as processed. It cannot be extended. The last label to be selected and removed from S is l_{10}. It is undominated and cannot be extended either. The while loop ends. d could be reached from o. The labels resident at d are l_8 and l_{10}. Both have the same values for costs and time, but they correspond to different paths. The predecessor label of l_8 is l_5, whose predecessor label is l_3, whose predecessor label is l_1, whose predecessor label is l_0, so the path corresponding to l_8 is $(o, a_1, i, a_3, j, a_5, i, a_4, d)$. Similarly, the path corresponding to l_{10} is $(o, a_1, i, a_3, j, a_5, i, a_3, j, a_6, d)$. Both Pareto-optimal paths are non-elementary. The shortest feasible elementary path is $(o, a_2, j, a_5, i, a_4, d)$.

2.3 Negative Cycles, Elementary Paths, and Complexity

A subtle difficulty with the above algorithm is that it does not compute shortest *elementary paths*, it computes shortest *walks*. When the underlying network does not contain any non-positive cost cycles, all undominated walks will be elementary paths. In the presence of negative cost cycles, there may be undominated non-elementary paths (see the above example). However, the algorithm will still be finite unless there is a non-positive time cycle. If there are both non-positive cost and time cycles, the algorithm may enter an infinite loop. For SPPRCs on general networks, this means that, in order to avoid such difficulties, it must be required that there be no non-positive cost cycles or at least one resource without non-positive cycles.

Formulation (2.1) is not valid if the traversal time function is $\tau^{tr} : A \to \mathbb{Z}$. Subtour elimination constraints involving only arc variables are necessary in this case; constraints (2.1c) are no longer sufficient. In addition, to require elementarity of paths, the constraints

$$\sum_{\{j \in V:(i,j) \in A\}} x_{ij} \leqq 1 \quad \forall\, i \in V \tag{2.4}$$

are necessary.

If there is at least one cycle with a resource consumption of zero for all considered resources, but no negative cycle for any of them, a label counter resource can be added to ensure finiteness: The labels are numbered consecutively in increasing order, and in the dominance check, if two labels have equal values for all other resources, the label with the lower number dominates the other one.

To solve the elementary SPPRC (ESPPRC) on general networks, Feillet et al. 2004 have extended an idea first presented in Beasley/Christofides 1989: the introduction of a binary resource for each vertex

(a *visitation counter*) indicating whether the vertex can still be visited. Partial paths are only extended in accordance with the visitation counter resources, and a path p_1 dominates a path p_2 only if p_1 can still visit any vertex that p_2 can. Their approach is used in this paper for the solution of the pricing problems in the branch-and-price approach for the TTRP and is described in detail in Chapter 5; in addition, the labelling algorithm used for solving the GDRPP in Chapter 3 is also based on visitation counter resources.

As far as the computational complexity of SPPRCs is concerned, it can be shown that they are, in general, $\mathcal{NP}$-hard, but on networks without negative cycles, or when the paths need not be elementary, SPPRCs can be solved in pseudopolynomial time (the above algorithm is such a one for the SPPTW). However, the ESPPRC on general networks is $\mathcal{NP}$-hard in the strong sense (Dror 1994); no pseudopolynomial algorithms are known.

The shortest path problem without resource constraints (SPP) (which corresponds to formulation 2.1a,b,d) is a special SPPRC with only one resource, the unconstrained resource 'cost'. The SPP is interesting from a computational complexity point of view, as it shows that the 'hardness' of a problem does not necessarily depend on the problem type or the instance size; it may also depend on the instance data: The SPP on networks with positive arc cost function is polynomially solvable; on general networks, it is $\mathcal{NP}$-hard (Garey/Johnson 1979, p. 213).

2.4 The `r_c_shortest_paths` Framework

As a part of this paper, a small framework for the solution of (E)SPPRCs was developed and implemented in C++ according to the paradigm of *generic programming*. The framework was accepted for inclusion into the Boost Graph library (BGL) and will be part of its 1.35.0 release. Boost (`boost.org`) is an online community that encourages development and peer review of free C++ libraries. This section describes the developed framework. Familiarity with the C++ programming language and the Standard Template Library (STL) is assumed. The section is rather technical, but the rest of the paper does not require knowledge of the material presented in the remainder of this chapter.

2.4.1 Fundamental Principles of Generic Programming

This subsection briefly reviews the main ideas of generic programming in the context of C++. Generic programming (GP) means programming with types as parameters (cf. Stroustrup 1998, p. 349). GP is 'a methodology for program design and implementation that separates data structures and algorithms through the use of abstract requirement specifications' (Siek et al. 2002, p. 19).

Important terms in generic programming are (cf. ib.):

- A *concept* is a set of requirements which a template argument must fulfil so that the class or function template will be compiled and executed correctly. Concepts are usually documented in source code comments or an external documentation. Instead of writing down the specification for a single type, a family of types is described, all of which have a common interface and semantic behaviour. Algorithms constructed in the generic style are applicable to *any* type that satisfies the requirements of the algorithm. The difference between a class and a concept is that a class is a singular, concrete data type with a unique interface, whereas a concept represents the commonalities of a set of types that may otherwise be completely different.
- A *model* describes the relationship between concrete types and the concepts fulfilled by them.
- A *refinement* is a concept which extends the requirements of another concept.

Sets of requirements of a concept are (ib., p. 28):

- *Valid expressions*: Expressions that must compile successfully in order for the types appearing in the expressions to be considered a model of the concept.

- *Associated types*: Auxiliary types related to a type modelling a concept.
- *Invariants*: Run-time characteristics of types that must be fulfilled at any time.
- *Complexity guarantees*: Worst and average case guarantees with respect to running time and memory requirements.

A central idea in software engineering is *polymorphism*, which means the ability to use many different types with the same variable or function parameter. There are two types of polymorphism:

(i) *Subtype polymorphism*:
In object-oriented programming (OOP), polymorphism is realized by virtual functions, inheritance (and overloading). Interface requirements of a concept are specified by (pure) virtual functions in an abstract base class. The concrete types which are derived from the abstract base class are called *subtypes*.

(ii) *Parametric polymorphism*:
In generic programming, polymorphism is realized through class or function templates (and overloading).

Subtype polymorphism uses run-time dispatch of function calls; parametric polymorphism uses compile-time dispatch. Whereas the former is sometimes (virtually) indispensable, the latter is more efficient, which is decisive in lower-level software components, such as mathematical algorithms. Note, however, that OOP and GP are complementing techniques, not competing ones.

2.4.2 Implementation Details

The implementation is designed for use with the Boost Graph library (BGL) (`boost.org/libs/-graph/doc/table_of_contents.html`). The BGL is a free, non-commercial, header-only library providing some general purpose graph classes and algorithms via a generic interface. Any graph library that implements this interface will be interoperable with the BGL generic algorithms and with other algorithms that also use this interface.

The implementation consists of overloaded template functions called `r_c_shortest_paths` and several underlying concepts.

The concepts are:

- ResourceContainer
  A type modelling the ResourceContainer concept is used to store the values of the resource variables of a label. It must be a refinement of the Assignable, LessThanComparable, and EqualityComparable concepts.
- Label
  This concept defines the interface for a label in the `r_c_shortest_paths` functions. As a design decision, the functions were not parameterized on the type of label. A concrete type parameterized on the graph and resource container type is used in the implementation. It stores the predecessor information necessary for reconstructing a path at the end of the algorithm.
- ResourceExtensionFunction
  A model of the ResourceExtensionFunction concept specifies how a label is extended along an arc. A type modelling this concept is likely to be a function or a function object.
- DominanceFunction
  A model of DominanceFunction is used to specify a dominance relation between two labels. A type modelling this concept will also probably be a function or a function object.

- ResourceConstrainedShortestPathsVisitor
  A design pattern extensively used in the BGL is that of an *algorithm visitor*. This is a generalization of the function object parameter used in many STL functions. An algorithm visitor for a BGL algorithm defines several functions that are called at certain *event points* during the algorithm.
  The ResourceConstrainedShortestPathsVisitor concept defines the visitor interface for the r_c_shortest_paths functions. The user can define a type with this interface and pass an object of this type to the r_c_shortest_paths functions in order to perform user-defined actions at the event points of the algorithm. The event points are when a label is selected and removed from the set of unprocessed labels, when a new label is known to be feasible or infeasible, and when a label is known to be dominated or undominated. A visitor can be used, for example, to count the number of generated labels for statistical comparisons or to mark a label as dominated according to information from outside the algorithm.

The functions are templated on the graph, resource container, resource extension function, dominance, label memory allocator and algorithm visitor type. There is a default type for the label memory allocator type and for the algorithm visitor type. Experience shows that, for 'small' resource containers, it may be useful to try a specialized small object allocator. For larger resource containers (i.e., for a large number of resources), the default allocator is the right choice.

The functions receive a graph object, an origin and a destination vertex, objects for resource extension and dominance, and containers where the solution is stored (Pareto-optimal labels and their corresponding paths). There are two optional parameters: one for an algorithm visitor object, and one for an object for allocating the memory for the labels.

It was a design decision not to parameterize the functions on the type of the container storing the set of unprocessed labels. Two different data structures were tried. The version submitted to Boost uses a priority queue. Another version of the functions uses buckets. The bucket version was approximately 10–40 % faster than the priority queue version. However, the priority queue version offers a simpler interface and is more generic. (To use buckets, there must be an integer-valued resource with strictly positive resource consumption. This resource need not be constrained, but an upper bound for its maximal value at any vertex must be known.)

The framework leaves a lot of work to the user. This, however, is a property inherent to the SPPRC. It is entirely up to the user to make sure that he stores the 'right' resources in his resource container object, that the resource extension function extends a label in the desired way, and that the dominance function declares the 'right' labels as dominated and not dominated.

The r_c_shortest_paths framework has been used to solve SPPs, SPPTWs, GDRPPs, and numerous variants of SPPRC and ESPPRC pricing problems in branch-and-price algorithms for the VRP, the VRPTW, and the TTRP. The framework has been used in the computational experiments described in Chapters 3 and 5.

Chapter 3

The Generalized Directed Rural Postman Problem

This chapter is concerned with the generalized directed rural postman problem (GDRPP). This problem has not yet been described in the literature, i.e., there are neither formulations nor heuristic or exact solution procedures, but, as its name implies, the problem is a straightforward generalization of the directed rural postman problem (DRPP). The DRPP consists in finding an optimal postman tour in a digraph, i.e., a least-cost cycle traversing each arc of a specified subset of the digraph's arcs at least once. The GDRPP is, to the DRPP, what the generalized travelling salesman problem (GTSP) is to the travelling salesman problem (TSP): In the GDRPP, there are several subsets (groups, classes, clusters) of arcs and the requirement is to find a least-cost cycle traversing at least one arc from each subset at least once. The subsets need neither be a partition of the arc set, nor need they be disjoint. The GDRPP is an interesting problem in its own right, but it is also important because many uncapacitated routing problems, especially arc routing problems (ARPs), can be modelled as GDRPPs. Moreover, there are several practically relevant constraints (e.g., turn penalties) which can be considered when modelling a problem as a GDRPP. Hence, the aim of the current chapter is to present formulations and solution procedures for the GDRPP as well as transformations of routing problems into GDRPPs, taking into account practically relevant constraints.

The chapter is structured as follows. In Section 3.1, some additional notation used in this chapter is introduced. In Section 3.2, integer programming formulations for the DRPP and the GDRPP are presented. The proposed transformations are described in Section 3.3, followed by solution procedures for the GDRPP in Section 3.4 and computational experiments in Section 3.5. The chapter ends with a brief conclusion. The relevant literature is discussed in the respective sections.

3.1 Notation

The graphs considered in this chapter need not be simple, as is usual in the arc routing context. Given an undirected, directed, or mixed graph $G = (V, L)$, $c : L \to \mathbb{Z}_+$ is a function denoting the *length* or the *costs of traversal of a link*. A graph $G = (V, L)$ is called *windy* if, instead of c, there are two functions $\overrightarrow{c} : L \to \mathbb{Z}_+ \cup \{\infty\}$ and $\overleftarrow{c} : L \to \mathbb{Z}_+ \cup \{\infty\}$ denoting the costs of traversal of a link from tail to head and from head to tail respectively. As the costs of traversal in one direction may be infinite, windy graphs are a generalization of mixed graphs.

Given a graph $G = (V, L)$, two arcs $a, a' \in A$ are called *antiparallel* if $ta_a = he_{a'}$ and $he_a = ta_{a'}$. For a given subset $V' \subseteqq V$, $L(V')$ is the link set of the vertex-induced subgraph of G induced by V'. Likewise, for a given subset $L' \subseteqq L$, $V(L')$ is the vertex set of the link-induced subgraph of G induced by L'. Furthermore, let $HE_{L'} := \{i \in V : \exists\, a \in L' \text{ with } he_a = i\}$, and let $TA_{L'}$ be defined accordingly for tails.

For graphs $G = (V, L)$ with turn penalties, there is a set T of *possible turns*, and there are three functions $il : T \to L$, called *inlink function*, $ol : T \to L$, called *outlink function*, and $tv : T \to V$, called *turnvertex function*. tv fulfils $tv_t \in \{ta_{il_t}, he_{il_t}\} \cap \{ta_{ol_t}, he_{ol_t}\}$ for each $t \in T$. Hence, a turn $t \in T$ can be denoted by (il_t, tv_t, ol_t). A graph with turn penalties is denoted by $G = (V, L, T)$. $pe_t : T \to [0, \infty]$ is the *turn penalty function*, and a turn t with $pe_t < \infty$ ($pe_t = \infty$) is *allowed* (*forbidden*). $S := (i_0, l_1, t_1, l_2, t_2, l_3, t_3, \ldots, l_{p-1}, t_{p-1}, i_p, l_p)$ is called *walk* if $i_0 \in \{ta_{l_1}, he_{l_1}\} \setminus \{tv_{t_1}\} \wedge i_p = tv_{t_{p-1}} \wedge l_q \in L, q = 1, \ldots, p \wedge il_{t_q} = l_q \wedge ol_{t_q} = l_{q+1}, q = 1, \ldots, p-1$. The *length* or *costs* of a walk S in a graph with turn penalties is/are equal to the sum of the traversal costs of the links in S in the direction the links are traversed in S plus the sum of the penalties of the turns in S minus the traversal costs of the last link in S in the direction the link is traversed in S. A walk is *feasible* if it performs no forbidden turns. The other terms defined so far for graphs without turn penalties apply analogously.

For 'generalized' problems on a graph $G = (V, L)$, with or without turn penalties, there are p_L groups $L_q \subseteqq L$, $q = 1, \ldots, p_L$, of links, and p_V groups $V_q \subseteqq V$, $q = 1, \ldots, p_V$, of vertices, whereof at least one element must be traversed/visited in any feasible solution. Such groups are referred to hereafter as *r-groups*, and the links (vertices) belonging to or *covering* an r-group are called *r-group links* (*r-group vertices*). W.l.o.g., it is assumed that every vertex is incident to an r-group link and that all links not belonging to an r-group are arcs. For graphs without turn penalties, it is assumed w.l.o.g. that there are no parallel links covering the same r-groups and no parallel non-r-group links. A link/vertex that will be traversed/visited in any optimal solution is called *required*. For simplicity of notation, let $HE_q := HE_{L_q}$, and let $TA_q := TA_{L_q}$. W.l.o.g., it is assumed that there is at least one *head-disjoint* r-group, i.e., an r-group $\tilde{q}$ with $HE_{\tilde{q}} \cap \bigcup_{q=1,\ldots,p_L, q \neq \tilde{q}} HE_q = \emptyset$. If this is not initially the case, an r-group q' with $|HE_{q'}| = \min_{q=1,\ldots,p_L}\{|HE_q|\}$ is selected. For each $i \in HE_{q'}$, one vertex v_i and two antiparallel arcs a and a' with $ta_a = he_{a'} = i$, $he_a = ta_{a'} = v_i$ and $c_a = c_{a'} = 0$ are added, and an additional r-group $\tilde{q}$, consisting of all arcs whose head is one of the newly added vertices, is created. In this way, the (optimal) objective function values of the modified and the original problem are equal. A(n optimal) postman tour for the original problem is easily obtained from a(n optimal) postman tour for the modified problem by simply removing from the latter all vertices in $HE_{\tilde{q}}$ and all arcs incident to one of these vertices. For most instances, in particular for instances representing real-world postman problems, $|HE_{q'}| \ll |V|$, so that the relative increase in instance size due to adding a head-disjoint r-group will be small. It is assumed w.l.o.g. that r-group 1 is a head-disjoint r-group with $|HE_1| \leqq |HE_q|$ for all head-disjoint r-groups $q \in \{1, \ldots, p_L\}$.

For non-generalized problems (i.e., problems where all r-groups have cardinality one) on a graph $G = (V, L)$, with or without turn penalties, $L = L^R \cup L^{NR}$, and L^R (L^{NR}) is the *set of required* (*non-required*) *links*. Likewise, $V = V^R \cup V^{NR}$, and V^R (V^{NR}) is the *set of required* (*non-required*) *vertices*. L^R induces $1 \leqq p_C \leqq |V|$ connected components in G, and the link-induced subgraph of G induced by L^R is called *component graph of* G and denoted by $G(L^R)$. The vertex and link sets of a connected component q of $G(L^R)$ are denoted by V_q^{CC} and L_q^{CC} respectively. W.l.o.g., it is assumed that every vertex is incident to a required link and that all non-required links are arcs. For graphs without turn penalties, it is assumed w.l.o.g. that there are no non-required links parallel to a required link with equal or lower costs and no parallel non-required links.

Traversing a non-required or a non-r-group arc, or traversing a required or an r-group arc without servicing it, is referred to as *deadheading*.

3.2 Integer Programming Formulations

All formulations in this section are based on a digraph $D = (V, A)$ with p_A r-groups $1, \ldots, p_A$ of arcs.

3.2.1 A Formulation for the DRPP

A possible formulation for the DRPP is:

(DRPP):

$$\sum_{a \in A} c_a x_a \rightarrow \min \text{ subject to} \tag{3.1a}$$

$$x_a \geqq 1 \quad \forall\, a \in A^R \tag{3.1b}$$

$$\sum_{\{a \in A : he_a = i\}} x_a - \sum_{\{a \in A : ta_a = i\}} x_a = 0 \quad \forall\, i \in V \tag{3.1c}$$

$$\sum_{\{a \in A : ta_a \in S \not\ni he_a\}} x_a \geqq 1 \quad \forall\, \emptyset \neq S \subsetneqq V, |S| \leqq \left\lfloor \frac{|V|}{2} \right\rfloor \tag{3.1d}$$

$$x_a \in \mathbb{Z}_+^0 \quad \forall\, a \in A \tag{3.1e}$$

x_a denotes the number of times arc a is traversed. (3.1b) ensures that all required arcs are covered, (3.1c) are flow conservation constraints, and (3.1d) are subtour elimination constraints.

3.2.2 Formulations for the GDRPP

A difficulty with 'generalized' ARPs is that not every vertex need be incident to a required link, because there need not be required links at all: If all r-groups have cardinality greater than one, it is impossible to tell in advance whether a certain link will be traversed in any optimal solution or not. If there is no r-group q with $\bigcap_{a \in A_q} \{ta_a, he_a\} \neq \emptyset$, there is not even a required vertex.

One possible formulation of subtour elimination constraints which takes this into account is

$$\sum_{\{a' \in A_q : he_{a'} = i\}} x_{a'} \geqq 1 \ \wedge \sum_{a'' \in A(S)} x_{a''} \geqq 1 \Rightarrow \sum_{\{a \in A : ta_a \in S \not\ni he_a\}} x_a \geqq 1$$

$$\forall\, q = 1, \ldots, p_A, i \in HE_q, \emptyset \neq S \subsetneqq V \setminus \{i\} \tag{3.2}$$

Constraints (3.2) are a sufficient, but not necessarily necessary condition for the elimination of subtours. At least two vertices $i \in V(A_q)$ are visited for each q. Constraints (3.2) are valid for *all* such vertices and *all* q, but they are already sufficient for *one* q.

Linearizing constraints (3.2) leads to the following formulation for the GDRPP:

(GDRPP1):

$$\sum_{a \in A} c_a x_a \rightarrow \min \text{ subject to} \tag{3.3a}$$

$$\sum_{a \in A_q} x_a \geqq 1 \quad \forall\, q = 1, \ldots, p_A \tag{3.3b}$$

$$\sum_{\{a \in A : he_a = i\}} x_a - \sum_{\{a \in A : ta_a = i\}} x_a = 0 \quad \forall\, i \in V \tag{3.3c}$$

$$x_a - (|A| + 1)\delta_i^1 \leqq 0 \quad \forall\, i \in HE_1, a \in A_1 \text{ with } he_a = i \tag{3.3d}$$

$$x_a - |A|(|A|+1)\delta_{iS}^2 \leqq 0 \quad \forall\, i \in HE_1, \emptyset \neq S \subseteqq V \setminus \{i\}, a \in A(S) \tag{3.3e}$$

$$\sum_{\{a \in A : ta_a \in S \not\ni he_a\}} x_a - \delta_i^1 - \delta_{iS}^2 \geqq -1 \quad \forall\, i \in HE_1, \emptyset \neq S \subseteqq V \setminus \{i\} \tag{3.3f}$$

$$x_a \in \mathbb{Z}_+^0 \quad \forall\, a \in A \tag{3.3g}$$

$$\delta_i^1 \in \{0,1\} \quad \forall\, i \in HE_1 \tag{3.3h}$$

$$\delta_{iS}^2 \in \{0,1\} \quad \forall\, i \in HE_1, \emptyset \neq S \subseteqq V \setminus \{i\} \tag{3.3i}$$

The weakness of this formulation is that there is not only an exponential number of subtour elimination constraints, but also an exponential number of δ_{iS}^2 variables.

A disaggregated form of subtour elimination constraints (3.2) can be stated as

$$\sum_{\{a' \in A_1 : he_{a'} = i\}} x_{a'} \geqq 1 \ \wedge\ x_{a''} \geqq 1 \Rightarrow \sum_{\{a \in A : ta_a \in S \not\ni he_a\}} x_a \geqq 1$$

$$\forall\, i \in HE_1, \emptyset \neq S \subseteqq V \setminus \{i\}, a'' \in A(S) \tag{3.4}$$

This leads to the following formulation:

(GDRPP2):

$$\sum_{a \in A} c_a x_a \to \min \text{ subject to} \tag{3.5a}$$

$$\sum_{a \in A_q} x_a \geqq 1 \quad \forall\, q = 1, \ldots, p_A \tag{3.5b}$$

$$\sum_{\{a \in A : he_a = i\}} x_a - \sum_{\{a \in A : ta_a = i\}} x_a = 0 \quad \forall\, i \in V \tag{3.5c}$$

$$x_a - (|A|+1)\delta_i \leqq 0 \quad \forall\, i \in HE_1, a \in A_1 \text{ with } he_a = i \tag{3.5d}$$

$$x_a - (|A|+1)\delta_a \leqq 0 \quad \forall\, a \in A \tag{3.5e}$$

$$\sum_{\{a' \in A : ta_{a'} \in S \not\ni he_{a'}\}} x_{a'} - \delta_i - \delta_a \geqq -1 \quad \forall\, i \in HE_1, \emptyset \neq S \subseteqq V \setminus \{i\}, a \in A(S) \tag{3.5f}$$

$$x_a \in \mathbb{Z}_+^0 \quad \forall\, a \in A \tag{3.5g}$$

$$\delta_i \in \{0,1\} \quad \forall\, i \in HE_1 \tag{3.5h}$$

$$\delta_a \in \{0,1\} \quad \forall\, a \in A \tag{3.5i}$$

It is possible to do without the δ_a variables and the constraints (3.5e) and (3.5i).

(GDRPP3):

$$\sum_{a \in A} c_a x_a \to \min \text{ subject to} \tag{3.6a}$$

$$\sum_{a \in A_q} x_a \geqq 1 \quad \forall\, q = 1, \ldots, p_A \tag{3.6b}$$

$$\sum_{\{a\in A:he_a=i\}} x_a - \sum_{\{a\in A:ta_a=i\}} x_a = 0 \quad \forall\, i \in V \tag{3.6c}$$

$$x_a - (|A|+1)\delta_i \leqq 0 \quad \forall\, i \in V, a \in A \text{ with } he_a = i \tag{3.6d}$$

$$\sum_{\{a'\in A:ta_{a'}\in S\not\ni he_{a'}\}} x_{a'} - \delta_i - \delta_{i'} \geqq -1 \quad \forall\, i \in HE_1, TA_q \ni i' \in S \subseteqq V \setminus \{i\},$$

$$q \in \{2,\ldots,p_A\} \tag{3.6e}$$

$$x_a \in \mathbb{Z}^0_+ \quad \forall\, a \in A \tag{3.6f}$$

$$\delta_i \in \{0,1\} \quad \forall\, i \in V \tag{3.6g}$$

Violated constraints (3.6e) can be separated by computing the maximum flow between all pairs of vertices i, i' with $HE_1 \ni i \neq i' \in TA_q, q \in \{2,\ldots,p_A\}$, in the support graph. For each pair i and i' where the maximum flow is less than one, the respective constraint (3.6e) must be checked and may be added if it is violated. However, this procedure does not work on graphs without head-disjoint r-groups. If a vertex i is the head of two arcs belonging to different r-groups, the maximum flow from i to i must be computed, which is impossible with standard maximum flow algorithms without modifying the graph.

In a digraph representing a GDRPP instance, there are feasible solutions which cannot be optimal, namely, postman tours with suboptimal deadheading between two r-group arcs, i.e., tours that do not always use the shortest path to move from one r-group arc to the next one. Such solutions are not optimal, even if the respective r-group arcs are selected for covering their r-groups and even if these arcs are in an optimal order. Such redundancies can be eliminated by the following preprocessing. $m-1$ duplicates of the common tail and head of m parallel r-group arcs are created, each duplicate tail-head pair is connected by one of the parallel arcs, shortest paths are computed between all pairs $(he_a, ta_{a'})$ with $he_a \neq ta_{a'}$ for any two r-group arcs a, a' covering different r-groups, the respective arcs are added (as r-group arcs for all r-groups being covered on the corresponding shortest path) with costs equal to the costs of the shortest path, all original non-r-group arcs are deleted, and non-r-group arcs with costs of zero are added between the duplicates of the head/tail of formerly antiparallel arcs. See Figure 3.1.

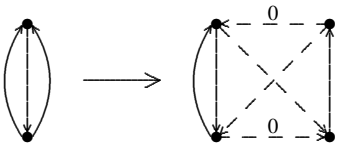


Figure 3.1: Making a GDRPP digraph simple

On the one hand, such a preprocessing will usually increase the number of arcs in the digraph, and, consequently, the number of variables. On the other hand, in any optimal solution on the preprocessed digraph, each arc will be traversed at most once, so that the traversal variables can be made binary. Binary variables are easier to handle than general integer variables. Hence, it is not immediately clear whether the computational effort to solve the instance will be reduced by such a preprocessing.

3.3 Transformations

In this section, various types of uncapacitated routing problem are considered. They can be classified along several orthogonal dimensions:

- type of routing application
  - vertex routing

  - arc routing
- graph type
  - undirected
  - directed
  - mixed
  - windy
- generalized, clustered, and hierarchical models
- additional real-world constraints
  - turn penalties
  - zigzag service
  - different costs for servicing and deadheading in different directions

The idea of transforming problems into other problems is not new. The contribution of this section consists in the fact that it extends and unifies existing transformation procedures under a common transformation target: the GDRPP. In the following, formulations such as 'problem A is transformed into (is modelled as) problem B' are used as shortcuts for 'a graph representing an instance of problem B is created from a graph representing an instance of problem A'. All of the presented transformations have to be interpreted as 'proof-of-concept'. They just describe one of several possibilities of how to (polynomially) transform one problem into another. No claim is made as to whether the presented transformations are the simplest or the most elegant ones.

3.3.1 Standard Arc Routing Problems

The archetypal arc routing problem is the Chinese postman problem (CPP) (Guan 1962), which seeks a least-cost cycle in a graph, traversing each link at least once. The problem of finding a least-cost cycle traversing only a specified subset of the links of a graph at least once is called rural postman problem (RPP) (Orloff 1974). Depending on the type of graph, these problems are referred to as undirected, directed, mixed, or windy CPP/RPP.

For routing problems on graphs containing links which may be traversed in both directions, one part of the solution consists in determining a direction of traversal for each such link which is to be used in the solution. This simply means the selection of one out of two alternatives. Hence, it is common in the literature to represent an edge or a windy link by two variables whose values indicate the number of traversals of the link in the respective direction, cf. Win 1987. Therefore, to model the above standard arc routing problems as GDRPPs, all links that can be traversed in both directions are replaced by two antiparallel arcs with the corresponding traversal costs. Each such pair of antiparallel arcs corresponding to a required link forms one r-group in the GDRPP, and each required original arc forms one r-group, too.

Laporte 1997 considers transformations of ARPs into asymmetric travelling salesman problems (ATSPs). He proceeds as follows. First, similar to Win 1987, required links that can be traversed in both directions are replaced by two antiparallel arcs. Each arc corresponding to a required link is then replaced by one vertex, and all non-required links and all original vertices are deleted. If v_a and $v_{a'}$ are two vertices corresponding to the two arcs a and a', v_a and $v_{a'}$ are connected by an arc with costs corresponding to the costs of a shortest path from he_a to $ta_{a'}$ in the original graph. The resulting problem is a generalized asymmetric travelling salesman problem (GATSP). This problem is transformed into an ATSP by a procedure due to Noon/Bean 1991. Laporte 1997 solves to optimality (transformations of) mixed CPPs with up to 440 arcs and 10 edges and mixed RPPs with up to 660 arcs and 5 edges.

3.3.2 The Generalized Travelling Salesman Problem

The best-known combinatorial optimization problem is the travelling salesman problem (TSP), which basically comes in two variants. On undirected graphs, it is called symmetric TSP (STSP); on directed graphs, it is called asymmetric TSP (ATSP). The generalized travelling salesman problem (GTSP) is a TSP on a graph whose vertex set is partitioned into disjoint clusters with the requirement that exactly (or, sometimes, at least) one vertex from each cluster be visited on any feasible tour. The asymmetric version, the GATSP, on a simple digraph $D = (V, A)$ with clusters $C_1, \ldots, C_K$ can be formulated as follows (Noon/Bean 1991):

(GATSP):

$$\sum_{a \in A} c_a x_a \rightarrow \min \text{ subject to} \tag{3.7a}$$

$$\sum_{\{a \in A : ta_a \notin C_k \ni he_a\}} x_a = 1 \quad \forall\, k \in \{1, \ldots, K\} \tag{3.7b}$$

$$\sum_{\{a \in A : ta_a \in C_k \not\ni he_a\}} x_a = 1 \quad \forall\, k \in \{1, \ldots, K\} \tag{3.7c}$$

$$\sum_{\{a \in A : ta_a = i\}} x_a - \sum_{\{a \in A : he_a = i\}} x_a = 0 \quad \forall\, i \in V \tag{3.7d}$$

$$\sum_{k \in S} \sum_{ta_a \in C_k} \sum_{k' \notin S} \sum_{he_a \in C_{k'}} x_a \geqq 1 \quad \forall\, S \subsetneqq \{1, \ldots, K\}, 2 \leqq |S| \leqq \left\lfloor \frac{K}{2} \right\rfloor \tag{3.7e}$$

$$x_a \in \{0, 1\} \quad \forall\, a \in A \tag{3.7f}$$

Constraints (3.7b) ((3.7c)) require that each cluster be entered (left) exactly once. Constraints (3.7d) are flow conservation constraints. Either constraints (3.7b) or (3.7c) are redundant, because of constraints (3.7d) and the handshaking lemma. Constraints (3.7e) are a generalized version of the Dantzig-Fulkerson-Johnson subtour elimination constraints. They also hold for non-disjoint clusters, as long as there is not a single vertex belonging to *every* cluster, in which case an optimal 'tour' consists only of one such vertex. But this is a trivial case that can be excluded w.l.o.g. If it is required that *at least* one vertex from each cluster be visited (which implies that a vertex may be visited more than once), (3.7b) and (3.7c) can be skipped if (3.7e) is modified so as to also allow subsets S of cardinality one.

3.3.2.1 Making the Clusters of a GTSP Disjoint

When solving the GTSP, it is interesting that existing algorithms (cf. Noon/Bean 1991, Dror/Langevin 1997) assume disjoint clusters. Noon/Bean 1991 state that this means no loss of generality, as a problem with overlapping clusters can be transformed into a problem with disjoint clusters. These authors, however, do not present the transformation, but refer to two unpublished reports. Presumably, they mean the following transformation (H. Gündüz 2004, personal communication). Consider Figure 3.2. Vertex v_3 is in the intersection of the two clusters C_1 and C_2. To make the clusters disjoint, a duplicate v'_3 of v_3 is created, v_3 is kept in C_1 and v'_3 is put in C_2, the two vertices are connected with two antiparallel arcs with costs of $-M$, the costs on the arcs (v_3, v_i) for all vertices $v_i \notin C_1 \cup C_2$ are kept, the costs on the arcs (v_i, v_3) for all vertices $v_i \notin C_1 \cup C_2$ are increased by M, arcs (v_3, v_i) and (v_i, v_3) with costs of $2M$ are introduced for all $v_i \in C_2 \setminus \{v'_3\}$, and a corresponding arc incident to v'_3 is introduced for each arc incident to v_3. All arcs not incident to v_3 or v'_3 keep their original costs. The antiparallel arcs between v_3 and v'_3 with costs of $-M$ make sure that the two vertices are visited consecutively on any optimal tour, and, together with the addition of M to the costs of all arcs entering v_3 and v'_3, that the correct costs are incurred when selecting v_3 or v'_3 as the vertex to be visited in clusters C_1 and C_2. The costs of $2M$ on the arcs (v_3, v_i) for $v_i \in C_2 \setminus \{v'_3\}$ and on their counterparts involving v'_3 avoid multiple visits of the original clusters C_1 and C_2.

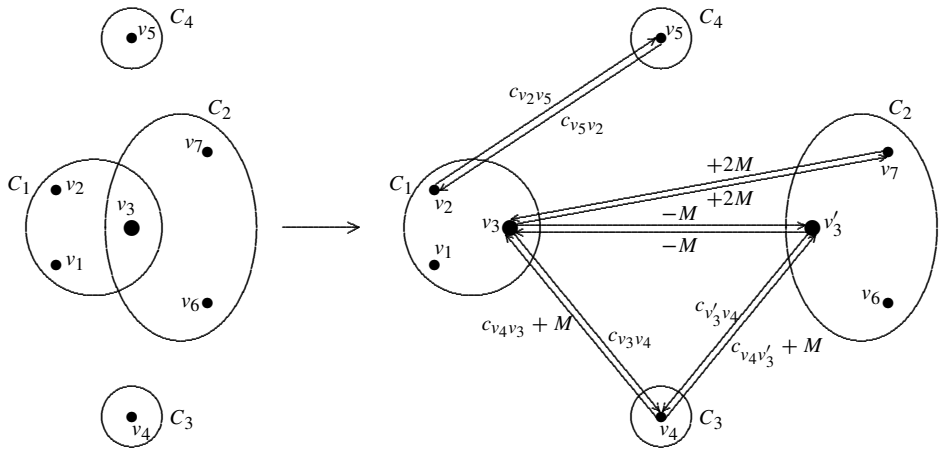


Figure 3.2: Making the clusters of a GTSP disjoint

The transformation works analogously if there are more than two overlapping clusters and/or if there is more than one vertex in the intersection.

3.3.2.2 Transforming a Generalized ATSP into a GDRPP, and Vice Versa

The transformation of a GATSP into a GDRPP works as follows. Each cluster of vertices is replaced by one r-group of arcs containing all arcs emanating from (or, equivalently, leading to) one of the vertices in the cluster. An alternative procedure is to replace each cluster of vertices by an r-group of arcs with costs of zero and with a one-to-one correspondence between the original vertices and the arcs replacing them, and to set the heads (tails) of all original arcs leading to (emanating from) an original vertex equal to the tail (head) of its corresponding newly added arc.

The inverse transformation is also very simple. To transform a GDRPP instance into a GATSP instance, each r-group arc is replaced by a vertex. Vertices v_a, $v_{a'}$ corresponding to arcs a, a' covering different r-groups are connected by two antiparallel arcs with costs equal to the costs of a shortest path from he_a to $ta_{a'}$ plus c_a, and from $he_{a'}$ to ta_a plus $c_{a'}$ respectively. All r-groups remain logically the same, the only difference is that they now contain vertices instead of arcs.

3.3.3 The Generalized Directed General Routing Problem

Another well-known problem is the general routing problem (GRP) (Orloff 1974). It is usually described for undirected graphs, but the directed version of the problem can be described as follows. Given a digraph, a specified subset of arcs and a specified subset of vertices, the problem is to find a least-cost cycle traversing/visiting each arc/vertex of the respective subsets at least once. The extension of this problem corresponding to the GDRPP is the generalized directed general routing problem (GDGRP). Adding

$$\sum_{i \in V_q} \sum_{\{a \in A : ta_a = i\}} x_a \geqq 1 \quad \forall\, q = 1, \ldots, p_V \tag{3.8}$$

to the formulations (3.3) or (3.5) for the GDRPP yields a formulation for the GDGRP. Evidently, constraints (3.8) are valid inequalities for all 'r-group tails', and they can just as well be formulated in terms of 'r-group heads'.

The GDGRP contains, as special cases, the symmetric and the asymmetric travelling salesman problem as well as their 'generalized' versions, and also the windy general routing problem examined in Corberán et al. 2005b. The transformation of a GDGRP into a GDRPP works as described for the GATSP. If at

least one element of a subset S of $V \cup A$ is to be visited/traversed, this can be modelled by adding all arcs a with $ta_a \in S \cap V$ to S.

Blais/Laporte 2003 solve the undirected, directed, and mixed general routing problem by transformations into GATSP, ATSP, and undirected, directed, and mixed RPP. The transformations into GATSP and ATSP work as described in Section 3.3.1. In addition, for each required vertex in the GRP, there is a corresponding vertex in the GATSP and the ATSP. The transformation of an undirected, directed, or mixed GRP into a corresponding RPP consists simply in replacing each required vertex by a corresponding link with costs of zero, as described in Section 3.3.2.2. Blais/Laporte 2003 solve to optimality directed GRPs with up to 4,000 required vertices and 3,000 required arcs, undirected GRPs with up to 90 required vertices and 10 required edges or 10 required vertices and 90 required edges, and mixed GRPs with up to 110 required vertices, 200 required arcs and 25 required edges.

3.3.4 The Clustered Travelling Salesman Problem

This problem was introduced by Chisman 1975. It is a TSP on a graph whose vertex set is partitioned into disjoint clusters with the requirement that all vertices in a cluster be visited consecutively in an arbitrary (preferably optimal) order. The clustered ATSP on a simple digraph $D = (V, A)$ with clusters $C_1, \ldots, C_K$ can be formulated as follows:

(Clustered ATSP):

$$\sum_{a \in A} c_a x_a \rightarrow \min \text{ subject to} \tag{3.9a}$$

$$\sum_{\{a \in A : he_a = i\}} x_a = 1 \quad \forall\, i \in V \tag{3.9b}$$

$$\sum_{\{a \in A : ta_a = i\}} x_a = 1 \quad \forall\, i \in V \tag{3.9c}$$

$$\sum_{ta_a \in S \not\ni he_a} x_a \geqq 1 \quad \forall\, S \subsetneqq V, 2 \leqq |S| \leqq \left\lfloor \frac{|V|}{2} \right\rfloor \tag{3.9d}$$

$$\sum_{k=1}^{K} \sum_{\{a \in A : ta_a \in C_k \not\ni he_a\}} x_a = K \tag{3.9e}$$

$$x_a \in \{0, 1\} \quad \forall\, a \in A \tag{3.9f}$$

Constraints (3.9a)–(3.9d) and (3.9f) are the usual ATSP formulation with Dantzig-Fulkerson-Johnson subtour elimination constraints. Constraint (3.9e) requires that any feasible solution use exactly K intercluster arcs. An alternative to constraint (3.9e) is

$$\sum_{\{a \in A : ta_a, he_a \in C_k\}} x_a = |C_k| - 1 \quad \forall\, k \in \{1, \ldots, K\} \tag{3.10}$$

In order to solve the clustered ATSP exactly, any branch-and-cut algorithm for the ATSP can be used and constraint (3.9e) or constraints (3.10) can be added at the root vertex of the branch-and-bound tree. Hence, when studying the clustered TSP, the focus lies mainly on heuristics exploiting the special structure of the problem.

The clustered TSP with non-disjoint clusters has feasible solutions only when no more than two clusters overlap or when the kth cluster, for $k \geqq 3$, is contained in/is a proper subset of the $(k-1)$th cluster. See Figure 3.3.

The requirement of disjoint clusters is not restrictive, as non-disjoint clusters can be made disjoint by a simple procedure (H. Gündüz 2004, personal communication). Consider Figure 3.4. It must be ensured

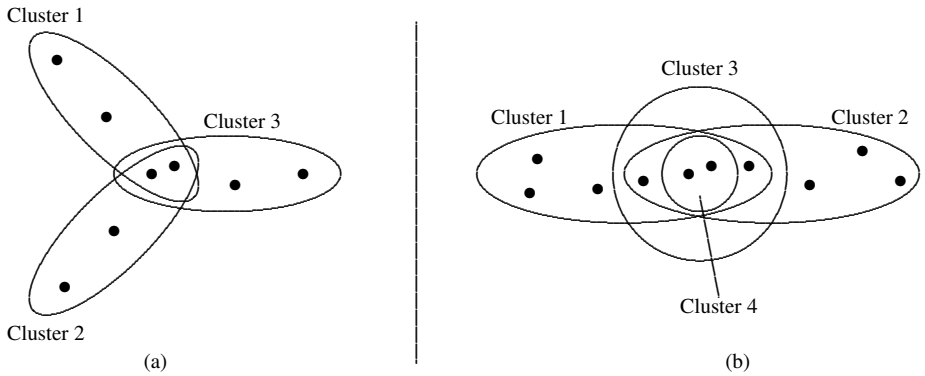


Figure 3.3: (a) Infeasible and (b) feasible clustered TSP with overlapping clusters

that the three clusters in (b) are visited consecutively either $1 \to 2 \to 3$ or $3 \to 2 \to 1$. This is achieved by adding M to all arcs between vertices in clusters 1 and 2 and between vertices in clusters 2 and 3, and by adding $2M$ to all other intercluster arcs. Alternatively, it is possible to solve the Hamiltonian path problem over the vertices in cluster 2 for each pair of vertices i, i' belonging to cluster 2, to compute shortest paths from h to i and from i' to j for each pair of vertices h belonging to cluster 1 and j belonging to cluster 3, and to connect h and j via an arc with costs equal to the minimal sum of the costs of a shortest h-i-path plus the costs of a shortest i'-j-path plus the costs of a shortest Hamiltonian path from i to i' over all pairs i, i'. Then, all arcs emanating from clusters 1 and 3 leading to clusters other than 3 and 1 respectively must be removed, as well as all vertices belonging to cluster 2 and all arcs incident to one of these vertices.

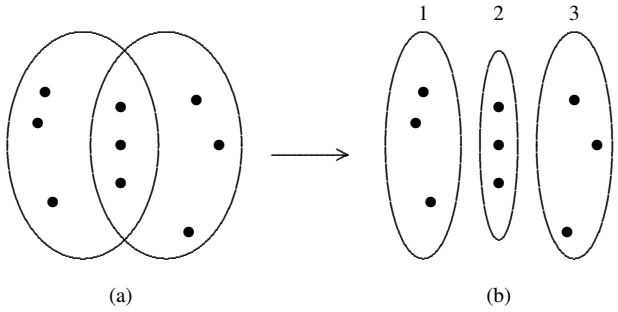


Figure 3.4: Clustered TSP (a) with and (b) without overlapping clusters

The clustered TSP can be modelled as a DRPP by replacing each vertex by a required arc, connecting these arcs with the remaining original arcs by setting the heads (tails) of all original arcs leading to (emanating from) an original vertex equal to the tail (head) of its corresponding newly added arc as described in the section on the GATSP, creating a corresponding cluster for every cluster of vertices in the original graph, and adding M to the costs of each intercluster arc. See Figure 3.5. The resulting DRPP can be called *clustered directed rural postman problem*. If an r-group for each arc corresponding to a vertex of the original clustered TSP is created, formulations (3.3) and (3.5) for the GDRPP are also correct for the clustered DRPP with disjoint and overlapping clusters.

3.3.5 The Clustered Directed Rural Postman Problem

The clustered DRPP with non-disjoint clusters can be transformed into the clustered DRPP with disjoint clusters as follows (see Figure 3.6). First, all clusters are made strongly connected by adding deadheading arcs corresponding to shortest paths. More precisely, a pair of antiparallel non-r-group arcs is added between any pair of vertices belonging to different components of a cluster. The costs of these arcs are set

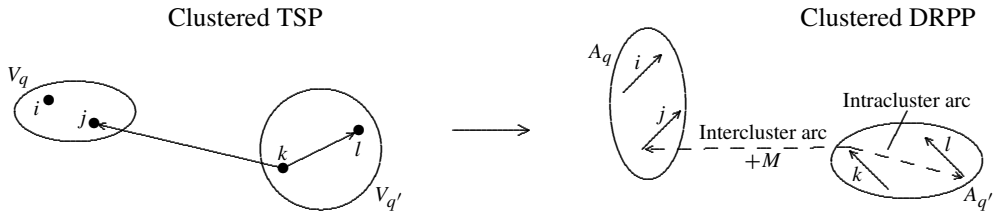


Figure 3.5: Clustered TSP and clustered DRPP

to the costs of the respective shortest paths. Then, each pair of overlapping clusters is separated/detached, as shown on the right hand side of the figure. Intercluster arcs are added, leading from cluster 1 to cluster 2, from cluster 2 to cluster 3, from cluster 3 to cluster 2, and from cluster 2 to cluster 1. The costs of these arcs are set to the costs of the shortest paths between the respective end vertices in the original graph. The clusters must be visited consecutively in either direction. Again, this is achieved by adding M to all intercluster arcs $1 \to 2$, $2 \to 3$, $3 \to 2$ and $2 \to 1$, and by adding $2M$ to all other intercluster arcs. Alternatively, the costs of an optimal cluster traversal (servicing the arcs in the cluster) for each pair of vertices i, i' incident to an arc belonging to cluster 2 can be computed. After that, for each pair of vertices h, j, where h is incident to an arc in cluster 1 and j is incident to an arc in cluster 3, or vice versa, a shortest path from h via any optimal traversal of cluster 2 to j is determined. h and j can then directly be connected by an arc with costs equal to the respective shortest path costs, and cluster 2 and all arcs leading to or emanating from a vertex in the cluster can be removed.

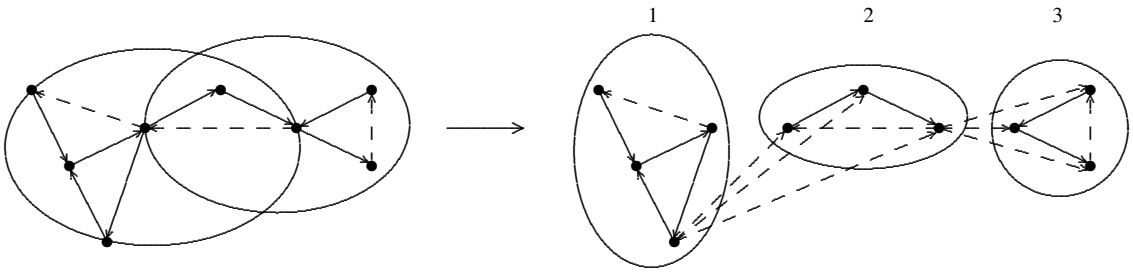


Figure 3.6: Clustered DRPP with and without overlapping clusters

Note that both transformations (for clustered TSP and clustered DRPP) extend straightforwardly to the case of nested clusters, as in Figure 3.3(b). The described procedures must then be applied in four directions: $1 \to 3 \to 4 \to 2$, $1 \to 4 \to 3 \to 2$, $2 \to 3 \to 4 \to 1$, and $2 \to 4 \to 3 \to 1$.

The alternative procedure above is an adaptation of the one used in Dror/Langevin 1997, who considered the clustered DRPP, transformed it into the GATSP, and solved it exactly by a Lagrangian-based method due to Noon/Bean 1991. The largest instances they solved had 581 vertices and 770 arcs.

3.3.6 Hierarchical Postman Problems

Several authors consider so-called hierarchical postman problems (HPPs), see, e.g., Ghiani/Improta 2000, Cabral et al. 2004. In HPPs, there are disjoint clusters of links as in the clustered version just described, and there is a given order (a 'hierarchy') in which the clusters have to be serviced, i.e., the sequence of clusters is fixed. This can be modelled as a special case of a clustered DRPP by adding M to all intercluster arcs for clusters that must be serviced consecutively, and by adding $2M$ to all other intercluster arcs.

3.3.7 The Clustered Generalized Directed Rural Postman Problem

As far as this author knows, this problem has also not yet been treated in the literature. It is encountered in the context of postal delivery: The streets (or street segments, or street segment sides) of a town are partitioned into (not necessarily connected) 'delivery sections' or 'service units' that must be serviced consecutively in an arbitrary order. The use of streets from different delivery sections for deadheading is allowed. Not all streets necessarily need be serviced. The problem bears some similarity to the clustered TSP, and therefore it is named *clustered generalized directed rural postman problem* (clustered GDRPP). It can be modelled as a GDRPP as follows. First, all undirected, windy or zigzag links are replaced by appropriate arcs and the appropriate r-groups are created. Then, all service units are made strongly connected by adding deadheading arcs corresponding to shortest paths, as described in the section on the Clustered DRPP. After that, for each service unit in the original graph, a 'super-cluster' containing all r-groups corresponding to links of the respective service unit is created. Finally, M is added to the costs of each arc connecting two super-clusters. See Figure 3.7.

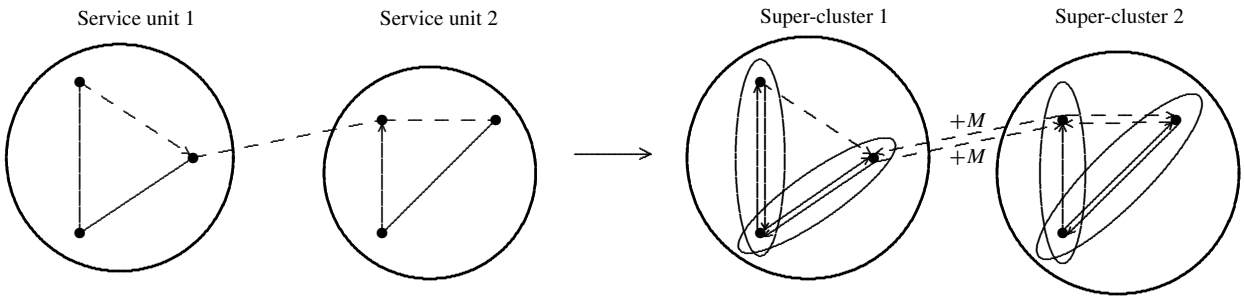


Figure 3.7: Clustered generalized DRPP

3.3.8 The Generalized Clustered Directed Rural Postman Problem

The generalized clustered DRPP is defined on a digraph with several disjoint groups of clusters of arcs and requires that at least one cluster from each group be serviced consecutively. As in the clustered GDRPP, the use of arcs belonging to different clusters for deadheading is allowed. To model the generalized clustered DRPP, additional decision variables and constraints can be added to the GDRPP formulations: variables y_k, where y_k is equal to one if cluster C_k is serviced consecutively and is otherwise equal to zero, and a group covering constraint for each group as well as constraints connecting the x_a variables to the y_k variables. Alternatively, the following procedure is possible. For each pair a, a' of arcs belonging to a cluster (with $a \neq a'$ for clusters of cardinality greater than one), the costs of an optimal cluster traversal (servicing the arcs in the cluster) starting with ta_a and ending with $he_{a'}$ are computed. Each cluster of each group is replaced by one arc for each possible pair of arcs a, a' (with the corresponding costs). Finally, the tails (heads) of these new arcs are connected with the heads (tails) of the other new arcs by deadheading arcs with costs equal to the costs of a shortest path between the corresponding start and end vertices in the original graph. See Figure 3.8.

3.3.9 Turn Penalties

There is a considerable amount of literature on ARPs with turn penalties, starting with the paper by Caldwell 1961. Other important contributions are Wattleworth/Shuldiner 1963, Kirby 1966, Kirby/Potts 1969, Wyskida/Gupta 1972, Bodin/Kursh 1978, Bodin/Kursh 1979, McBride 1982, Roy/Rousseau 1989, Añez et al. 1996, Benavent/Soler 1999, Bousonville/Kopfer 2000, Clossey et al. 2001, Corberán et al.

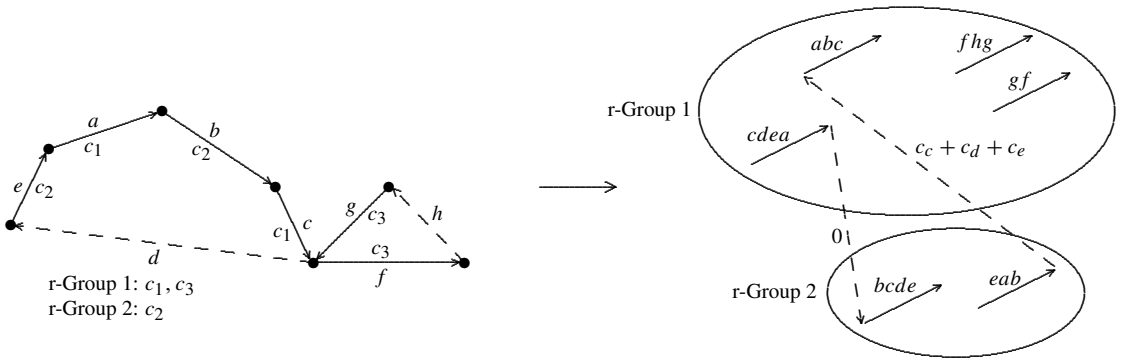


Figure 3.8: Generalized clustered DRPP

2002, Winter 2002, Arkin et al. 2005. Some authors develop solution algorithms working directly on the graphs with turn penalties, others use transformations into arc or vertex routing problems without turn penalties. A detailed review would go beyond the scope of this paper. In the following, two formulations for ARPs with turn penalties and the three basic possibilities of transformations into graphs without turn penalties are presented.

This author is not aware of any publications considering *direct exact* approaches for ARPs with turn penalties. The following two formulations are the first ones to use the concept of *turn variables*.

A possible formulation for the windy rural postman problem with turn penalties (WRPPTP) on a graph $G_{TP} = (V, L, T)$ is as follows:

(WRPPTP1):

$$\sum_{l \in L} (\vec{c}_l \vec{x}_l + \overleftarrow{c}_l \overleftarrow{x}_l) + \sum_{t \in T} pe_t y_t \rightarrow \min \text{ subject to} \tag{3.11a}$$

$$\vec{x}_l + \overleftarrow{x}_l \geqq 1 \quad \forall\, l \in L^R \tag{3.11b}$$

$$\sum_{\{l \in L : he_l = i\}} (\vec{x}_l - \overleftarrow{x}_l) + \sum_{\{l \in L : ta_l = i\}} (\overleftarrow{x}_l - \vec{x}_l) = 0 \quad \forall\, i \in V \tag{3.11c}$$

$$\sum_{\{l \in L : ta_l \in \cup_{q \in S} V_q^{CC} \not\ni he_l\}} \vec{x}_l + \sum_{\{l \in L : ta_l \notin \cup_{q \in S} V_q^{CC} \ni he_l\}} \overleftarrow{x}_l \geqq 1 \quad \forall\, \emptyset \neq S \subsetneqq \{1, \ldots, p_C\}, |S| \leqq \left\lfloor \frac{p_C}{2} \right\rfloor \tag{3.11d}$$

$$\vec{x}_l - \sum_{\{t \in T : il_t = l, tv_t = he_l\}} y_t = 0 \quad \forall\, l \in L \tag{3.11e}$$

$$\vec{x}_l - \sum_{\{t \in T : tv_t = ta_l, ol_t = l\}} y_t = 0 \quad \forall\, l \in L \tag{3.11f}$$

$$\overleftarrow{x}_l - \sum_{\{t \in T : il_t = l, tv_t = ta_l\}} y_t = 0 \quad \forall\, l \in L \tag{3.11g}$$

$$\overleftarrow{x}_l - \sum_{\{t \in T : tv_t = he_l, ol_t = l\}} y_t = 0 \quad \forall\, l \in L \tag{3.11h}$$

$$\vec{x}_l \in \mathbb{Z}_+^0 \quad \forall\, l \in L \tag{3.11i}$$

$$\overleftarrow{x}_l \in \mathbb{Z}_+^0 \quad \forall\, l \in L \tag{3.11j}$$

$$y_t \in \mathbb{Z}_+^0 \quad \forall\, t \in T \tag{3.11k}$$

$\vec{x}_l$ ($\overleftarrow{x}_l$) are *link variables* denoting the number of times link $l \in L$ is traversed from tail to head (from head to tail). y_t are *turn variables* denoting the number of times turn t is performed. The objective function strives to minimize the sum of link traversal costs and turn penalties.

Constraints (3.11b) guarantee covering of the required links. (3.11c) ensure flow conservation. (3.11d) ensure that a link between any subset of the p_C required components and the complement of this subset is used. It would not be sufficient to require this for every single required component, because then subtours would be possible. (3.11e)–(3.11h) perform the coupling between the two types of variable. If a link is traversed in a certain direction, there must be a turn using this link as inlink and a turn using this link as outlink, and these turns must traverse the link in the right direction.

Are subtour elimination constraints for the turns needed, too? Consider the so-called *hourglass graph* depicted in Figure 3.9 (a). If all six depicted arcs are required, there is exactly one closed trail constituting a postman tour. If there are no turn penalties and only arc variables are used, no subtours arise. With turn penalties and turn variables, though, it is also possible to traverse all six arcs when the indicated turns are performed, but this leads to two subtours.

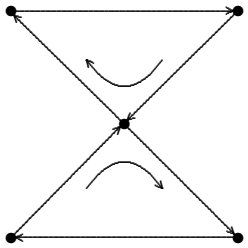


Figure 3.9: Hourglass graph

Constraints that avoid such subtours are

$$\sum_{\{t \in T : il_t \in L(S), ol_t \notin L(S)\}} y_t \geqq 1 \quad \forall\, S \subsetneqq V, L^R(S) \neq \emptyset \tag{3.12}$$

(3.12) require that, for each proper subset S of the vertex set V for which a required link exists whose head and tail vertices are elements of S, there must be a turn leaving S. In conjunction with flow conservation constraints, (3.12) also ensure that, for each such S, there is a turn entering S, too. For subsets of V with only one vertex, the constraints coupling the arc and the turn variables make sure that there is always a turn leaving the respective subset.

However, consider the *inverse hourglass graph* in Figure 3.10. If all arcs are required, performing the indicated turns fulfils constraints (3.12), but again, there are two subtours.

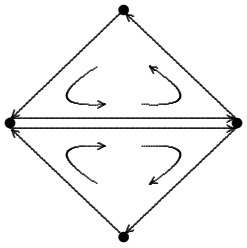


Figure 3.10: Inverse hourglass graph

To prevent such subtours, the following constraints can be used:

$$\sum_{t' \in T'} y_{t'} \geqq 1 \Rightarrow \sum_{\{t \in T : t \notin T', \exists \tilde{t} \in T' \text{ with } ol_{\tilde{t}} = il_t, tv_{\tilde{t}} \neq tv_t\}} y_t \geqq 1 \quad \forall\, \emptyset \neq T' \subsetneqq T, |T'| \leqq \left\lfloor \frac{|T|}{2} \right\rfloor \tag{3.13}$$

(3.13) state that for each non-empty proper subset T' of the set T of turns containing not more than half of all turns, the following must hold: If a turn $t' \in T'$ is performed, another turn t not belonging to T' must be performed, and there must be a turn $\tilde{t} \in T'$ whose outlink is the inlink of t, and whose turnvertex is not the turnvertex of t. Thus, constraints (3.13) must be added to (3.11). Constraints (3.11d) are then redundant.

With constraints (3.13), a formulation using only turn variables can be stated as follows:

(WRPPTP2):

$$\sum_{t \in T} (\tilde{c}_{il_t} + pe_t) y_t \rightarrow \min \text{ subject to} \tag{3.14a}$$

$$\sum_{\{t \in T : il_t = l\}} y_t \geqq 1 \quad \forall\, l \in L^R \tag{3.14b}$$

$$\sum_{\{t \in T : il_t = l, tv_t = he_l\}} y_t - \sum_{\{t \in T : ol_t = l, tv_t = ta_l\}} y_t = 0 \quad \forall\, l \in L \tag{3.14c}$$

$$\sum_{\{t \in T : il_t = l, tv_t = ta_l\}} y_t - \sum_{\{t \in T : ol_t = l, tv_t = he_l\}} y_t = 0 \quad \forall\, l \in L \tag{3.14d}$$

$$\sum_{t' \in T'} y_{t'} \geqq 1 \Rightarrow \sum_{\{t \in T : t \notin T', \exists \tilde{t} \in T' \text{ with } ol_{\tilde{t}} = il_t, tv_{\tilde{t}} \neq tv_t\}} y_t \geqq 1 \quad \forall\, \emptyset \neq T' \subsetneqq T, |T'| \leqq \left\lfloor \frac{|T|}{2} \right\rfloor \tag{3.14e}$$

$$y_t \in \mathbb{Z}_+^0 \quad \forall\, t \in T \tag{3.14f}$$

$\tilde{c}_{il_t}$ in the objective function indicates the costs of traversal of the inlink of t in the direction in which il_t is traversed in t. The meaning of the constraints should be clear from the preceding explanations.

To transform a graph G_{TP} with turn penalties into a graph G without, there are three basic possibilities:

(i) A graph G is created which contains one arc for each possible direction of traversal of each link of G_{TP}. These arcs form the set S. Then, additional non-r-group arcs from the head of each arc $a \in S$ to the tail of each other arc $a' \in S$ are inserted if and only if the corresponding links are incident or identical in G_{TP}, i.e., if and only if there is a turn in G_{TP} with inlink equal to the link corresponding to a, outlink equal to the link corresponding to a', and finite turn penalty. The costs of these additional arcs are set to the corresponding turn penalty. An example (adapted from Grünert/Irnich 2005b, p. 615) is given in Figure 3.11. As can be seen from the figure, an equivalent transformation is to create a digraph D with one vertex for each possible direction of traversal of each link of G_{TP} and one arc for each turn in G_{TP}. The costs of an arc a in D must then include the corresponding costs of traversal of the link in G_{TP} corresponding to he_a in D.
This transformation is henceforth referred to as *no_sfw transformation*.

(ii) A graph G is created which contains one r-group arc for each possible direction of traversal of each required link in G_{TP}, and additional arcs leading from the head of each original r-group arc a to the tail of each original r-group arc a' are inserted. The costs of these additional arcs are set to the costs of the shortest feasible walk from ta_a via a to $he_{a'}$ via a' minus c_a minus $c_{a'}$. (This takes into account the turn penalties.) The first turn penalty of the shortest feasible walk must not be subtracted. To compute such feasible walks, a slightly modified version of the Dijkstra algorithm can be used, see Benavent/Soler 1999.
This transformation is henceforth referred to as *sfw transformation*.

(iii) A graph G is created which contains one vertex for each possible direction of traversal of each link in G_{TP} and one arc for each turn. The tail (head) of an arc in G is the vertex corresponding to the inlink (outlink) of the turn corresponding to the arc (see Figure 4.5 on page 79). Essentially, the formulation (3.14) on G_{TP} is a formulation on G where the y_t are arc variables. The presumably first paper on routing problems with turn penalties (Caldwell 1961) was already proposing such a transformation, and Añez et al. 1996 call the resulting graph G the *dual graph of* G_{TP}.

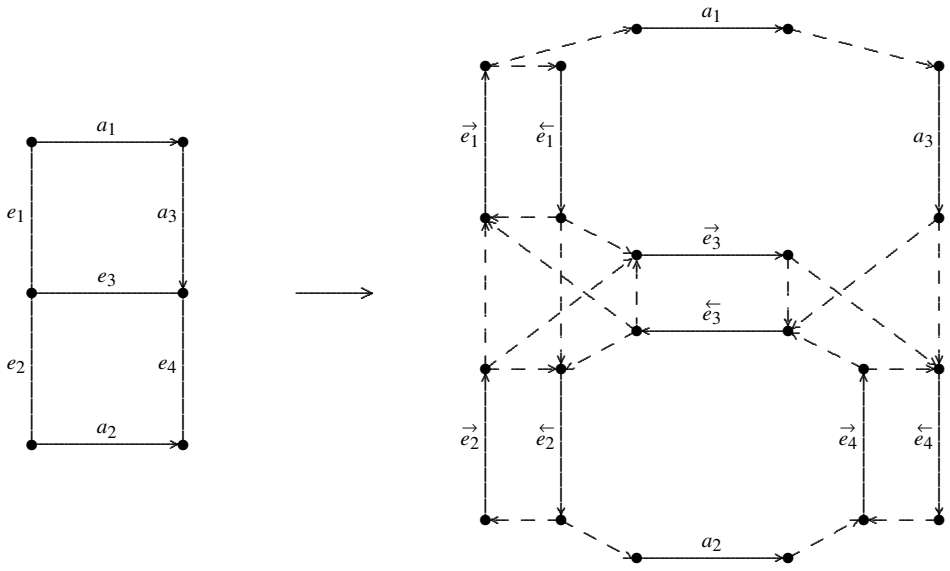


Figure 3.11: Transformation of a graph with turn penalties into one without

3.3.10 Zigzag Service

The issue of zigzag service also arises in the context of postal delivery. Imagine a postman who must deliver mail in a street segment between two road junctions. There are houses on both sides of the street segment. If there is not too much traffic in the segment, the postman has several possibilities for servicing it: He may service one side at a time (and not necessarily both sides contiguously), which means that he must traverse the street segment at least twice, and he may service both sides simultaneously (in 'zigzag mode'), which means that he must traverse the street segment only once. See Figure 3.12.

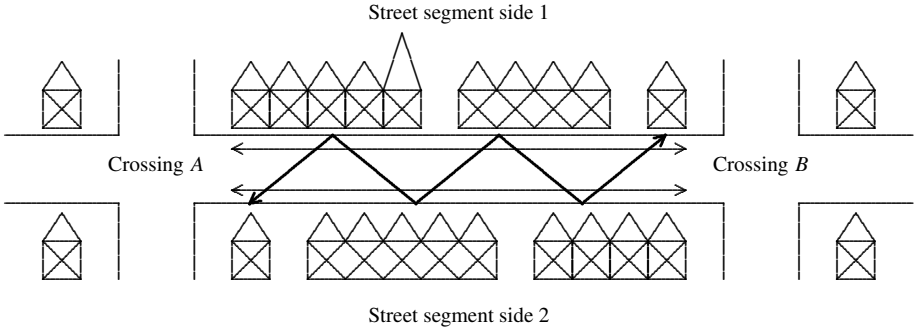


Figure 3.12: Street with zigzag service option

Thus, zigzag service of a required link means that it can be serviced by traversing it once in each direction, by traversing it twice from tail to head, by traversing it twice from head to tail, or by traversing it once in any direction in 'zigzag mode'. Figure 3.13 shows how this can be modelled by two non-disjoint r-groups. For a potential zigzag service street segment, two vertices, eight arcs, and two r-groups are

introduced. There are two non-r-group deadheading arcs, one for each direction, two antiparallel r-group arcs for each street segment side, and two antiparallel r-group arcs for zigzagging. To service the street segment, the postman must either use one of the zigzag arcs or two 'street segment side arcs', one for each street segment side. Grouping the arcs in two non-disjoint r-groups, one for each street segment side, such that the two zigzag arcs belong to the intersection of the r-groups, yields the desired result.

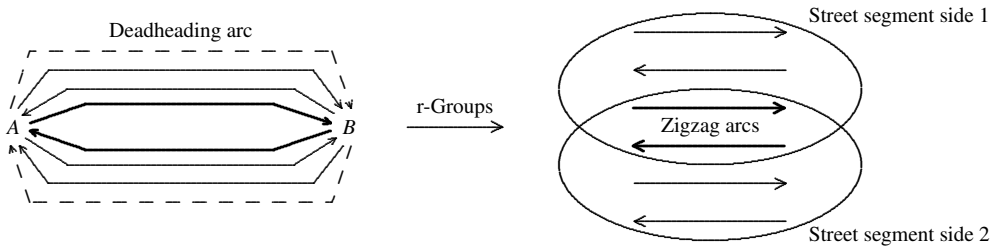


Figure 3.13: Modelling of zigzag service in a GDRPP

An alternative approach for considering zigzag service is proposed by Irnich 2005.

3.3.11 Different Costs for Servicing and Deadheading in Different Directions

An extension recently considered by Lacomme et al. 2004 is the possibility of two different costs per link for traversal with and without servicing. For windy graphs, this must be generalized to four different costs, depending on the direction of traversal and on whether the link is being serviced or not. This, too, can be modelled within a GDRPP, by introducing two service and two deadheading antiparallel arcs for each such windy link, where the two service arcs form one r-group. Note that, for uncapacitated problems, or, to be precise, for problems without resource constraints, on a windy graph $G = (V, L)$, it is necessary to distinguish between costs for traversal with and without service if and only if the condition

$$\vec{c}_l^{\,d} - \vec{c}_l^{\,s} = \overleftarrow{c}_l^{\,d} - \overleftarrow{c}_l^{\,s} \tag{3.15}$$

is violated, where $\vec{c}_l^{\,d}$ and $\overleftarrow{c}_l^{\,d}$ are the costs for deadheading through a link $l \in L$ from tail to head and from head to tail respectively, and where $\vec{c}_l^{\,s}$ and $\overleftarrow{c}_l^{\,s}$ are the costs for servicing l from tail to head and from head to tail. This can be seen as follows. The overall cost contribution of a link l to a postman tour is

$$\vec{c}_l^{\,d}\vec{x}_l^{\,d} + \vec{c}_l^{\,s}\vec{x}_l^{\,s} + \overleftarrow{c}_l^{\,d}\overleftarrow{x}_l^{\,d} + \overleftarrow{c}_l^{\,s}\overleftarrow{x}_l^{\,s}, \tag{3.16}$$

where $\vec{x}_l^{\,d}$ is the number of deadheading traversals of l from tail to head, and where $\vec{x}_l^{\,s}$, $\overleftarrow{x}_l^{\,d}$, and $\overleftarrow{x}_l^{\,s}$ are defined analogously. For fixed values of $\vec{x}_l^{\,d} + \vec{x}_l^{\,s}$ and $\overleftarrow{x}_l^{\,d} + \overleftarrow{x}_l^{\,s}$, assume w.l.o.g. $\vec{x}_l^{\,s} = 1$ and $\overleftarrow{x}_l^{\,s} = 0$. If this is to be changed to $\vec{x}_l^{\,s} = 0$ and $\overleftarrow{x}_l^{\,s} = 1$, (3.16) changes by $\vec{c}_l^{\,d} - \vec{c}_l^{\,s} + \overleftarrow{c}_l^{\,s} - \overleftarrow{c}_l^{\,d}$, which is equal to zero if $\vec{c}_l^{\,d} = \vec{c}_l^{\,s}$ and $\overleftarrow{c}_l^{\,d} = \overleftarrow{c}_l^{\,s}$. If the costs of traversal and of servicing are no longer the same, (3.16) remains unchanged if and only if (3.15) holds.

3.3.12 Complexity Results

After the detailed description of the transformations, some remarks on the difficulty of the considered problems are appropriate, as the interest of transformations from one problem type into another is, in general, limited to $\mathcal{NP}$-hard problems. With the exception of the undirected and directed CPP, for which Edmonds/Johnson 1973 give polynomial algorithms, all considered arc routing problems are $\mathcal{NP}$-hard (Dror 2000a). All considered vertex routing and rural postman problems are even $\mathcal{NP}$-hard in the strong sense. The former is shown in Johnson/Papadimitriou 1985, the latter is easily seen by reducing the STSP to the undirected RPP (URPP): A graph of an STSP instance is polynomially transformed into a graph of a URPP instance by creating a duplicate of each vertex and connecting each original vertex with its duplicate by an edge with costs of zero. The additional edges are required; all original edges are non-required. Moreover, Corberán et al. 2002 prove that postman problems with turn penalties are $\mathcal{NP}$-hard in the strong sense. Consequently, the GDRPP itself is strongly $\mathcal{NP}$-hard as well.

3.4 Solution Approaches

The transformations presented in the previous section show the genericity and flexibility of the GDRPP model, thus suggesting that it is worthwhile to develop good algorithms for it. However, as the GDRPP is a hard problem, heuristics are needed to be able to solve large real-world instances in reasonable time. So, in this section, exact as well as heuristic algorithms are presented.

3.4.1 A Labelling Algorithm for the GDRPP

Noon/Bean 1991 mention several early papers (which were not accessible to this author) that tackle the TSP and the GTSP by dynamic programming algorithms 'in which a state is defined by the sets already visited' (ib., p. 624). The ideas of these authors can be adapted to construct a labelling algorithm for the GDRPP.

GDRPP-Exact-Labelling (GDRPP-EL):
Given: a digraph $D = (V, A)$ defining a GDRPP instance.
Select an r-group of minimal cardinality. Let this r-group be q_1 (w.l.o.g.).
For each arc a^{curr} in q_1:

> Modify D by adding an *artificial destination vertex* d' and an arc from $ta_{a^{curr}}$ to d' with costs of $c_{a^{curr}}$.
>
> Solve an SPPRC on the modified digraph by a labelling algorithm. To this end, introduce a cardinally scaled, unconstrained resource r^{cost} measuring the costs and one nominally scaled binary resource r^q, indicating whether r-group q must still be covered, for each r-group, except for q_1 and all other r-groups covered by a^{curr}. Set the resource windows for all resources at all vertices to $[0, +\infty[$, except for d', where all resource windows for r-group covering are set to $[0, 0]$. Use the following resource variables and resource extension functions: For r^{cost}, use a resource variable σ_i^{cost} at vertex i and the REF $f^{r^{cost}}$ with $f_a^{r^{cost}}(\sigma_{ta_a}^{cost}) = \sigma_{ta_a}^{cost} + c_a$. For r-group q, use a resource variable σ_i^q at vertex i and the REF f^{r^q} with $f_a^{r^q}(\sigma_{ta_a}^q) = \max\{0, \sigma_{ta_a}^q - \delta_a^q\}$, where δ_a^q is equal to one if and only if $a \in A_q$ and is otherwise equal to zero. Start with a first label l_1, resident at $he_{a^{curr}}$, set l_1's cost resource variable to zero and all its other resource variables to one (true). Seek a shortest walk from $he_{a^{curr}}$ to d' using the following dominance procedure: Label l dominates label l' if and only if l has covered all r-groups l' has covered and has not accumulated greater costs than l'. If l has covered an r-group which l' has not and vice versa, the two labels are incomparable. (To avoid two labels dominating each other, the labels can be numbered and the rule can be added that, in case of a tie, the label with the smaller number dominates the other label.)

The solution of the SPPRC with the smallest objective function value yields an optimal solution to the GDRPP instance, when the arc d' is replaced by the corresponding arc a.

It is not sufficient to compute a shortest path from $o := he_a$ to $d := ta_a$ for each arc a from r-group q_1, because, as can be seen in Figure 3.14, there may be instances where it is necessary to visit d before having covered all other r-groups. Therefore, the artificial destination vertex d' is introduced.

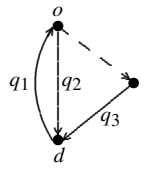


Figure 3.14: GDRPP instance showing necessity of artificial vertex

Note that it is not a good idea to introduce an artificial source and sink vertex, to connect the source with the tails of the r-group arcs, and to connect all heads, i.e., all vertices, with the sink. This is because a postman *tour* is sought, not an open walk. Hence, a label would need to store the first 'real' vertex reached, and at the last vertex before the artificial sink, the shortest path distance back to this first real vertex would have to be added to the current costs. This is not a real difficulty; but what *does* make this approach unfavourable is the fact that the number of labels would increase considerably; in principle, an SPPRC for each tail of each r-group arc would be solved.

For computational purposes, a resource counting the number of already covered r-groups can be added, similar to the proposal of Feillet et al. 2004.

If more than one SPPRC must be solved, the optimal objective function value over all previously solved SPPRCs can be used as a bound in each SPPRC. To use the bound, the value of the cost resource of a label is checked when it is selected for extension. If this value is not strictly less than the bound, the label can be discarded. If the labels to be extended are selected from the set of unprocessed labels in non-decreasing order of their costs, the current SPPRC can be terminated/bounded altogether as soon as a label is selected for extension whose cost resource value exceeds the bound. To implement this strategy in the `r_c_shortest_path` functions, a specialized visitor type can be used which accordingly defines the function for the event point when a label is selected for extension.

Similar to the GTSP, the number of r-groups strongly influences the difficulty of an instance. Therefore, an alternative concerning the selection of the arcs for which an SPPRC is solved is to compute, for each arc, the number of remaining r-groups in the corresponding SPPRC and to select the r-group q where the maximal number of remaining r-groups, taken over all arcs covering q, is minimal. Consider, for example, an instance with ten r-groups, one r-group with two arcs, each of them covering only this r-group, and one r-group with three arcs, each of them covering four r-groups. The alternatives are then to solve two SPPRCs with nine r-groups or three SPPRCs with six r-groups; the latter is clearly preferable. Of course, this argument is of no use when the r-groups are disjoint. For graphs modelling road networks, not much is to be gained.

How can the dominance procedure be implemented? It is natural to store the information on which r-groups must still be covered in a container of Boolean variables. To decide whether a label/container l dominates a label/container l', one possibility is to compare the two containers element by element. Another possibility is based on the following observation: If the number of already covered r-groups is stored, the difference in the number of covered r-groups between l and l' can be computed. If this difference is negative, 'l does not dominate l'' can be returned directly. Otherwise, this difference is updated after each pairwise comparison of the 'covered' values of the two containers for one r-group. If l has already covered some r-group q and l' has not, the difference value is decreased by one. As soon as the updated difference value becomes negative, the procedure is stopped and 'l does not dominate l'' is returned. A theoretical argument as to which procedure should be faster in the average case is as follows. Consider two containers l, l' of size n of Boolean variables storing realizations of independent and identically distributed random variables taking on the values 0 and 1 with probability $\frac{1}{2}$. The expected value of the number of bits set in each container is then $\frac{n}{2}$. The simple dominance check can be stopped at the first index i with $l[i] < l'[i]$. The probability for this event is $\frac{1}{4}$, and therefore, on average, four pairwise comparisons must be made. The probability of having to traverse both sequences until index n equals

$$1 - \sum_{i=0}^{n-2} \left(\frac{3}{4}\right)^i \frac{1}{4}.$$

With the second comparison strategy, the expected value of the difference in the number of covered r-groups is zero. This means that the comparison can be terminated at the first index i with $l[i] < l'[i]$, and also at the first index i' where it is the other way round, because then there must be an index $i'' \neq i'$

with $l[i''] < l'[i'']$. In both cases, l cannot dominate l'. The probability that $l[i] < l'[i]$ or $l[i] > l'[i]$ is $\frac{1}{2}$, and therefore, on average, two pairwise comparisons must be made. The probability of having to traverse both sequences until index n equals

$$1 - \sum_{i=0}^{n-2} \left(\frac{1}{2}\right)^i \frac{1}{2}.$$

However, there is an additional overhead incurred by the more complex checking, as there are two additional operations for each index: updating of the difference value and checking it against zero. The expected number of elementary operations with the first strategy is $4 \cdot 1 = 4$, the expected number with the second strategy is $2 \cdot (1+1+1) = 6$. Consequently, the second strategy does not seem advantageous. Computational experiments have indeed shown that, for the GDRPP, the more sophisticated check is outperformed by the simpler pairwise comparison.

The approach of solving the GDRPP by a labelling algorithm has several advantages:

- The usual case in the literature for routing problems is to assume a non-negative link cost function, so, the SPPRC for the GDRPP need be solved only on a digraph without negative cycles. As explained in Chapter 2, SPPRCs on digraphs with negative cycles are much more difficult to solve. Strictly speaking, on digraphs with negative cycles, no minimal cost postman tour exists. To obtain a tour covering all r-groups, the number of traversals of each arc must be limited. This requires additional resources, and these resources increase the number of possible labels and the number of label incompatibilities.
- Solving the GDRPP by transforming it to a TSP requires adding deadheading arcs corresponding to shortest paths. When solving it as an SPPRC, this is not necessary. Instead, it is possible to operate directly on the original digraph. There is no need for preprocessing (other than adding the artificial destination vertex). More importantly, TSP algorithms always operate on complete graphs, whereas the labelling algorithm benefits if the original graph is sparse (which is the case for graphs representing road networks).
- It is possible to handle non-disjoint r-groups without any modification.
- Some of the transformations described above require the introduction of degenerate cost structures (through the M constants). This might lead to numerical problems for LP-based methods, whereas such cost structures are not critical for labelling algorithms.

The drawback of this approach is that, for dense graphs (e.g., for transformed TSP instances), and, of course, for graphs with many r-groups, there may be very many incompatible labels. In fact, a worst-case analysis of the space and time complexity of GDRPP-EL on a digraph $D = (V, A)$ yields the following results: The number of SPPRCs to be solved is at most $|A|$. In each SPPRC, at each vertex, there can be $|A|(|A|-1)2^{p_A}$ different undominated labels. This is because each time a vertex is reached via the same arc, there will be at least one additional covered r-group, or else the resulting label will be dominated. With each new label, a dominance check is performed by comparing it with every other label resident at the same vertex. Hence, the space complexity of the labelling algorithm is $\mathcal{O}(|V||A|^2 2^{p_A})$, and the time complexity is $\mathcal{O}(|A|^5 2^{2p_A})$, which is exponential—but this was to be expected for a strongly $\mathcal{NP}$-hard problem.

Also, the clustered ATSP can be solved via an SPPRC algorithm. The principle is as follows. A resource is introduced for each cluster C. At each vertex, the resource windows corresponding to a cluster the vertex does not cover are set to $[0, 0]$, and the resource windows corresponding to a cluster the vertex does cover are set to $[0, |C|]$, where $|C|$ is the number of vertices belonging to C. The algorithm is started with a label l at a vertex from an arbitrary cluster C_1 containing at least two vertices, and all of l's resource variables are set to M, except for the resource variable for C_1, which is set to one. When a

vertex of cluster C_i is reached, the variable for C_i is increased by 1, and the variables for all other clusters are set to M. After having reached all vertices in C_i, the variables for all other clusters not yet covered are set to zero.

3.4.2 Heuristics

It is also possible to use the labelling algorithm to solve the GDRPP heuristically. To this end, the check of the covered r-groups in the dominance procedure is omitted. A label l dominates a label l' already if l has either lower costs and covered at least as many r-groups as l' or l has the same costs and covered more r-groups than l'. This simplified dominance procedure makes the algorithm faster by several orders of magnitude: It is then of polynomial complexity. However, the simplified algorithm is no longer exact, because it is then possible that a path is dominated by another path although both cover different r-groups. The dominated path could possibly have covered r-groups which the dominating path still has to cover, and it could have done so in a much more efficient manner, and the dominating path could have covered r-groups which the dominated path still has to visit in a very inefficient manner. Consider, for example, Figure 3.15 (a). The arc a_1 from h to i is the only arc covering r-group q_1. Hence, h is connected with the artificial destination vertex d', and the algorithm seeks a shortest path from i to d' covering all r-groups except for q_1. If the costs of a_3 are higher than the costs of a_2, the label corresponding to the path (i, a_3, j) is dominated by the label corresponding to the path (i, a_2, j). However, in order to cover r-group q_3, the algorithm has to loop once around the digraph to yield a label corresponding to the path $(i, a_2, j, a_4, k, a_5, h, a_1, i)$. The path $(i, a_2, j, a_4, k, a_5, h, a_1, i, a_3, j)$ will then not be dominated, because it has covered more r-groups than any other path for which there is a label at vertex j. The solution will be the path $(i, a_2, j, a_4, k, a_5, h, a_1, i, a_3, j, a_4, k, a_5, h, a', d')$, although the evident optimal solution is $(i, a_3, j, a_4, k, a_5, h, a', d')$. Moreover, the algorithm may even fail altogether, i.e., may not return a feasible solution although one exists. This may occur when, at a certain vertex, each label that has covered a certain r-group is dominated by another label that has not covered this r-group. The algorithm will then terminate without having created a label/path covering all r-groups, and, hence, without having reached d'. An example is provided in Figure 3.15 (b). The labels corresponding to the paths $(o, a_1, h, a_3, i, a_5, j)$ and $(o, a_1, h, a_3, i, a_4, o, a_2, j)$, which cover q_2, are dominated by the label corresponding to the path $(o, a_2, j, a_6, k, a_7, j)$, which covers q_3, as all paths cover one r-group, but the third one is the shortest. On the other hand, the label corresponding to the path $(o, a_2, j, a_6, k, a_7, j, a_6, k, a_8, o)$, which also covers q_3, is dominated by the path $(o, a_1, h, a_3, i, a_4, o)$ for the same reason.

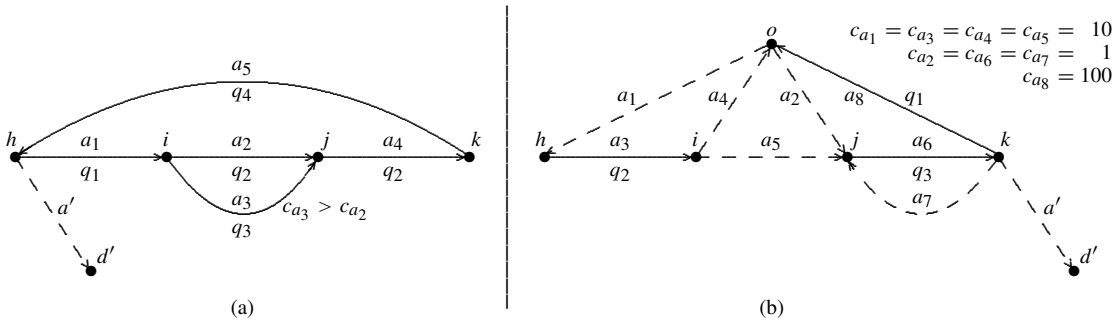


Figure 3.15: GDRPP-HL (a) is a heuristic and (b) may fail

In addition to the heuristic labelling algorithm (henceforth called GDRPP-HL), the following generic constructive heuristic is proposed. Contrary to GDRPP-HL, it guarantees to compute a feasible solution for any feasible instance.

GDRPP-Constructive (GDRPP-C):
Given: a digraph defining a GDRPP instance.
Compute the shortest paths from the head of each r-group arc to the tail of each r-group arc.
Initiate the solution cycle with a trail consisting of one r-group arc which is a shortest one among all r-group arcs covering the highest number of r-groups.
Mark all r-groups covered by this arc as covered.
While not all r-groups are covered:

Select an arc a.

Extend the current trail by adding (in the right sequence) the arcs on the shortest path from the head of the last arc in the current trail to the tail of a and the arc a itself.

Mark all previously uncovered r-groups covered by one of the newly added arcs as covered.

To obtain a cycle, extend the current trail by adding the arcs on the shortest path from the head of the last arc added to the tail of the first arc added.

Two different criteria for the arc selection step are proposed:

GDRPP-Nearest-Neighbour (GDRPP-NN):
Select an arc a covering the highest number of not yet covered r-groups among all *closest* r-group arcs. a is a closest one if the costs of the shortest path from the head of the last arc in the current trail to the tail of a plus c_a are minimal over all r-group arcs covering a not yet covered r-group.

GDRPP-Best-Covering (GDRPP-BC):
Select a closest arc a among all r-group arcs covering the highest number of not yet covered r-groups.

For disjoint r-groups, both heuristics yield the same solution (if both are coded in the same way with respect to ordering, tie-breaking etc.).

The following two improvement heuristics are proposed:

GDRPP-Swap-Arcs (GDRPP-SA):
Given: a cycle constituting a feasible GDRPP solution.
For each r-group arc a:

Remove a at one of its *most isolated* positions. Remove here means that the arcs between the head of the preceding r-group arc and the tail of the succeeding r-group arc are replaced by the arcs of a shortest path between these two vertices. A position of a is a most isolated one if the distance in the cycle from the head of the preceding r-group arc to the tail of the succeeding r-group arc is maximal over all appearances of a in the cycle.

For each r-group q that a covers:

If there is no other arc that covers q:

Insert into the cycle, at a *best possible position* between two successive r-group arcs a_1, a_2, another arc a' that covers q. Insert here means that all non-r-group arcs (if any) between a_1 and a_2 are replaced by the arcs of a shortest path from the head of a_1 to the tail of a' via a' itself to the tail of a_2. A best possible position is one leading to the lowest increase in the costs of the cycle.

If the resulting cycle is shorter than the current best one, replace the current best one by the resulting cycle.

GDRPP-Swap-r-Groups (GDRPP-SrG):
Given: a cycle constituting a feasible GDRPP solution.
For each r-group q:
Remove all arcs covering q. (Remove is defined as above.)
Insert into the cycle, at a best possible position, an arc that covers q. (Insert and best possible position are defined as above.)
If the resulting cycle is shorter than the current best one, replace the current best one by the resulting cycle (although, if there are non-disjoint r-groups, the resulting cycle may not cover all r-groups).
For each r-group q:
If q is not covered in the current cycle:
Insert into the cycle, at a best possible position, an arc covering q.

Note that the two improvement procedures as described above perform a 'tightened first improvement search': Each improving solution immediately becomes the new incumbent solution, and the search based on a new incumbent solution is continued with the next candidate object for removal (arc or r-group) instead of starting the examination of each new incumbent with the first candidate object. However, the removed objects are re-inserted at a *best* possible position. Nevertheless, the two procedures sacrifice solution quality in favour of execution speed.

3.4.3 A Branch-and-Cut Algorithm for the GDRPP

There are many papers that consider branch-and-cut approaches for uncapacitated ARPs. An overview is given in Eglese/Letchford 2000 and Benavent et al. 2000.

Computational experiments were performed with a simple branch-and-cut algorithm for formulation (3.6). The LP relaxation was initially solved without the subtour elimination constraints (3.6e).

3.4.3.1 Valid Inequalities

A class of simple static cuts which are obviously valid is formed by the so-called δ_i **Cuts**:

$$\sum_{i \in HE_q} \delta_i \geqq 1 \quad \forall\, q \in \{1, \ldots, p_A\}. \tag{3.17}$$

These constraints require that, for each r-group q, the number of visited vertices which are heads of an arc in A_q must be greater than or equal to one. (*Static* cuts are added to the formulation at the root vertex of the branch-and-bound tree at the beginning of the algorithm. *Dynamic* cuts are separated in the course of the algorithm.)

3.4.3.2 Branching and Enumeration Strategies

The default branching strategy of the employed branch-and-cut framework (automatic selection of branching variable) was used. As enumeration strategy, 'best-bound' was used. This strategy chooses the vertex in the branch-and-bound tree with the best objective function value of the associated LP relaxation.

3.4.3.3 Upper Bounding

The heuristics described in Section 3.4.2 were used to compute upper bounds. All heuristics run quite fast, so at the beginning of the branch-and-cut algorithm, all construction heuristics were used to compute a feasible initial solution. The best solution was then used as input to GDRPP-SA. The resulting solution was then used as input to GDRPP-SrG, and the objective function value of the solution output by GDRPP-SrG was used as an upper bound.

3.5 Computational Experiments

The algorithms described in Section 3.4 were implemented in C++ using the Boost Graph library (`boost.org`). The heuristic and the exact labelling algorithm used the `r_c_shortest_paths` framework. The branch-and-cut algorithm was implemented with ILOG Concert Technology (`www.ilog.com/products/optimization/tech/concert.cfm`), which is included as part of ILOG CPLEX (`www.ilog.com/products/cplex`).

3.5.1 Test Instances

Three classes of random test instances were created. Each class consists of several types differing with respect to the number of vertices, arcs, and r-groups.

The first class consists of random GDRPP instances. The characteristics of the created instance types are shown in Table 3.1.

No. of created instances	No. of vertices	No. of arcs	No. of r-groups	Max. % r-group arcs covering one r-group	Max. % r-groups covered by one arc
10	50	500	10	2	10
10	50	500	15	2	10
10	50	500	20	2	10
30	50	500	25	2	10
30	50	500	50	2	5
30	100	1,000	100	1	4

Table 3.1: Characteristics of created random GDRPP instances

For all types, n instances with disjoint and n instances with not necessarily disjoint r-groups were created, where n is the value in the first column of Table 3.1. The entries in the last two columns refer only to the instances with not necessarily disjoint r-groups. The instances with 10, 15, and 20 r-groups were created by deleting 15, 10 and 5 r-groups from the instances with 25 r-groups. All instances had 50 % r-group arcs. The arc costs were selected randomly from [1; 100].

The second class consists of transformed WRPPTP instances. Both the no_sfw and the sfw transformation were used. Three types of WRPPTP instance were created: 10 vertices and 40 links, 20 vertices and 80 links, and 30 vertices and 120 links. For each type, 30 instances were created. Each instance had 50 % required links. 50 % of the links were symmetric edges, and 25 % of the links were arcs. The link costs were selected randomly from [1; 100], and the turn penalties were selected randomly from [0; 100]. 3 % of the turns were forbidden.

The third class consists of transformed instances of windy rural postman problems with zigzag service (WRPPZZ) with different costs for servicing and deadheading, so that (3.15) was violated. Three types of WRPPZZ instance were created: 30 vertices and 100 links, 40 vertices and 150 links, and 50 vertices and 200 links. For each type, 30 instances were created. Each instance had 50 % required links. 50 % of the links were symmetric edges, 25 % of the links were arcs, and 25 % of the links were zigzag links. The deadheading costs were selected randomly from [1; 100]. The costs for one-sided service were computed by multiplying the deadheading costs by a random factor of between 1 and 2. The costs for zigzag service were computed by summing up the costs for the two one-sided services of the respective link in the respective direction and by multiplying the sum by a random factor of between 1 and 2.

As mentioned above, the procedure for the separation of violated subtour elimination constraints (3.6e) does not work on graphs without head-disjoint r-groups. For this reason, a head-disjoint r-group was

added to all test instances without prior testing for whether an instance already contained such an r-group.

3.5.2 System Parameters

The system parameters used in the computational experiments are shown in the following table. All CPLEX parameters not shown in the table were at their default values. The selection of the algorithm for solving the LP relaxations was left to CPLEX. The maximum flow problems for the separation of the subtour elimination constraints were solved with the Edmonds-Karp algorithm provided by the BGL.

Parameter	Setting
CPU frequency	1 GHz
Main memory	1 GB
CPLEX version	9.1
Concert Technology version	2.1
Wall-clock time limit for branch-and-cut	3,900 seconds
Min. violation of subtour elimination constraints to be considered violated	10^{-3}
`IloCplex::EpAGap`	10^{-4}
`IloCplex::EpInt`	10^{-4}
`IloCplex::CutLo`	-10^{75}
`IloCplex::EpGap`	10^{-4}
`IloCplex::ObjDif`	0
`IloCplex::RelObjDif`	0
`IloCplex::CutUp`	10^{75}

Table 3.2: System parameters GDRPP

Due to the used operating system, all reported running times (in this chapter as well as in the chapters on the VRPTT and the TTRP) are wall-clock times and, hence, are subject to considerable variations. However, as a high number of instances was used for each instance class and type, and as all experiments were performed on a 'dedicated' PC that was not used otherwise during the experiments, the variations should be quite balanced on average.

3.5.3 Computational Results

In the following tables, the random GDRPP instance types are denoted $v\_a\_r\_s$, where v indicates the number of vertices, a indicates the number of arcs, r indicates the number of r-groups, and $s \in \{y, n\}$, where y means that the r-groups are disjoint, and n means that they are not necessarily disjoint. The transformed WRPPTP instances are denoted v_a_s, where v denotes the number of vertices in the WRPPTP instances, a denotes the number of arcs in the WRPPTP instances, and $s \in \{no\_sfw, sfw\}$ denotes the type of transformation. The transformed WRPPZZ instances are denoted v_a, where v and a denote the number of vertices and arcs respectively in the WRPPZZ instances. Here and in the following chapters, table entries of the form x / y / z denote the minimal, average, and maximal value respectively.

3.5.3.1 Results for the Exact Labelling Algorithm

The complexity analysis of the exact labelling algorithm in section 3.4.1 has shown that the algorithm's time and space complexity are determined by the number of r-groups. As was to be expected from this analysis, GDRPP-EL was able to solve only the smallest instances, i.e., those with the smallest number of r-groups. The following table shows how the running times of GDRPP-EL compare with those of the

branch-and-cut algorithm. The rows with the heading 'Rank time' indicate the rank of the respective procedure in a ranking of running time over all procedures. For example, 1/2.3/4 means that for at least one instance, no other procedure was faster; the average rank over all instances is 2.3, and for all instances, the procedure was at least the fourth-fastest.

		Branch-and-cut algorithm			
	GDRPP-EL	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
50_500_10_n					
Running time [s]	20 / 69 / 137	0 / 53 / 257	0 / 40 / 261	0 / 44 / 146	0 / 26 / 125
Rank time	3 / 4.2 / 5	1 / 2.7 / 5	1 / 2.1 / 5	2 / 3.3 / 5	1 / 2.7 / 4
50_500_10_y					
Running time [s]	20 / 47 / 89	0 / 32 / 260	0 / 19 / 130	0 / 38 / 263	0 / 23 / 188
Rank time	1 / 4.5 / 5	1 / 2.5 / 4	1 / 2.1 / 4	1 / 3.0 / 5	1 / 2.5 / 4
50_500_15_n					
Running time [s]	774 / 8,161 / 24,733	0 / 22 / 125	0 / 192 / 753	0 / 61 / 319	0 / 99 / 601
Rank time	5 / 5.0 / 5	1 / 1.6 / 3	1 / 2.7 / 4	1 / 2.5 / 4	1 / 3.2 / 4
50_500_15_y					
Running time [s]	1,326 / 10,231 / 20,254	0 / 6 / 22	0 / 10 / 29	0 / 6 / 15	0 / 10 / 29
Rank time	5 / 5.0 / 5	1 / 2.0 / 4	1 / 2.4 / 3	1 / 2.3 / 4	2 / 3.2 / 4
All 40 above					
Running time [s]	20 / 4,627 / 24,733	0 / 28 / 260	0 / 65 / 753	0 / 37 / 319	0 / 40 / 601
Rank time	1 / 4.7 / 5	1 / 2.2 / 5	1 / 2.3 / 5	1 / 2.8 / 5	1 / 2.9 / 4

Table 3.3: Comparison of running times GDRPP-EL vs. branch-and-cut (min. / avg. / max.)

3.5.3.2 Results for the Heuristics

The following tables present the computational results obtained with the heuristics. The line headings in the tables have the following meanings:

- % Solution above LB: percentage by which objective function value obtained with heuristic (OFH) exceeds optimal objective function value or best lower bound as obtained with branch-and-cut algorithm (BLB): $(OFH - BLB)/BLB \cdot 100$ (see Section 3.5.3.3 for the number of instances of each type that were solved optimally)
- % Improvement: percentage by which objective function value of improved solution (OFI) is below objective function value of best solution before improvement (BBI): $(BBI - OFI)/BBI \cdot 100$
- No. of times best: number of times no other heuristic yielded a better objective function value
- Rank quality: rank of heuristic in ranking of solution quality over all heuristics; example: 1/2.3/4 means that, for at least one instance, no other heuristic yielded a better objective function value; the average rank over all instances is 2.3, and for all instances, the objective function value obtained with the heuristic was at least the fourth-best
- % Time above fastest: percentage by which running time of heuristic (RTH) exceeds running time of fastest heuristic (RTF) (by instance): $(RTH - RTF)/RTF \cdot 100$
- No. of times fastest: number of times no other heuristic needed less computation time

The column headings indicate the sequence in which the constructive and improvement heuristics were applied; for example, 'GDRPP-NN-SA-SrG' means that GDRPP-NN was used as a constructive heuristic, GDRPP-SA was used to improve the solution obtained with GDRPP-NN, and GDRPP-SrG was used to improve the solution obtained with GDRPP-SA.

In the following tables, computational results are displayed only for the largest instance types of each class (as a representative of the respective class) and for all 510 instances together. In Table 3.7, 'GDRPP-All' means that all construction heuristics are run and a solution with the best objective function value is used as input to the improvement procedures.

Heuristic	GDRPP-NN	GDRPP-NN-SA	GDRPP-NN-SrG	GDRPP-NN-SA-SrG	GDRPP-NN-SrG-SA
100_1,000_100_n					
% Solution above LB	40 / 50 / 64	40 / 49 / 60	40 / 50 / 64	40 / 49 / 60	40 / 49 / 60
% Improvement	0 / 0 / 0	0 / 1 / 5	0 / 0 / 0	0 / 1 / 5	0 / 1 / 5
No. of times best	0	0	0	0	0
Rank quality	8 / 12.9 / 16	8 / 9.8 / 13	8 / 12.9 / 16	8 / 9.8 / 13	8 / 9.8 / 13
% Time above fastest	28 / 31 / 35	36 / 41 / 74	33 / 36 / 39	42 / 83 / 182	42 / 46 / 50
No. of times fastest	0	0	0	0	0
Rank time	4 / 5.4 / 6	6 / 7.6 / 10	5 / 6.4 / 7	8 / 9.3 / 10	7 / 8.4 / 10
100_1,000_100_y					
% Solution above LB	28 / 36 / 44	27 / 35 / 43	28 / 36 / 44	27 / 35 / 43	27 / 35 / 43
% Improvement	0 / 0 / 0	0 / 0 / 1	0 / 0 / 0	0 / 0 / 1	0 / 0 / 1
No. of times best	0	0	0	0	0
Rank quality	8 / 10.4 / 14	8 / 8.0 / 8	8 / 10.4 / 14	8 / 8.0 / 8	8 / 8.0 / 8
% Time above fastest	29 / 45 / 52	43 / 53 / 69	32 / 53 / 66	48 / 68 / 131	40 / 57 / 70
No. of times fastest	0	0	0	0	0
Rank time	4 / 5.4 / 6	6 / 7.1 / 8	5 / 7.0 / 9	9 / 9.4 / 10	7 / 8.1 / 10
30_120_no_sfw					
% Solution above LB	20 / 33 / 47	20 / 32 / 47	20 / 33 / 47	20 / 32 / 47	20 / 32 / 47
% Improvement	0 / 0 / 0	0 / 0 / 4	0 / 0 / 1	0 / 0 / 4	0 / 0 / 4
No. of times best	3	5	3	5	5
Rank quality	1 / 9.9 / 14	1 / 7.0 / 14	1 / 9.7 / 14	1 / 6.8 / 8	1 / 6.8 / 8
% Time above fastest	0 / 0 / 1	1 / 3 / 12	0 / 4 / 43	0 / 5 / 24	1 / 5 / 12
No. of times fastest	22	0	8	0	0
Rank time	1 / 1.3 / 2	2 / 5.8 / 10	1 / 4.5 / 10	2 / 5.8 / 10	6 / 8.2 / 10
30_120_sfw					
% Solution above LB	20 / 32 / 47	15 / 24 / 34	13 / 24 / 34	11 / 22 / 31	13 / 22 / 33
% Improvement	0 / 0 / 0	1 / 6 / 10	1 / 6 / 9	1 / 7 / 11	1 / 7 / 14
No. of times best	0	0	2	2	4
Rank quality	15 / 15.9 / 16	5 / 11.3 / 14	1 / 10.9 / 14	1 / 8.1 / 12	1 / 7.8 / 12
% Time above fastest	0 / 0 / 3	1 / 2 / 4	0 / 4 / 26	0 / 1 / 3	1 / 7 / 11
No. of times fastest	28	0	1	1	0
Rank time	1 / 1.2 / 5	2 / 3.4 / 4	1 / 3.7 / 10	1 / 2.3 / 4	3 / 5.3 / 9
50_200					
% Solution above LB	12 / 15 / 19	6 / 10 / 12	7 / 10 / 12	5 / 8 / 10	6 / 8 / 11
% Improvement	0 / 0 / 0	3 / 5 / 7	3 / 5 / 7	4 / 6 / 8	4 / 6 / 10
No. of times best	0	0	0	0	1
Rank quality	15 / 16.0 / 16	8 / 12.1 / 15	9 / 12.5 / 15	6 / 9.0 / 12	1 / 7.9 / 13
% Time above fastest	42 / 53 / 66	82 / 104 / 131	79 / 97 / 109	120 / 152 / 210	121 / 142 / 159
No. of times fastest	0	0	0	0	0
Rank time	2 / 3.0 / 4	6 / 7.5 / 8	6 / 6.8 / 8	8 / 9.6 / 10	8 / 9.2 / 10
All 510 instances					
% Solution above LB	5 / 35 / 85	5 / 30 / 83	3 / 31 / 84	2 / 29 / 83	0 / 29 / 83
% Improvement	0 / 0 / 0	0 / 4 / 26	0 / 3 / 24	0 / 4 / 29	0 / 4 / 24
No. of times best	12	22	15	37	41
Rank quality	1 / 13.2 / 17	1 / 9.4 / 15	1 / 11.3 / 16	1 / 8.0 / 14	1 / 8.0 / 15
% Time above fastest	0 / 24 / 66	0 / 36 / 143	0 / 34 / 113	0 / 50 / 219	0 / 46 / 304
No. of times fastest	119	12	78	18	6
Rank time	1 / 3.7 / 10	1 / 6.2 / 10	1 / 5.3 / 17	1 / 7.2 / 12	1 / 7.5 / 17

Table 3.4: Computational results for GDRPP-NN heuristics (min. / avg. / max.)

Heuristic	GDRPP-BC	GDRPP-BC-SA	GDRPP-BC-SrG	GDRPP-BC-SA-SrG	GDRPP-BC-SrG-SA
100_1,000_100_n					
% Solution above LB	36 / 52 / 70	34 / 51 / 67	36 / 52 / 70	34 / 51 / 67	34 / 51 / 67
% Improvement	0 / 0 / 0	0 / 1 / 3	0 / 0 / 0	0 / 1 / 3	0 / 1 / 3
No. of times best	0	0	0	0	0
Rank quality	8 / 13.7 / 16	8 / 10.7 / 13	8 / 13.7 / 16	8 / 10.7 / 13	8 / 10.7 / 13
% Time above fastest	0 / 0 / 0	4 / 7 / 10	7 / 85 / 180	9 / 16 / 30	12 / 20 / 48
No. of times fastest	30	0	0	0	0
Rank time	1 / 1.0 / 1	2 / 2.1 / 3	2 / 6.7 / 10	3 / 3.4 / 5	3 / 4.4 / 8
100_1,000_100_y					
% Solution above LB	28 / 36 / 44	27 / 35 / 43	28 / 36 / 44	27 / 35 / 43	27 / 35 / 43
% Improvement	0 / 0 / 0	0 / 0 / 1	0 / 0 / 0	0 / 0 / 1	0 / 0 / 1
No. of times best	0	0	0	0	0
Rank quality	8 / 10.4 / 14	8 / 8.0 / 8	8 / 10.4 / 14	8 / 8.0 / 8	8 / 8.0 / 8
% Time above fastest	0 / 3 / 10	0 / 6 / 28	0 / 89 / 256	0 / 15 / 66	2 / 13 / 37
No. of times fastest	16	12	2	2	0
Rank time	1 / 1.8 / 4	1 / 1.9 / 5	1 / 7.1 / 10	1 / 3.5 / 8	2 / 3.4 / 5
30_120_no_sfw					
% Solution above LB	20 / 33 / 47	20 / 32 / 47	20 / 33 / 47	20 / 32 / 47	20 / 32 / 47
% Improvement	0 / 0 / 0	0 / 0 / 4	0 / 0 / 1	0 / 0 / 4	0 / 0 / 4
No. of times best	3	5	3	5	5
Rank quality	1 / 9.9 / 14	1 / 7.0 / 14	1 / 9.7 / 14	1 / 6.8 / 8	1 / 6.8 / 8
% Time above fastest	1 / 2 / 5	1 / 2 / 5	1 / 2 / 9	1 / 4 / 8	2 / 5 / 18
No. of times fastest	0	0	0	0	0
Rank time	3 / 5.1 / 9	2 / 4.0 / 10	2 / 4.5 / 9	4 / 7.5 / 10	6 / 7.9 / 10
30_120_sfw					
% Solution above LB	20 / 32 / 47	15 / 24 / 34	13 / 24 / 34	11 / 22 / 31	13 / 22 / 33
% Improvement	0 / 0 / 0	1 / 6 / 10	1 / 6 / 9	1 / 7 / 11	1 / 7 / 14
No. of times best	0	0	2	2	4
Rank quality	15 / 15.9 / 16	5 / 11.3 / 14	1 / 10.9 / 14	1 / 8.1 / 12	1 / 7.8 / 12
% Time above fastest	9 / 11 / 25	9 / 9 / 9	9 / 10 / 17	9 / 11 / 20	10 / 12 / 19
No. of times fastest	0	0	0	0	0
Rank time	6 / 8.0 / 10	5 / 5.9 / 7	5 / 7.1 / 10	7 / 8.4 / 10	8 / 9.5 / 10
50_200					
% Solution above LB	25 / 32 / 39	7 / 10 / 14	7 / 11 / 14	6 / 9 / 13	6 / 8 / 11
% Improvement	0 / 0 / 0	12 / 17 / 22	11 / 16 / 20	13 / 18 / 23	14 / 18 / 22
No. of times best	0	0	0	0	0
Rank quality	17 / 17.0 / 17	9 / 13.4 / 15	10 / 14.1 / 15	4 / 9.4 / 14	3 / 8.5 / 13
% Time above fastest	0 / 0 / 0	37 / 46 / 70	40 / 104 / 479	79 / 88 / 102	78 / 89 / 99
No. of times fastest	30	0	0	0	0
Rank time	1 / 1.0 / 1	2 / 2.2 / 4	2 / 5.2 / 10	4 / 4.9 / 6	4 / 5.3 / 6
All 510 instances					
% Solution above LB	6 / 39 / 109	0 / 30 / 92	4 / 32 / 94	0 / 29 / 92	0 / 30 / 92
% Improvement	0 / 0 / 0	0 / 6 / 42	0 / 5 / 44	0 / 7 / 42	0 / 7 / 44
No. of times best	12	24	16	45	47
Rank quality	1 / 13.7 / 17	1 / 9.8 / 16	1 / 11.8 / 16	1 / 8.2 / 15	1 / 8.2 / 15
% Time above fastest	0 / 3 / 94	0 / 13 / 70	0 / 54 / 696	0 / 26 / 215	0 / 25 / 135
No. of times fastest	287	26	22	10	3
Rank time	1 / 3.2 / 10	1 / 3.8 / 10	1 / 5.4 / 17	1 / 5.4 / 17	1 / 5.8 / 10

Table 3.5: Computational results for GDRPP-BC heuristics (min. / avg. / max.)

Heuristic	GDRPP-HL	GDRPP-HL-SA	GDRPP-HL-SrG	GDRPP-HL-SA-SrG	GDRPP-HL-SrG-SA
100_1,000_100_n					
% Solution above LB	12 / 19 / 23	12 / 19 / 23	12 / 19 / 23	12 / 19 / 23	12 / 19 / 23
% Improvement	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times best	27	30	27	30	30
Rank quality	1 / 1.5 / 6	1 / 1.0 / 1	1 / 1.5 / 6	1 / 1.0 / 1	1 / 1.0 / 1
% Time above fastest	373 / 541 / 897	343 / 557 / 1,001	379 / 534 / 897	382 / 545 / 899	372 / 524 / 877
No. of times fastest	0	0	0	0	0
Rank time	11 / 12.8 / 15	12 / 14.1 / 17	12 / 12.8 / 14	13 / 14.2 / 15	11 / 11.1 / 12
100_1,000_100_y					
% Solution above LB	12 / 16 / 21	12 / 16 / 20	12 / 16 / 21	12 / 16 / 20	12 / 16 / 20
% Improvement	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times best	27	30	27	30	30
Rank quality	1 / 1.5 / 6	1 / 1.0 / 1	1 / 1.5 / 6	1 / 1.0 / 1	1 / 1.0 / 1
% Time above fastest	315 / 426 / 820	316 / 434 / 836	329 / 434 / 853	321 / 452 / 862	311 / 422 / 797
No. of times fastest	0	0	0	0	0
Rank time	11 / 12.2 / 14	11 / 13.2 / 15	12 / 13.5 / 15	14 / 14.7 / 15	11 / 11.2 / 14
30_120_no_sfw					
% Solution above LB	13 / 22 / 38	13 / 22 / 38	13 / 22 / 38	13 / 22 / 38	13 / 22 / 38
% Improvement	0 / 0 / 0	0 / 0 / 2	0 / 0 / 0	0 / 0 / 2	0 / 0 / 2
No. of times best	6	29	6	29	29
Rank quality	1 / 5.5 / 16	1 / 1.4 / 13	1 / 5.5 / 16	1 / 1.4 / 13	1 / 1.4 / 13
% Time above fastest	40 / 70 / 157	43 / 78 / 173	44 / 73 / 173	41 / 77 / 176	38 / 68 / 158
No. of times fastest	0	0	0	0	0
Rank time	12 / 13.5 / 15	13 / 15.9 / 17	12 / 15.1 / 17	13 / 16.1 / 17	11 / 11.7 / 15
30_120_sfw					
% Solution above LB	12 / 21 / 36	11 / 18 / 25	11 / 17 / 25	10 / 17 / 23	11 / 16 / 23
% Improvement	0 / 0 / 0	0 / 3 / 10	0 / 3 / 8	0 / 3 / 10	0 / 4 / 13
No. of times best	1	4	14	13	19
Rank quality	1 / 9.5 / 15	1 / 5.3 / 12	1 / 3.8 / 14	1 / 3.1 / 9	1 / 2.3 / 11
% Time above fastest	198 / 243 / 334	196 / 249 / 337	196 / 247 / 330	196 / 255 / 357	191 / 235 / 322
No. of times fastest	0	0	0	0	0
Rank time	12 / 13.4 / 15	12 / 14.0 / 17	12 / 14.7 / 17	12 / 15.7 / 17	11 / 11.2 / 13
50_200					
% Solution above LB	6 / 9 / 13	5 / 7 / 9	5 / 7 / 10	4 / 6 / 9	4 / 6 / 8
% Improvement	0 / 0 / 0	1 / 2 / 4	1 / 2 / 3	1 / 3 / 5	1 / 3 / 5
No. of times best	0	0	0	12	18
Rank quality	7 / 10.5 / 15	3 / 5.4 / 9	3 / 5.9 / 11	1 / 2.5 / 6	1 / 1.9 / 4
% Time above fastest	1,667 / 1,943 / 2,325	1,697 / 1,998 / 2,424	1,701 / 2,146 / 3,818	1,733 / 2,073 / 2,562	1,764 / 1,969 / 2,351
No. of times fastest	0	0	0	0	0
Rank time	11 / 11.3 / 13	11 / 12.9 / 17	13 / 14.5 / 17	14 / 15.1 / 17	12 / 12.4 / 15
All 510 instances					
% Solution above LB	1 / 16 / 44	1 / 14 / 44	0 / 14 / 44	0 / 14 / 44	0 / 14 / 44
% Improvement	0 / 0 / 0	0 / 1 / 12	0 / 1 / 10	0 / 2 / 13	0 / 2 / 13
No. of times best	224	346	260	398	423
Rank quality	1 / 5.7 / 17	1 / 2.8 / 14	1 / 3.8 / 16	1 / 1.8 / 13	1 / 1.6 / 15
% Time above fastest	17 / 516 / 2,325	17 / 525 / 2,424	18 / 537 / 3,818	20 / 547 / 2,562	17 / 514 / 2,351
No. of times fastest	0	0	0	0	0
Rank time	10 / 12.7 / 17	10 / 13.6 / 17	10 / 13.9 / 17	11 / 14.4 / 17	10 / 11.8 / 17

Table 3.6: Computational results for GDRPP-HL heuristics (min. / avg. / max.)

Heuristic	GDRPP-All-SA-SrG	GDRPP-All-SrG-SA
100_1,000_100_n		
% Solution above LB	12 / 19 / 23	12 / 19 / 23
% Improvement	0 / 0 / 0	0 / 0 / 0
No. of times best	30	30
Rank quality	1 / 1.0 / 1	1 / 1.0 / 1
% Time above fastest	461 / 615 / 972	459 / 613 / 968
No. of times fastest	0	0
Rank time	16 / 16.7 / 17	15 / 16.1 / 17
100_1,000_100_y		
% Solution above LB	12 / 16 / 20	12 / 16 / 20
% Improvement	0 / 0 / 0	0 / 0 / 0
No. of times best	30	30
Rank quality	1 / 1.0 / 1	1 / 1.0 / 1
% Time above fastest	418 / 541 / 920	417 / 541 / 916
No. of times fastest	0	0
Rank time	16 / 16.7 / 17	16 / 16.2 / 17
30_120_no_sfw		
% Solution above LB	13 / 22 / 38	13 / 22 / 38
% Improvement	0 / 0 / 2	0 / 0 / 2
No. of times best	30	30
Rank quality	1 / 1.0 / 1	1 / 1.0 / 1
% Time above fastest	40 / 56 / 81	40 / 56 / 80
No. of times fastest	0	0
Rank time	11 / 12.9 / 15	11 / 12.5 / 16
30_120_sfw		
% Solution above LB	10 / 17 / 23	11 / 16 / 23
% Improvement	0 / 3 / 10	0 / 4 / 13
No. of times best	13	19
Rank quality	1 / 3.1 / 9	1 / 2.3 / 11
% Time above fastest	202 / 237 / 282	202 / 237 / 282
No. of times fastest	0	0
Rank time	12 / 14.6 / 17	11 / 14.3 / 17
50_200		
% Solution above LB	4 / 6 / 9	4 / 6 / 8
% Improvement	1 / 3 / 5	1 / 3 / 5
No. of times best	12	18
Rank quality	1 / 2.5 / 6	1 / 1.9 / 4
% Time above fastest	1,819 / 2,103 / 2,543	1,815 / 2,009 / 2,500
No. of times fastest	0	0
Rank time	14 / 16.0 / 17	14 / 15.7 / 17
All 510 instances		
% Solution above LB	0 / 14 / 42	0 / 14 / 42
% Improvement	0 / 1 / 13	0 / 2 / 13
No. of times best	399	426
Rank quality	1 / 1.7 / 12	1 / 1.6 / 11
% Time above fastest	18 / 574 / 2,543	18 / 574 / 2,500
No. of times fastest	0	0
Rank time	10 / 15.5 / 17	10 / 15.3 / 17

Table 3.7: Computational results for GDRPP-All heuristics (min. / avg. / max.)

The following observations can be made in Tables 3.4–3.7:

- Solution quality:
  - The solution quality of all heuristics is rather poor. Even the GDRPP-All heuristics are on average 14 % above the lower bound.
  - The solution quality varies widely, both within and between instance types. Even for the GDRPP-All heuristics, the relative deviation from the lower bound ranges from 0 to 42 %, and the average relative deviation from the lower bound ranges from 6 % for the WRPPZZ instances to 22 % for the WRPPTP instances without shortest deadheading walks.
  - The GDRPP-HL heuristics clearly outperform the GDRPP-NN and GDRPP-BC approaches.
  - The best solution quality on average is obtained for the WRPPZZ instances.
  - The solution quality for the random GDRPP instances with disjoint r-groups is better than for those with not necessarily disjoint r-groups, particularly for the GDRPP-NN and GDRPP-BC heuristics.
  - For the WRPPTP instances, the sfw transformation yields a better solution quality than the no_sfw transformation.
  - The line 'Rank quality' in Table 3.7 for 'All 510 instances' reports a maximum rank of 12 for GDRPP-All-SA-SrG and a maximum rank of 11 for GDRPP-All-SrG-SA. This means that, for at least one instance, 11 and 10 other heuristics yielded a better solution than GDRPP-All-SA-SrG and GDRPP-All-SrG-SA. This is possible, because the solution with the best objective function value among all construction heuristics is used as input to the improvement heuristics, but it may happen that a worse initial solution can be improved more.
- Running time:
  - As mentioned above, the running times were measured in wall-clock time. Therefore, the evaluations of the running times are not completely consistent. For example, it is not sensible that a heuristic with improvement phase is the fastest one for an instance (as in Table 3.4 for the 30_120_no_sfw instances). The heuristic without improvement phase should have been faster. Likewise, it is not sensible that GDRPP-HL-SrG-SA is on average faster than GDRPP-HL (as in Table 3.6).
  - The GDRPP-HL constructive heuristic is by far the most time-consuming one. It takes one order of magnitude more time than any other constructive or improvement heuristic. So, the usual trade-off between solution quality and running time can be observed here, too.

These ad hoc observations give rise to the following hypotheses that were tested for their statistical significance:

- GDRPP-HL is the best constructive heuristic. The expected value of the relative deviation from the best lower bound for GDRPP-HL, $E(\text{rel\_dev}_{\text{HL}})$, is lower than the corresponding values for GDRPP-NN, $E(\text{rel\_dev}_{\text{NN}})$, and for GDRPP-BC, $E(\text{rel\_dev}_{\text{BC}})$.
  Wilcoxon signed rank test (Golden/Stewart 1985, p. 209 ff.):
  $H_0 : E(\text{rel\_dev}_{\text{HL}}) = E(\text{rel\_dev}_{\text{NN}})$ against $H_1 : E(\text{rel\_dev}_{\text{HL}}) < E(\text{rel\_dev}_{\text{NN}})$
  Result: Reject H_0 (and, consequently, accept H_1) for $\alpha = 0.0001$.
  $H_0 : E(\text{rel\_dev}_{\text{HL}}) = E(\text{rel\_dev}_{\text{BC}})$ against $H_1 : E(\text{rel\_dev}_{\text{HL}}) < E(\text{rel\_dev}_{\text{BC}})$
  Result: Reject H_0 (and, consequently, accept H_1) for $\alpha = 0.0001$.
- Over all heuristics, the expected value of the relative deviation from the best lower bound for the WRPPZZ instances is lower than the corresponding value for the other instances.
  Approximate two-sample Gauß test (Bamberg/Baur 1989, p. 193 f.):
  $H_0 : E(\text{rel\_dev}_{\text{WRPPZZ}}) = E(\text{rel\_dev}_{\text{other\_heurs}})$ against $H_1 : E(\text{rel\_dev}_{\text{WRPPZZ}}) < E(\text{rel\_dev}_{\text{other\_heurs}})$
  Result: Reject H_0 (and, consequently, accept H_1) for $\alpha = 0.0001$.

- Over all heuristics, the expected value of the relative deviation from the best lower bound for the sfw transformation of the WRPPTP instances into GDRPP instances, E(WRPPTP-SFW), is lower than the corresponding expected value for the no_sfw transformation, E(WRPPTP-No-SFW).
  Wilcoxon signed rank test:
  $H_0 : E(\text{WRPPTP-SFW}) = E(\text{WRPPTP-No-SFW})$ against
  $H_1 : E(\text{WRPPTP-SFW}) < E(\text{WRPPTP-No-SFW})$
  Reject H_0 (and, consequently, accept H_1) for $\alpha = 0.0001$.

Hence, all three hypotheses could be verified at a very high level of significance.

All transformed WRPPTP instances have disjoint r-groups, so for these instances as well as for the random GDRPP instances with disjoint r-groups, the results with respect to solution quality are identical for the GDRPP-NN and the GDRPP-BC heuristics.

Why are the results obtained with the sfw transformation better than those obtained with the no_sfw transformation? It follows from the construction of the GDRPP-NN heuristic that the solutions for transformed WRPPTP instances will be the same for the sfw and the no_sfw transformation. Also, the results with the GDRPP-HL heuristic were very similar for the sfw and the no_sfw instances. This means that the sfw transformation leaves more opportunities for the improvement heuristics. This is probably due to the following fact: When an arc is removed in the no_sfw transformation, the solution cycle is closed again by the arcs on the shortest path from the head of the preceding r-group arc to the tail of the succeeding r-group arc. If this path contains the removed arc, the latter will immediately be inserted again and the solution will remain unchanged. However, when an arc is removed in the sfw transformation, the cycle is closed by the non-r-group deadheading arc representing the shortest path from the head of the preceding r-group arc to the tail of the succeeding r-group arc. If the removed arc at its former position in the cycle was the best way to cover its r-group, it will be inserted again at this position. If it was not, a better arc will be inserted at a better position.

The running times of the heuristics ranged from 200 milliseconds to slightly less than one minute. The average over all 510 instances was 2.46 seconds for the ten combinations of GDRPP-NN and GDRPP-BC with improvement heuristics (Tables 3.4 and 3.5), and 7.82 seconds for the seven combinations of GDRPP-HL and GDRPP-All with improvement heuristics (Tables 3.6 and 3.7). However, the implementation of the heuristics was not optimized for speed, so it should be possible to reduce the running times by one order of magnitude.

Figure 3.16 depicts a Pareto diagram of the 17 considered combinations of constructive and improvement heuristics. There are six Pareto-optimal combinations. GDRPP-HL-SrG-SA is best with respect to solution quality, GDRPP-BC is best with respect to running time. No combination comes close to the 'perfect' heuristic, whose position is marked with an asterisk, and the relatively sharp trade-off between solution quality and running time becomes visible: Most combinations are quite close to the line from (1; 17) (best rank with respect to time, last rank with respect to quality) to (17; 1) (vice versa). Along this line, any increase in solution quality leads to an equivalent increase in running time. On the other hand, almost all combinations are below this line, which means that the trade-off is less than 1-to-1.

3.5.3.3 Results for the Branch-and-Cut Algorithm

To test the usefulness of the δ_i cuts and the upper bounding by the heuristic solution, all test instances were tackled with the following four set-ups:

(i) no cuts, no upper bounding

(ii) with δ_i cuts, no upper bounding

(iii) no δ_i cuts, with upper bounding

(iv) with δ_i cuts, with upper bounding

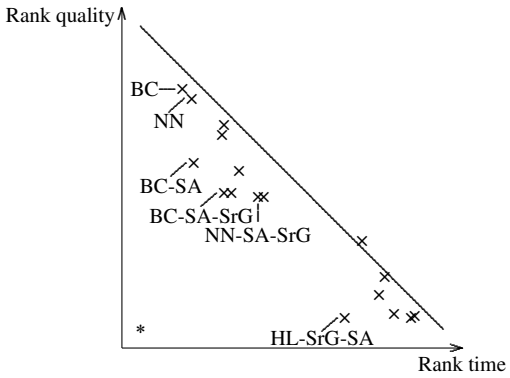


Figure 3.16: Pareto diagram GDRPP heuristics

The following table presents the computational results obtained with the branch-and-cut algorithm. The line headings in the table have the following meanings:

- No. of tried / feasible / optimal: number of instances that were tackled with the respective branch-and-cut set-up, number of instances for which a feasible solution was found (before reaching the time limit or running out of memory), and number of instances that were solved to optimality
- % Gap at root: percentage by which heuristic upper bound (HUB) exceeds lower bound at the root vertex of the branch-and-bound tree (RLB): $(HUB - RLB)/RLB \cdot 100$
- % Gap at end: percentage by which best feasible solution (BFS) exceeds best lower bound at the end of the optimization (BLB) (zero if optimal solution is found, not counted if no feasible solution is found): $(BFS - BLB)/BLB \cdot 100$
- No. of times UB pruned: number of times a vertex of the branch-and-bound tree could be pruned because of the heuristic upper bound

In all columns, only the running times for the branch-and-cut algorithm itself are indicated; the time needed to compute the upper bounds is not included in the running times given in the pertinent columns. As before, the running times were measured in wall-clock time, and hence, there are again some small inconsistencies.

	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
50_500_10_n				
No. of vertices / arcs / r-groups		58 / 516 / 10		
No. of flow vars. / δ_i vars. / constraints		516 / 58 / 585		
No. of tried / feasible / optimal	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10
Running time [s]	0 / 53 / 257	0 / 40 / 261	0 / 44 / 146	0 / 26 / 125
% Gap at root	–	–	3 / 15 / 25	7 / 15 / 25
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 45 / 231	0 / 8 / 62
No. of B&B vertices	1 / 595 / 1,884	1 / 212 / 1,168	1 / 569 / 1,980	1 / 154 / 473
No. of separated SECs	4 / 994 / 3,579	2 / 989 / 3,921	4 / 701 / 2,126	2 / 775 / 2,833
50_500_10_y				
No. of vertices / arcs / r-groups		55 / 510 / 10		
No. of flow vars. / δ_i vars. / constraints		510 / 55 / 576		
No. of tried / feasible / optimal	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10
Running time [s]	0 / 32 / 260	0 / 19 / 130	0 / 38 / 263	0 / 23 / 188
% Gap at root	–	–	6 / 14 / 27	5 / 14 / 25
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 15 / 115	0 / 7 / 40
No. of B&B vertices	1 / 375 / 2,180	1 / 122 / 699	1 / 428 / 2,180	1 / 130 / 839
No. of separated SECs	0 / 480 / 3,097	1 / 549 / 2,778	0 / 540 / 3,097	1 / 543 / 3,550

(continued on next page)

(continued from previous page)

	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
50_500_15_n				
No. of vertices / arcs / r-groups		58 / 516 / 15		
No. of flow vars. / δ_i vars. / constraints		516 / 58 / 589		
No. of tried / feasible / optimal	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10
Running time [s]	0 / 22 / 125	0 / 192 / 753	0 / 61 / 319	0 / 99 / 601
% Gap at root	–	–	11 / 18 / 36	11 / 18 / 34
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 36 / 295	0 / 2 / 15
No. of B&B vertices	1 / 255 / 1,124	1 / 268 / 824	1 / 337 / 1,552	1 / 204 / 630
No. of separated SECs	27 / 518 / 2,610	17 / 2,582 / 7,819	27 / 698 / 2,881	17 / 1,739 / 7,819
50_500_15_y				
No. of vertices / arcs / r-groups		55 / 510 / 15		
No. of flow vars. / δ_i vars. / constraints		510 / 55 / 580		
No. of tried / feasible / optimal	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10
Running time [s]	0 / 6 / 22	0 / 10 / 29	0 / 6 / 15	0 / 10 / 29
% Gap at root	–	–	5 / 14 / 29	5 / 14 / 28
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 2 / 24	0 / 3 / 33
No. of B&B vertices	1 / 114 / 461	1 / 81 / 327	1 / 108 / 461	1 / 80 / 327
No. of separated SECs	9 / 247 / 837	16 / 484 / 1,199	9 / 231 / 837	16 / 474 / 1,199
50_500_20_n				
No. of vertices / arcs / r-groups		58 / 515 / 20		
No. of flow vars. / δ_i vars. / constraints		515 / 58 / 594		
No. of tried / feasible / optimal	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10
Running time [s]	0 / 11 / 44	1 / 108 / 705	0 / 10 / 44	1 / 68 / 350
% Gap at root	–	–	5 / 16 / 31	5 / 16 / 31
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 4	0 / 6 / 49
No. of B&B vertices	1 / 247 / 1,076	1 / 302 / 1,060	1 / 226 / 1,076	1 / 259 / 751
No. of separated SECs	11 / 284 / 1,051	44 / 2,017 / 9,013	11 / 276 / 1,051	44 / 1,472 / 4,600
50_500_20_y				
No. of vertices / arcs / r-groups		54 / 508 / 20		
No. of flow vars. / δ_i vars. / constraints		508 / 54 / 584		
No. of tried / feasible / optimal	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10	10 / 10 / 10
Running time [s]	1 / 46 / 388	1 / 9 / 33	1 / 35 / 258	1 / 11 / 42
% Gap at root	–	–	5 / 16 / 29	5 / 16 / 29
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 3 / 15	0 / 0 / 3
No. of B&B vertices	7 / 999 / 8,662	1 / 48 / 158	7 / 878 / 7,142	1 / 51 / 188
No. of separated SECs	0 / 265 / 1,380	1 / 337 / 1,331	0 / 302 / 1,663	1 / 355 / 1,331
50_500_25_n				
No. of vertices / arcs / r-groups		58 / 515 / 25		
No. of flow vars. / δ_i vars. / constraints		515 / 58 / 599		
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 28	30 / 30 / 29	30 / 30 / 29
Running time [s]	0 / 85 / 1,892	0 / 458 / 3,931	0 / 151 / 3,900	0 / 418 / 3,900
% Gap at root	–	–	1 / 16 / 24	1 / 16 / 24
% Gap at end	0 / 0 / 0	0 / 1 / 16	0 / 1 / 30	0 / 0 / 8
No. of times UB pruned	–	–	0 / 20 / 556	0 / 7 / 160
No. of B&B vertices	1 / 482 / 7,078	1 / 639 / 3,173	1 / 454 / 6,255	1 / 697 / 3,173
No. of separated SECs	0 / 771 / 7,266	1 / 3,354 / 22,415	0 / 807 / 8,467	1 / 3,064 / 15,013

(continued on next page)

(continued from previous page)

	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
50_500_25_y				
No. of vertices / arcs / r-groups	54 / 509 / 25			
No. of flow vars. / δ_i vars. / constraints	509 / 54 / 589			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
Running time [s]	0 / 41 / 755	0 / 93 / 1,901	0 / 47 / 1,001	0 / 67 / 1,132
% Gap at root	–	–	4 / 15 / 29	4 / 15 / 29
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 6	0 / 0 / 7
No. of B&B vertices	1 / 489 / 6,501	1 / 332 / 5,267	1 / 535 / 8,144	1 / 262 / 3,162
No. of separated SECs	0 / 410 / 4,914	0 / 893 / 9,203	0 / 390 / 5,120	0 / 866 / 8,393
50_500_50_n				
No. of vertices / arcs / r-groups	52 / 503 / 50			
No. of flow vars. / δ_i vars. / constraints	503 / 52 / 606			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
Running time [s]	0 / 5 / 15	0 / 5 / 21	0 / 5 / 15	0 / 5 / 21
% Gap at root	–	–	10 / 17 / 43	9 / 17 / 43
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	1 / 66 / 215	1 / 54 / 319	1 / 66 / 215	1 / 54 / 319
No. of separated SECs	0 / 56 / 184	0 / 61 / 305	0 / 56 / 184	0 / 61 / 305
50_500_50_y				
No. of vertices / arcs / r-groups	51 / 502 / 50			
No. of flow vars. / δ_i vars. / constraints	502 / 51 / 605			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
Running time [s]	0 / 5 / 29	0 / 4 / 19	0 / 5 / 29	0 / 4 / 19
% Gap at root	–	–	9 / 15 / 21	9 / 15 / 21
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	1 / 80 / 612	1 / 59 / 342	1 / 80 / 612	1 / 59 / 342
No. of separated SECs	0 / 34 / 253	0 / 32 / 197	0 / 34 / 253	0 / 32 / 197
100_1000_100_n				
No. of vertices / arcs / r-groups	101 / 1,002 / 100			
No. of flow vars. / δ_i vars. / constraints	1,002 / 101 / 1,205			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 29	30 / 30 / 30	30 / 30 / 29
Running time [s]	1 / 96 / 299	1 / 394 / 3,900	1 / 93 / 287	1 / 394 / 3,900
% Gap at root	–	–	12 / 20 / 25	12 / 20 / 25
% Gap at end	0 / 0 / 0	0 / 0 / 2	0 / 0 / 0	0 / 0 / 2
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	1 / 408 / 1,586	1 / 651 / 4,158	1 / 408 / 1,586	1 / 651 / 4,158
No. of separated SECs	0 / 431 / 1,993	2 / 1,508 / 10,373	0 / 431 / 1,993	2 / 1,508 / 10,373
100_1000_100_y				
No. of vertices / arcs / r-groups	101 / 1,002 / 100			
No. of flow vars. / δ_i vars. / constraints	1,002 / 101 / 1,204			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
Running time [s]	21 / 151 / 650	14 / 164 / 1,154	20 / 136 / 577	14 / 164 / 1,154
% Gap at root	–	–	13 / 17 / 21	13 / 17 / 21
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	55 / 594 / 1,884	56 / 553 / 1,931	55 / 594 / 1,884	56 / 553 / 1,931
No. of separated SECs	11 / 452 / 2,068	17 / 584 / 3,834	11 / 452 / 2,068	17 / 584 / 3,834

(continued on next page)

(continued from previous page)

	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
10_40_no_sfw				
No. of vertices / arcs / r-groups	142 / 586 / 19			
No. of flow vars. / δ_i vars. / constraints	586 / 142 / 748			
No. of tried / feasible / optimal	30 / 29 / 29	30 / 30 / 30	30 / 29 / 29	30 / 30 / 30
Running time [s]	0 / 53 / 609	0 / 153 / 1,872	0 / 47 / 526	0 / 153 / 1,872
% Gap at root	–	–	6 / 19 / 35	7 / 19 / 35
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 5	0 / 0 / 0
No. of B&B vertices	1 / 78 / 603	1 / 128 / 795	1 / 78 / 610	1 / 128 / 795
No. of separated SECs	20 / 905 / 5,959	21 / 1,499 / 9,940	20 / 884 / 5,351	21 / 1,499 / 9,940
10_40_sfw				
No. of vertices / arcs / r-groups	69 / 1,203 / 19			
No. of flow vars. / δ_i vars. / constraints	1,203 / 69 / 1,293			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
Running time [s]	0 / 16 / 97	1 / 19 / 234	0 / 13 / 61	1 / 19 / 234
% Gap at root	–	–	3 / 13 / 28	3 / 13 / 28
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 3	0 / 0 / 0
No. of B&B vertices	1 / 64 / 767	1 / 57 / 458	1 / 59 / 619	1 / 57 / 458
No. of separated SECs	21 / 286 / 1,252	17 / 338 / 2,620	21 / 274 / 890	17 / 338 / 2,620
20_80_no_sfw				
No. of vertices / arcs / r-groups	282 / 1,197 / 42			
No. of flow vars. / δ_i vars. / constraints	1,197 / 282 / 1,522			
No. of tried / feasible / optimal	30 / 29 / 9	30 / 27 / 10	30 / 29 / 9	30 / 27 / 10
Running time [s]	14 / 3,036 / 3,906	7 / 2,934 / 3,906	12 / 3,057 / 3,906	7 / 2,934 / 3,906
% Gap at root	–	–	16 / 24 / 44	16 / 23 / 43
% Gap at end	0 / 4 / 13	0 / 4 / 16	0 / 4 / 13	0 / 4 / 16
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	34 / 204 / 640	1 / 243 / 753	34 / 213 / 640	1 / 243 / 753
No. of separated SECs	65 / 11,336 / 16,808	33 / 10,989 / 18,338	65 / 11,533 / 16,808	33 / 10,989 / 18,338
20_80_sfw				
No. of vertices / arcs / r-groups	148 / 5,495 / 42			
No. of flow vars. / δ_i vars. / constraints	5,495 / 148 / 5,686			
No. of tried / feasible / optimal	30 / 30 / 19	30 / 30 / 19	30 / 30 / 19	30 / 30 / 19
Running time [s]	9 / 2,101 / 3,917	8 / 1,965 / 3,928	8 / 2,120 / 3,934	8 / 1,965 / 3,928
% Gap at root	–	–	10 / 18 / 29	10 / 18 / 29
% Gap at end	0 / 1 / 6	0 / 1 / 8	0 / 1 / 6	0 / 1 / 8
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	1 / 354 / 985	1 / 372 / 946	1 / 356 / 985	1 / 372 / 946
No. of separated SECs	136 / 5,869 / 10,769	33 / 5,687 / 10,798	136 / 5,912 / 10,617	33 / 5,687 / 10,798
30_120_no_sfw				
No. of vertices / arcs / r-groups	420 / 1,776 / 59			
No. of flow vars. / δ_i vars. / constraints	1,776 / 420 / 2,256			
No. of tried / feasible / optimal	16 / 10 / 0	30 / 25 / 0	16 / 10 / 0	30 / 25 / 0
Running time [s]	3,900 / 3,902 / 3,905	3,900 / 3,905 / 3,933	3,900 / 3,902 / 3,905	3,900 / 3,905 / 3,933
% Gap at root	–	–	16 / 24 / 38	14 / 23 / 38
% Gap at end	3 / 8 / 11	0 / 10 / 25	3 / 8 / 11	0 / 10 / 25
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	41 / 97 / 289	35 / 106 / 238	41 / 97 / 289	35 / 106 / 238
No. of separated SECs	7,633 / 10,815 / 12,753	7,548 / 11,311 / 14,777	7,633 / 10,815 / 12,753	7,548 / 11,311 / 14,777

(continued on next page)

(continued from previous page)

	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
30_120_sfw				
No. of vertices / arcs / r-groups	207 / 10,696 / 59			
No. of flow vars. / δ_i vars. / constraints	10,696 / 207 / 10,963			
No. of tried / feasible / optimal	30 / 24 / 5	30 / 19 / 5	30 / 24 / 5	30 / 19 / 5
Running time [s]	736 / 3,434 / 3,975	742 / 3,373 / 3,985	736 / 3,434 / 3,975	742 / 3,373 / 3,985
% Gap at root	–	–	11 / 18 / 25	12 / 19 / 25
% Gap at end	0 / 3 / 10	0 / 4 / 12	0 / 3 / 10	0 / 4 / 12
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	41 / 239 / 640	33 / 191 / 459	41 / 239 / 640	33 / 191 / 459
No. of separated SECs	1,593 / 6,965 / 10,136	2,016 / 6,621 / 10,061	1,593 / 6,965 / 10,136	2,016 / 6,621 / 10,061
30_100				
No. of vertices / arcs / r-groups	31 / 371 / 97			
No. of flow vars. / δ_i vars. / constraints	371 / 31 / 500			
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
Running time [s]	0 / 7 / 48	0 / 4 / 26	0 / 6 / 41	0 / 4 / 26
% Gap at root	–	–	2 / 6 / 9	2 / 6 / 9
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	1 / 487 / 3,834	1 / 314 / 2,351	1 / 487 / 3,834	1 / 314 / 2,351
No. of separated SECs	0 / 0 / 3	0 / 0 / 3	0 / 0 / 3	0 / 0 / 3
40_150				
No. of vertices / arcs / r-groups	41 / 559 / 148			
No. of flow vars. / δ_i vars. / constraints	559 / 41 / 749			
No. of tried / feasible / optimal	30 / 30 / 26	30 / 30 / 26	30 / 30 / 26	30 / 30 / 26
Running time [s]	0 / 593 / 3,900	0 / 612 / 3,900	0 / 591 / 3,900	0 / 612 / 3,900
% Gap at root	–	–	4 / 7 / 13	4 / 7 / 12
% Gap at end	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	1 / 25,390 / 173,086	1 / 27,081 / 187,357	1 / 26,787 / 196,731	1 / 27,081 / 187,357
No. of separated SECs	0 / 0 / 1	0 / 0 / 1	0 / 0 / 1	0 / 0 / 1
50_200				
No. of vertices / arcs / r-groups	51 / 736 / 195			
No. of flow vars. / δ_i vars. / constraints	736 / 51 / 983			
No. of tried / feasible / optimal	30 / 30 / 21	30 / 30 / 21	30 / 30 / 21	30 / 30 / 21
Running time [s]	1 / 1,420 / 3,900	0 / 1,550 / 3,901	1 / 1,419 / 3,901	0 / 1,550 / 3,901
% Gap at root	–	–	4 / 6 / 9	4 / 6 / 9
% Gap at end	0 / 0 / 0	0 / 0 / 1	0 / 0 / 0	0 / 0 / 1
No. of times UB pruned	–	–	0 / 0 / 0	0 / 0 / 0
No. of B&B vertices	10 / 39,842 / 119,503	1 / 43,410 / 111,369	10 / 40,079 / 123,005	1 / 43,410 / 111,369
No. of separated SECs	0 / 0 / 1	0 / 0 / 1	0 / 0 / 1	0 / 0 / 1
All 510 instances				
No. of vertices / arcs / r-groups	113 / 1,628 / 62			
No. of flow vars. / δ_i vars. / constraints	1,628 / 113 / 1,805			
No. of tried / feasible / optimal	496 / 482 / 409	510 / 491 / 408	496 / 482 / 408	510 / 491 / 409
Running time [s]	0 / 723 / 3,975	0 / 829 / 3,985	0 / 728 / 3,975	0 / 823 / 3,985
% Gap at root	–	–	1 / 15 / 44	1 / 15 / 43
% Gap at end	0 / 1 / 13	0 / 1 / 25	0 / 1 / 30	0 / 1 / 25
No. of times UB pruned	–	–	0 / 3 / 556	0 / 1 / 160
No. of B&B vertices	1 / 4,333 / 173,086	1 / 4,547 / 187,357	1 / 4,435 / 196,731	1 / 4,543 / 187,357
No. of separated SECs	0 / 1,883 / 16,808	0 / 2,431 / 22,415	0 / 1,895 / 16,808	0 / 2,379 / 18,338

Table 3.8: Computational results for branch-and-cut algorithm (min. / avg. / max.)

The following observations can be made in Table 3.8:

- The difficulty of the instances varies widely. For each instance type (with the exception of the 30_120 WRPPTP instances), there are very easy and very hard instances.
- In general, the δ_i cuts are not useful. Indeed, over all instances, the percentage by which the lower bound at the root vertex of the branch-and-bound tree when the cuts are used exceeds the lower bound when they are not used by a negligible 0.02 %. The average running times, number of vertices in the branch-and-bound trees, and number of separated SECs even increase when the cuts are used, although there are instances types where this is not the case.
- The upper bounding procedure is not helpful for most instances. The quality of the heuristic solutions is apparently too poor.
- The number of r-groups is not decisive for the difficulty of an instance type. This is in marked contrast to the results obtained with the exact labelling algorithm and with the results obtained by Noon/Bean 1991 on the GATSP.
- There is no clear evidence that instances with non-disjoint r-groups are more difficult or easier than instances with disjoint r-groups.
- The computational results by Laporte 1997 and Blais/Laporte 2003 show that instances with many required edges, i.e., instances with a 'significant generalized component', are much more difficult to solve with the solution approaches these authors use than instances of 'purely' directed problems are. Although the transformed WRPPZZ instances, which contain a considerable number of r-groups of cardinality greater than one to represent edges and zigzag links, seem to be more difficult than the random GDRPP instances, this effect is not so evident in the above computational results.
- Similar to the heuristics, also the branch-and-cut algorithm yields much better results for the sfw transformation of the WRPPTP instances than for the no_sfw transformation, even though the number of variables and constraints is very much higher with the sfw transformation.
- For the transformed WRPPZZ instances, the number of vertices in the branch-and-bound tree is extremely high and varies enormously. The results for this instance class are already quite good (considering the number of optimally solved instances), but for this class in particular, there seems to be a great potential for improvement through better lower and upper bounding procedures. It is also remarkable that there are almost no violated SECs in any of the transformed WRPPZZ instances.

The following table displays the percentage of instances of each type that were solved to optimality by the branch-and-cut algorithm.

Instance type	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
50_500_10_n	100	100	100	100
50_500_10_y	100	100	100	100
50_500_15_n	100	100	100	100
50_500_15_y	100	100	100	100
50_500_20_n	100	100	100	100
50_500_20_y	100	100	100	100
50_500_25_n	100	93	97	97
50_500_25_y	100	100	100	100
50_500_50_n	100	100	100	100
50_500_50_y	100	100	100	100
100_1000_100_n	100	97	100	97
100_1000_100_y	100	100	100	100

(continued on next page)

(continued from previous page)

Instance type	No cuts, no UB	Cuts, no UB	No cuts, UB	Cuts, UB
10_40_no_sfw	97	100	97	100
10_40_sfw	100	100	100	100
20_80_no_sfw	30	33	30	33
20_80_sfw	63	63	63	63
30_120_no_sfw	0	0	0	0
30_120_sfw	17	17	17	17
30_100	100	100	100	100
40_150	87	87	87	87
50_200	70	70	70	70
All 510 instances	82	80	82	80

Table 3.9: Percentage of instances solved to optimality by branch-and-cut

3.6 Conclusions

This chapter has treated the generalized directed rural postman problem (GDRPP). Different formulations have been given, and it has been shown that various uncapacitated routing problems can be modelled as GDRPPs and that several important practical constraints can be taken into account. Most of the presented transformations are simple and straightforward to implement. Also, different exact and heuristic solution approaches have been described, and computational experiments with these approaches have been presented. The results verify the applicability of the GDRPP model as a transformation target and confirm that it is an interesting alternative to the generalized asymmetric travelling salesman problem.

The main advantage of the GDRPP model is its genericity. It constitutes a unified model for many types of uncapacitated routing problem and is therefore able to serve as a flexible framework for representing and solving practical problems.

Chapter 4

The Vehicle Routing Problem with Trailers and Transshipments

In this chapter, the vehicle routing problem with trailers and transshipments (VRPTT), a generalization of the well-known vehicle routing problem (VRP), is introduced. This problem possesses some interesting characteristics which have not yet been treated in the literature. In the VRPTT, the vehicle fleet consists of autonomous vehicles able to move on their own, and of non-autonomous vehicles which must be accompanied by an autonomous vehicle to be able to move. Moreover, both autonomous and non-autonomous vehicles are either collection or support vehicles. Collection vehicles are used to collect the supplies of the customers. Support vehicles are used as mobile depots by the collection vehicles.

The chapter is structured as follows. In the next section, a detailed description of the VRPTT is given, and, based on this description, the essential new aspects of the VRPTT are identified and discussed. Section 4.2 reviews the relevant literature. In Section 4.3, three formulations for the VRPTT are developed. In Section 4.4, the possibility of solving the VRPTT by branch-and-price and the difficulties arising in such an approach are evaluated, in Section 4.5, a branch-and-cut algorithm is described, and in Section 4.6, computational experiments with this branch-and-cut algorithm are presented and analyzed. The chapter ends with a conclusion.

4.1 Problem Description

The research on the VRPTT was motivated by the following real-world problem: There is a set of *customers* with a known, *deterministic supply of a single good*, a set of *service stations* necessary for *unloading* and *maintenance*, and a set of *intermediate locations* that may be used for *parking* and *load transfer*. Service stations may also be used for parking. All customers, service stations and intermediate locations have an *arbitrary number of time windows* associated with them. These time windows may represent the same time of day on different days. Some *customers may have to be visited more than once*. The supply of a customer is the same on each visit. There may be restrictions on the minimal time between two consecutive visits at a customer, as well as restrictions specifying the number of visits that must occur in a subset of a customer's time windows. To do the collecting of the good, a *fleet of heterogeneous*, *limited-capacity vehicles* stationed at *vehicle depots* is available.

There are six main criteria in which the vehicles differ:

(i) First of all, the fleet is comprised of *lorries* and *trailers*. On a more abstract level, one can say that the fleet consists of two types of vehicle: *Autonomous* ones, able to move in time and space on their own (the lorries), and *non-autonomous* ones, able to move in time on their own, but requiring an autonomous object to move in space (the trailers).

(ii) Second, the lorries may be technically equipped to service customers (*collection lorries*) or not (*support lorries*).

(iii) Third, each vehicle may have a different *capacity*. There may be support lorries with a capacity of zero. All trailers and all collection lorries have positive capacity.

(iv) Fourth, each vehicle may have different fixed and variable *costs*. If a vehicle is not used, its fixed costs are not incurred.

(v) Fifth, the collection lorries may have different *load transfer times*. This is the time necessary to collect a given amount of load at a customer or to transfer load to another vehicle.

(vi) Sixth, the collection lorries may be subject to *accessibility constraints*, i.e., they may not be allowed to visit some customers. The driving speeds are assumed to be identical for all lorries, whether or not they pull a trailer, and whether or not they carry load.

Figure 4.1 depicts the four relevant types of vehicle as they are encountered on European streets and motorways. A lorry with an attached trailer is referred to as a *lorry-trailer combination* (*LTC*). Typically, lorries able to form jointed lorry-trailer combinations are used as collection lorries, and saddle lorries (which have a capacity of zero) are used as support lorries.

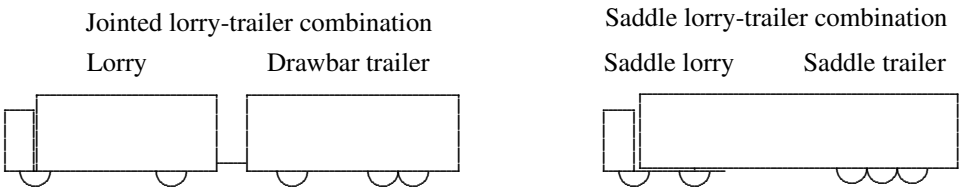


Figure 4.1: VRPTT fleet

Essentially, the trailers and the support vehicles are used to extend the capacity of the collection lorries, so that the latter need not go to an unloading station so often. Obviously, for a trailer to be useful, it must be pulled by a lorry. However, not every combination of lorry and trailer is allowed. Trailers which can be pulled only by support lorries are called *support trailers*, the others are called *collection trailers*. Support vehicles are simply subject to special accessibility constraints at customers. In principle, a collection vehicle can also work as a pure support vehicle, but not the other way round. However, a collection lorry would never be used as a pure support lorry, because a pure support lorry is considerably cheaper. Moreover, the distinction between collection and support vehicles is really not a gradual one, because load transfer locations may be visited by both types of vehicle, but to perform a load transfer, (the technical equipment of) a collection lorry is needed.

Intermediate locations can be used either for parking (*parking locations*) or for load transfer (*load transfer locations* or *transshipment locations*).

There is no fixed depot for a vehicle. A vehicle may start its itinerary at any depot. At the end of the planning horizon, it always returns to the depot it started from. As with intermediate locations, there are two types of depot. *Parking depots* can be used only for parking a vehicle. *Transshipment depots* offer the additional possibility of transferring load to a vehicle which is still in the depot.

Using a parking or transshipment location or depot may incur fixed setup costs per vehicle. Typically, these costs are zero for using parking locations or depots. Service stations do not incur setup costs.

Transshipment locations, transshipment depots and service stations have two types of time window: The *visiting time windows* determine the times when these locations may be used for parking, the *service time*

windows determine the times when load transfers, respectively, unloading or maintenance, are allowed. The visiting time windows contain the service time windows at the respective locations, i.e., load transfers, unloading and maintenance are allowed only at times when parking is allowed also.

Coupling and decoupling a trailer incurs certain costs; fleet owners, dispatchers and drivers consider it undesirable to separate a lorry and a trailer 'too often'. The coupling and decoupling process takes time, but this time is negligible compared to the total driving and loading time.

Customers that can only be visited by a lorry without a trailer (a *single* lorry) are called *lorry customers*. The other customers, which can be visited by a lorry with or without a trailer, are called *trailer customers*. As mentioned above, there may be additional vehicle-specific accessibility constraints at customers. Split collection is not allowed; hence, there are no customers with a supply exceeding the capacity of the largest lorry (or lorry-trailer combination, in the case of trailer customers). The statement that split collection is not allowed holds for individual customers. However, it is sometimes useful to perform a customer aggregation when modelling a real-world problem, in particular when the number of individual customers is too large from a computational point of view, and when the customer locations form clearly separable clusters. In practice, split collection of these aggregated customers is allowed, but this is not considered here.

The vehicles are used as follows. The standard case is a collection lorry with a trailer. Most customers are lorry customers, so normally the trailer is parked at an intermediate location and the lorry does some collecting. Then, it either transfers its load to the trailer and continues collecting, or re-couples the trailer, parks it elsewhere, decouples, and does some more collecting before transferring load. When a trailer customer is visited by a lorry with a trailer, the supply of the customer can directly be loaded completely or partly into the trailer, and at most customers, any additional load the lorry has already collected can be transferred from lorry to trailer at such a location, too. (Some customers, though, may not want any kind of unnecessary operation performed on their premises.) After collecting at least one customer's supply, both the lorry and the trailer go to an unloading station for unloading. They may then start another tour. In addition, a collection lorry may also operate singly.

There is no fixed assignment of a trailer to a lorry. Any lorry may pull any compatible trailer for some time. What is more, any vehicle may transfer load to any other vehicle. In order to do so, the equipment of a collection lorry must be used. That is, for a load transfer from a trailer to another trailer or to a support lorry, a collection lorry must be present. For unloading a trailer or a support lorry at an unloading station, no collection lorry is necessary. Figure 4.2 depicts a possible route plan with three collection lorries and one trailer. In the plan, lorry 1, together with the trailer, starts at the depot, goes to a transshipment location, decouples the trailer there, visits two lorry customers, returns to the trailer, transfers some load, leaves the trailer there and returns to the depot via some lorry and some trailer customers. Lorry 2 starts at the depot, visits some lorry customers, couples the trailer (after lorry 1 has performed its load transfer), visits a trailer customer, decouples the trailer at another transshipment location, possibly performs a load transfer, visits some lorry customers, returns to the parked trailer, re-couples it and pulls it back to the depot via a trailer customer. Meanwhile, lorry 3 also starts at the depot, visits some lorry customers, transfers some load to the trailer while lorry 2 is visiting the three lorry customers bottom right, and returns to the depot via another lorry customer. The two transshipment locations in the centre of the figure are not used.

All vehicles must undergo maintenance a specified number of times. As with consecutive visits to customers, there are restrictions specifying the minimal and maximal time between two consecutive maintenance processes. One maintenance process takes an amount of time dependent on the maintenance station and the vehicle. Some unloading stations, called *combi stations*, can also be used as maintenance stations. At any point in time, only one lorry, trailer, or lorry-trailer combination, can unload at an unloading station or be maintained at a maintenance station. At a combi station, only one type of

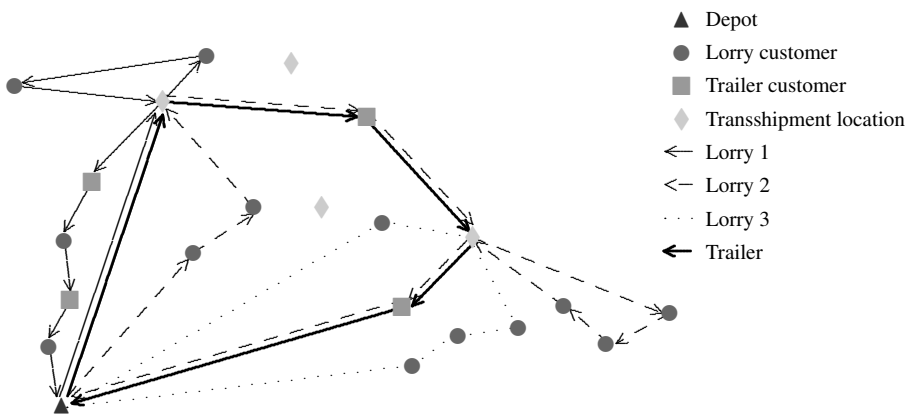


Figure 4.2: Example of a VRPTT route plan

service (unloading, maintenance) can be performed at a time. To undergo maintenance, a vehicle must be empty. A lorry-trailer combination that unloads at an unloading or combi station has a total unloading time which is equal to the sum of the unloading times of the lorry and the trailer, but it is technically possible to maintain the two vehicles of a lorry-trailer combination at a maintenance or combi station *in parallel*.

A certain amount of the customers' supply must be delivered to a certain unloading station. There may also be restrictions on the minimum amount of load delivered to an unloading station until a certain point in time.

Moreover, there may be time intervals where no customer may be visited, i.e., the planning horizon may contain (at least two) disjoint *meta time windows*. All customer time windows then lie within one such meta time window. There are no intermediate locations or service stations with a time window stretching over more than two consecutive meta time windows (i.e., stretching over more than one time interval between two meta time windows). Between visits at customers during two consecutive meta time windows, a vehicle must be maintained exactly once and return to the real-world location where its depot is situated. This means that the maintenance frequency of all vehicles equals the number of meta time windows.

The task is to devise routings of minimal total costs for all vehicles (some of which may not be needed), such that the complete supply of all customers is collected and delivered to the unloading stations. This includes the assignment of each vehicle which is used to a depot.

4.1.1 What is New?

Several of the aspects relevant in the problem presented in the previous section are neither novel nor complicated. What really *is* new is the problem as a whole, and, in particular, the simultaneous consideration of the following three points in one problem:

(i) A trailer may be pulled by different lorries on its itinerary.

(ii) Load transfers are possible between arbitrary vehicles.

(iii) There are support vehicles, which cannot visit any customers.

4.1.2 The Central Question

The main difficulty of the problem is the close interdependency between the vehicles: With few exceptions, the whole road network may be used by all vehicles. The parking and transshipment locations are

there precisely to be visited more than once, and by more than one vehicle. This is not usually the case in vehicle routing problems. Moreover, the fact that trailers cannot move on their own and the load transfer possibility give the problem not only a routing, but also a scheduling aspect: The itineraries of trailers and the lorries pulling them and the load transfer processes must be synchronized. All in all, a fourfold synchronization of the vehicles is required:

(i) *Customer Covering Synchronization*
All customers must be visited exactly the required number of times, and on each visit by exactly one collection lorry.

(ii) *Spacial Synchronization*
The vehicle itineraries must be synchronized, because a trailer must be pulled by a lorry to move in space.

(iii) *Temporal Synchronization*

- Spacial synchronization evidently implies temporal synchronization for common movements of a lorry and a trailer.
- In addition, temporal synchronization is necessary at transshipment locations, where vehicles may meet that reach these locations from different other locations. A vehicle can perform a load transfer to another vehicle at a transshipment location only if both vehicles are present at the respective location during the time needed for the load transfer.

(iv) *Load Synchronization*
For each transshipment operation, it must be decided how much load is to be transferred. The load one vehicle unloads is exactly equal to the load the other vehicle receives. No load gets lost.

Temporal and load synchronization are themselves interdependent because of the load-dependent load transfer times.

These deliberations are summarized in the central question that must be answered when solving a VRPTT:

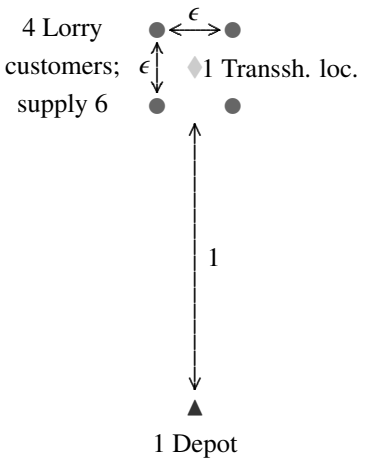
Which vehicle transfers how much load when where into which other vehicle?

4.1.3 Potential Savings Through the Use of Trailers

Another question that immediately springs to mind is how much can actually be saved by using trailers. Interestingly, the potential savings are not bounded from above. Consider an instance with one depot, n lorry customers, each with a supply of one, located at a distance of one unit from the depot and arbitrarily close to one another, and with one transshipment location arbitrarily close to each of the customers. If there are n identical lorries with fixed costs of zero, distance-dependent costs of one, and a capacity of one, the optimal solution evidently consists of using one lorry for each customer, and this solution incurs costs of $2n$. If there is a trailer with fixed and distance-dependent costs of zero and a capacity of $n-1$, the optimal solution consists in using one lorry and the trailer, and this solution incurs costs of 2 (if the load transfer speed is assumed to be infinite or if there are no time-dependent costs). As n reaches infinity, so do the potential absolute savings, and the relative savings reach 100 %.

A less contrived case is given by an instance with one depot, four lorry customers, each with a supply of six, located at a distance of one unit from the depot and arbitrarily close to one another, and with one transshipment location arbitrarily close to each of the customers. If the vehicle fleet consists of identical lorries with fixed costs of zero, distance-dependent costs of one and a capacity of ten, the optimal solution incurs costs of $4 \cdot (1+1) \cdot 1 = 8$. If there is a trailer available with fixed costs of zero, distance-dependent costs of one tenth of the distance-dependent costs of a lorry, and with a capacity of 15, the optimal solution incurs costs of $2 \cdot 1 + 2 \cdot 0.1 = 2.2$, which means a saving of 72.5 %; cf. Figure 4.3. (The case

of a lorry-trailer combination where the lorry has a capacity of about 10 units (tons) and where the trailer has a capacity of about 15 units (tons) is common in Europe. See also the section on computational experiments.)



Fleet:
4 lorries,
capacity 10,
fixed costs 0, variable costs 1
$\Rightarrow$ minimal cost: $4 \cdot (1 + 1) \cdot 1 = 8$
One additional trailer,
capacity 15,
fixed costs 0, variable costs 0.1
$\Rightarrow$ minimal cost: $2 \cdot 1 + 2 \cdot 0.1 = 2.2$
$\Rightarrow$ savings of 72.5 %

Figure 4.3: Potential savings through the use of trailers

4.2 Literature Review

Notwithstanding the considerable potential for cost savings through the use of trailers, the literature on vehicle routing with trailers is scarce. To the best of the author's knowledge, the subsequent formulations are the first ones that simultaneously take the above aspects into account. Following is a brief review of the few papers there are.

Brunswicker 1986, Vahrenkamp 1989, and Scheuerer 2004 consider the problem of raw milk collection at farmyards. The milk is collected at storage tanks on the farmyards every or every other day and must be transported to dairy plants. Some of the farmyards cannot be visited by a lorry-trailer combination because of space restrictions. The first two authors allow only one transshipment location per trailer and a fixed lorry-trailer assignment. This is not adequate for the VRPTT. Brunswicker 1986 develops a heuristic sequential solution approach for this type of problem (clustering of customers, determination of one transshipment location per trailer, routing). Vahrenkamp 1989 proceeds similarly.

Semet/Taillard 1993 and Gerdessen 1996 consider the task of food delivery to supermarkets, some of which lie in inner-city areas where there is not enough space for manoeuvring with a lorry-trailer combination. Additionally, Gerdessen 1996 considers the distribution of animal food to farmyards, which essentially is the delivery case corresponding to the raw milk collection case just described.

Gerdessen 1996 also allows only one transshipment location per trailer and a fixed lorry-trailer assignment. The paper presents several construction heuristics and intra- and inter-tour exchange improvement procedures and mentions an unpublished technical report containing an MIP formulation.

Semet/Taillard 1993, Semet 1995, Chao 2002, and Scheuerer 2004 (see also Scheuerer 2006) allow that trailers be parked several times at different locations, but they identify the potential trailer parking locations with the trailer customers (i.e., any trailer can be parked at any trailer customer location but nowhere else), they assume a fixed lorry-trailer assignment, and they do not allow a load transfer from a lorry to a trailer other than its assigned one. Consequently, they do not consider support vehicles. Semet /Taillard 1993 consider time windows and heterogeneous lorries, but identical trailers. Semet/Taillard 1993 and Semet 1995 also consider accessibility constraints for the lorries. Chao 2002 and Scheuerer 2004 impose a restriction on the maximum length or duration of a tour. Chao 2002 and Scheuerer 2004

call this problem the *truck-and-trailer routing problem* (TTRP). Scheuerer 2004 extends the approaches of Semet 1995 and Chao 2002 and considers a multi-period and a multi-depot version of the problem (but not a combined multi-period and -depot problem). All authors solve their respective problems by sophisticated heuristic procedures. Semet/Taillard 1993 use tabu search to improve the solutions obtained by a constructive method based on clustering methods and the solution of a generalized assignment problem. Semet 1995 uses a cluster-first-route-second method. In the clustering step, customers and trailers are simultaneously assigned to lorries by optimally solving an extended generalized assignment problem through a branch-and-bound algorithm where the lower bounds are computed by Lagrangean relaxation. In the routing step, a TSP is solved for the single lorry clusters and an extension of the TSP for the lorry-and-trailer clusters. Chao 2002 presents a three-stage constructive heuristic (customer clustering, routing, making routes feasible by means of inter-tour exchanges) and a tabu search improvement procedure. Scheuerer 2004 presents two new construction heuristics (a clustering-based sequential insertion procedure and an adaptation of the well-known sweep heuristic by Gillett/Miller 1974) and a tabu search improvement procedure. Moreover, the author adapts these procedures to the multi-depot and the multi-period version of the problem. The clustering methods used by these authors are variants or extensions of the generalized assignment heuristic developed by Fisher/Jaikumar 1981. Semet 1995 and Scheuerer 2004 give mathematical programming formulations of the problems they consider (0-1 IP formulations). Their choice of variables is such that it is not possible to consider the above generalizations, so their formulations cannot be used as a basis for a VRPTT model.

Very recently, Hoff/Løkketangen 2006 have presented a case study for milk collection in Norway. The problem they consider is essentially a multi-depot, multi-period TTRP with heterogeneous vehicles and without trailer customers. They propose a sophisticated tabu search algorithm for solving their problem and report successful solution of real-world instances, improving on the existing tour plans used by their industry partner.

Another recent paper is the one by Tan et al. 2006. The authors consider a problem of moving containers to and from a seaport by lorries and trailers. The underlying application is, in essence, a pickup-and-delivery problem, but with lorries with a capacity of zero and trailers with a capacity of one, so each pickup sub-task is followed by its corresponding delivery. This means that every task can be viewed as a single customer in a VRP sense. The authors consider homogeneous lorries and two types of trailer, but they do not identify the trailers; they rather regard them as interchangeable, limited resources that a lorry needs in order to perform tasks. Each task requires that the lorry performing it pull a trailer of the correct type. The trailers/resources are located at 'trailer exchange points' with a dynamically changing stock of trailers, Moreover, not all tasks need be performed by the fleet; it is as well possible to outsource tasks to external service providers at specified costs (which correspond to a penalty for not performing a task). The authors view their problem from a scheduling perspective, not so much from a routing point of view. They consider task time windows and handling times. The objective is two-dimensional: minimize the total distance travelled by lorries and minimize the number of lorries, and the problem is to find a Pareto-optimal schedule for the lorries. The authors propose an evolutionary algorithm for solving the problem. Overall, their problem is, in some respects, very similar to the VRPTT, and very different in others.

The rest of this section discusses some problems which do not consider trailers, but which are related to the VRPTT for other reasons.

Location-routing problems (LRPs, cf. Engele 1980, Laporte 1988, Daskin 1995, p. 339 ff., Nagy/Salhi 1996, Albareda-Sambola et al. 2005) combine facility location with vehicle routing and simultaneously address the following five questions (Daskin 1995, p. 339 f.): (i) how many facilities should be located, (ii) where should these facilities be located, (iii) which customers should be assigned to which facility, (iv) which customers should be visited on one and the same vehicle route starting and ending at a certain facility, and (v) in what sequence should the customers on one route be visited. It is evident that LRPs

have a lot in common with VRPs with trailers. Section 6.4 treats LRPs and their relationships to the VRPTT and the TTRP in more detail.

Bodin/Levy 2000 (see also Assad/Golden 1995) report on work on a problem in the context of postal delivery, the so-called *park-and-loop problem*. On each working day, postmen leave the post office by car to reach their delivery regions. At appropriate parking places, they may park the car and distribute part of the mail on foot. The postmen may return to their cars to refill their delivery bags several times, and they may change the parking place during the day. There is a fixed assignment of postmen to cars. In some streets, the mail may be delivered by car. Hence, the postmen correspond to the lorries, the cars correspond to the trailers, and the streets or street segments correspond to the customers. Streets that must be serviced on foot correspond to lorry customers; streets that may be serviced by car correspond to trailer customers. The problems (actually, the authors consider several variants) are solved by multi-stage heuristics mostly based on the cluster-first-route-second principle.

Another related problem is the so-called *vehicle and crew scheduling problem* (VCSP), considered, among others, by Haase et al. 2001 and Freling et al. 2003: Given a set of scheduled bus trips, the task is to find a set of minimum cost driver and bus schedules covering all bus trips and respecting a certain set of side constraints, such as breaks for drivers. Haase et al. 2001 develop an exact and a heuristic branch-and-price(-and-cut) procedure based on a set partitioning formulation for the driver scheduling problem incorporating side constraints for the buses. An optimal bus assignment is derived afterwards (in polynomial time). Freling et al. 2003 use Lagrangean relaxation in combination with column generation. The VCSP bears some similarity to the VRPTT, but there are decisive differences. Both problems consider autonomous and non-autonomous objects. In the VRPTT, the lorries are the autonomous objects, and the trailers are the non-autonomous ones. In the VCSP, the drivers are the autonomous objects, which are allowed to visit customers (perform trips) and have a capacity of zero, and the buses are the non-autonomous objects. In both problems, there is no fixed assignment of an autonomous object to a non-autonomous object. In the VCSP, a *relief point* is a point in space and time where and when a change of driver is allowed. The beginning and the end of a bus trip are such relief points. Haase et al. 2001 also allow relief points other than the beginning and the end of a trip ('inner' relief points). The VCSP requires that all tasks of a trip be performed by one and the same bus, but not necessarily the same driver. Hence, the Haase et al. 2001 version of the VCSP is not a VRPTT. Freling et al. 2003, on the other hand, consider the VCSP with no inner relief points. This is a special case of the VRPTT where all customers are trailer customers and must be visited at a specified point in time. The VCSP with no relief points at all is simply a vehicle scheduling problem. The most striking difference between the two problems lies in the fact that, unlike the VRPTT, in the VCSP, there is a fixed schedule for the bus trips/customers. This makes the VRPTT much more difficult. What is more, the trailers in the VRPTT need not visit any customers, but *may* visit some of them during a time window of positive length. Hence, there is an additional 'generalized' component in the VRPTT, and consequently, the VRPTT possesses yet an additional degree of complexity. Moreover, due to the fixed schedule in the VCSP, the time-dependent costs are much easier to handle there than in the VRPTT. Finally, the capacity constraints and the load transfer issue do not arise in the VCSP.

Planning problems in rail transport are also in some way related to the VRPTT because of the analogy of locomotives and waggons to lorries and trailers. The survey by Cordeau et al. 1998 gives an overview of the work in this area. More recent papers are Cordeau et al. 2000, Cordeau et al. 2001a, Cordeau et al. 2001b. The many different types of planning problem can be categorized into strategic, tactical, and operational problems, and into train routing, scheduling, and assignment problems. Routing problems consider the routing of locomotives, waggons, consists (groups of waggons), and complete trains. Scheduling problems are concerned with devising and maintaining timetables. Assignment problems address the assignment of locomotives to waggons, consists, or trains. The usual procedure is to decompose the overall problem a rail company faces into smaller problems which deal only with short- or medium- or long-term planning problems or only with train routing or assignment or scheduling problems. The

author is not aware of any papers in this area that address the four synchronization tasks of the VRPTT (task covering, spacial, temporal, and load synchronization) simultaneously and/or at a comparable level of detail.

Ioachim et al. 1998 and Ioachim et al. 1999 consider an *aircraft routing and scheduling problem*. In the application treated in these papers, there are scheduled flights (non-autonomous objects) required to depart at the same time on different days. These flights are known in advance, and the aircraft (autonomous objects) performing these flights have to be determined. To model this, the authors use so-called 'same departure time constraints'. Such constraints also arise in the VRPTT, but with two additional degrees of freedom: (i) It is not clear in advance which transshipment locations should be used and how many load transfers should be performed at each location, and (ii) the amounts of load transferred have to be synchronized, too.

To point out the general relevance of the concept of autonomous and non-autonomous vehicles/objects, the following Table 4.1 summarizes the above-mentioned applications that implicitly make use of this concept.

Problem	Autonomous vehicle/object	Non-autonomous vehicle/object
truck-and-trailer routing	lorry	trailer
park-and-loop	postman	car
vehicle and crew scheduling	bus driver	bus
train routing and scheduling	locomotive	waggon, consist, train
aircraft routing and scheduling	aircraft	scheduled flight
location-routing	lorry	abstract, virtual object
		(no corresponding real-world object)

Table 4.1: Applications of the concept of autonomous and non-autonomous 'vehicles'

Another problem which is of relevance when studying the VRPTT is the *split delivery vehicle routing problem* (SDVRP) (Dror/Trudeau 1989). In this problem, the usual VRP requirement that each customer be visited exactly once is abandoned. It is possible to serve a customer by more than one visit, each performed by a different vehicle. Dror/Trudeau 1989 say that this is a *relaxation* of the VRP. It is also correct to say that both problems are special cases of a more general VRP where each customer's demand must be exactly met by at most n^{visits} visits with different vehicles. The SDVRP can be considered a special case of this problem where n^{visits} is equal to the number of vehicles, and the usual VRP where split delivery is not allowed is a special case with $n^{visits} = 1$. The SDVRP is related to the VRPTT, because in the VRPTT, there are also locations that may be visited more than once: the parking and the transshipment locations. In the SDVRP, it must be decided what fraction of a customer's demand is satisfied by a certain vehicle. In the VRPTT, it must be decided what fraction of a trailer's capacity is used in a certain transshipment process.

The paper by Del Pia/Filippi 2006 must be mentioned here, too. The authors consider and solve a special real-world capacitated arc routing problem which they call CARP-MD, where MD stands for 'mobile depots'. They study a waste collection application with two types of (autonomous) vehicle, namely, large and small ones, and they allow load transfers from the small ones to the large ones. They develop a heuristic procedure for solving real-world instances. The distinguishing feature of their work is that, in their application, the need for temporal synchronization of vehicles at transshipment points arises. Their solution algorithm is based on a local search procedure for the CARP proposed by Hertz/Mittaz 2001. They modify this algorithm by including a procedure for the temporal synchronization of the vehicles. The basic ideas of their procedure are as follows. As an input to the procedure, they fix the total collection

amount of the small vehicles on their routes and the part of the capacity of the large vehicles reserved for transshipments from the small vehicles. Then, they compute a route plan serving all edges of their network with the vehicles. After that, they make the resulting routes feasible by arranging transshipments. This step is performed sequentially: First, they choose a small vehicle for a transshipment. For this vehicle, they determine a part of its tour during which a transshipment is reasonable (because the vehicle has already collected some load) and necessary (because at the end of this subtour the vehicle is full). Second, they determine a large vehicle and a vertex on its tour such that the total increase in the duration of both routes, when performing a load transfer at this vertex, is minimal. The route plan is updated accordingly and the procedure repeats until all routes are feasible. The problem considered by Del Pia/Filippi 2006 differs from the VRPTT in the following respects. First, there are no non-autonomous vehicles; hence, no routing synchronization along edges/arcs in the network is necessary. Second, only one vehicle serves each street/visits each customer; hence, there is no question as to how to divide up the supply of a trailer customer between a lorry and a trailer. Third, the authors assume that a small vehicle always transfers all of its load during a transshipment operation; hence, there is no question as to how much load to transfer at transshipment vertices. The authors do not develop a mathematical programming formulation.

For the sake of completeness, it must be mentioned that Ghiani/Laporte 2001 cite a technical report (which was not accessible to this author) that considers an arc routing problem in the context of garbage collection, where transshipments from one type of (small) vehicle to another type of (large) vehicle at 'transfer stations' must be planned. The large vehicles in this application apparently correspond to support vehicles in the VRPTT.

As seen, no model presented in the literature so far is appropriate for the VRPTT. Therefore, in the following section, a formulation for the complete problem described in the previous subsection is devised from scratch. In particular, except for the multi-period problem Scheuerer 2004 considers and the multi-objective model of Tan et al. 2006, the subsequent formulation contains the VRPs with trailers treated by the above-mentioned authors as special cases.

4.3 Mixed Integer Programming Formulations

4.3.1 Representation of the Data

As explained in Chapter 1, a system of sets and functions is used to represent the data defining a problem instance.

There are the following basic sets:

- a set RWL of *real-world locations*,
- a set V of *vertices*,
- a set A of *arcs*,
- and a set F of *vehicles* (the *fleet*).

RWL simply contains (at least) one element for each relevant real-world location. It is composed of four disjoint subsets. As described in the introduction, there is a set RWL_S of service stations, a set RWL_T of transshipment locations, a set RWL_P of parking locations, and a set $RWL_C := RWL_{C_L} \dot\cup RWL_{C_{LT}}$ of real-world customer locations. RWL_{C_L} is the set of real-world lorry customers, $RWL_{C_{LT}}$ is the set of real-world trailer customers. If, for example, a real-world location rwl is the location of a customer, and if rwl can also be used as a transshipment location, RWL contains two elements corresponding to rwl: one in RWL_C and one in RWL_T.

$T^{max} \in \mathbb{R}_+$ is the *length of the planning horizon*. $TW := \{[a, b] : a, b \in [0, T^{max}], a \leqq b\}$.

RWL has the following attribute:

- $tw : RWL \to \mathscr{P}(TW)$. tw_l is the *set of visiting time windows of* l. All time windows in tw_l are disjoint, i.e., tw_l fulfils $[a, b], [a', b'] \in tw_l \Rightarrow b < a' \vee b' < a$.

$RWL_S \cup RWL_T$ has the following attribute:

- $tw^{service} : RWL_S \cup RWL_T \to \mathscr{P}(TW)$. $tw_l^{service}$ is the *set of service time windows of* l. For each time window in $tw_l^{service}$, there is a corresponding *surrounding visiting time window* in tw_l, i.e. $tw_l^{service}$ fulfils $\exists\, [a', b'] \in tw_l$ with $a' \leqq a, b \leqq b'\ \forall\, [a, b] \in tw_l^{service}$.

RWL_C has the following attributes:

- $cs : RWL_C \to \mathbb{N}$. cs_l is the *supply of* l.
- $vf : RWL_C \to \mathbb{N}$. vf_l is the *visiting frequency of* l.
- $vf^{tw} : RWL_C \to \{(S, n) \in \mathscr{P}(TW) \times \mathbb{N}_0\}$. vf_l^{tw} is the *set of all pairs of subsets* S *of* l*'s visiting time window set and numbers* n *of visits of* l *that must occur within the time windows in* S.
- $vd^{min} : RWL_C \to [0, T^{max}]$. vd_l^{min} is the *minimal temporal distance between two consecutive visits of* l. Note that a *maximal* temporal distance is implicitly given by the visiting frequency.

An element of the vertex set V represents a combination of a real-world location, a time window and possibly a vehicle. There are the following subsets of V:

- V_D, the *set of depot vertices*. A vertex in this set represents a combination of a real-world location $l \in RWL$, one of its time windows, and a vehicle $k \in F$. There is not necessarily a vertex in V_D for each pair $(l, k) \in RWL \times F$. V_D itself is comprised of two disjoint subsets. The first is V_{D_P}, the set of *parking depot vertices*. An element of this subset represents a potential depot where no load transfer is possible. For each real-world parking location and for each real-world service station and each of their respective visiting time windows, there is a parking depot vertex for each vehicle that may use this real-world location as its depot. The second is V_{D_T}, the set of *transshipment depot vertices*. At a potential depot represented by a vertex in this subset, a load transfer can be performed. For each real-world transshipment location and each service station and each of their respective service time windows, there is a transshipment depot vertex for each vehicle that may use this real-world location as its depot.
  The formulation is supposed to select exactly one vertex of V_D as the actual depot vertex for each vehicle used. If there is only one vertex for each physical depot location, a load transfer at a depot vertex is very hard to model.

- V_I, the *set of intermediate vertices*. Like V_D, V_I can be decomposed into two disjoint subsets: V_{I_P}, the *set of parking intermediate vertices*, and V_{I_T}, the *set of transshipment intermediate vertices*. For each real-world parking location and each real-world service station and each of their respective visiting time windows, there is one parking intermediate vertex. For each real-world transshipment location and each of its service time windows, there is a transshipment intermediate vertex for each vehicle.
  Similar to the case of the depot locations, if there is only one vertex for each transshipment location, the temporal synchronization of the load transfers is overly complicated.
  Parking depot vertices and parking intermediate vertices are henceforth referred to as *parking vertices*. Likewise, the term *transshipment vertices* is used to denote transshipment depot vertices and transshipment intermediate vertices.

- V_C, the *set of customer vertices*. For each visiting time window of each customer, there is at least one element in V_C. If a customer's visiting time windows, visiting frequency, and minimal temporal distance between two consecutive visits are such that more than one visit within one visiting time window is possible, there are as many vertices for each such time window as there may be visits within this time window. This is because the total number of visits at a customer vertex by all lorries is supposed to be less than or equal to one.
  V_C consists of two disjoint subsets, V_{C_L}, the *set of lorry customer vertices*, and $V_{C_{LT}}$, the *set of trailer customer vertices*.

- V_S, the *set of service station vertices*. For each real-world service station and each of its service time windows, there is a vertex in V_S. V_S is comprised of the three disjoint subsets V_U, the *set of unloading station vertices*, V_M, the *set of maintenance station vertices*, and V_{UM}, the *set of combi station vertices*. V_U contains the vertices corresponding to service stations where unloading is possible and maintenance is not, V_M contains the vertices corresponding to service stations where unloading is not possible but maintenance is, and V_{UM} contains the vertices corresponding to service stations where both unloading and maintenance are possible.

The arc set A represents transitions from one vertex to another. It follows from the definition of the vertex set that these transitions may be spacial as well as temporal. A contains an element for each ordered pair $(i, j) \in V \times V$ with $i \neq j$. Thus, (V, A, ta, he) constitutes a simple, complete digraph D. (It may be impossible to visit a vertex j directly after a vertex i, e.g., due to time window constraints, but for modelling purposes, this does not matter.) Due to the construction of the vertex set, D may be called a *time-space-vehicle network*.

The set of vehicles, F, contains an element for each available vehicle. F can be decomposed as follows. $F = F_L \cup F_T$, where F_L is the *set of lorries* and F_T is the *set of trailers*. Also, $F = F_C \cup F_S$, where F_C is the *set of collection vehicles*, and F_S is the *set of support vehicles*. Furthermore, $F_{CL} := F_L \cap F_C$ is the *set of collection lorries*, $F_{CT} := F_T \cap F_C$ is the *set of collection trailers*, $F_{SL} := F_L \cap F_S$ is the *set of support lorries*, and $F_{ST} := F_T \cap F_S$ is the *set of support trailers*.

For all $k \in F$, let V^k (A^k) be the set of vertices (arcs) that can be reached (traversed) by vehicle k. For all subsets of V (A) defined in this chapter, the superscript k denotes the intersection of the respective subset with V^k (A^k).

There are the following vertex attributes:

- $loc : V \to RWL$. loc_i is the *real-world location corresponding to vertex* i.
  loc represents one part of the relationship between the vehicle and the real-world location represented by a vertex in $V_D \cup V_{I_T}$.
  For all subsets $S \subseteqq V$, S^l denotes the subsets of S whose elements correspond to identical real-world locations l: $S^l := \{i \in S : loc_i = l\} \; \forall \, l \in RWL$.

- $twv : V \to \{[a, b] : a, b \in [0, T^{max}], a \leqq b\}$. $[twa_i, twb_i] := twv_i$, and twa_i (twb_i) is the *opening* (*closing*) *time of* i.
  twv_i is one of the time windows of loc_i, and it is called *static* or *fixed time window of* i. For transshipment vertices, there are also *dynamic* or *variable time windows*, see below.

- $vl : V_D \cup V_{I_T} \to F$. vl_i is the *vehicle associated with* i.
  As stated above, each vertex $i \in V_D \cup V_{I_T}$ represents a combination of a real-world location and a vehicle. vl represents one part of this relationship.
  At any transshipment vertex i, the vehicle vl_i is considered the *passive* vehicle, the one to or from which load is transferred, and the other vehicles entering i are the *active* ones; they are responsible for performing the load transfer. Hence, it does not matter whether vl_i is a collection lorry or not. For a load transfer at i, a collection lorry other than vl_i must be present.

It is useful to further subdivide V_D and V_{I_T} into subsets representing the potential depot and intermediate vertices for a fixed vehicle k: $V_D^k := \{i \in V_D : vl_i = k\}$, and $V_{I_T}^k := \{i \in V_{I_T} : vl_i = k\}$. Also define $V_{D_P}^k := \{i \in V_{D_P} : vl_i = k\}$.

- $su : V \to \mathbb{Z}_+^0$. su_i is the *supply of vertex* i, and, equivalently, the *supply per visit of* rwl_i. su is defined on the complete vertex set for convenience. $su_i := 0$ for all $i \notin V_C$.
- $us : V_U \cup V_{UM} \to \mathbb{Z}_+^0$. us_i is the *unloading speed at unloading or combi station* i.
- $dus : V_U \cup V_{UM} \to \mathbb{Z}_+^0$. dus_i is the *minimum amount of load that must be delivered to unloading or combi station* i.
- $c^{fix,L} : V_{D_T} \cup V_{I_T} \to \mathbb{Z}_+^0$. $c^{fix,L}$ are the *setup costs of transshipment vertex* i.
- $c^{dec} : V_I \to \mathbb{Z}_+^0$. c_i^{dec} are the *decoupling costs at intermediate vertex* i.
  It is assumed that each time a trailer reaches a parking vertex, it is decoupled, and it may also be decoupled at its transshipment vertices and at service stations. c^{dec} models the 'inconvenience costs' incurred by decoupling a trailer and coupling it again later.

There are the following arc attributes:

- $\tau^{tr} : A \to \mathbb{Z}_+^0$. τ_{ij}^{tr} is the *traversal time of arc* (i, j).
  For all arcs (i, j) connecting vertices representing the same physical location, i.e. with $loc_i = loc_j$, τ_{ij}^{tr} is equal to zero.
- $d^{tr} : A \to \mathbb{Z}_+^0$. d_{ij}^{tr} is the *length of* (i, j).
  Similarly to τ^{tr}, for all arcs (i, j) connecting vertices representing the same physical location, d_{ij}^{tr} is equal to zero.

There are the following fleet attributes:

- $q : F \to \mathbb{Z}_+^0$. q_k is the *capacity of vehicle* k.
- $acc : F \to \mathscr{P}(V)$. acc is the *vertex accessibility function*. For a vehicle k, acc_k denotes the subset of vertices it can reach. acc is constructed such that customers are not accessible to support vehicles, lorry customers are not accessible to trailers, and parking depot vertices of trailers are not accessible to other trailers. Any additional accessibility constraints for certain vehicles at customers (e.g., due to the size of a vehicle or the fact that the capacity of a lorry is less than the supply of a lorry customer) are also considered in the definition of acc.
- $com : F \to \mathscr{P}(F)$. com is the *vehicle compatibility function*. For a lorry k, com_k is the *set of trailers which lorry* k *can pull*; for a trailer k', $com_{k'}$ is the *set of lorries which can pull trailer* k'.
  If it is desired that a trailer be always pulled by one and the same lorry, com can be defined accordingly.
- $\tau^{lt} : F_L \to \mathbb{Z}_+$. τ_k^{lt} is the *load transfer time of lorry* k *per unit of load*. For support lorries k, $\tau_k^{lt} = T^{max}$.
- $mf : F \to \mathbb{Z}_+^0$. mf_k is the *maintenance frequency of* k.
- $td^{min} : F \to \mathbb{Z}_+$. td_k^{min} is the *minimal temporal distance between two consecutive maintenance processes for* k.
- $c^{fix,F} : F \to \mathbb{Z}_+^0$. $c_k^{fix,F}$ are the *fixed costs of* k.
- $c^{time} : F_L \to \mathbb{Z}_+^0$. c_k^{time} are the *time-dependent costs of* k or the *costs of* k *per unit of time*.
- $c^{dist} : F \to \mathbb{Z}_+$. c^{dist} are the *distance-dependent costs of* k or the *costs of* k *per unit of length*.

The idea behind the last three attributes is that $c^{fix,F}$ indicates the costs of *making a vehicle available during the planning horizon*, whereas c^{time} represents the costs determined by the *temporal length of use during the planning horizon*, and c^{dist} models the costs that depend on the *spacial length of each vehicle's itinerary*.

There is the following composite attribute:

- $\tau^m : F \times V_{UM} \cup V_M \to \mathbb{Z}_+$. τ^m_{ki} is the *maintenance time of k at combi or maintenance station i*, i.e., the temporal duration of a maintenance process.

Some remarks are appropriate:

A trailer must normally be pulled by some compatible lorry when traversing an arc, but there are two notable exceptions:

(i) A very tricky aspect of transshipment locations is the following: A load transfer to a trailer at one of these locations is only possible during a service time window. However, the trailer may *park* at such a location before and after the service time window. This means that the lorry pulling the trailer to and the lorry pulling the trailer away from such a location may reach the location before, respectively, after the service time window. For the corresponding vertices, this implies that the trailer and the lorry pulling it may reach such vertices before the opening and after the closing time. Therefore, in the model, a trailer being pulled to a transshipment location before this location's service time window (this can be sensible because the lorry pulling the trailer may be needed for something else afterwards) is parked at the respective parking vertex and moves to the transshipment vertex independently when the service time begins (and a load transfer is to be performed). If the trailer is coupled at a transshipment location after the service time window, in the model it moves to the respective parking vertex at the end of the service time window. Arcs connecting vertices with disjoint time windows cannot be traversed by a single trailer. Thus, it is clear which parking vertex will be used by the model for correctly routing a trailer. For transshipment depot vertices, if a load transfer is performed, this implies that the receiving trailer is assigned to the respective depot and must leave the vertex anyway. Hence, if the trailer is to be coupled after the closing time, it leaves the transshipment depot vertex for the corresponding parking vertex.

(ii) For depot vertices, it is assumed that a vehicle vl_i may return to its assigned depot vertex i at any time, even after twb_i. This implies that the lorry k pulling a trailer vl_i back to its assigned depot vertex i must be allowed to enter i after twb_i, too. This problem is circumvented by allowing trailers vl_i to traverse arcs (h, i) leading from a parking intermediate vertex h with $loc_h = loc_i$ to vl_i's assigned depot vertex i without being pulled by a lorry. It must then be ensured that, for each physical location for which a depot vertex is created, there is also a parking vertex with a closing time of T^{max}.

In essence, trailers k' are allowed to traverse arcs (h, i) and (i, j) for $i \in V^{k'}_{I_T}$, $h, j \in V_{I_P}$ with $rwl_i = rwl_h = rwl_j$ and arcs (h, i) for $h \in V_{I_P}$, $i \in V^{k'}_D$, $rwl_h = rwl_i$ without being pulled by a lorry. The set of such arcs for a trailer k' is $A^{single}_{k'}$.

One little problem that remains is the following. Simultaneous load transfer of two different vehicles to a lorry and its attached trailer at the same physical transshipment location is possible: One vehicle transfers load to the lorry, the other vehicle transfers load to the trailer. It is not necessary to decouple the trailer from the lorry in this case. In the model, however, this is represented by the lorry decoupling the trailer at the latter's corresponding transshipment vertex, moving singly to its own transshipment vertex, and re-coupling the trailer again afterwards, incurring decoupling costs. This is a minor defect and not worth further modelling efforts.

'Congestion' at depots or intermediate locations, i.e., the possibility of having too many vehicles at a physical location at the same time, is not considered. However, congestion *is* modelled as far as load transfer and unloading processes are concerned.

The issue of a maximum tour duration is not considered in the formulation presented subsequently. This formulation allows multiple use of vehicles, i.e., the possibility of a vehicle unloading at a depot and starting a new tour. The planning horizon is typically one or two days (24 or 48 hours), which means that it is too long for one driver to drive a vehicle during the complete planning horizon. Nevertheless, the planning horizon limits the time en route of a vehicle. This means that there must be a change of driver at some point in time. In practice, such a change occurs in accordance with legal working time regulations. It has no effect on the costs of a tour.

4.3.2 A Formulation for the Complete Problem Based on Turn Variables

4.3.2.1 Assumptions

There are two fundamental assumptions in the formulation presented in this section:

(i) The first assumption is that *there is a transshipment vertex for each combination of* (*pertinent*) *physical location*, *time window*, *and vehicle*, and that *each vehicle visits each of its transshipment vertices at most once*. To see why this is a good idea, consider the possible alternatives:

- If there are as many transshipment vertices as there are combinations of location, time window, and passive vehicle, allowing that each transshipment vertex be visited by any trailer (but at most one) requires introducing many additional variables (compared to the subsequent formulation), namely, variables modelling vehicle movements to and from all transshipment vertices i are necessary for vehicles $k \neq vl_i$.
- Having one transshipment vertex for each time window of each physical location means that each transshipment vertex may have more than one passive vehicle. It is then difficult to say to which passive vehicle an active vehicle transfers its load.

If it is considered too restrictive an assumption that each vehicle visits each of its transshipment vertices at most once during each of the respective location's time windows, duplicates of the relevant vertices can be introduced.

(ii) The second assumption is that *every arc is traversed by each vehicle at most once*. The significance of this postulate is discussed in Section 4.3.5. If a vehicle shall be allowed to travel from vertex i to vertex j more than once, parallel arcs can be introduced. D is assumed to be simple only for simplicity.

4.3.2.2 Variables

The concepts of turns and of turn variables have been introduced in the previous chapter. The networks used in this chapter are simple, so the notation for turns can be simplified: In a simple digraph, a turn is uniquely described by an ordered sequence of three vertices. If a vehicle traverses an arc (i, j) immediately after an arc (h, i), it performs a turn t in vertex i, and this turn is denoted $t = (h, i, j)$. As before, T is used to denote the set of turns in D.

The formulation uses the following variables:

- $x^k_{hij} \in \{0, 1\} \; \forall \; k \in F, (h, i), (i, j) \in A$.
  $x^k_{hij} = \begin{cases} 1, \text{ vehicle } k \text{ traverses arc } (i, j) \text{ immediately after arc } (h, i) \\ 0, \text{ otherwise} \end{cases}$
  The x^k_{hij} are the *turn variables*.

- $x^k \in \{0,1\}\ \forall\, k \in F$.
  $x^k = \begin{cases} 1, \text{ vehicle } k \text{ is used} \\ 0, \text{ otherwise} \end{cases}$

- $x_{ij}^{kk'} \in \{0,1\}\ \forall\, k \in F_L, k' \in com_k, (i,j) \in A$.
  $x_{ij}^{kk'} = \begin{cases} 1, \text{ lorry } k \text{ pulls trailer } k' \text{ over arc } (i,j) \\ 0, \text{ otherwise} \end{cases}$
  The $x_{ij}^{kk'}$ are called *lorry-trailer arc variables*.

- $x_{hi}^{dec,k'} \in \{0,1\}\ \forall\, k' \in F_T, i \in \{i' \in V_{I_T} : k' = vl_{i'}\} \cup V_S, (h,i) \in A$.
  $x_{hi}^{dec,k'} = \begin{cases} 1, \text{ trailer } k' \text{ is decoupled at vertex } i \text{ after traversing arc } (h,i) \\ 0, \text{ otherwise} \end{cases}$

- $y_{hi}^{k,m} \in \{0,1\}\ \forall\, k \in F, m = 1,\ldots,mf_k, i \in V_{UM} \cup V_M, (h,i) \in A$.
  $y_{hi}^{k,m} = \begin{cases} 1, \text{ vehicle } k \text{ undergoes its } m\text{th maintenance at vertex } i \text{ coming from vertex } h \\ 0, \text{ otherwise} \end{cases}$

- $l_{hi}^k \in \mathbb{R}_+^0\ \forall\, k \in F, (h,i) \in A$.
  The amount of load vehicle k is carrying immediately after traversing arc (h,i).

- $t_{hi}^k \in \mathbb{R}_+^0\ \forall\, k \in F, (h,i) \in A$.
  The point in time when vehicle k begins its service at vertex i after traversing arc (h,i).

- $t_{hi}^{lt,k'} \in \mathbb{Z}_+\ \forall\, i \in V_{D_T} \cup V_{I_T}, vl_i \neq k' \in F_T, (h,i) \in A$.
  The load transfer time of trailer k' at transshipment vertex i, after k' has reached i via arc (h,i). There is exactly one lorry k that has pulled k' to i via (h,i). By the $x_{hi}^{kk'}$ variables, k then determines the load transfer time of k' at i.

- $t^{k,m} \in \mathbb{R}_+^0\ \forall\, k \in F, m = 1,\ldots,mf_k$.
  The point in time when the mth maintenance of k begins.

- $t_{ij}^{d,k} \in \mathbb{R}_+^0\ \forall\, k \in F, (i,j) \in A$.
  The point in time when vehicle k departs from vertex i heading for vertex j.

- $z_{hi}^{k+} \in \mathbb{R}_+^0\ \forall\, k \in F, i \in V_{D_T} \cup V_{I_T} \cup V_U \cup V_{UM}, (h,i) \in A$.
  For $i \in V_{D_T} \cup V_{I_T}$, the amount of load transferred from vehicle k to vehicle vl_i at vertex i after vehicle k has reached i via arc (h,i).
  For $i \in V_U \cup V_{UM}$, the amount of load unloaded from vehicle k at service station i after vehicle k has reached i via arc (h,i).

- $z_{hi}^{k-} \in \mathbb{R}_+^0\ \forall\, k \in F, i \in V_{D_T} \cup V_{I_T} \cup V_U \cup V_{UM}, (h,i) \in A$.
  For $i \in V_{D_T} \cup V_{I_T}$, the amount of load transferred from vehicle vl_i to vehicle k at vertex i after vehicle k has reached i via arc (h,i).
  For $i \in V_U \cup V_{UM}$, the amount of load collected by vehicle k at service station i after vehicle k has reached i via arc (h,i).
  These variables model the possibility of a load transfer from a support lorry or a trailer to a collection lorry or a trailer currently pulled by a collection lorry as well as the possibility of transports between unloading and combi stations.

- $z_i \in \mathbb{R}_+^0\ \forall\, i \in V_{C_{LT}}$.
  The amount of load transferred at trailer customer i. One index is sufficient here because at most one lorry and one trailer visit such a vertex.

All turn variables x_{hij}^k and all lorry-trailer arc variables $x_{ij}^{kk'}$ where h, i, or j are not accessible to the vehicles k and k' are equal to zero. Hence, these variables are not necessary; they were defined only for convenience.

The load and time variables are also called *resource variables*.

The basic idea behind the time variables t_{hi}^k is the so-called *implicit waiting concept*: At each vertex a vehicle reaches, the latter must perform a service. This service may consist in doing nothing and/or may take no time. For example, a vehicle parked at a parking vertex actually does nothing, but it does so for a positive amount of time. A lorry re-coupling a trailer does perform a service, but one which is considered to take no time. Every vertex accessible to a certain vehicle can be reached by the vehicle at any time. If the vertex is reached before its opening time, the vehicle waits until it can begin its service. The time a vehicle spends waiting is included in the t_{hi}^k variables; it is not computed explicitly. There is no distinction between the arrival time at a vertex and the beginning of the service at this vertex. Additional variables would be necessary for this. The total waiting time of a vehicle can be easily computed from the total duration of its itinerary, the total driving and the total service time. Therefore, the $t_{ij}^{d,k}$ variables can also be interpreted as the point in time when vehicle k ends its service at vertex i. If $x_{hij}^k = 1$, k will leave i for j, but the point in time when it does so may be later than $t_{ij}^{d,k}$. If any waiting occurs, it is not specified *where* this waiting takes place: at i, at j, or somewhere on the arc (i, j).

It can be sensible to unload a vehicle only partially at an unloading or combi station, to unload only the lorry or only the trailer of a completely loaded lorry-trailer combination that reaches an unloading or combi station, or to perform direct transports between unloading stations, and it can even be *necessary* to do so if there are minimum demands at unloading or combi stations that must be satisfied. Hence, the $z_{hi}^{k\pm}$ variables are also defined for unloading and combi stations.

The following objects completely describe a load transfer:

- the location where the load transfer takes place
- the point in time when the load transfer begins
- the passive vehicle, the one to or from which load is transferred
- the active vehicle, the one which initiates the load transfer
- the vehicle providing the technical load transfer equipment; this vehicle determines the load transfer speed
- the amount of load transferred; this quantity, together with the load transfer speed, determines the duration and, hence, the end of a load transfer

How is this information represented in the data and the formulation? The location and the passive vehicle are represented by the set V_{I_T} of transshipment intermediate vertices. The other quantities are decision variables: The t_{hi}^k variables indicate the beginning, the k index of the $z_{hi}^{k\pm}$ variables implies the active vehicle, the $z_{hi}^{k\pm}$ variables themselves indicate the amount of load transferred and the direction of the transfer, and the vehicle determining the load transfer speed is either vehicle k, if it is a (collection) lorry, or the collection lorry pulling vehicle (trailer) k to i over the arc (h, i) (there are constraints for ensuring this, see below).

The itinerary of a vehicle k is a directed cycle C in D. When the time variables are taken into account, it starts and ends at the unique vertex $i \in V_D$ of C with $vl_i = k$.

4.3.2.3 Objective Function

The following objective function applies:

$$\sum_{i \in V_{D_T} \cup V_{I_T}} c_i^{fix,L} \sum_{(h,i,j) \in T} x_{hij}^{vl_i} + \sum_{k' \in F_T} \sum_{i \in \{i' \in V_{I_T} : k' = vl_{i'}\} \cup V_S} \sum_{(h,i) \in A} c_i^{dec} x_{hi}^{dec,k'} + \sum_{k' \in F_T} \sum_{i \in V_{I_P}} c_i^{dec} \sum_{(h,i,j) \in T} x_{hij}^{k'}$$

$$+\sum_{k\in F} c_k^{fix,F} x^k + \sum_{k\in F}\sum_{(i,j)\in A} d_{ij}^{tr} c_k^{dist} \sum_{(h,i,j)\in T} x_{hij}^k + \sum_{k\in F_L}\left[\sum_{i\in V_D^k}\left(\sum_{(h,i)\in A} t_{hi}^k - \sum_{(i,j)\in A} t_{ij}^{d,k}\right)\right] c_k^{time} \to \min \qquad (4.1)$$

The first line of the objective function contains the location-dependent costs, the second line contains the vehicle-dependent costs.

The last summand in the above term deserves explanation. The total 'working time' of a lorry is what is relevant for the time-dependent costs. This total time is not necessarily equal to the value of the t_{hi}^k variable for the arc (h, i) via which the lorry returns to its assigned depot vertex i, because the lorry may have started its itinerary later than time zero. (It will not start earlier than necessary precisely because of the time-dependent costs, which make it desirable to minimize the total time en route.) The factor to the left of c_k^{time} takes this into account.

4.3.2.4 Constraints

The following constraints apply:

Each vehicle must be present to be used, and it must be assigned to exactly one depot:

$$\sum_{\{i\in V_D : vl_i = k\}} \sum_{(h,i,j)\in T} x_{hij}^k \leqq x^k \quad \forall\, k \in F \qquad (4.2)$$

Flow conservation:

$$\sum_{(h,i,j)\in T} x_{hij}^k - \sum_{(i,j,j')\in T} x_{ijj'}^k = 0 \quad \forall\, k \in F, (i,j) \in A \qquad (4.3)$$

Only if the vehicle vl_i leaves the potential depot vertex i can another vehicle enter i:

$$x_{hij}^k \leqq \sum_{(h',i,j')\in T} x_{h'ij'}^{vl_i} \quad \forall\, k \in F, i \in V_D, (h,i,j) \in T \qquad (4.4)$$

Each vehicle uses each of its transshipment vertices at most once:

$$\sum_{(h,i,j)\in T} x_{hij}^{vl_i} \leqq 1 \quad \forall\, i \in V_{I_T} \qquad (4.5)$$

Only if the vehicle vl_i reaches its transshipment vertex i can another vehicle enter i:

$$x_{hij}^k \leqq \sum_{(h',i,j')\in T} x_{h'ij'}^{vl_i} \quad \forall\, k \in F, i \in V_{I_T}, (h,i,j) \in T \qquad (4.6)$$

Each customer vertex is visited by at most one lorry:

$$\sum_{k\in F_L} \sum_{(h,i,j)\in T} x_{hij}^k \leqq 1 \quad \forall\, i \in V_C \qquad (4.7)$$

Each customer l is visited exactly vf_l times:

$$\sum_{i\in V_C^l} \sum_{k\in F_L} \sum_{(h,i,j)\in T} x_{hij}^k = vf_l \quad \forall\, l \in RWL_C \qquad (4.8)$$

Constraints on the number of visits at customer l in specified subsets of l's time windows:

$$\sum_{\{i\in V_C^l : twv_i \in S\}} \sum_{k\in F_L} \sum_{(h,i,j)\in T} x_{hij}^k \geqq n \quad \forall\, l \in RWL_C, (S,n) \in vf_l^{tw} \qquad (4.9)$$

Constraints on the minimal time between two visits at a customer:

$$x^k_{hij} = 1 \ \wedge \ x^{\tilde{k}}_{h'i'j'} = 1 \Rightarrow t^k_{hi} + vd^{min}_l \leqq t^{\tilde{k}}_{h'i'} \quad \forall\, k, \tilde{k} \in F_L, i, i' \in V_C, loc_i = loc_{i'}, \\ twb_i < twa_{i'}, (h, i, j), (h', i', j') \in T \tag{4.10}$$

These constraints are equivalent to (omitting the '∀' section for simplicity)

$$x^k_{hij} + x^{\tilde{k}}_{h'i'j'} = 2 \Rightarrow t^k_{hi} + vd^{min}_l \leqq t^{\tilde{k}}_{h'i'}, \tag{4.10a}$$

which can be linearized as follows:

$$t^k_{hi} + vd^{min}_l + T^{max}(x^k_{hij} + x^{\tilde{k}}_{h'i'j'} - 2) \leqq t^{\tilde{k}}_{h'i'}. \tag{4.10b}$$

In standard formulations for VRPs, i.e., formulations using arc variables, subtour elimination constraints are necessary to include the depot in each vehicle's tour(s). Are such constraints also needed when using turn variables? Consider Figure 4.4. If all four vertices must be visited, all four depicted arcs must be traversed, and there is exactly one cycle constituting a tour. With arc variables, no subtours arise. With turn variables, though, it is possible to visit all four vertices when the indicated turns t_1, t_2, t_3 are performed, but this leads to two subtours. However, it is impossible to perform both t_2 and t_3 if the sum of the travel times on the arcs and the service times at the vertices of any cycle is strictly positive. Consequently, if this is the case, no subtours are possible and no subtour elimination constraints are necessary. Unfortunately, in the formulation considered in this section, there may be cycles with a spacial and temporal length of zero. For example, a lorry k with an attached trailer k' might reach a parking vertex i, decouple k', go to an unloading station i' with $loc_{i'} = loc_i$, couple a trailer $\tilde{k}$, return to i, decouple $\tilde{k}$, couple k', and leave for somewhere else (cf. Figure 4.4). With the arc attributes τ^{tr} and d^{tr} defined as above, the lorry can perform this cycle in zero time units and travel zero length units. If all three vehicles are empty, the load variables are not affected either.

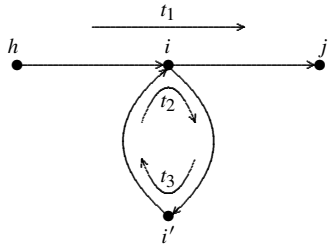


Figure 4.4: Turn subtour

There are three ways to deal with this situation:

(i) One way is to set τ^{tr} and d^{tr} to very small positive values ϵ for all arcs (i, j) connecting vertices representing the same physical location, i.e., with $loc_i = loc_j$.

(ii) The second way is to leave everything as it is and accept the possibility of turn subtours. Such subtours can occur only for cycles with a length of zero, so they do not affect the optimal solution value, and neither do they affect any other variables. The arcs any vehicle traverses remain the same. Only if a vehicle's itinerary is to be reconstructed from the values of the x^k_{hij} variables of a solution must care be taken. The Hierholzer Algorithm (Hierholzer 1873) can be applied in this case: For each vehicle, its assigned depot vertex must be found out (which is not difficult), the arc emanating from this vertex used by the vehicle must be traversed, and (the) other arc(s) emanating from the head of the first arc it uses must be traversed in the sequence the time and the turn variables prescribe. At vertices the vehicle leaves more than once at the same time, subtours may occur. Such vertices are marked. If a cycle is generated that does not contain all arcs the vehicle uses according to the x^k_{hij} variable values, the zero-length subtours starting at marked vertices are successively included in the tour containing the assigned depot vertex.

(iii) The third way is to include subtour elimination constraints for the turns. To do this, it must be observed that the depot (vertex) for each vehicle is not known in advance; it is determined during the solution. Hence, to ensure connectivity of a vehicle's itinerary, it must be ensured that, for each subset of vertices not including the respective assigned depot vertex, if the vehicle performs a turn involving two vertices of this subset (traverses an arc between two vertices of this subset), then it also performs a turn whose first two vertices belong to this subset and whose third vertex does not (traverses an arc leaving this subset). Constraints to ensure this are very complicated to formulate, so the other two alternatives are clearly superior.

Linking of turn and lorry-trailer arc variables:
A trailer k' must normally be pulled by some compatible lorry when traversing an arc, with the exception of the arcs in $A_{k'}^{single}$:

$$\sum_{(h,i,j)\in T} x_{hij}^{k'} = \sum_{k\in com_{k'}} x_{ij}^{kk'} \quad \forall\, k' \in F_T, (i,j) \in A \setminus A_{k'}^{single} \tag{4.11}$$

$$\sum_{(h,i,j)\in T} x_{hij}^{k} \geqq \sum_{k'\in com_{k}} x_{ij}^{kk'} \quad \forall\, k \in F_L, (i,j) \in A \tag{4.12}$$

Linking of lorry-trailer arc and time variables:

$$x_{hi}^{kk'} = 1 \Rightarrow t_{hi}^{k} \leqq t_{hi}^{k'} \quad \forall\, k' \in F_T, k \in com_{k'}, (h,i) \in A \tag{4.13}$$

$$x_{ij}^{kk'} = 1 \Rightarrow t_{ij}^{d,k} = t_{ij}^{d,k'} \quad \forall\, k' \in F_T, k \in com_{k'}, (i,j) \in A \tag{4.14}$$

The first implication can be rewritten as

$$t_{hi}^{k} - (1 - x_{hi}^{kk'})T^{max} \leqq t_{hi}^{k'}. \tag{4.13a}$$

The second implication can be expressed as

$$x_{ij}^{kk'} = 1 \Rightarrow t_{ij}^{d,k} \leqq t_{ij}^{d,k'} \wedge t_{ij}^{d,k} \geqq t_{ij}^{d,k'}, \tag{4.14a}$$

or

$$t_{ij}^{d,k} - T^{max}(1 - x_{ij}^{kk'}) \leqq t_{ij}^{d,k'} \wedge t_{ij}^{d,k} + T^{max}(1 - x_{ij}^{kk'}) \geqq t_{ij}^{d,k'}. \tag{4.14b}$$

Linking of turn, load, and time variables:

$$x_{hij}^{k} = 0 \Rightarrow l_{ij}^{k} = t_{ij}^{k} = t_{ij}^{d,k} = 0 \quad \forall\, k \in F, (h,i,j) \in T \tag{4.15}$$

Decoupling a trailer k' is not allowed at depot vertices and transshipment intermediate vertices i with $vl_i \neq k'$:

$$x_{hij}^{k'} = 1 \;\wedge\; x_{hi}^{kk'} = 1 \Rightarrow x_{hij}^{k} = 1 \;\wedge\; x_{ij}^{kk'} = 1 \quad \forall\, i \in V_D \cup V_{I_T}, vl_i \neq k' \in F_T, k \in com_{k'}, (h,i,j) \in T \tag{4.16}$$

x_{hij}^{k} and $x_{ij}^{kk'}$ are binary variables. Therefore, the '=' signs to the right of the implication can be replaced by '$\geqq$', and the constraints are equivalent to

$$x_{hij}^{k'} + x_{hi}^{kk'} - 1 \leqq x_{hij}^{k} \wedge x_{hij}^{k'} + x_{hi}^{kk'} - 1 \leqq x_{ij}^{kk'}. \tag{4.16a}$$

These constraints also make sure that, if there is a load transfer from a trailer k' to another vehicle, the lorry k (which is assumed to be a collection lorry, see above) which has pulled k' to the respective vertex is present during the load transfer.

Decoupling is also not allowed at customer vertices. This need not be explicitly required: A trailer which is used must return to its assigned depot vertex. If a trailer is decoupled at a trailer customer, it cannot be coupled again, as at most one lorry visits a customer vertex.

Decoupling and coupling take place at parking vertices:

$$x_{hij}^{k'} = 1 \Rightarrow x_{hi}^{dec,k'} = 1 \quad \forall\, k' \in F_T, i \in V_{I_P}, (h,i,j) \in T \tag{4.17}$$

Decoupling and coupling may take place at transshipment intermediate vertices i for trailer vl_i and at service stations :

$$x_{hij}^{k'} = 1 \ \wedge\ x_{hi}^{kk'} = 1 \ \wedge\ x_{ij}^{kk'} = 0 \Rightarrow x_{hi}^{dec,k'} = 1 \quad \forall\, k' \in F_T, k \in com_{k'},$$
$$i \in \{i' \in V_{I_T} : k' = vl_{i'}\} \cup V_S, (h,i,j) \in T \tag{4.18}$$

These constraints are equivalent to

$$x_{hij}^{k'} + x_{hi}^{kk'} + (1 - x_{ij}^{kk'}) - 2 \leqq x_{hi}^{dec,k'}. \tag{4.18a}$$

Capacity constraints:

$$l_{hi}^{k} \leqq q_k \sum_{(h,i,j)\in T} x_{hij}^{k} \quad \forall\, k \in F, (h,i) \in A \tag{4.19}$$

Static time windows:

$$x_{hij}^{k} = 1 \Rightarrow twa_i \leqq t_{hi}^{k} \quad \forall\, k \in F, (h,i,j) \in T \tag{4.20}$$

$$x_{hij}^{k} = 1 \Rightarrow t_{hi}^{k} \leqq t_{ij}^{d,k} \quad \forall\, k \in F, (h,i,j) \in T \tag{4.21}$$

$$t_{ij}^{d,k} \leqq twb_i \quad \forall\, k \in F, (i,j) \in A \tag{4.22}$$

$$t_{hi}^{vl_i} \leqq T^{max} \quad \forall\, i \in V_D, (h,i) \in A \tag{4.23}$$

These last constraints make sure that all vehicles return to their assigned depot vertices before the end of the planning horizon.

Dynamic time windows:
A transshipment intermediate vertex i must be 'open' before any vehicle can perform a load transfer there, respectively, a vehicle can perform a load transfer at i only after i has 'opened', i.e., after vehicle vl_i has arrived at i:

$$x_{h'ij}^{k} = 1 \Rightarrow t_{hi}^{vl_i} \leqq t_{h'i}^{k} \quad \forall\, k \in F, i \in V_{I_T}, (h,i), (h',i) \in A, (h',i,j) \in T \tag{4.24}$$

Similarly, any load transfer must be finished by the time i 'closes', i.e., by the time vl_i leaves i, respectively, vl_i leaves i only after all vehicles have finished their service there:

$$x_{hij}^{vl_i} = 1 \Rightarrow t_{ij}^{d,vl_i} \geqq t_{ij'}^{d,k} \quad \forall\, k \in F, i \in V_{D_T} \cup V_{I_T}, (i,j), (i,j') \in A,$$
$$(h,i,j) \in T \tag{4.25}$$

For a transshipment intermediate vertex i, the time interval $[\max_{(h,i)\in A}\{t_{hi}^{vl_i}\}, \max_{(i,j)\in A}\{t_{ij}^{d,vl_i}\}]$ between the arrival of vl_i at i and the departure of vl_i from i is called *dynamic time window of* i. The dynamic time window of a transshipment depot vertex i is $[twa_i, \max_{(i,j)\in A}\{t_{ij}^{d,vl_i}\}]$.

A vehicle which is used undergoes maintenance exactly mf_k times:

$$\sum_{i \in V_{UM} \cup V_M} \sum_{(h,i) \in A} y_{hi}^{k,m} = x^k \quad \forall\, k \in F, m = 1, \ldots, mf_k \tag{4.26}$$

Restrictions on the minimal temporal distance between two consecutive maintenance processes for the same vehicle:

$$y_{hi}^{k,m} = 1 \Rightarrow t^{k,m} = t_{hi}^k \quad \forall\, k \in F, m = 1, \ldots, mf_k, (h,i) \in A \tag{4.27}$$

$$t^{k,m-1} + td_k^{min} \leqq t^{k,m} \quad \forall\, k \in F, m = 2, \ldots, mf_k \tag{4.28}$$

The lorry pulling a trailer $k' \neq vl_i$ to a transshipment vertex i determines k''s load transfer time at i:

$$x_{hi}^{kk'} = 1 \Rightarrow t_{hi}^{lt,k'} = \tau_k^{lt} \quad \forall\, k \in com_{vl_i}, i \in V_{D_T} \cup V_{I_T}, (h,i) \in A \tag{4.29}$$

No load transfer is possible from a support lorry $k \neq vl_i$ at a transshipment vertex i:

$$z_{hi}^{k+} = z_{hi}^{k-} = 0 \quad \forall\, k \in F_{SL}, i \in V_{D_T} \cup V_{I_T}, (h,i) \in A \tag{4.30}$$

Constraints for the minimum demand of the unloading stations:

$$\sum_{k \in F} \sum_{(h,i) \in A} z_{hi}^{k+} \geqq dus_i \quad \forall\, i \in V_U \cup V_{UM} \tag{4.31}$$

If, for the unloading stations, there are also restrictions on the amount of load to be delivered until a specified time, it is best to introduce an unloading station vertex for each real unloading station and each subperiod, and to modify the twv vertex attribute accordingly.

Bounds for the load transfer variables:

- At transshipment depot and intermediate vertices:

$$x_{hij}^k = 1 \Rightarrow z_{hi}^{k+} \leqq l_{hi}^k \quad \forall\, i \in V_{D_T} \cup V_{I_T}, k \in F \setminus \{vl_i\}, (h,i,j) \in T \tag{4.32}$$

$$x_{hij}^k = 1 \ \wedge\ x_{h'ij'}^{vl_i} = 1 \Rightarrow z_{hi}^{k-} \leqq l_{h'i}^{vl_i} \quad \forall\, i \in V_{D_T} \cup V_{I_T}, k \in F \setminus \{vl_i\}, (h,i,j), (h',i,j') \in T \tag{4.33}$$

  These constraints state that the amount of load transferred from a passive vehicle to an active one is not more than the load of the passive vehicle when it reaches the transshipment depot or intermediate vertex i.

- At unloading and combi stations:

$$x_{hij}^k = 1 \Rightarrow z_{hi}^{k+} \leqq l_{hi}^k \quad \forall\, k \in F, i \in V_U \cup V_{UM}, (h,i,j) \in T \tag{4.34}$$

- At trailer customer vertices:

$$x_{hij}^k = 1 \Rightarrow z_i \leqq l_{hi}^k \quad \forall\, k \in F_L, i \in V_{C_{LT}}, (h,i,j) \in T \tag{4.35}$$

  For customers i where load transfers are forbidden (if any), $z_i \leqq su_i$ is required instead.

Update of load variables:

- At depot vertices:
  When a vehicle returns to its depot, it is empty:

$$l_{hi}^{vl_i} = 0 \quad \forall\, i \in V_D, (h,i) \in A \tag{4.36}$$

When a vehicle starts from a parking depot vertex, it is empty:

$$l_{ij}^{vl_i} = 0 \quad \forall\, i \in V_{D_P}, (i, j) \in A \tag{4.37}$$

When it starts from a transshipment depot vertex:

$$x_{hij}^{vl_i} = 1 \Rightarrow l_{ij}^{vl_i} = \sum_{k \in F \setminus \{vl_i\}} \sum_{(h,i) \in A} (z_{hi}^{k+} - z_{hi}^{k-}) \quad \forall\, i \in V_{D_T}, (h, i, j) \in T \tag{4.38}$$

Vehicles k at transshipment depot vertex i with $k \neq vl_i$:

$$x_{hij}^{k} = 1 \Rightarrow l_{hi}^{k} - z_{hi}^{k+} + z_{hi}^{k-} = l_{ij}^{k} \quad \forall\, i \in V_{D_T}, k \in F \setminus \{vl_i\}, (h, i, j) \in T \tag{4.39}$$

- At parking intermediate vertices:

$$x_{hij}^{k} = 1 \Rightarrow l_{hi}^{k} = l_{ij}^{k} \quad \forall\, k \in F, i \in V_{I_P}, (h, i, j) \in T \tag{4.40}$$

- At transshipment intermediate vertices:

$$x_{hij}^{vl_i} = 1 \Rightarrow l_{hi}^{vl_i} + \sum_{k \in F} \sum_{(h',i) \in A} (z_{h'i}^{k+} - z_{h'i}^{k-}) = l_{ij}^{vl_i} \quad \forall\, i \in V_{I_T}, (h, i, j) \in T \tag{4.41}$$

$$x_{hij}^{k} = 1 \Rightarrow l_{hi}^{k} - z_{hi}^{k+} + z_{hi}^{k-} = l_{ij}^{k} \quad \forall\, k \in F \setminus \{vl_i\}, i \in V_{I_T}, (h, i, j) \in T \tag{4.42}$$

- At lorry customers:

$$x_{hij}^{k} = 1 \Rightarrow l_{hi}^{k} + su_i = l_{ij}^{k} \quad \forall\, k \in F_L, i \in V_{C_L}, (h, i, j) \in T \tag{4.43}$$

Independent of the acc fleet attribute, these constraints additionally model the possibility that a customer cannot be served by a lorry because the lorry's capacity may be less than the customer's supply.

- At trailer customers:
  - Lorries:

$$x_{hij}^{k} = 1 \Rightarrow l_{hi}^{k} + su_i - z_i = l_{ij}^{k} \quad \forall\, k \in F_L, i \in V_{C_{LT}}, (h, i, j) \in T \tag{4.44}$$

  - Trailers:

$$x_{hij}^{k'} = 1 \Rightarrow l_{hi}^{k'} + z_i = l_{ij}^{k'} \quad \forall\, k' \in F_T, i \in V_{C_{LT}}, (h, i, j) \in T \tag{4.45}$$

- At unloading and combi stations:

$$x_{hij}^{k} = 1 \Rightarrow l_{hi}^{k} - z_{hi}^{k+} + z_{hi}^{k-} = l_{ij}^{k} \quad \forall\, k \in F, i \in V_U \cup V_{UM}, (h, i, j) \in T \tag{4.46}$$

It is assumed that there is always enough supply at unloading and combi stations for a vehicle to increase its load there, in spite of the above constraints for the minimum demand of unloading and combi stations.

- At combi stations:
  A vehicle is empty when it undergoes maintenance:

$$y_{hi}^{k,m} = 1 \Rightarrow z_{hi}^{k+} = l_{hi}^{k} \quad \forall\, k \in F, m = 1, \ldots, mf_k, i \in V_{UM}, (h, i) \in A \tag{4.47}$$

- At maintenance stations:

$$x^k_{hij} = 1 \Rightarrow l^k_{hi} = l^k_{ij} \quad \forall\, k \in F, i \in V_M, (h,i,j) \in T \tag{4.48}$$

A vehicle is empty when it undergoes maintenance:

$$y^{k,m}_{hi} = 1 \Rightarrow l^k_{hi} = 0 \quad \forall\, k \in F, m = 1,\ldots,mf_k, i \in V_M, (h,i) \in A \tag{4.49}$$

Note that the load constraints are formulated as equality constraints. This accurately models reality. Given a solution to an instance, the information given by these variables can be used directly. On the other hand, it is sometimes preferable to use inequality instead of equality constraints, depending on the solution method. The load update constraints can also be formulated as inequalities. For example, the load of a vehicle after leaving a parking location is, in reality, exactly equal to the load at the moment of arrival. But it is logically sufficient to require the load to be no smaller. It will not be any larger (except if there is slack) because of the objective function.

Update of time variables:
The timing constraints are inequalities because of the implicit waiting concept.

For each depot vertex i, the point in time when vehicle vl_i departs from it must not be later than the point in time when vl_i returns there:

$$x^{vl_i}_{hij} = 1 \Rightarrow t^{vl_i}_{hi} \geqq t^{d,vl_i}_{ij} \quad \forall\, i \in V_D, (h,i,j) \in T \tag{4.50}$$

For every vehicle and every arc it traverses, the departure time of the vehicle at the tail of the arc plus the travel time along the arc is less than or equal to the beginning of the service of the vehicle at the head of the arc:

$$x^k_{hij} = 1 \Rightarrow t^{d,k}_{hi} + \tau^{tr}_{hi} \leqq t^k_{hi} \quad \forall\, k \in F, i \in V^k \setminus V_D, (h,i,j) \in T \tag{4.51}$$

The beginning of the service of a vehicle at a vertex plus the time the service takes is less than or equal to the departure time of the vehicle at that vertex:

$$x^k_{hij} = 1 \Rightarrow t^k_{hi} + t^{s,k}_{hi} \leqq t^{d,k}_{ij} \quad \forall\, k \in F, i \in V^k \setminus V_D, (h,i,j) \in T \tag{4.52}$$

$t^{s,k}_{hi}$ is not a variable, it is just a placeholder for the service time of vehicle k at vertex i after k has reached i via the arc (h,i). At the different vertex types, it is defined as follows:

- At parking depot vertices:

$$t^{s,k}_{hi} = 0 \quad \forall\, vl_i \neq k \in F_L \tag{4.52a}$$

The above constraints are not applicable for vehicle vl_i and for trailers.

- At parking intermediate vertices:

$$t^{s,k}_{hi} = 0 \quad \forall\, k \in F \tag{4.52b}$$

- At transshipment vertices:

$$t^{s,vl_i}_{hi} = 0 \tag{4.52c}$$

$$t^{s,k}_{hi} = (z^{k+}_{hi} + z^{k-}_{hi})\tau^{lt}_k \quad \forall\, vl_i \neq k \in F_L \tag{4.52d}$$

$$t^{s,k'}_{hi} = (z^{k'+}_{hi} + z^{k'-}_{hi})t^{lt,k'}_{hi} \quad \forall\, vl_i \neq k' \in F_T \tag{4.52e}$$

The service time is equal to zero for vehicle vl_i, because vl_i is the passive vehicle in a load transfer at i, and its service time is determined by the active vehicles that perform the load transfer.

- At lorry customers:

$$t_{hi}^{s,k} = su_i \tau_k^{lt} \quad \forall\, k \in F_L \tag{4.52f}$$

- At trailer customers:

$$t_{hi}^{s,k} = \max\{su_i \tau_k^{lt}, z_i \tau_k^{lt}\} \quad \forall\, k \in F_L, \tag{4.52g}$$

  i.e., for each $k \in F_L$, one constraint is necessary with $t_{hi}^{s,k} = su_i \tau_k^{lt}$, and one is necessary with $t_{hi}^{s,k} = z_i \tau_k^{lt}$. As stated above, at a trailer customer, a load transfer can be performed from lorry to trailer. The service time therefore depends on su_i, if the amount of load transferred is less than or equal to su_i; otherwise, it depends on z_i.

$$t_{hi}^{s,k'} = z_i \tau_k^{lt} \quad \forall\, k' \in F_T \tag{4.52h}$$

  For a trailer, the service time at a customer vertex does not really matter; arrival and departure at customers are determined by the pulling lorry. So, the service time can be set to zero as well.

- At unloading stations:

$$t_{hi}^{s,k} = (z_{hi}^{k+} + z_{hi}^{k-}) us_i \quad \forall\, k \in F \tag{4.52i}$$

- At maintenance stations:

$$t_{hi}^{s,k} = y_{hi}^{k} \tau_{ki}^{m} \quad \forall\, k \in F \tag{4.52j}$$

- At combi stations:

$$t_{hi}^{s,k} = (z_{hi}^{k+} + z_{hi}^{k-}) us_i + y_{hi}^{k} \tau_{ki}^{m} \quad \forall\, k \in F \tag{4.52k}$$

Scheduling/synchronization constraints at transshipment vertices:
At any point in time, at most one vehicle can transfer load to another vehicle. This means that if, for example, a lorry k arrives at a (currently open) transshipment vertex i at time t and wants to transfer an amount of load that will take 10 units of time to transfer, and if another lorry $\tilde{k}$ arrives 5 units of time after lorry k, lorry $\tilde{k}$ must wait 5 units of time until it can transfer load.

Such relationships are covered by the concept of implicit waiting, i.e., by the (meaning of the) t_{hi}^{k} variables. With these variables, it must be forced that the beginning of a load transfer from a vehicle k to another vehicle vl_i at transshipment vertex i plus the load transfer time do not overlap with the beginning of a third vehicle's load transfer, i.e.

$$t_{hi}^{k} + \tau_k^{lt}(z_{hi}^{k+} + z_{hi}^{k-}) \leqq t_{\tilde{h}i}^{\tilde{k}} \;\vee\; t_{\tilde{h}i}^{\tilde{k}} + \tau_{\tilde{k}}^{lt}(z_{\tilde{h}i}^{\tilde{k}+} + z_{\tilde{h}i}^{\tilde{k}-}) \leqq t_{hi}^{k}$$
$$\forall\, k, \tilde{k} \in F_L, k \neq \tilde{k}, k, \tilde{k} \neq vl_i, i \in V_{D_T} \cup V_{I_T},$$
$$(h,i), (\tilde{h},i) \in A \tag{4.53}$$

If one of the vehicles that want to transfer load is a trailer, the load transfer time constant in the above constraints must be replaced by the respective load transfer time variable.

The constraints stated as yet do not exclude the possibility that an arc may be traversed by two or more different lorries (though never more than once by the same lorry or trailer). This poses no problem. In the above constraints, however, $h \neq \tilde{h}$ must not be required therefore.

The above constraints are disjunctive constraints. They are rewritten as follows (cf. Williams 1999, p. 169 ff.). For each '$\leqq$'-constraint, a binary indicator variable δ is introduced which is equal to one if and only if the associated constraint holds:

$$t_{hi}^{k} + \tau_k^{lt}(z_{hi}^{k+} + z_{hi}^{k-}) + T^{max}\delta_{kh\tilde{k}\tilde{h}}^{i} \leqq T^{max} + t_{\tilde{h}i}^{\tilde{k}} \tag{4.53a}$$

Then, for each of the above disjunctions,

$$\delta^i_{kh\tilde{k}\tilde{h}} + \delta^i_{\tilde{k}\tilde{h}kh} \geqq 1 \tag{4.53b}$$

is required.

Scheduling/synchronization constraints for service stations:
They are modelled similarly to those for transshipment vertices. The start time of a vehicle's service at a service station plus the service time must not overlap with another vehicle's start time there:

$$t^k_{hi} + t^{s,k}_{hi} \leqq t^{\tilde{k}}_{\tilde{h}i} \ \vee \ t^{\tilde{k}}_{\tilde{h}i} + t^{s,\tilde{k}}_{\tilde{h}i} \leqq t^k_{hi} \quad \forall\, k, \tilde{k} \in F, k \neq \tilde{k}, i \in V_S, (h,i), (\tilde{h},i) \in A \tag{4.54}$$

Here, too, a binary indicator variable δ is introduced.

Parallel maintenance of a lorry-trailer combination is incorporated in the formulation as follows. For a combi station, an additional vertex for the maintenance part of the station is introduced; this vertex has only one predecessor and one successor, namely, the corresponding unloading vertex of the station. The corresponding arcs have a length and a travel time of zero and can be traversed by single trailers. Then, decoupling at 'maintenance part' vertices is forbidden, and the previous constraints are relaxed for the relevant lorry-trailer combinations.

The meta time windows can be modelled as follows. Let the number of meta time windows be m^{meta}. The customer vertices are grouped into m^{meta} groups V_{C_m}, for $m = 1, \ldots, m^{meta}$, according to their time windows. All arcs and respective variables between customer vertices of different groups can be deleted. For each real-world location l where a depot is situated and for each meta time window $m = 1, \ldots, m^{meta}-1$, a vertex v_{lm} with a time window starting at the end of m and ending at the beginning of $m+1$ is introduced. All depot vertices close before the end of the first meta time window.

Then,

$$y^{k,m}_{hi} = 1 \Rightarrow t^k_{hi} \geqq t^k_{h'i'} \quad \forall\, k \in F, m = 1, \ldots, m^{meta}, i \in V_{UM} \cup V_M, \\ i' \in V_{C_m}, (h,i), (h',i') \in A \tag{4.55}$$

is required for the maintenance operations, and

$$x^k_{hij} = 1 \Rightarrow \sum_{(h,v_{lm},j)\in T} x^k_{hv_{lm}j} \geqq 1 \quad \forall\, k \in F, l \in RWL, i \in V^k_D \text{ with } loc_i = l, \\ (h,i,j) \in T, m = 1, \ldots, m^{meta} \tag{4.56}$$

is needed to guarantee that a vehicle returns to the real-world location of its assigned depot vertex between each meta time window.

4.3.2.5 Connection Between Turn Variables and Arc Variables

Single vehicle arc variables x^k_{ij} with

$x^k_{ij} \in \{0,1\}\ \forall\, k \in F, (i,j) \in A,$
$$x^k_{ij} = \begin{cases} 1, \text{ vehicle } k \text{ traverses arc } (i,j) \\ 0, \text{ otherwise} \end{cases}$$

could easily be introduced in the above formulation by the following constraints:

$$x^k_{hi} - \sum_{(i,j)\in A} x^k_{hij} = 0 \quad \forall\, k \in F, (h,i) \in A \tag{4.57}$$

and

$$x_{ij}^k - \sum_{(h,i)\in A} x_{hij}^k = 0 \quad \forall\, k \in F, (i,j) \in A. \tag{4.58}$$

However, a complete substitution (in all constraints) of the turn variables by an expression containing only arc variables is *not* possible. This means that the turn variables are indispensable for the above formulation, whereas single vehicle arc variables are not needed.

In essence, the digraph D considered in this section is equivalent to a digraph D' with one vertex for each arc of D and one arc for each turn in D. The idea of this transformation is illustrated in Figure 4.5. The above formulation based on turn variables corresponds to an equivalent formulation in D' based on arc variables. (D' is the dual network of D, cf. p. 29).

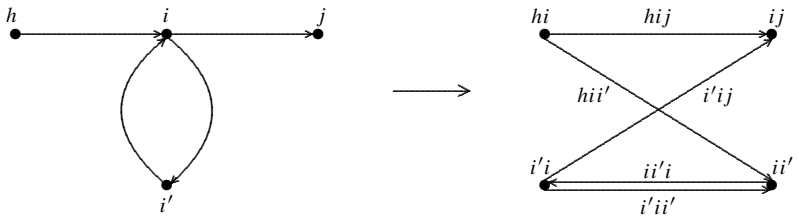


Figure 4.5: Transformation of a turn-based into an arc-based network

4.3.2.6 Critical Appraisal of the 'Complete' Problem

The formulation presented in this section is extremely complicated and cannot be solved exactly. Although the problem encompasses quite a lot of detailed technical aspects (which, however, is necessary for operative planning), it was not foreseeable that modelling the VRPTT would be so involved. In particular, there is a large number of logical constraints (implications and disjunctions), and the logic of load transfers between arbitrary vehicles is very intricate. The number of different variable and constraint types necessary to describe this logic is astonishing. Therefore, sections 4.3.1 and 4.3.2 mainly serve three purposes:

(i) they provide a detailed and systematic data model and a graph-theoretical representation of the complete VRPTT

(ii) they highlight the difficulty of considering trailers and transshipments

(iii) they show that and how turn variables can be used for modelling vehicle routing problems

Considering the intractability of the complete problem, the remainder of this chapter (and the entire Chapter 5) strive to answer the following question:

Which aspects of the VRPTT can be considered in a model that can still be solved exactly?

4.3.3 A Formulation for a 'Core' Problem Based on Arc Variables

In this section, a 'core' VRPTT is considered, which is confined to central aspects of the 'complete' problem. The notation of section 4.3.1 is used unless otherwise specified.

4.3.3.1 Simplifications

The following simplifications are made in this section:

- All vehicles start from the same fixed depot and return to it at the end of the planning horizon. There are no restrictions on the minimum amount of load to be delivered to the depot until a certain point in time.
- The planning horizon is only one period. This means that maintenance is not considered. It is assumed that a vehicle is maintained after it has returned to the depot. Hence, the time costs for maintenance are not relevant. As in the complete problem above, there are no restrictions on the duration of a tour other than the length of the planning horizon. A potentially necessary change of driver is not considered.
- Each customer may have several disjoint time windows, but must be visited exactly once.
- There are no parking locations, there are only transshipment locations with one or more disjoint time windows.
- There is no distinction between visiting and service time windows.
- Meta time windows are not considered.
- Each physical location corresponding to a trailer customer can also be used as a transshipment location.
- There are no unloading or combi stations.
- Load transfer is only possible from a collection lorry to a trailer, and the load transfer time per unit of load is the same for all lorries.

4.3.3.2 Underlying Network

The formulation is based on a *time-space-operation-vehicle network* $D = (V, A)$. V is comprised as follows. There is one start depot vertex o and one end depot vertex d, both with a time window of $[0, T^{max}]$. For each customer c and each of its time windows, there is one vertex. For each combination of physical transshipment location (*pure* transshipment location or trailer customer location), time window, and trailer, there are (at least) two vertices representing the operations decoupling, transshipment, and coupling. To be precise, for each transshipment location l and each trailer k', there are $n_{k'l}^{TS}$ vertices $v_{k'l}^{decouple}$, $v_{k'l}^{transfer,1}$, ..., $v_{k'l}^{transfer,n_{k'l}^{TS}-2}$, $v_{k'l}^{couple}$. $V_{decouple}$ is the set of decoupling vertices, $V_{transfer}$ is the set of transfer vertices, and V_{couple} is the set of coupling vertices. $V_{decouple}^{k'}$ is the set of decoupling vertices of trailer k', and $V_{transfer}^{k'}$ and $V_{couple}^{k'}$ are defined analogously. $V_{I_T} := V_{decouple} \cup V_{transfer} \cup V_{couple}$. The idea behind this separation of transshipment locations and processes is the following: A vertex $v_{k'l}^{decouple} \in V_{decouple}$ can only be reached by a lorry pulling the corresponding trailer k'. The lorry then leaves the vertex singly, the trailer moves on to $v_{k'l}^{transfer,1}$, the first pertinent transshipment vertex. A vertex $v_{k'l}^{transfer,i} \in V_{transfer}$ can only be reached and left by a single lorry and by the corresponding trailer k' (where the latter comes from the preceding transshipment vertex $v_{k'l}^{transfer,i-1}$ and moves on to the succeeding transshipment vertex $v_{k'l}^{transfer,i+1}$). A vertex $v_{k'l}^{couple} \in V_{couple}$ can only be reached by a single lorry and by the corresponding trailer (where the latter comes from the preceding transshipment vertex $v_{k'l}^{transfer,n_{k'l}^{TS}-2}$), and be left by a lorry pulling the corresponding trailer.

Lorries can visit all vertices of D, except for decoupling and coupling vertices of incompatible trailers and inaccessible customers (e.g., lorry customers with too much supply for a certain lorry). Trailers can only visit their corresponding transshipment vertices, trailer customers, and, of course, the start and the end depot vertex.

A contains the following arcs:

- (o, d) (for unused vehicles)

- (o, v_c) and (v_c, d) for all customer vertices $v_c \in V_C$
- (o, v_{I_T}) for all decoupling and coupling vertices $v_{I_T} \in V_{decouple} \cup V_{couple}$
- $(v_c, v_{c'})$ for all customer vertices $v_c, v_{c'} \in V_C$ with $c \neq c'$
- $(v_{k'l}^{decouple}, j)$ for all trailers $k' \in F_T$, all transshipment locations $l \in RWL_T$, and all other vertices $j \in V \setminus (\{o\} \cup V_{decouple} \cup \{v_{k'l}^{transfer,2}, \ldots, v_{k'l}^{transfer,n_{k'l}^{TS}-2}\})$
- $(v_{k'l}^{transfer,i}, j)$ for all $i = 1, \ldots, n_{k'l}^{TS} - 3$, all trailers $k' \in F_T$, all transshipment locations $l \in RWL_T$, and all other vertices $j \in V \setminus (\{o\} \cup V_{decouple} \cup \{v_{k'l}^{transfer,i'} : i' \neq i+1\} \cup \{v_{k'l}^{couple}\})$
- $(v_{k'l}^{transfer,n_{k'l}^{TS}-2}, j)$ for all trailers $k' \in F_T$, all transshipment locations $l \in RWL_T$, and all other vertices $j \in V \setminus (\{o\} \cup V_{decouple} \cup \{v_{k'l}^{transfer,i'} : i' \leqq i\})$
- $(v_{k'l}^{couple}, j)$ for all trailers $k' \in F_T$, all transshipment locations $l \in RWL_T$, and all vertices $j \in V_{decouple}^{k'} \setminus \{v_{k'l}^{decouple}\} \cup V_{C_{LT}} \cup \{d\}$
- (i, j) for all $i \in V_{C_{LT}}$, $j \in V_{I_T}$
- (i, j) for all $i \in V_{C_L}$, $j \in V_{transfer} \cup V_{couple}$

Figure 4.6 visualizes the subnetworks for the different vehicle types. To keep the figure clear and concise, there is only one arc for each arc type present in the subnetwork of the respective vehicle type. For example, in the LTC subnetwork, one arc from the left trailer customer to the right one is depicted to indicate that LTCs can freely move from any trailer customer to any other trailer customer (unless capacity, time window, or accessibility constraints prohibit this). The absence of an arc from the right trailer customer to the left one in the figure does not mean that there is no such arc in the network. Tables 4.2 and 4.3 also give an overview of the arcs in A.

		0 start depot	1 lorry c.	2 trailer c.	3 decoupling	4 transfer	5 coupling	6 end depot
0	start depot	–	✓	✓	✓	✓	✓	✓
1	lorry c.	–	✓	✓	–	✓	✓	✓
2	trailer c.	–	✓	✓	✓	✓	✓	✓
3	decoupling	–	✓	✓	–	✓	✓	✓
4	transfer	–	✓	✓	–	✓	✓	✓
5	coupling	–	–	✓	✓	–	–	✓
6	end depot	–	–	–	–	–	–	–

Table 4.2: Lorry arcs in VRPTT

		0 start depot	1 lorry c.	2 trailer c.	3 decoupling	4 transfer	5 coupling	6 end depot
0	start depot	–	–	✓	✓	–	–	✓
1	lorry c.	–	–	–	–	–	–	–
2	trailer c.	–	–	✓	✓	–	–	✓
3	decoupling	–	–	–	–	(✓)	(✓)	–
4	transfer	–	–	–	–	(✓)	(✓)	–
5	coupling	–	–	✓	✓	–	–	✓
6	end depot	–	–	–	–	–	–	–

Table 4.3: Trailer arcs in VRPTT

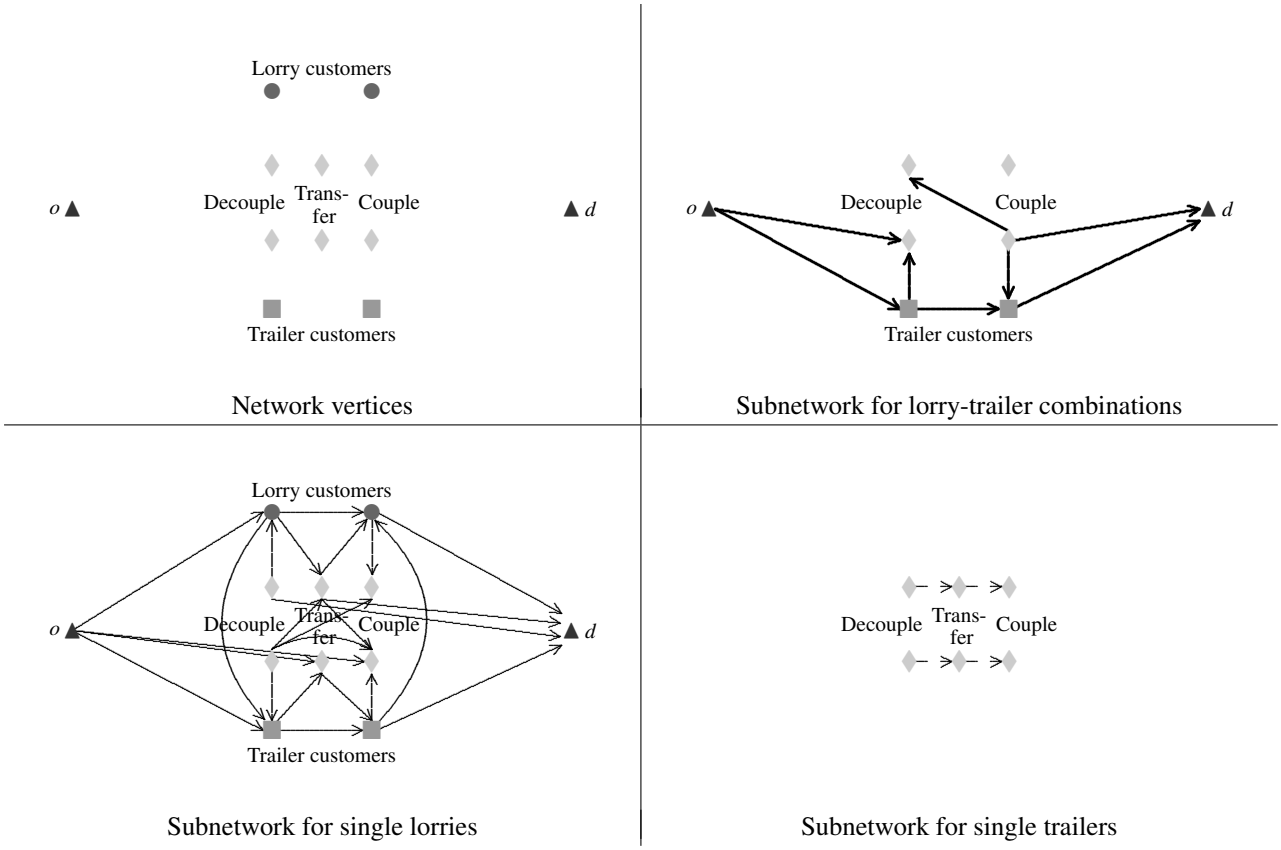


Figure 4.6: VRPTT core problem subnetworks

The arcs $(v_{k'l}^{decouple}, v_{k'l}^{couple})$ model the possibility that a lorry can wait for a trailer at a transshipment location while other lorries perform a load transfer. It is not necessary to introduce arcs $(v_{k'l}^{transfer,i}, v_{k'l}^{couple})$ for $i \in \{1, \ldots, n_{k'l}^{TS} - 3\}$, as it is not restrictive to require that the last transshipment operation involving a trailer k' at a trailer location l be always performed by the lorry k that will eventually pull k' away from l. This is because it is not sensible to allow k to wait at one of the transshipment intermediate vertices of k' at l, because k' can be pulled away from l only after all transshipment operations involving k' at l have been performed.

A^L is the set of arcs of D that can be traversed by lorries. Lorries cannot traverse arcs $(v_{k'l}^{decouple}, v_{k'l}^{transfer,1})$, $(v_{k'l}^{transfer,i}, v_{k'l}^{transfer,i+1})$, $\ldots$, $(v_{k'l}^{transfer,n_{k'l}^{TS}-2}, v_{k'l}^{couple})$. The corresponding trailers k' can traverse these arcs, and these arcs, together with the arc (o, d), form the set $A_{k'}^{single}$.

4.3.3.3 Assumptions

For the formulation presented subsequently, the following two fundamental assumptions are made:

(i) *Each vertex is visited by each vehicle at most once*.

(ii) *Each trailer uses each transshipment location at most once*, and *each transshipment vertex is reached by at most one lorry*.

Thus, for each trailer, there are at most $n_{k'l}^{TS}$ transshipment operations at each transshipment location, and it is clear which lorry performs each of these transshipments. This means that, on the one hand, scheduling constraints such as (4.53) in the turn variable formulation are not necessary. On the other hand, the number of vertices is smaller in the turn variable formulation, in which there is always only one vertex for each combination of location, time window, and trailer. As for the 'correct' choice of $n_{k'l}^{TS}$, note that, in the worst case, a *feasible* solution to an instance may exist only when, for each trailer, there

are as many intermediate and coupling vertices as there are lorry customers, because it may be necessary to perform a load transfer after each visit to a lorry customer. Consequently, an *optimal* solution obtained with the formulation may only be an optimal solution to an underlying problem instance if there are as many opportunities for a load transfer *at the right transshipment location* as there are lorry customers. Depending on the instance data, however, better values for n^{TS} can be computed.

4.3.3.4 The Formulation

There are the following variables:

- $x_{ij}^k \in \{0,1\}\ \forall\, k \in F, (i,j) \in A^k$.
  $x_{ij}^k = \begin{cases} 1, \text{ vehicle } k \text{ traverses arc } (i,j) \\ 0, \text{ otherwise} \end{cases}$
- $l_i^k \in \mathbb{R}_+^0\ \forall\, k \in F, i \in V^k$.
  The amount of load vehicle k is carrying when reaching vertex i, before k begins its service at i.
- $t_i^k \in \mathbb{R}_+^0\ \forall\, k \in F, i \in V^k$.
  The point in time when vehicle k begins its service at vertex i.

The resulting formulation is:

(VRPTT):

$$\sum_{k \in F} \sum_{(i,j) \in A^k} c_{ij}^k x_{ij}^k + \sum_{k \in F_L} c_k^{time}(t_d^k - t_o^k) \rightarrow \min \tag{4.59}$$

subject to

$$\sum_{i \in V_C^l} \sum_{k \in F_L} \sum_{(h,i) \in A^k} x_{hi}^k = 1 \qquad \forall\, l \in RWL_C \tag{4.60a}$$

$$\sum_{(h,i) \in A^{vl_i}} x_{hi}^{vl_i} \leqq 1 \qquad \forall\, i \in V_{I_T} \tag{4.60b}$$

$$\sum_{k \in F_L} \sum_{(h,i) \in A^k} x_{hi}^k - \sum_{(h',i) \in A^{vl_i}} x_{h'i}^{vl_i} \leqq 0 \qquad \forall\, i \in V_{I_T} \tag{4.60c}$$

$$x_{ij}^{k'} - \sum_{k \in com_{k'} \cap \{\tilde{k} \in F_L : (i,j) \in A^{\tilde{k}}\}} x_{ij}^k \leqq 0 \qquad \forall\, k' \in F_T, (i,j) \in A^{k'} \setminus A_{k'}^{single} \tag{4.60d}$$

$$x_{hi}^k = 1 \Rightarrow t_i^{k'} \leqq t_i^k \qquad \forall\, k' \in F_T, i \in (V_{C_{LT}} \cup V_{I_T}) \cap V^{k'}, k \in F_L, (h,i) \in A^k \tag{4.60e}$$

$$x_{hi}^{k'} = 1 \Rightarrow t_i^k \leqq t_i^{k'} \qquad \forall\, k' \in F_T, i \in (V_{C_{LT}} \cup V_{I_T}) \cap V^{k'}, k \in F_L, (h,i) \in A^{k'} \tag{4.60f}$$

$$x_{ij}^k = 1 \wedge x_{ij}^{k'} = 1 \Rightarrow l_i^k + l_i^{k'} + su_i \leqq l_j^k + l_j^{k'} \qquad \forall\, i \in V_{C_T}, k \in F_L, k' \in com_k, (i,j) \in A^k \cap A^{k'} \tag{4.60g}$$

$$x_{ij}^{vl_i} = 1 \wedge x_{ij'}^k = 1 \Rightarrow l_i^{vl_i} + (l_i^k - l_{j'}^k) \leqq l_j^{vl_i} \forall\, i \in V_{decouple}, k \in com_{vl_i}, (i,j) \in A^{vl_i}, (i,j') \in A^k \tag{4.60h}$$

$$x_{ij}^{vl_i} = 1 \wedge x_{ij'}^k = 1 \Rightarrow l_i^{vl_i} + (l_i^k - l_{j'}^k) \leqq l_j^{vl_i} \forall\, i \in V_{transfer}, k \in F_L, (i,j) \in A^{vl_i}, (i,j') \in A^k \tag{4.60i}$$

$$x_{ij}^{vl_i} = 1 \wedge x_{ij}^k = 1 \Rightarrow l_i^{vl_i} + (l_i^k - l_j^k) \leqq l_j^{vl_i} \ \forall\, i \in V_{couple}, k \in com_{vl_i}, (i,j) \in A^{vl_i} \cap A^k \tag{4.60j}$$

$$\sum_{(i,d)\in A^k} x_{id}^k = 1 \qquad \forall\, k \in F \tag{4.61a}$$

$$\sum_{(h,i)\in A^k} x_{hi}^k - \sum_{(i,j)\in A^k} x_{ij}^k = 0 \qquad \forall\, k \in F, i \in V^k \setminus \{o,d\} \tag{4.61b}$$

$$x_{ij}^{vl_i} = 1 \Rightarrow t_i^{vl_i} + (l_j^{vl_i} - l_i^{vl_i})\tau^{lt} \leqq twb_i \qquad \forall\, i \in V_{I_T}, (i,j) \in A^{vl_i} \tag{4.61c}$$

$$x_{ij}^k = 1 \Rightarrow l_i^k + su_i \leqq l_j^k \qquad \forall\, k \in F_L, i \in V_{C_L}, (i,j) \in A^k \tag{4.61d}$$

$$x_{ij}^k = 1 \Rightarrow l_i^k \leqq l_j^k \qquad \forall\, k \in F, i \in \{o\} \cup V_{C_{LT}}, (i,j) \in A^k \tag{4.61e}$$

$$x_{ij}^k = 1 \Rightarrow l_i^k + su_i - 2q_k \sum_{k' \in com_k \cap \{\tilde{k}' \in F_T : (i,j) \in A^{\tilde{k}'}\}} x_{ij}^{k'} \leqq l_j^k$$

$$\forall\, k \in F_L, i \in V_{C_{LT}}, (i,j) \in A^k \tag{4.61f}$$

$$x_{ij}^k = 1 \Rightarrow l_j^k \leqq l_i^k \qquad \forall\, k \in F_L, i \in V_{I_T}, (i,j) \in A^k \tag{4.61g}$$

$$x_{ij}^{vl_i} = 1 \Rightarrow l_i^{vl_i} \leqq l_j^{vl_i} \qquad \forall\, i \in V_{transfer}, (i,j) \in A^{vl_i} \tag{4.61h}$$

$$x_{oi}^k = 1 \Rightarrow t_o^k + \tau_{oi}^{tr} \leqq t_i^k \qquad \forall\, k \in F, (o,i) \in A^k \tag{4.61i}$$

$$x_{ij}^k = 1 \Rightarrow t_i^k + \tau_{ij}^{tr} \leqq t_j^k \qquad \forall\, k \in F_L, i \in V_C, (i,j) \in A^k \tag{4.61j}$$

$$x_{ij}^k = 1 \Rightarrow t_i^k + (l_i^k - l_j^k)\tau^{lt} + \tau_{ij}^{tr} \leqq t_j^k \qquad \forall\, k \in F_L, i \in V_{I_T}, (i,j) \in A^k \tag{4.61k}$$

$$x_{ij}^{vl_i} = 1 \Rightarrow t_i^{vl_i} + (l_j^{vl_i} - l_i^{vl_i})\tau^{lt} + \tau_{ij}^{tr} \leqq t_j^{vl_i} \forall\, i \in V_{I_T}, (i,j) \in A^{vl_i} \tag{4.61l}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, k \in F, (i,j) \in A^k \tag{4.62a}$$

$$0 \leqq l_i^k \leqq q_k \qquad \forall\, k \in F, i \in V^k \tag{4.62b}$$

$$twa_i \leqq t_i^k \leqq twb_i \qquad \forall\, k \in F, i \in V^k \tag{4.62c}$$

(4.59) is the objective function, which minimizes total costs. The cost coefficient c_{ij}^k indicates the distance-dependent costs of vehicle k traversing arc (i,j). For the arcs emanating from the start depot vertex (except for the arc (o,d)), the fixed costs for using vehicle k are also contained in c_{ij}^k. It is assumed that all arc costs and arc travel times are strictly positive. However, movements between vertices corresponding to the same physical location incur only very small positive costs ϵ^c and take only a very small amount of time ϵ^t. t_o^k denotes the point in time when vehicle k leaves the start depot. This may be later than time zero.

Constraints (4.60) are logically coupling constraints (i.e., constraints involving more than one vehicle). (4.60a)–(4.60d) are the routing synchronization constraints, (4.60e) and (4.60f) are temporal synchronization constraints (observance of dynamic time windows), and (4.60g)–(4.60j) are load synchronization constraints. (4.61) are non-coupling constraints, and (4.62) determine the ranges of the variables.

(4.60a) are the usual customer covering constraints. (4.60b) ensure that each transshipment vertex is visited at most once by the corresponding trailer. (4.60c) make sure that a transshipment vertex i is only visited by a lorry if the corresponding trailer visits this vertex, and, hence, that at most one lorry visits a transshipment vertex. (4.60a)–(4.60c) imply that each vertex in $V \setminus \{o,d\}$ is visited by at most one lorry

and one trailer. (4.60d) guarantee that a trailer k' is pulled by a compatible lorry if k' traverses a spacial arc. (4.60e) and (4.60f) state that the beginning of the service at a trailer customer or transshipment vertex is the same for a trailer and a lorry that both visit the vertex. For $i \in (V_{C_{LT}} \cup V_{decouple}) \cap V^{k'}$, it is sufficient to require these two constraints for $k \in com_{k'}$, $(h, i) \in A^{k'} \cap A^k$. Constraints (4.60e) and (4.60f) are similar to the 'same departure time constraints' used in aircraft routing and scheduling problems, cf. Section 4.2. (4.60g)–(4.60j) are the load update constraints for both types of vehicle at trailer customer and transshipment vertices. (4.60g) state that the supply of a trailer customer is divided up arbitrarily between the lorry and the trailer visiting the customer.

(4.61a) require that each vehicle reach the end depot (possibly via the arc (o, d)). (4.61b) are the flow conservation constraints, and (4.61c) make sure that the static time windows are met. (4.61c) take the service time at transshipment vertices into account: The load transfer must be finished by the end of the time window. Note that, because of (4.60e), (4.60f), and (4.60h)–(4.60j), it is sufficient to require (4.61c) for vl_i. It is assumed that the service time at customer vertices i is included in the traversal times of the arcs (i, j). By shifting the original twb_i value backwards by the service time, the time windows for customer vertices can be adapted such that the collection of the supply is finished by the end of the time window.

The next constraints, (4.61d), are the load update constraints for the lorries at lorry customer vertices. Constraints (4.61e) state that, at a trailer customer vertex, the amount of load transferred to the trailer is not more than the customer supply. This is a sensible requirement, because, as stated above, for each trailer customer location, there is also a set of transshipment vertices, and movements between vertices corresponding to the same physical location incur virtually no costs and take virtually no time. Constraints (4.61f) are for the correct update of the load variables at trailer customer vertices visited by a single lorry. Constraints (4.61g) make sure that no load transfer from trailer to lorry is possible, respectively, that the amount of load transferred from a lorry to a trailer is non-negative. Without this constraint, negative load transfer times can result. (4.61h) make sure that the load of a trailer does not decrease at transshipment intermediate vertices not visited by a lorry.

Constraints (4.61i)–(4.61l) are the constraints for the update of the time variables: (4.61i) are for the arcs emanating from the start depot, (4.61j) are for lorries on arcs emanating from customer vertices, (4.61k) are for lorries at transshipment vertices, and (4.61l) are for the trailers at their transshipment vertices.

When $n^{TS}_{k'l} \geqq 4$, a solution where, for example, a lorry visits $v^{transfer,1}_{k'l}$ after $v^{transfer,2}_{k'l}$ for some $k'l$ represents the same real-world process as a solution where the lorry visits $v^{transfer,2}_{k'l}$ after $v^{transfer,1}_{k'l}$. Such symmetries with respect to the sequence of transshipment subtours, i.e., of visits of transshipment intermediate vertices of one real-world location by one and the same lorry (*transshipment subtour symmetries*), are avoided in the formulation by constraints (4.61l). However, if two different lorries visit two transshipment vertices for some $k'l$, it *does* make a difference which lorry visits which vertex: A solution where lorry k_1 visits vertex $v^{transfer,i}_{k'l}$ and lorry k_2 visits $v^{transfer,i+1}_{k'l}$ is structurally different from (represents a different real-world process than) a solution where it is the other way round, because the first solution implies that lorry k_2 can start its transshipment process only after the transshipment process of lorry k_1 is finished; in the second solution it is vice versa.

Constraints of the form

$$t^k_i - T^{max} \sum_{(h,i)\in A^k} x^k_{hi} \leqq 0 \qquad \forall\, k \in F, i \in V^k, \tag{4.63}$$

are not necessary, but may be helpful in an implementation (by 'guiding' the solution algorithm).

Formulation (4.59)–(4.62) includes the TTRP as described by Chao 2002 and Scheuerer 2004.

The consideration of support vehicles is trivial. The assumption that a load transfer is only allowed from a collection lorry to a trailer implies that all support lorries can be assumed to have a capacity of zero (which will most often be the case in reality anyway, see the introduction). Hence, it is sufficient not to introduce any variables for support vehicles on arcs leading to or emanating from customer vertices.

Multiple use of vehicles can be modelled in the network used for the core problem by introducing subperiods and additional depot vertices and connecting them to the original vertices, taking into account load transfer and driving times. A depot vertex for period t is connected to its neighbouring depot vertices of periods $t-1$ and $t+1$. All vehicles can move in time along the arcs connecting these depot vertices. Thus, a support lorry can park a support trailer somewhere, return to a depot vertex, couple another support trailer, park it elsewhere, re-couple the first support trailer and pull it to a later depot vertex and so on. Similarly, a collection lorry (with or without a trailer) can unload at such a depot vertex and continue collection afterwards. These additional intermediate depot vertices can be visited by more than one lorry and more than one trailer, but at most once by each vehicle. The formulation for the core problem has to be modified slightly. In particular, if V_{depot} is the set of these additional depot vertices, (4.60e) have to be supplemented by

$$x_{hi}^k = 1 \wedge x_{hi}^{k'} = 1 \Rightarrow t_i^{k'} \leqq t_i^k \qquad \forall\, k' \in F_T, h, i \in V_{depot}, k \in com_{k'}, (h,i) \in A^k \cap A^{k'}, \quad (4.64)$$

(4.60f) need an analogous supplement, and the load and time update at the depot vertices must be

$$x_{ij}^k = 1 \Rightarrow l_j^k \leqq 0 \qquad \forall\, k \in F, i \in V_{depot}, (i,j) \in A^k, \quad (4.65)$$

and

$$x_{ij}^k = 1 \Rightarrow t_i^k + l_i^k \tau^{lt} + \tau_{ij}^{tr} \leqq t_j^k \qquad \forall\, k \in F, i \in V_{depot}, (i,j) \in A^k. \quad (4.66)$$

A two-period planning horizon can be modelled by an 'end-of-day-one' depot vertex. All customer and transshipment vertices with a time window lying entirely in day one are connected to this vertex, and all customer and transshipment vertices with a time window lying entirely in day two can be reached only from this vertex and from other vertices of day two, depending on the respective time windows and driving times, but not from vertices of day one. Time windows stretching over more than one day lead to two disjoint time windows, one for each day.

To compute an optimal solution to a real-world VRPTT instance, the above-mentioned extensions must be considered. However, although these extensions are important from a practical point of view, they are not new from a modelling perspective, and the above formulation is still complicated enough without them. Hence, they are not considered in the core problem.

4.3.4 A Formulation for the Core Problem Based on Path Variables

In this section, a reformulation of (4.59)–(4.62) that could be used in a branch-and-price algorithm is developed. The subsequent exposition follows Desaulniers et al. 1998.

The usual decomposition approach for VRPs can also be applied to the VRPTT. The logically coupling constraints (4.60), i.e., the constraints involving more than one vehicle, define the master program; the non-coupling constraints (4.61), i.e., the vehicle-specific constraints, define the sub- or pricing problems. Constraints (4.60b) are essentially non-coupling constraints in that each one of them includes only variables concerning one trailer. Nevertheless, it is just as well possible to put them in the master program.

4.3.4.1 The Master Program

The formulation presented in the previous section uses arc variables for each relevant combination of lorry or trailer k and arc (i,j). Branch-and-price approaches for vehicle routing problems use path variables in the master program: There is one variable for each feasible o-d-path of each vehicle. As for

VRPs without trailers, also for the VRPTT, each feasible tour of a vehicle in the network used in the previous section is an elementary path from the start depot vertex to the end depot vertex through the respective subnetwork. This is true for the lorries as well as for the trailers. In reality, though, the vehicle itineraries will contain cycles starting and ending at transshipment locations.

It follows from the flow decomposition theorem (cf. Ahuja et al. 1993, p. 80 f.) that the extreme points of the convex hull of all points fulfilling the pricing problems' constraints (4.61)–(4.62) correspond to paths from o to d in D, some of which may be non-elementary, because D contains cycles. However, non-elementary paths can never be part of a feasible solution due to the master program constraints.

The pricing problems' extreme points are described by flow and resource vectors

$$(x_p^k, l_p^k, t_p^k) = (x_{ijp}^k, l_{ip}^k, t_{ip}^k) \quad \forall\, k \in F,\, p \in P^k,\, (i,j) \in A^k, \tag{4.67}$$

where P^k is the set of extreme points for vehicle $k \in F$.

Any solution to (4.61)–(4.62) can then be expressed as a convex combination of these extreme points:

$$x_{ij}^k = \sum_{p \in P^k} x_{ijp}^k \lambda_p^k \quad \forall\, k \in F,\, (i,j) \in A^k \tag{4.67a}$$

$$x_{ij}^k \in \{0,1\} \quad \forall\, k \in F,\, (i,j) \in A^k \tag{4.67b}$$

$$l_i^k = \sum_{p \in P^k} l_{ip}^k \lambda_p^k \quad \forall\, k \in F,\, i \in V^k \tag{4.67c}$$

$$t_i^k = \sum_{p \in P^k} t_{ip}^k \lambda_p^k \quad \forall\, k \in F,\, i \in V^k \tag{4.67d}$$

$$\sum_{p \in P^k} \lambda_p^k \leqq 1 \quad \forall\, k \in F \tag{4.67e}$$

$$\lambda_p^k \geqq 0 \quad \forall\, k \in F,\, p \in P^k \tag{4.67f}$$

In constraints (4.67e), '$\leqq$' instead of '$=$' is used. This is possible, if not using a vehicle does not incur any (fixed) costs, as is assumed here. Therefore, the arc (o,d) is not necessary and is removed.

The integer master program (IMP) is then (linearizing all implications in (4.60) by using appropriate constants):

$$\sum_{k \in F} \sum_{p \in P^k} \left(\sum_{(i,j) \in A^k} c_{ij}^k x_{ijp}^k \right) \lambda_p^k + \sum_{k \in F_L} \sum_{p \in P^k} c_k^{time} (t_{dp}^k - t_{op}^k) \lambda_p^k \to \min \tag{4.68}$$

subject to

$$\sum_{k \in F_L} \sum_{p \in P^k} \left(\sum_{i \in V_C^l} \sum_{(h,i) \in A^k} x_{hip}^k \right) \lambda_p^k = 1 \qquad \forall\, l \in RWL_C \tag{4.69a}$$

$$\sum_{p \in P^{vl_i}} \left(\sum_{(h,i) \in A^{vl_i}} x_{hip}^{vl_i} \right) \lambda_p^{vl_i} \leqq 1 \qquad \forall\, i \in V_{I_T} \tag{4.69b}$$

$$\sum_{k \in F_L} \sum_{p \in P^k} \left(\sum_{(h,i) \in A^k} x_{hip}^k \right) \lambda_p^k - \sum_{p \in P^{vl_i}} \left(\sum_{(h',i) \in A^{vl_i}} x_{h'ip}^{vl_i} \right) \lambda_p^{vl_i} \leqq 0$$

$$\forall\, i \in V_{I_T} \tag{4.69c}$$

$$\sum_{p \in P^{k'}} x_{ijp}^{k'} \lambda_p^{k'} - \sum_{k \in com_{k'} \cap \{\tilde{k} \in F_L : (i,j) \in A^{\tilde{k}}\}} \sum_{p \in P^k} x_{ijp}^k \lambda_p^k \leqq 0$$

$$\forall\, k' \in F_T, (i,j) \in A^{k'} \setminus A_{k'}^{single} \tag{4.69d}$$

$$\sum_{p \in P^{k'}} t_{ip}^{k'} \lambda_p^{k'} - \sum_{p \in P^k} t_{ip}^k \lambda_p^k + T^{max} \sum_{p \in P^k} x_{hip}^k \lambda_p^k \leqq T^{max}$$

$$\forall\, k' \in F_T, i \in (V_{C_{LT}} \cup V_{I_T}) \cap V^{k'}, k \in F_L, (h,i) \in A^k \tag{4.69e}$$

$$\sum_{p \in P^k} t_{ip}^k \lambda_p^k - \sum_{p \in P^{k'}} t_{ip}^{k'} \lambda_p^{k'} + T^{max} \sum_{p \in P^{k'}} x_{hip}^{k'} \lambda_p^{k'} \leqq T^{max}$$

$$\forall\, k' \in F_T, i \in (V_{C_{LT}} \cup V_{I_T}) \cap V^{k'}, k \in F_L, (h,i) \in A^{k'} \tag{4.69f}$$

$$\sum_{p \in P^k} \left(l_{ip}^k - l_{jp}^k\right) \lambda_p^k$$

$$+ \sum_{p \in P^{k'}} \left(l_{ip}^{k'} - l_{jp}^{k'}\right) \lambda_p^{k'} + (q_k + q_{k'} + su_i) \left(\sum_{p \in P^k} x_{ijp}^k \lambda_p^k + \sum_{p \in P^{k'}} x_{ijp}^{k'} \lambda_p^{k'} \right) \leqq 2q_k + 2q_{k'} + su_i$$

$$\forall\, i \in V_{C_{LT}}, k \in F_L, k' \in com_k, (i,j) \in A^k \cap A^{k'} \tag{4.69g}$$

$$\sum_{p \in P^{vl_i}} \left(l_{ip}^{vl_i} - l_{jp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p \in P^k} \left(l_{ip}^k - l_{j'p}^k\right) \lambda_p^k + \sum_{p \in P^{vl_i}} \left(q_{vl_i} x_{ijp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p \in P^k} \left(q_{vl_i} x_{ij'p}^k\right) \lambda_p^k \leqq 2q_{vl_i}$$

$$\forall\, i \in V_{decouple}, k \in com_{vl_i}, (i,j) \in A^{vl_i}, (i,j') \in A^k \tag{4.69h}$$

$$\sum_{p \in P^{vl_i}} \left(l_{ip}^{vl_i} - l_{jp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p \in P^k} \left(l_{ip}^k - l_{j'p}^k\right) \lambda_p^k + \sum_{p \in P^{vl_i}} \left(q_{vl_i} x_{ijp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p \in P^k} \left(q_{vl_i} x_{ij'p}^k\right) \lambda_p^k \leqq 2q_{vl_i}$$

$$\forall\, i \in V_{transfer}, k \in F_L, (i,j) \in A^{vl_i}, (i,j') \in A^k \tag{4.69i}$$

$$\sum_{p \in P^{vl_i}} \left(l_{ip}^{vl_i} - l_{jp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p \in P^k} \left(l_{ip}^k - l_{jp}^k\right) \lambda_p^k + \sum_{p \in P^{vl_i}} \left(q_{vl_i} x_{ijp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p \in P^k} \left(q_{vl_i} x_{ijp}^k\right) \lambda_p^k \leqq 2q_{vl_i}$$

$$\forall\, i \in V_{couple}, k \in com_{vl_i}, (i,j) \in A^{vl_i} \cap A^k \tag{4.69j}$$

$$\sum_{p \in P^k} \lambda_p^k \leqq 1 \qquad \forall\, k \in F \tag{4.69k}$$

$$\lambda_p^k \geqq 0 \qquad \forall\, k \in F, p \in P^k \tag{4.69l}$$

$$x_{ij}^k = \sum_{p \in P^k} x_{ijp}^k \lambda_p^k \qquad \forall\, k \in F, (i,j) \in A^k \tag{4.69m}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, k \in F, (i,j) \in A^k \tag{4.69n}$$

The last two constraints are necessary, because the binary restrictions on the arc variables cannot be replaced by binary restrictions on the path variables, cf. Desaulniers et al. 1998, p. 75.

4.3.4.2 The Pricing Problems

As is the case for all vehicle routing problems, also the pricing problems for the VRPTT are elementary shortest path problems with resource constraints. When the fleet is heterogeneous, to solve the LP relaxation of the master problem exactly, (at least) one ESPPRC for each vehicle (or vehicle class, see the next section) must be solved exactly. Because of the dual prices coming from the master problem, the pricing problem networks usually have negative cost cycles. This makes the problem $\mathscr{NP}$-hard in the strong sense as discussed in section 2.3. There is a considerable amount of literature on the solution of (E)SPPRCs as pricing problems in branch-and-price approaches for VRP(TW)s and related problems. A review is given in Section 5.3.1.1.

The sub- or pricing problems in Dantzig-Wolfe decomposition and branch-and-price approaches use the variables of the 'original' formulation. As the reformulation in this section is based on (4.59)–(4.62), this means that the pricing problems corresponding to the above master program use arc variables.

As mentioned above, the pricing problems' constraints are (4.61)–(4.62).

The objective functions (for lorries and trailers) can be derived as follows:

Introducing dual variables

$$(\alpha, \beta, \gamma) := (\alpha_l^1, \alpha_i^2, \alpha_i^3, \alpha_{k'ij}^4, \beta_{k'khi}^1, \beta_{k'khi}^2, \beta_{kk'ij}^3, \beta_{kijj'}^4, \beta_{kijj'}^5, \beta_{kij}^6, \gamma^k) \tag{4.70}$$

for constraints (4.69a)–(4.69k), and with

$$\delta_{k'vl_i}^1 = \begin{cases} 1, k' = vl_i \\ 0, \text{otherwise} \end{cases} \quad \forall\, k' \in F_T, i \in V_{I_T},$$

and

$$\delta_{kk'ij}^2 = \begin{cases} 1, k \in com_{k'} \cap \{\tilde{k} \in F_L : (i,j) \in A^{\tilde{k}}\} \\ 0, \text{otherwise} \end{cases} \quad \forall\, k' \in F_T, k \in F_L, (i,j) \in A^k \setminus A_{k'}^{single}$$

the reduced costs of path p are shown in Table 4.4.

$\tilde{c}_p^k(\alpha,\beta,\gamma)\|_{k\in F_L} =$	$\tilde{c}_p^{k'}(\alpha,\beta,\gamma)\|_{k'\in F_T} =$
$\sum_{(i,j)\in A^k} c_{ij}^k x_{ijp}^k + c_k^{time}(t_{dp}^k - t_{op}^k)$	$\sum_{(i,j)\in A^{k'}} c_{ij}^{k'} x_{ijp}^{k'}$
$-\sum_{l\in RWL_C}\left(\sum_{i\in V_C^l}\sum_{(h,i)\in A^k} x_{hip}^k\right)\alpha_l^1$	
	$-\sum_{i\in V_{I_T}}\left(\sum_{(h,i)\in A^{k'}} x_{hip}^{k'}\right)\alpha_i^2\delta_{k'vl_i}^1$
$-\sum_{i\in V_{I_T}}\left(\sum_{(h,i)\in A^k} x_{hip}^k\right)\alpha_i^3$	$+\sum_{i\in V_{I_T}}\left(\sum_{(h',i)\in A^{k'}} x_{hip}^{k'}\right)\alpha_i^3\delta_{k'vl_i}^1$
$+\sum_{k'\in F_T}\sum_{(i,j)\in A^{k'}\setminus A_{k'}^{single}} x_{ijp}^k\alpha_{k'ij}^4\delta_{kk'ij}^2$	$-\sum_{(i,j)\in A^{k'}\setminus A_{k'}^{single}} x_{ijp}^{k'}\alpha_{k'ij}^4$
$+\sum_{k'\in F_T}\sum_{i\in(V_{C_{LT}}\uplus V_{I_T})\cap V^{k'}}\sum_{(h,i)\in A^k}(t_{ip}^k - T^{max}x_{hip}^k)\beta_{k'khi}^1$	$-\sum_{k\in F_L}\sum_{i\in(V_{C_{LT}}\uplus V_{I_T})\cap V^{k'}}\sum_{(h,i)\in A^k} t_{ip}^{k'}\beta_{k'khi}^1$

(continued on next page)

(continued from previous page)

$$-\sum_{k'\in F_T}\sum_{i\in(V_{C_{LT}}\cup V_{I_T})\cap V^{k'}}\sum_{(h,i)\in A^{k'}} t^k_{ip}\beta^2_{k'khi}$$

$$-\sum_{k'\in com_k}\sum_{i\in V_{C_{LT}}}\sum_{(i,j)\in A^k\cap A^{k'}} [l^k_{ip}-l^k_{jp}+(q_k+q_{k'}+su_i)x^k_{ijp}]\beta^3_{kk'ij}$$

$$-\sum_{i\in V_{decouple}}\sum_{(i,j)\in A^{vl_i}}\sum_{(i,j')\in A^k} (l^k_{ip}-l^k_{j'p}-q_{vl_i}x^k_{ij'p})\beta^4_{kijj'}\delta^2_{kvl_iij'}$$

$$-\sum_{i\in V_{transfer}}\sum_{(i,j)\in A^{vl_i}}\sum_{(i,j')\in A^k} (l^k_{ip}-l^k_{j'p}-q_{vl_i}x^k_{ij'p})\beta^5_{kijj'}$$

$$-\sum_{i\in V_{couple}}\sum_{(i,j)\in A^{vl_i}\cap A^k} (l^k_{ip}-l^k_{jp}-q_{vl_i}x^k_{ijp})\beta^6_{kij}\delta^2_{kvl_iij}$$

$$-\gamma^k$$

$$+\sum_{k\in F_L}\sum_{i\in(V_{C_{LT}}\cup V_{I_T})\cap V^{k'}}\sum_{(h,i)\in A^{k'}}(t^{k'}_{ip}-T^{max}x^{k'}_{hip})\beta^2_{k'khi}$$

$$-\sum_{k\in com_{k'}}\sum_{i\in V_{C_{LT}}}\sum_{(i,j)\in A^k\cap A^{k'}} [l^{k'}_{ip}-l^{k'}_{jp}+(q_k+q_{k'}+su_i)x^{k'}_{ijp}]\beta^3_{kk'ij}$$

$$-\sum_{k\in com_{k'}}\sum_{i\in V_{decouple}}\sum_{(i,j)\in A^{k'}}\sum_{(i,j')\in A^k} (l^{k'}_{ip}-l^{k'}_{jp}-q_{k'}x^{k'}_{ijp})\beta^4_{kijj'}\delta^1_{k'vl_i}$$

$$-\sum_{k\in F_L}\sum_{i\in V_{transfer}}\sum_{(i,j)\in A^{k'}}\sum_{(i,j')\in A^k} (l^{k'}_{ip}-l^{k'}_{jp}-q_{k'}x^{k'}_{ijp})\beta^5_{kijj'}\delta^1_{k'vl_i}$$

$$-\sum_{k\in com_{k'}}\sum_{i\in V_{couple}}\sum_{(i,j)\in A^{k'}\cap A^k} (l^{k'}_{ip}-l^{k'}_{jp}-q_{k'}x^{k'}_{ijp})\beta^6_{kij}\delta^1_{k'vl_i}$$

$$-\gamma^{k'}$$

Table 4.4: Reduced costs of paths for lorries and trailers

Hence, the reduced costs on lorry and trailer arcs are as shown in Tables 4.5 to 4.8.

weighted dual price	on arcs between vertex types
$-\alpha^1_l$	$\{0,\ldots,4\}\times\{1,2\}$; $(5,2)$
$-\alpha^3_j$	$\{0,2\}\times\{3,4,5\}$; $\{1,3,4\}\times\{4,5\}$
$+\alpha^4_{k'ij}\delta^2_{kk'ij}$	$(0,2)$; $(0,3)$; $\{2,5\}\times\{2,3,6\}$
$+(t^k_j-T^{max})\beta^1_{k'kij}$	$\{0,2\}\times\{2,\ldots,5\}$; $\{3,4\}\times\{4,5\}$; $(5,2)$; $(5,3)$
$-t^k_j\beta^2_{k'kij}$	$\{0,2\}\times\{2,3\}$; $\{3,4\}\times\{4,5\}$; $(5,2)$; $(5,3)$
$-[l^k_i-l^k_j+(q_k+q_{k'}+su_i)]\beta^3_{kk'ij}$	$(2,2)$; $(2,3)$; $(2,6)$
$-(l^k_i-l^k_{j'}-q_{vl_i})\beta^4_{kijj'}\delta^2_{kvl_iij'}$	$(3,1)$; $(3,2)$; $(3,4)$; $(3,5)$; $(3,6)$
$-(l^k_i-l^k_{j'}-q_{vl_i})\beta^5_{kijj'}$	$(4,1)$; $(4,2)$; $(4,4)$; $(4,5)$; $(4,6)$
$-(l^k_i-l^k_j-q_{vl_i})\beta^6_{kij}\delta^2_{kvl_iij}$	$(5,2)$; $(5,3)$; $(5,6)$
$-\gamma^k$	$(0,6)$; $(1,6)$; $(2,6)$; $(3,6)$; $(4,6)$; $(5,6)$

Table 4.5: Reduced costs on lorry arcs in VRPTT

weighted dual price	on arcs between vertex types
$-\alpha_j^2 \delta_{k'vl_j}^1$	(0, 3); (2, 3); {3, 4} × {4, 5}; (5, 3)
$+\alpha_j^3 \delta_{k'vl_j}^1$	(0, 3); (2, 3); {3, 4} × {4, 5}; (5, 3)
$-\alpha_{k'ij}^4$	{0, 2, 5} × {2, 3}; (2, 6); (5, 6)
$-t_j^{k'} \beta_{k'kij}^1$	{0, 2} × {2, 3}; {3, 4} × {4, 5}; (5, 2); (5, 3)
$+(t_j^{k'} - T^{max})\beta_{k'kij}^2$	{0, 2} × {2, 3}; {3, 4} × {4, 5}; (5, 2); (5, 3)
$-[l_i^{k'} - l_j^{k'} + (q_k + q_{k'} + su_i)]\beta_{kk'ij}^3$	(2, 2); (2, 3); (2, 6)
$-(l_i^{k'} - l_j^{k'} - q_{k'})\beta_{kijj'}^4 \delta_{k'vl_i}^1$	(3, 4); (3, 5)
$-(l_i^{k'} - l_j^{k'} - q_{k'})\beta_{kijj'}^5 \delta_{k'vl_i}^1$	(4, 4); (4, 5)
$-(l_i^{k'} - l_j^{k'} - q_{k'})\beta_{kij}^6 \delta_{k'vl_i}^1$	(5, 2); (5, 3); (5, 6)
$-\gamma^{k'}$	(0, 6); (2, 6); (3, 6); (4, 6); (5, 6)

Table 4.6: Reduced costs on trailer arcs in VRPTT

	α^1	α^2	α^3	α^4	β^1	β^2	β^3	β^4	β^5	β^6	γ
(0, 1)	✓										
(0, 2)	✓			✓	✓	✓					
(0, 3)			✓	✓	✓	✓					
(0, 4)			✓		✓						
(0, 5)			✓		✓						
(0, 6)											✓
(1, 1)	✓										
(1, 2)	✓										
(1, 4)			✓								
(1, 5)			✓								
(1, 6)											✓
(2, 1)	✓										
(2, 2)	✓			✓	✓	✓	✓				
(2, 3)			✓	✓	✓	✓	✓				
(2, 4)			✓		✓						
(2, 5)			✓		✓						
(2, 6)				✓			✓				✓
(3, 1)	✓							✓			
(3, 2)	✓							✓			
(3, 4)			✓		✓	✓		✓			
(3, 5)			✓		✓	✓		✓			
(3, 6)								✓			✓
(4, 1)	✓								✓		
(4, 2)	✓								✓		
(4, 4)			✓		✓	✓			✓		
(4, 5)			✓		✓	✓			✓		
(4, 6)									✓		✓
(5, 2)	✓			✓	✓	✓				✓	
(5, 3)				✓	✓	✓				✓	
(5, 6)				✓						✓	✓

Table 4.7: Reduced costs on lorry arcs in VRPTT

	α^1	α^2	α^3	α^4	β^1	β^2	β^3	β^4	β^5	β^6	γ
(0, 2)				✓	✓	✓					
(0, 3)		✓	✓	✓	✓	✓					
(0, 6)											✓
(2, 2)				✓	✓	✓	✓				
(2, 3)		✓	✓	✓	✓	✓	✓				
(2, 6)				✓			✓				✓
(3, 4)		✓	✓		✓	✓		✓			
(3, 5)		✓	✓		✓	✓		✓			
(4, 4)		✓	✓		✓	✓			✓		
(4, 5)		✓	✓		✓	✓			✓		
(5, 2)				✓	✓	✓				✓	
(5, 3)		✓	✓	✓	✓	✓				✓	
(5, 6)				✓						✓	✓

Table 4.8: Reduced costs on trailer arcs in VRPTT

The objective functions of the pricing problems for lorries and trailers can therefore be written as shown in Table 4.9.

$$OF^{sub,k}|_{k\in F_L} =$$

$$\sum_{(i,j)\in A^k} c_{ij}^k x_{ij}^k + c_k^{time}(t_d^k - t_o^k)$$

$$- \sum_{l\in RWL_C} \sum_{i\in V_C^l} \sum_{(h,i)\in A^k} \alpha_l^1 x_{hi}^k$$

$$- \sum_{i\in V_{I_T}} \sum_{(h,i)\in A^k} \alpha_i^3 x_{hi}^k$$

$$+ \left(\sum_{k'\in F_T} \sum_{(i,j)\in A^{k'}\setminus A_{k'}^{single}} \alpha_{k'ij}^4 x_{ij}^k \right) \delta_{kk'ij}^2$$

$$+ \sum_{k'\in F_T} \sum_{i\in(V_{C_{LT}}\cup V_{I_T})\cap V^{k'}} \sum_{(h,i)\in A^k} \left(\beta_{k'khi}^1 t_i^k - \beta_{k'khi}^1 T^{max} x_{hi}^k\right)$$

$$- \sum_{k'\in F_T} \sum_{i\in(V_{C_{LT}}\cup V_{I_T})\cap V^{k'}} \sum_{(h,i)\in A^{k'}} \beta_{k'khi}^2 t_i^k$$

$$- \sum_{k'\in com_k} \sum_{i\in V_{C_{LT}}} \sum_{(i,j)\in A^k\cap A^{k'}} \left[\beta_{kk'ij}^3 l_i^k - \beta_{kk'ij}^3 l_j^k + \beta_{kk'ij}^3 (q_k + q_{k'} + su_i) x_{ij}^k\right]$$

$$OF^{sub,k'}|_{k'\in F_T} =$$

$$\sum_{(i,j)\in A^{k'}} c_{ij}^{k'} x_{ij}^{k'}$$

$$- \sum_{i\in V_{I_T}} \left(\sum_{(h,i)\in A^{k'}} \alpha_i^2 x_{hi}^{k'} \right) \delta_{k'vl_i}^1$$

$$+ \sum_{i\in V_{I_T}} \left(\sum_{(h',i)\in A^{k'}} \alpha_i^3 x_{hi}^{k'} \right) \delta_{k'vl_i}^1$$

$$- \sum_{(i,j)\in A^{k'}\setminus A_{k'}^{single}} \alpha_{k'ij}^4 x_{ij}^{k'}$$

$$- \sum_{k\in F_L} \sum_{i\in(V_{C_{LT}}\cup V_{I_T})\cap V^{k'}} \sum_{(h,i)\in A^k} \beta_{k'khi}^1 t_i^{k'}$$

$$+ \sum_{k\in F_L} \sum_{i\in(V_{C_{LT}}\cup V_{I_T})\cap V^{k'}} \sum_{(h,i)\in A^{k'}} \left(\beta_{k'khi}^2 t_i^{k'} - \beta_{k'khi}^2 T^{max} x_{hi}^{k'}\right)$$

$$- \sum_{k\in com_{k'}} \sum_{i\in V_{C_{LT}}} \sum_{(i,j)\in A^k\cap A^{k'}} \left[\beta_{kk'ij}^3 l_i^{k'} - \beta_{kk'ij}^3 l_j^{k'} + \beta_{kk'ij}^3 (q_k + q_{k'} + su_i) x_{ij}^{k'}\right]$$

(continued on next page)

(continued from previous page)

$-\sum_{i\in V_{decouple}}\sum_{(i,j)\in A^{vl_i}}\sum_{(i,j')\in A^k}$	$-\left(\sum_{k\in com_{k'}}\sum_{i\in V_{decouple}}\sum_{(i,j)\in A^{k'}}\sum_{(i,j')\in A^k}\right)$
$\left(\beta^4_{kijj'}l^k_i-\beta^4_{kijj'}l^k_{j'}-\beta^4_{kijj'}q_{vl_i}x^k_{ij'}\right)\delta^2_{kvl_iij'}$	$\left(\beta^4_{kijj'}l^{k'}_i-\beta^4_{kijj'}l^{k'}_j-\beta^4_{kijj'}q_{k'}x^{k'}_{ij}\right)\delta^1_{k'vl_i}$
$-\sum_{i\in V_{transfer}}\sum_{(i,j)\in A^{vl_i}}\sum_{(i,j')\in A^k}$	$-\sum_{k\in F_L}\sum_{i\in V_{transfer}}\sum_{(i,j)\in A^{k'}}\sum_{(i,j')\in A^k}$
$\left(\beta^5_{kijj'}l^k_i-\beta^5_{kijj'}l^k_{j'}-\beta^5_{kijj'}q_{vl_i}x^k_{ij'}\right)$	$\left(\beta^5_{kijj'}l^{k'}_i-\beta^5_{kijj'}l^{k'}_j-\beta^5_{kijj'}q_{k'}x^{k'}_{ij}\right)\delta^1_{k'vl_i}$
$-\sum_{i\in V_{couple}}\sum_{(i,j)\in A^{vl_i}\cap A^k}$	$-\sum_{k\in com_{k'}}\sum_{i\in V_{couple}}\sum_{(i,j)\in A^{k'}\cap A^k}$
$\left(\beta^6_{kij}l^k_i-\beta^6_{kij}l^k_j-\beta^6_{kij}q_{vl_i}x^k_{ij}\right)\delta^2_{kvl_iij}$	$\left(\beta^6_{kij}l^{k'}_i-\beta^6_{kij}l^{k'}_j-\beta^6_{kij}q_{k'}x^{k'}_{ij}\right)\delta^1_{k'vl_i}$
$-\gamma^{\,k}$	$-\gamma^{\,k'}$

Table 4.9: Objective functions of pricing problems in VRPTT

The dual prices for the logically coupling constraints (4.60e)–(4.60j), respectively, the values of the dual variables β in (4.70), induce linear costs on the vertices. These costs depend on the time and load variables. Instead of considering them at the vertices, they can be put on the arcs incident to the respective vertices. Hence, the decisive observation is:

The costs of traversing an arc (i, j) in a pricing problem subnetwork are a function of the service start time at i and the vehicle load at i and j.

4.3.4.3 Identical Pricing Problems

If all lorries are identical, i.e., if there is only one type of lorry, the problem is *symmetric* with respect to the lorries, i.e., the subnetworks for the lorries are structurally identical, and only one lorry pricing problem has to be considered. There are no effects on the trailer pricing problems.

If all trailers are identical, the subnetworks for the trailers are structurally identical as well. However, it is not possible to exploit this fact. Neither the size of a lorry or trailer pricing problem subnetwork nor the number of trailer pricing problems to be considered can be reduced. This is because the above formulation is based on a network with one vertex per physical transshipment location, time window, and trailer, and the number of these transshipment vertices specifies the number of times a load transfer at a physical transshipment location can be performed. Consequently, the number of transshipment vertices in the lorry pricing problem must remain the same whether or not the trailers are identical, and each trailer still has its own distinct subnetwork.

In short:

The VRPTT is easier if all lorries are the same, but it is not easier if all trailers are the same.

If all lorries are identical, and whether or not all trailers are identical, let P_L be the common set of extreme points for lorries, let c^L_{ij} and c^{time}_L be the objective function coefficients for lorries for the distance- and time-dependent costs, let x^L_{ij} be the common binary arc variable for the lorries on arc (i, j), let x^L_{ijp} be the common binary arc variable for the lorries on arc (i, j) for path p, let t^L_{ip} be the common time variable for the lorries at vertex i on path p, and define

$$\lambda^L_p := \sum_{k\in F_L}\lambda^k_p \quad \forall\ p \in P_L. \tag{4.71}$$

The IMP can then be formulated as follows:

$$\sum_{p\in P_L}\left(\sum_{(i,j)\in A^L} c_{ij}^L x_{ijp}^L\right)\lambda_p^L + \sum_{k'\in F_T}\sum_{p\in P^{k'}}\left(\sum_{(i,j)\in A^{k'}} c_{ij}^{k'} x_{ijp}^{k'}\right)\lambda_p^{k'} + \sum_{p\in P_L} c_L^{time}(t_{dp}^L - t_{op}^L)\lambda_p^L \to \min \tag{4.72}$$

subject to

$$\sum_{p\in P_L}\left(\sum_{i\in V_C^l}\sum_{(h,i)\in A^L} x_{hip}^L\right)\lambda_p^L = 1 \qquad \forall\, l \in RWL_C \tag{4.73a}$$

$$\sum_{p\in P^{vl_i}}\left(\sum_{(h,i)\in A^{vl_i}} x_{hip}^{vl_i}\right)\lambda_p^{vl_i} \leqq 1 \qquad \forall\, i \in V_{I_T} \tag{4.73b}$$

$$\sum_{p\in P_L}\left(\sum_{(h,i)\in A^L} x_{hip}^L\right)\lambda_p^L - \sum_{p\in P^{vl_i}}\left(\sum_{(h',i)\in A^{vl_i}} x_{h'ip}^{vl_i}\right)\lambda_p^{vl_i} \leqq 0$$

$$\forall\, i \in V_{I_T} \tag{4.73c}$$

$$\sum_{p\in P^{k'}} x_{ijp}^{k'}\lambda_p^{k'} - \sum_{p\in P_L} x_{ijp}^L\lambda_p^L \leqq 0 \qquad \forall\, k' \in F_T, (i,j) \in A^{k'} \setminus A_{k'}^{single} \tag{4.73d}$$

$$\sum_{p\in P^{k'}} t_{ip}^{k'}\lambda_p^{k'} - \sum_{p\in P_L} t_{ip}^L\lambda_p^L + T^{max}\sum_{p\in P_L} x_{hip}^L\lambda_p^L \leqq T^{max}$$

$$\forall\, k' \in F_T, i \in (V_{C_{LT}} \cup V_{I_T}) \cap V^{k'}, (h,i) \in A^L \tag{4.73e}$$

$$\sum_{p\in P_L} t_{ip}^L\lambda_p^L - \sum_{p\in P^{k'}} t_{ip}^{k'}\lambda_p^{k'} + T^{max}\sum_{p\in P^{k'}} x_{hip}^{k'}\lambda_p^{k'} \leqq T^{max}$$

$$\forall\, k' \in F_T, i \in (V_{C_{LT}} \cup V_{I_T}) \cap V^{k'}, (h,i) \in A^{k'} \tag{4.73f}$$

$$\sum_{p\in P_L}\left(l_{ip}^L - l_{jp}^L\right)\lambda_p^L$$

$$+\sum_{p\in P^{k'}}\left(l_{ip}^{k'} - l_{jp}^{k'}\right)\lambda_p^{k'} + (q_L + q_{k'} + su_i)\left(\sum_{p\in P_L} x_{ijp}^L\lambda_p^L + \sum_{p\in P^{k'}} x_{ijp}^{k'}\lambda_p^{k'}\right) \leqq 2q_L + 2q_{k'} + su_i$$

$$\forall\, i \in V_{C_{LT}}, k' \in F_T, (i,j) \in A^L \cap A^{k'} \tag{4.73g}$$

$$\sum_{p\in P^{vl_i}}\left(l_{ip}^{vl_i} - l_{jp}^{vl_i}\right)\lambda_p^{vl_i} + \sum_{p\in P_L}\left(l_{ip}^L - l_{j'p}^L\right)\lambda_p^L + \sum_{p\in P^{vl_i}}\left(q_{vl_i}x_{ijp}^{vl_i}\right)\lambda_p^{vl_i} + \sum_{p\in P_L}\left(q_{vl_i}x_{ij'p}^L\right)\lambda_p^L \leqq 2q_{vl_i}$$

$$\forall\, i \in V_{decouple}, (i,j) \in A^{vl_i}, (i,j') \in A^L \tag{4.73h}$$

$$\sum_{p\in P^{vl_i}}\left(l_{ip}^{vl_i} - l_{jp}^{vl_i}\right)\lambda_p^{vl_i} + \sum_{p\in P_L}\left(l_{ip}^L - l_{j'p}^L\right)\lambda_p^L + \sum_{p\in P^{vl_i}}\left(q_{vl_i}x_{ijp}^{vl_i}\right)\lambda_p^{vl_i} + \sum_{p\in P_L}\left(q_{vl_i}x_{ij'p}^L\right)\lambda_p^L \leqq 2q_{vl_i}$$

$$\forall\, i \in V_{transfer}, (i,j) \in A^{vl_i}, (i,j') \in A^L \tag{4.73i}$$

$$\sum_{p\in P^{vl_i}} \left(l_{ip}^{vl_i} - l_{jp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p\in P_L} \left(l_{ip}^{L} - l_{jp}^{L}\right) \lambda_p^{L} + \sum_{p\in P^{vl_i}} \left(q_{vl_i} x_{ijp}^{vl_i}\right) \lambda_p^{vl_i} + \sum_{p\in P_L} \left(q_{vl_i} x_{ijp}^{L}\right) \lambda_p^{L} \leqq 2q_{vl_i}$$

$$\forall\, i \in V_{couple}, (i,j) \in A^{vl_i} \cap A^L \tag{4.73j}$$

$$\lambda_p^L = \sum_{k\in F_L} \lambda_p^k \qquad \forall\, p \in P_L \tag{4.73k}$$

$$\lambda_p^L \geqq 0 \qquad \forall\, p \in P_L \tag{4.73l}$$

$$\sum_{p\in P_L} \lambda_p^k \leqq 1 \qquad \forall\, k \in F_L \tag{4.73m}$$

$$\sum_{p\in P^{k'}} \lambda_p^{k'} \leqq 1 \qquad \forall\, k' \in F_T \tag{4.73n}$$

$$\lambda_p^k \geqq 0 \qquad \forall\, k \in F, p \in P^k \tag{4.73o}$$

$$x_{ij}^L = \sum_{p\in P_L} x_{ijp}^L \lambda_p^L \qquad \forall\, (i,j) \in A^L \tag{4.73p}$$

$$x_{ij}^L \in \{0,1\} \qquad \forall\, (i,j) \in A^L \tag{4.73q}$$

$$x_{ij}^{k'} = \sum_{p\in P^{k'}} x_{ijp}^{k'} \lambda_p^{k'} \qquad \forall\, k' \in F_T, (i,j) \in A^{k'} \tag{4.73r}$$

$$x_{ij}^{k'} \in \{0,1\} \qquad \forall\, k' \in F_T, (i,j) \in A^{k'} \tag{4.73s}$$

The dual variables in (4.70), the reduced costs for paths and arcs, and the objective functions for the pricing problems given in Tables 4.4–4.9 then lose their k indices (for the lorries), as (4.73k) and (4.73m) imply

$$\sum_{p\in P_L} \lambda_p^L = \sum_{p\in P_L} \sum_{k\in F_L} \lambda_p^k = \sum_{k\in F_L} \sum_{p\in P_L} \lambda_p^k \leqq \sum_{k\in F_L} 1 = |F_L|, \tag{4.74}$$

cf. Desaulniers et al. 1998, p. 85; i.e., (4.73k) and (4.73m) can be replaced by (4.74). Indeed, the right hand side in constraint (4.74) can be set to an arbitrarily high value, respectively, can be dropped altogether, reflecting a situation where the number of lorries is unlimited.

Note that all x_{ij}^L variables remain binary, because the arc (o,d) was removed. This means that, in any feasible solution, no arc is traversed by more than one lorry, and this is why the third summand in the above objective function correctly accounts for the time-dependent costs.

The LP relaxation of (4.72)–(4.73), however, is weaker than that of (4.68)–(4.69), because the disaggregated constraints (4.69e)–(4.69j) are stronger than the aggregated constraints (4.73e)–(4.73j).

In practice, there are often two types of lorry and two types of trailer, and only one lorry type can pull both trailer types. Generally speaking, if there are several subsets or *classes* of identical lorries, the aggregation performed in (4.72)–(4.73) can be performed for each class separately. The L indices must then be sub-indexed, and slight modifications of (4.72)–(4.73) and a little more writing effort are necessary (therefore, this is not fully carried out here): Let K_L be the number of lorry classes, and let $\{L_1, \ldots, L_{K_L}\}$ be the set of lorry classes. Then, modified vehicle compatibility functions $\widetilde{com}_{L_r}$ for the

lorry classes and $\widetilde{\widetilde{com}}_{k'}$ for the trailers are necessary. The second term on the left hand side of (4.73d), for example, becomes

$$- \sum_{L_r \in \widetilde{\widetilde{com}}_{k'} \cap \{\tilde{L}_r \in \{L_1,\ldots,L_{K_L}\}:(i,j)\in A^{\tilde{L}_r}\}} \sum_{p \in P_{L_r}} x_{ijp}^{L_r} \lambda_p^{L_r}.$$

It is also possible to aggregate over identical vehicles in the arc variable formulation (4.59)–(4.62). However, the underlying network must then be modified to make sure that the resource variables for load and time can be updated correctly at the end of each vehicle's itinerary. This is not a problem at customer or transshipment vertices, because only one lorry and one trailer (at most) visit such vertices, but the end depot vertex is reached by all vehicles. Hence, an end depot vertex for each trailer and for each lorry must be introduced (which means that the number of vehicles cannot be arbitrary). These new depot vertices replace the single depot vertex, and they can be reached from all vertices that have emanating arcs leading to the end depot vertex in the original network. Moreover, an arc from each trailer depot vertex $d^{k'}$ for trailer k' to each lorry depot vertex d^k for lorry k with $k \in com_{k'}$ must be added. No arcs emanate from the lorry depot vertices. The resulting modifications to formulation (4.59)–(4.62) are straightforward.

4.3.5 Modified Formulations

The turn variable formulation requires that each arc is traversed at most once, the arc variable and the path variable formulations require that each vertex is visited at most once. Without such postulates, unless good upper bounds for the number of traversals of each arc by each vehicle, respectively, for the maximum number of visits of each vehicle at each vertex, can be determined, it is impossible to state a mathematical programming formulation: If it is not clear how many times an arc may be traversed or a vertex may be visited, there is no way to index the load and time variables. This section presents possible modifications to the arc variable formulation that overcome this difficulty, albeit at the cost of an enlarged network and at the cost of loss of precision. The decisive ideas are

- to discretize the load transfer amounts, i.e., to specify all possible load transfer amounts at each transshipment location, and
- to introduce a fixed schedule network for the trailers (or even for lorries and trailers) to get rid of the 'same departure time constraints' (4.60e) and (4.60f).

The discretization approach is a technique that is also used in the context of the split delivery vehicle routing problem, where the fraction of a certain customer's demand satisfied by a certain vehicle is restricted to a discrete number of possible values, cf. Dror/Langevin 2000, p. 315. Acyclic fixed-schedule time-space networks are used in many papers on time-constrained vehicle routing and scheduling problems (see the survey by Desaulniers et al. 1998).

If only the load transfer amounts are discretized, (4.59)–(4.62) changes as follows:

- S^{plta}, the *set of possible load transfer amounts*, is specified. To keep the network size limited, $|S^{plta}| \leqq 4$. For each triple (k', l, τ), i.e., for each transshipment vertex in (4.59)–(4.62), there are $|S^{plta}|$ transshipment vertices $v_{\tau s}^{k'l}$, $s \in S^{plta}$, each of which represents a load transfer of s units of load to trailer k' at location l in time window τ. The load of a lorry visiting such a transshipment vertex then decreases by s; the load of the pertinent trailer increases by s. The arc traversal times of arcs leaving transshipment vertices are changed to reflect the now known service times at their tails, similar to the arcs leaving customer vertices.

  If the definition of the supply function, su, is changed such as to reflect the discretized load transfer amounts, the following constraints are affected:

  – (4.60h) become

$$x_{ij}^{k} = 1 \Rightarrow l_i^k - su_i \leqq l_j^k \qquad \forall\, i \in V_{decouple}, k \in com_{vl_i}, (i,j) \in A^k \tag{4.75a}$$

$$x_{ij}^{vl_i} = 1 \Rightarrow l_i^{vl_i} + su_i \leqq l_j^{vl_i} \qquad \forall\, i \in V_{decouple}, (i,j) \in A^{vl_i}, \tag{4.75b}$$

and (4.60i) and (4.60j) change accordingly.

- (4.61c) become redundant.
- (4.61g) and (4.61h) become redundant, because of the modified constraints (4.60h), (4.60i), and (4.60j).
- (4.61k) and (4.61l) become redundant, as they can be subsumed under constraints (4.61j) by also considering $i \in V_{I_T}$ there, respectively, by considering (4.61j) also for $i \in V_{I_T}$, $(i,j) \in A^{vl_i}$.

- For the load update constraints at trailer customers, several options exist:
  - Assume the complete supply of the customer is loaded onto the lorry.
  - Assume the complete supply of the customer is loaded onto the trailer.
    This and the previous option are restrictive in the sense that a lorry-trailer combination may not be able to visit a trailer customer, although the overall remaining capacity of the combination is still sufficient to collect the customer's supply, because the remaining capacity of the lorry or trailer is too small. These options may even lead to infeasibilities if there are trailer customers with a supply greater than the capacity of the biggest available lorry or trailer.
  - Assume several allowed partitions of the supply to divide it up between lorry and trailer.

Equivalently, the original VRPTT network can be enlarged by adding $|S^{plta}| - 1$ parallel arcs for each arc emanating from a transshipment vertex (and possibly from a trailer customer vertex).

The problem when only these changes are made is that the same departure time constraints (4.60e) and (4.60f) remain. To avoid them, the network can be further changed as follows. For each customer and transshipment vertex resulting from the above modifications, the associated time window is interpreted differently: For a vertex i with a time window of $[twa_i, twb_i]$, twb_i is viewed as *the point in time where any vehicle visiting i leaves i*. Waiting is still allowed. This makes the time variables redundant. The service times for all services to be performed at any vertex are simply added to the traversal times of the entering or emanating arcs. The planning horizon is then partitioned into subperiods, and, considering the original time windows of the locations, one vertex for each customer and transshipment vertex of the original network is introduced. This leads to an acyclic network.

This constitutes a very restrictive approach, too. The inaccuracy thus introduced depends largely on the number of subperiods. The more subperiods there are, the more accurate the resulting network is. However, there is a trade-off in that more subperiods lead to an increased network size. The subperiod length desirable for realistic instances, together with the length of the planning horizon, may result in a network of prohibitively large size.

A possible compromise consists in making fixed schedule vertices only for the transshipment locations and the trailer customers and keeping the time variables for the lorries. The update of the time variables at the fixed schedule vertices then works as follows. If the departure time at a fixed schedule vertex i is t at the outset, the time window of i is set to $[twa_i, t - t_i^{service}]$, where $t_i^{service}$ corresponds to the time needed to collect the customer supply, respectively, the time needed to transfer the fixed amount of load. The traversal time of all arcs emanating from i is then increased by $t_i^{service}$. Consider the example situation depicted in Figure 4.7. Vertices i_1, i_2, i_3 and the unnamed vertices correspond to one physical transshipment location l and one trailer k' during different subperiods/time windows ([8:01, 8:20], [8:21, 8:40], and [8:41, 9:00]) and for different load transfer amounts (0, 5 and 10 units). In the figure, lorry k_1 pulls trailer k' to transshipment vertex i_1, which has a twenty-minute time window of [8:01, 8:20] and

corresponds to a load transfer amount of zero. k_1 decouples k' and moves on along arc a_1. k' performs a temporal and logical move to vertex i_2, which corresponds to l during the time window [8:21, 8:40] and to a load transfer amount of five units. This load is transferred by lorry k_2, which arrives at i_2 via arc a_2 and leaves location l along arc a_3. By definition, k' is at vertex i_2 at 8:21, and the load transfer may start as soon as k_2 arrives, as long as it arrives soon enough to complete the transfer of five units of load by 8:40. Hence, if a load transfer of five units takes five minutes, the time window for i_2 is set to [8:21, 8:35]. Trailer k' performs yet another temporal and logical move to vertex i_3 and is coupled there at 8:41 by lorry k_3, which arrives along arc a_4 and moves on along arc a_5, pulling k' with it. The only additional change to (4.59)–(4.62) is that the beginning of the service of lorry k at trailer customer or transshipment vertex i is exactly equal to the modified right time window bound twb_i.

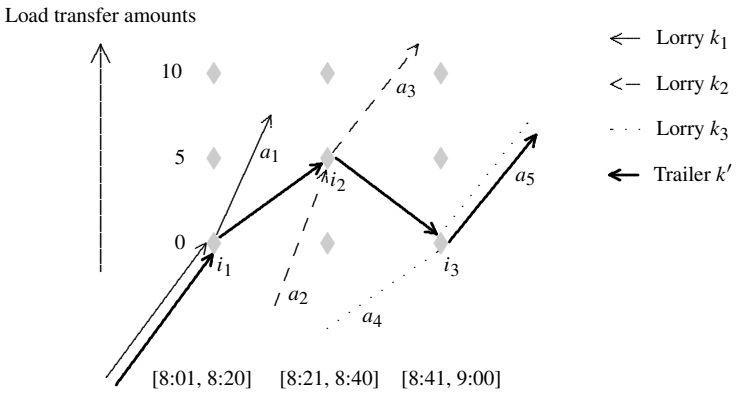


Figure 4.7: Transshipment vertices with discretized load transfer amounts and fixed schedule

The following cases are possible:

- A trailer is only parked. No load transfer is performed. The trailer is coupled again by the lorry that parked it or by a different one. This is represented by the trailer moving in time between two or more vertices corresponding to a load transfer amount of zero.
- The trailer is parked, but not decoupled; only a load transfer is performed. This is represented by the trailer moving directly with a lorry to a vertex corresponding to a positive load transfer amount, and leaving this vertex again along the same arc as the lorry.
- The lorry pulling the trailer to the transshipment location performs a load transfer, decouples, and a different lorry re-couples the trailer later. This is represented by the trailer moving directly with a lorry to a vertex corresponding to a positive load transfer amount, and then moving logically in time to a vertex corresponding to a load transfer amount of zero.
- Several lorries perform a load transfer to one trailer at one transshipment location. This is schematically depicted in Figure 4.8. The depicted lorry arcs could all represent one and the same lorry, or they could represent up to four different lorries.

As mentioned at the beginning of this section, the above approach can, strictly speaking, no longer be called exact. From a practical point of view, it may be argued that it is preferable to have a heuristic solution to an exact formulation, i.e., to a formulation that represents reality as correctly and precisely as possible, than to have an optimal solution to a less exact formulation. However, the idea of discretizing the load transfer amounts maintains exactness of the formulation in cases where the goods to be collected at customers are not homogeneous.

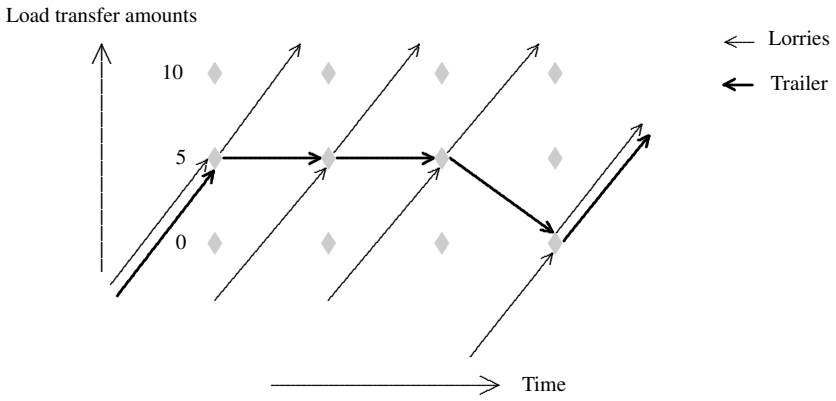


Figure 4.8: Several lorries perform a load transfer to one trailer at one transshipment location

4.4 A Branch-and-Price Algorithm for the VRPTT?

The path variable formulation given above can, in principle, be used to solve the VRPTT by branch-and-price. However, the interdependencies between lorries and trailers cause considerable difficulties for the solution of the pricing problems. These difficulties are discussed in this section.

As mentioned above, the pricing problems for the VRPTT are ESPPRCs, and for each vehicle, an ESPPRC must be solved. If these ESPPRCs are to be solved by a labelling algorithm, the resources for vehicle k are:

- an unconstrained, cardinally scaled resource for costs and
- one cardinally scaled resource for each of the two resource variables used in (4.59)–(4.62), i.e., for load and time

The unconstrained resource measuring the costs is denoted by r^{cost}. The corresponding resource variable is σ_i^{cost}, and the REF is $f^{r^{cost}}$. The resource for time is r^{time} with resource variable σ_i^{time} and REF $f^{r^{time}}$, and the resource for load is r^{load} with resource variable σ_i^{load} and REF $f^{r^{load}}$. The resource windows at a vertex i are $]-\infty, +\infty[$ for cost, $[twa_i, twb_i]$ for time, and $[0, q_k]$ for load. The REFs for the VRPTT pricing problem for lorry $k \in F$ for all arcs $(i, j) \in A^k$ are as follows:

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}, \sigma_o^{time}, \sigma_j^{time}, \sigma_i^{load}, \sigma_j^{load}) = \sigma_i^{cost} + \tilde{c}_{ij}^k(\sigma_j^{time}, \sigma_i^{load}, \sigma_j^{load}) + (\sigma_j^{time} - \sigma_o^{time})c_k^{time} \tag{4.76}$$

$$f_{ij}^{r^{time}}(\sigma_i^{time}, \sigma_i^{load}, \sigma_j^{load}) = \begin{cases} \max\left\{twa_j, \sigma_i^{time} + \tau_{ij}^{tr}\right\} & , i \in \{o\} \cup V_C \\ \max\left\{twa_j, \sigma_i^{time} + \tau_{ij}^{tr} + (\sigma_i^{load} - \sigma_j^{load})\tau^{lt}\right\}, & i \in V_{I_T} \end{cases} \tag{4.77}$$

$$f^{r^{load}}(\sigma_i^{load}) = \begin{cases} \sigma_i^{load} & , i \in \{o\} \cup V_{C_{LT}} \\ \sigma_i^{load} + su_i, & i \in V_{C_L} \\ 0 & , i \in V_{I_T} \end{cases} \tag{4.78}$$

The REFs for the VRPTT pricing problem for trailer $k' \in F_T$ on arc $(i, j) \in A^{k'}$ are as follows:

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}, \sigma_i^{time}, \sigma_j^{time}, \sigma_i^{load}, \sigma_j^{load}) = \sigma_i^{cost} + \tilde{c}_{ij}^k(\sigma_j^{time}, \sigma_i^{load}, \sigma_j^{load}) \tag{4.79}$$

$$f_{ij}^{r^{time}}(\sigma_i^{time}, \sigma_i^{load}, \sigma_j^{load}) = \begin{cases} \max\left\{twa_j, \sigma_i^{time} + \tau_{ij}^{tr}\right\} & , i \in \{o\} \cup V_{C_{LT}} \\ \max\left\{twa_j, \sigma_i^{time} + \tau_{ij}^{tr} + (\sigma_j^{load} - \sigma_i^{load})\tau^{lt}\right\}, & i \in V_{I_T} \end{cases} \tag{4.80}$$

$$f^{r^{load}}(\sigma_i^{load}) = \sigma_i^{load} \tag{4.81}$$

$\tilde{c}_{ij}^k(\sigma_j^{time}, \sigma_i^{load}, \sigma_j^{load})$ is the function specifying the reduced costs on arc (i, j) according to Tables 4.5 and 4.6.

The first difficulty that arises when trying to solve the lorry pricing problems by a labelling algorithm is the time-dependent costs (the last term in the cost REF for lorries). With time-dependent costs and the possibility of having to wait at a vertex (due to the static and dynamic time windows), it need no longer be optimal to leave the start depot vertex at time zero, i.e., to set $\sigma_o^{time} = 0$. Desaulniers/Villeneuve 2000 have shown that such a problem can be solved as an (E)SPPRC by introducing two additional resources that take into account the time costs. The *definition* and *interpretation* of these resources is rather involved. The reader interested in the details is referred to the above paper. Suffice it here to say that the *implementation* (in a computer code) is straightforward, so if it were only for the time-dependent costs, the lorry pricing problem in the VRPTT could be solved by a labelling algorithm without much programming or computational overhead.

However, there are more serious issues to consider: To solve (E)SPPRCs by a labelling algorithm, it is highly desirable that all REFs possess the following two properties (Desaulniers et al. 1998, p. 82):

(i) All REFs for an arc (i, j) should depend only on the resource vector at i.
If this property holds, intermediate resource levels can be computed which provide (good) lower bounds for the values of the resource variables.

(ii) All REFs should be non-decreasing.
If this property holds, the lowest costs at vertex j are always attained at the lowest feasible values of the resource variables.

Unfortunately, neither property is fulfilled in the VRPTT pricing problems. To see why this is the case and why this is a difficulty, first consider the REFs for cost, time, and load in the VRPTW:

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}) = \sigma_i^{cost} + \tilde{c}_{ij}$$

$$f_{ij}^{r^{time}}(\sigma_i^{time}) = \max\{twa_j, \sigma_i^{time} + \tau_{ij}^{tr}\}$$

$$f_{ij}^{r^{load}}(\sigma_i^{load}) = \sigma_i^{load} + su_i$$

It is easy to see that these REFs fulfil the above two properties. For example, the load after having visited customer i is at least as much as the load at the point in time when customer i was reached plus customer i's supply, i.e., $\sigma_i^{load} + su_i$ is a valid lower bound. But it is more than that: It is a *sharp* bound. The load after having visited customer i can safely be assumed to be *equal* to $\sigma_i^{load} + su_i$. When extending a label at i along the arc (i, j), it is not necessary to create a label with a different value for σ_j^{load}. The same holds for the other two resources.

In the VRPTT, things are different. To see that the first property is violated, observe that constraints (4.60e)–(4.60j), which determine the update of the time and load variables at trailer customer and transshipment vertices, are logically coupling constraints. This means that the REFs for time and load of a vehicle at such vertices depend on the load of other vehicles. Such REFs are not possible. However, as indicated in (4.76)–(4.81), valid lower bounds can still be computed. So, property (i) is not 'really' violated in a narrow sense, but the bounds cannot directly be used for the resource updates in a labelling algorithm. Consider, for example, the lorry REFs. All three of them are interdependent. The value of the REF for time at intermediate vertices, i.e., the value of the resource variable for time at vertex j, σ_j^{time}, depends not only on σ_i^{time}, but also on the values of the resource variables for load at the tail and head of an arc (i, j). The value of the REF for cost, in turn, depends on the values of the resource variables for time. A sequential computation of the REFs is possible in a labelling algorithm, so the REF for load

can be computed first, then the REF for time, and then the REF for cost. But, whereas the REF value for load at lorry customer vertices is sharp as in the VRPTW, the REF value for load at trailer customer and intermediate vertices is useless. The good to be collected in the VRPTT is assumed to be homogeneous, so there is an infinite number of possible extensions of a label at such vertices. It is not clear which of these extensions could lead to Pareto-optimal o-d-paths, respectively, it is not clear which of these extensions are dominated. The computation of the REF for time thus becomes useless as well as impossible. Useless, because the useless value of σ_i^{load} is an input to the REF, and impossible, because there are infinitely many possible values for σ_j^{load}. The same holds for the evaluation of the REF for cost.

To see that the second property is also violated, consider the objective functions of the lorry and trailer subproblems in Table 4.9. The dual prices for the logically coupling constraints (4.60e)–(4.60j), respectively, the values of the dual variables β in (4.70), induce linear costs on the vertices. These costs depend on the time and load variables, and they are *not* the same for all vertices, contrary to the time-dependent costs. (This is the decisive point that makes time-dependent costs easy to consider in a labelling algorithm.) Holding other things equal, if all dual prices at the vertices are positive, it is always best to reach a vertex at the earliest possible time with the least possible load. If all dual prices are negative, it is the other way round. However, dual prices may take positive as well as negative values. Thus, vertices with positive as well as negative dual prices for time and/or load may appear in a path. This means that for a vertex i with a time window of $[twa_i, twb_i]$ and a vehicle k with a capacity of q_k, the following is possible. For every service start time t_i^k with $twa_i \leqq t_i^k < t' < twb_i$, the costs may decrease with increasing t_i^k; for every service start time with $t' \leqq t_i^k \leqq twb_i$, the costs may increase with increasing t_i^k. Similarly, for every load l_i^k with $0 \leqq l_i^k < q < q_k$, the costs may decrease with increasing values of l_i^k, and for every load ranging from q to q_k, the costs may increase with increasing load. In other words, the cost REF is no longer non-decreasing, and it is no longer generally optimal to set the resource variables to their lowest feasible values. Essentially, the determination of a cost-optimal schedule and load plan for a *fixed* path becomes an optimization problem in itself.

As an example, consider a path (o, i, j) with $[twa_o, twb_o] = [0, 0]$, $[twa_i, twb_i] = [10, 20]$, $[twa_j, twb_j] = [30, 50]$, $c_{oi} = c_{ij} = 0$, $\tau_{oi}^{tr} = 10$, $\tau_{ij}^{tr} = 20$, and dual prices $\beta_o = 0$, $\beta_i = -2$, and $\beta_j = +1$. Disregarding load and time-dependent costs, the reduced cost functions along arcs (o, i) and (i, j) are

$$\tilde{c}_{oi}^k(\sigma_i^{time}) = -2\sigma_i^{time} \text{ for } \sigma_i^{time} \in [10, 20]$$

and

$$\tilde{c}_{ij}^k(\sigma_j^{time}) = \sigma_j^{time} \text{ for } \sigma_j^{time} \in [30, 50]$$

respectively. Thus, the cost REF along arc (o, i) is

$$f_{oi}^{r^{cost}}(\sigma_o^{cost}, \sigma_i^{time}) = 0 + \tilde{c}_{oi}^k = -2\sigma_i^{time} \text{ for } \sigma_i^{time} \in [10, 20],$$

and the cost REF along arc (i, j) is

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}, \sigma_j^{time}) = \sigma_i^{cost} + \tilde{c}_{ij}^k = f_{oi}^{r^{cost}} + \tilde{c}_{ij}^k = -2\sigma_i^{time} + \sigma_j^{time} \text{ for } \sigma_j^{time} \in [30, 50].$$

For any $\sigma_i^{time}, \sigma_j^{time} \geqq \sigma_i^{time} + \tau_{ij}^{tr} = \sigma_i^{time} + 20$. (Remember that an REF specifies lower bounds for the resource consumption along an arc.) Because of the negative reduced costs β_i, it is best to choose σ_i^{time} as large as possible, i.e., $\sigma_i^{time} = \min\{20, \sigma_j^{time} - 20\}$. Hence, the REF for cost along arc (i, j) is

$$f_{ij}^{r^{cost}}(\sigma_j^{time}) = \begin{cases} -\sigma_j^{time} + 40 \text{ for } \sigma_j^{time} \in [30, 40] \\ \sigma_j^{time} - 40 \text{ for } \sigma_j^{time} \in]40, 50] \end{cases}.$$

This means that, until a service start time of 40 at vertex j, the costs of path (o, i, j) decrease with increasing service start time at j, and for a service start time of between 40 and 50 at j, the costs increase with increasing service start time at j.

A non-decreasing REF (of costs against costs) along an arc (i, j) is depicted in Figure 4.9(a). It says that, holding other things equal, i.e., for fixed time and load, traversing arc (i, j) reduces the costs by 20. Observe that the cumulated resource consumption after extension is less than before the extension. This is not the point with non-decreasing REFs. The point is that the cumulated resource consumption after extension is higher, the higher the cumulated resource consumption before extension is. In Figure 4.9(b), a not non-decreasing REF (of costs against time) is depicted (it also contains increasing parts, so it cannot be called decreasing). It says that, holding other things equal, i.e., for fixed costs and load, the total costs after extension decrease for any service start time up to t', and increase for any service start time later than t''.

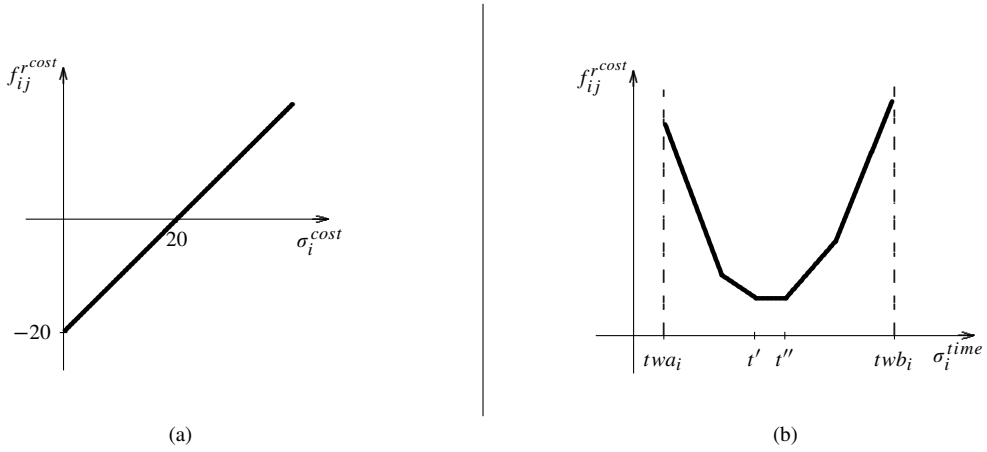


Figure 4.9: (a) Non-decreasing and (b) not non-decreasing REF

The issue is discussed thoroughly in Ioachim et al. 1998 for the case of linear time costs at the vertices. The authors show that the cost function is piecewise linear with a finite number of pieces depending linearly on the number of vertices in the path, and that the general shape of the cost function is as depicted in Figure 4.9(b) (possibly with $t' = t''$). In the VRPTT, the situation is even more complicated, because there are linear time *and* load costs. The cost REF for an arc (i, j) depends on the service start time at i and on the load variables at i and j; hence, there are four dimensions to consider (the function graph is four-dimensional).

Summing up, violation of REF properties (i) and (ii) means that it is neither clear how to extend a label along an arc, nor when a label at a vertex may be discarded because it is dominated by one or more other labels. It is necessary to consider all feasible extensions, but there are uncountably many. The standard resource concept as described in Chapter 2 is not applicable.

In view of these arguments, how can the pricing problem be solved? The following alternatives exist:

- Modify/simplify the problem to avoid not non-decreasing REFs.
  The issue with the load costs in the VRPTT also arises in the split delivery vehicle routing problem. Dror/Langevin 2000, p. 305 ff., present a column generation approach for this problem, where they discretize the number of allowed split deliveries and create one label for each allowed quantity. As has been described in Section 4.3.5, the same is possible in the VRPTT, but the resulting algorithm is no longer really exact.

- Extend the approach of Gendreau et al. 2005.
  Gendreau et al. 2005 use a sophisticated pairwise dominance algorithm based on duality theory for solving the split delivery vehicle routing problem without imposing any restrictions on the split delivery options. It is as yet unclear whether it is possible to adapt this approach to the VRPTT.

- Extend the approach of Ioachim et al. 1998.
  As mentioned above, Ioachim et al. 1998 consider the SPPRC with linear vertex (time) costs. The authors solve the problem with a labelling algorithm by representing the set of feasible resource vectors by its lower envelope (using adequate resources for storing the description of the lower envelope), and by proving that all labels/paths that do not contribute to the lower envelope at a vertex are dominated and can be discarded. (This is an example of *non-pairwise dominance*.)
  What these authors do, in effect, is to extend the resource concept described in Chapter 2: They consider REFs that are not real-valued, but whose values are *intervals* (possible values for time and load) or even *functions* (costs as a function of time and load) instead of scalars. The dominance procedure then consists in comparing sets of functions.
  This seems to be a promising approach. However, Ioachim et al. 1998 describe the lower envelope of the set of feasible resource vectors by a piecewise linear cost function with at most $p+1$ pieces for a path of length p. The corresponding four-dimensional envelope for the VRPTT must have $(p+1)^3$ 'pieces', and in general, if there are r resources, there are $(p+1)^r$ 'pieces'; i.e., the complexity of the lower envelope increases exponentially with the number of resources.

- Use results and algorithms from computational geometry and polyhedral theory.
  For the VRPTT, the set of feasible resource vectors at the end of a (partial) path is a convex polytope. If the polytope corresponding to a label can be successfully described by its extreme points, and if the description of the polytope resulting from extending its corresponding label along an arc can be computed, it is possible to use a point-in-polyhedron algorithm (see, e.g., O'Rourke 1998, p. 245 ff.) for (not necessarily pairwise) dominance: If the polytope of a path p is completely contained in the polytope of another path, or if the polytope is contained in the union of the polytopes of at least two other paths, path p is dominated and can be discarded. Indeed, both tasks (polytope description by extreme points and computing polytopes/labels resulting from path extension steps) can be done in a manner similar to the one used by Ioachim et al. 1998 (see previous item). These authors describe their piecewise linear cost functions of (partial) paths by the functions' breakpoints and provide a procedure for computing the breakpoints of a function after a path extension step. In theory, a point-in-polyhedron test is rather easy for convex polyhedra (needed for pairwise dominance), but somewhat more involved for non-convex polyhedra (needed for non-pairwise dominance); cf. again O'Rourke 1998, ib.
  A similar approach may be possible when the polytopes corresponding to labels are described by their facets.
  However, due to numerical problems on today's digital computers with limited-precision floating-point arithmetic, even an implementation of the approach of Ioachim et al. 1998 in two dimensions is non-trivial, and 'reliable CAD in three dimensions is still ahead of us' (K. Mehlhorn, 2006). Thus, a robust implementation of a point-in-polyhedron algorithm for non-convex four-dimensional polyhedra constitutes a challenge in itself. This is all the more so as such an algorithm is called thousands of times in each iteration of the column generation process at each vertex of the branch-and-price tree, so it would also have to run very fast to be useful.

- Solve the ESPPRC by branch-and-cut.
  The author is aware of only one publication that takes this approach (cf. Bramel/Simchi-Levi 2002, p. 91 f.; the authors describe an unpublished technical report where the pricing problem for the VRP is solved by branch-and-cut). One potential reason for this fact is that, in column generation, it is mostly useful to add more than one negative reduced cost column at each iteration. A labelling algorithm always returns all Pareto-optimal solutions (and hence, often more than one negative reduced cost column), whereas a branch-and-cut algorithm only returns one column (the one with most negative reduced cost).

It was impossible to implement the branch-and-price algorithm for the VRPTT in the time available for this paper. However, it is doubtful whether an implementation of a branch-and-price algorithm would be able to solve instances of more realistic size than the branch-and-cut algorithm, even if a reasonably fast

and robust procedure for the solution of the pricing problems were available. This is because the usefulness of the dual prices may be limited: The dual prices for the logically coupling constraints are added to the original costs of the subnetworks' arcs. In the VRPTW, the only logically coupling constraints are the ones for customer covering. In the VRPTT, there are several types of logically coupling constraint whose dual prices may give conflictive indications. If the dual prices for, e.g., the temporal synchronization constraints, are negative, and the dual prices for, e.g., the load synchronization constraints, are positive, the former may be offset by the latter and virtually no dual information may be present in the subproblems to produce 'useful' paths, thus leading to an aggravation of the tailing-off effect often encountered in column generation.

After these elaborations, it should be evident that the question mark in this section's headline is not a typing error.

4.5 A Branch-and-Cut Algorithm for the Core Problem

In this section, a branch-and-cut algorithm for the core problem, i.e., the arc variable formulation (4.59)–(4.62), is proposed. The valid inequalities used in the algorithm and the employed branching and enumeration strategies are described.

4.5.1 Valid Inequalities

The following inequalities are valid for (4.59)–(4.62).

Supply Collection Cut:

$$\sum_{k \in F} l_d^k \geqq \sum_{l \in RWL_C} cs_l \tag{4.82}$$

This inequality requires that the total customer supply be brought to the depot. In a fractional solution, this might not automatically be the case because of the flow variables that appear in the load update constraints (4.60g)–(4.60j) and (4.61d)–(4.61h). To see that the cut constitutes a valid inequality, note that the following is true in any feasible solution to (4.60)–(4.62): For each lorry (or lorry-trailer combination), the complete load after visiting a vertex i and traversing an arc (i, j) (and an arc (i, j')) is greater than or equal to the complete load of the lorry (or lorry-trailer pair) when reaching vertex i plus the supply of vertex i (because of the load update constraints). Each vehicle reaches the end depot vertex exactly once (because of constraints (4.61a)) and does not leave it again (because there is no arc emanating from the end depot). The '$\geqq$' sign can also be replaced by '$=$', because in reality, it is not possible to collect more supply than there is. Strictly speaking, the inequality is then no longer valid, because feasible solutions are cut off. However, none of these solutions can be optimal if the load transfer times and the time costs are strictly positive.

Trailer Flow Cut:

$$\sum_{k' \in F_T} \sum_{\{(i,d) \in A^{k'} : i \neq o\}} x_{id}^{k'} \geqq num^{trailers} \tag{4.83}$$

This inequality requires that the flow of trailers into the end depot over arcs other than (o, d) be at least equal to the minimum number of necessary trailers, $num^{trailers}$. As the total customer supply must be brought to the depot (see the supply collection cut), the inequality is valid. A lower bound for $num^{trailers}$ can be computed as follows. First, $su^{trailer}$, the total customer supply that must be brought to the end depot by a trailer, is computed:

$$su^{trailer} = \sum_{l \in RWL_C} cs_l - \sum_{k \in F_L} q_k. \tag{4.84}$$

Then, all trailers are sorted by non-increasing total capacity and the trailer capacities are summed up until the sum exceeds $su^{trailer}$. For each trailer that contributes to the sum, $num^{trailers}$ is increased by one.

Lorry Flow Cut:

$$\sum_{k \in F_L} \sum_{\{(i,d) \in A^k : i \neq o\}} x_{id}^k \geqq num^{lorries} \tag{4.85}$$

This inequality is also valid because of the supply collection cut. Similar to the lower bound for the number of necessary trailers, a lower bound for the number of necessary lorries, $num^{lorries}$, can be determined by computing su^{lorry}, the total customer supply that must be brought to the end depot by a lorry, instead of $su^{trailer}$, as

$$su^{lorry} = \sum_{l \in RWL_C} cs_l - \sum_{k' \in F_T} q_{k'}. \tag{4.86}$$

However, the resulting bound is not very strong. A better bound can be computed as follows. All possible lorry-trailer combinations and single lorries are sorted by non-increasing total capacity and the capacities are summed up until the sum exceeds the total customer supply. For each summand, $num^{lorries}$ is increased by one. As trailers are usually compatible with more than one lorry, all lorry-trailer combinations which are no longer possible because the respective lorry or trailer or both has/have already been used with another lorry or trailer must be deleted during the summation.

Connectivity Cuts:

$$\sum_{\{a \in A^k : ta_a \notin S \cup \{i\}, he_a \in S\}} x_a^k - x_{hi}^k \geqq 0 \quad \forall\, k \in F, S \subsetneqq V^k \setminus \{d\}, (h,i) \in A^k, h \in S \tag{4.87}$$

These cuts require that, for each vehicle and for each subset of the vertex set not containing the end depot vertex, if the vehicle uses an arc (h,i) whose tail is in the subset, there must be another arc a entering this subset whose tail must not be i. This is implied by (4.61a).

An exact procedure for the separation of violated connectivity cuts works as follows. For each $x_{hi}^k > 0$, a maximum flow problem from i to d is solved in the support graph where the arc capacities are equal to the values of the flow variables of vehicle k, but where vertex h, and, hence, all arcs incident to it, are deleted. This yields a cut with i on one side and d on the other. If the value of the maximum flow is less than x_{hi}^k, a violated connectivity cut has been found.

This procedure is correct, because in any feasible solution to the mixed binary problem, $x_{hi}^k > 0$ means $x_{hi}^k = 1$, and h is on the same side as i in any i-d-cut, because each vertex is visited by each vehicle at most once. In other words, there is no feasible solution where h is on the unique path of vehicle k from i to d, if arc (h,i) is used by vehicle k. This means that there must be a flow of at least x_{hi}^k from the i side of the cut to the d side of the cut, but this flow must not pass through h.

Cuts (4.87) can still be lifted. If one of the incident vertices h, i of an arc (h,i) is a coupling vertex, the corresponding decoupling and intermediate vertices cannot be visited any more, and, hence, any potential cut arc emanating from or leading to such a vertex can be discarded. If h or i is an intermediate vertex, the same holds for arcs incident to the corresponding decoupling vertex.

Generalized κ-Path Cuts:
κ-path cuts are well-known valid inequalities for the VRP(TW), cf. Kohl et al. 1999. In the VRPTW with single time windows for each customer and with homogeneous fleet (in particular, without trailers), the

idea of κ-path cuts for $\kappa \geqq 1$ is that, for any subset S of the customer vertices with a cumulated supply of more than $\kappa - 1$ times the vehicle capacity, at least κ vehicles must visit this subset, respectively, at least κ arcs must enter this subset. (The cuts can be strengthened by computing the minimal number of vehicles needed to collect the supply of the customers in the subset. This number may be strictly greater than $\lceil \sum_{i \in S} su_i/q \rceil$.)

Generalized 1-Path Cuts:

$$\sum_{k \in F_L} \sum_{\{a \in A^k : loc_{ta_a} \notin S \ni loc_{he_a}\}} x_a^k \geqq 1 \quad \forall\, S \subseteqq RWL_C \tag{4.88}$$

These inequalities require that there be a lorry flow of at least one into each subset of the customer vertices containing all vertices corresponding to any customer with a vertex in the subset. They are a straightforward generalization of 1-path cuts for the case where there may be more than one vertex per customer.

If each customer has only one time window (in which case there is only one vertex per customer), violated inequalities of this type can be separated by solving a maximum flow problem in the support graph considering only the lorry flow variables, where the source is a customer vertex and the sink is the end depot vertex. In a minimal capacity cut, the customer vertex is on one side and the end depot vertex on the other. If the maximum flow value is less than one, a violated inequality has been found. Because of the flow conservation constraints, it is not useful to compute also the maximum flows from the start depot vertex to a customer vertex. If there are customers with several time windows and, hence, several vertices, a condensed support graph must be considered, where the vertices corresponding to a customer are shrunk into one and the flow values on the remaining arcs are equal to the sum of the corresponding flow values of the original graph.

Every 1-path cut where the subset of real-world customer locations consists only of one customer location $l \in RWL_C$ can be expressed as the sum of one or more connectivity cuts, by summing up all inequalities where the subset of $V^k \setminus \{d\}$ is $\{i \in V^k : loc_i = l\}$ over all lorries and all predecessors of i. This is because the customer covering constraints (4.60a) force the sum of the x_{hi}^k terms to be equal to one in this case. For a subset S of real-world customer locations with $|S| \geqq 2$, the sum of connectivity cuts,

$$\sum_{k \in F_L} \sum_{\{i \in V : loc_i \in S\}} \sum_{\{a \in A^k : loc_{ta_a} \notin S \ni loc_{he_a} = i\}} x_a^k \geqq |S| - \sum_{k \in F_L} \sum_{\{(h,i) \in A^k : loc_h, loc_i \in S\}} x_{hi}^k, \tag{4.89}$$

is stronger than the 1-path cut corresponding to this subset. (The left hand sides are equal, the right hand side of (4.89) is greater than or equal to the right hand side of the corresponding 1-path cut, and, depending on the second summand, the right hand side may be strictly greater.) Also, the disaggregated connectivity cuts (4.87) are stronger than the aggregated ones (4.89).

Generalized 2-Path Cuts:
Kohl et al. 1999 have used 2-path cuts in a branch-and-price-and-cut algorithm for the VRPTW. Their results clearly demonstrate the strength of these cuts. However, there is a trade-off to be observed when using the cuts (ib., p. 106 f.). In the presence of time windows, it is difficult to compute the minimal number of vehicles needed to service a certain subset of customers, because not only the vehicle capacity must be taken into account, but also the time windows. Using only capacity considerations in the computation will lead to weak cuts, especially when the time windows are rather tight. Considering also the time windows requires the (exact) solution of a travelling salesman problem with time windows (TSPTW) for (exact) separation of violated 2-path cuts, and the (exact) solution of a VRPTW for (exact) separation of violated κ-path cuts for $\kappa \geqq 3$ (where the objective functions strive to minimize the number of vehicles). Both these problems are $\mathcal{NP}$-hard in the strong sense (ib., p. 107).

For vehicle routing problems with heterogeneous fleet as well as for the VRPTT, 2-path cuts can be generalized as follows. For each subset $S \subseteqq RWL_C$, let $\kappa_k(S)$ be (a lower bound on) the minimal

number of tours of vehicle $k \in F$ necessary to collect the complete supply of the customers in S when no other vehicle is used. For example, if there are no time windows, the above-mentioned bound of $\lceil \sum_{i \in S} su_i / q_k \rceil$ can be used. Then, the following inequality is valid:

$$\sum_{\{k \in F : \kappa_k(S) \geqq 2\}} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} x_{hi}^k + 2 \sum_{\{k \in F : \kappa_k(S) = 1\}} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} x_{hi}^k \geqq 2$$

$$\forall\, S \subsetneqq RWL_C. \tag{4.90}$$

The validity of (4.90) follows from the customer covering constraints (4.60a), the vehicle capacity constraints (4.62b), and the load update constraints (4.60g)–(4.60j) and (4.61d)–(4.61h).

It is an open question under which conditions generalized κ-path cuts for $\kappa \geqq 3$ are valid for heterogeneous fleet VRPs. However, the following disjunction is *not* valid:

$$\bigvee_{k \in F} \left(\sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} x_{hi}^k \geqq \kappa_k(S) \right) \tag{4.91}$$

This disjunction (which can be linearized to become a (set of) linear constraint(s), cf. Williams 1999, p. 169 ff.) states that, if $\kappa_{k_1}(S)$ vehicles of class k_1 and $\kappa_{k_2}(S)$ vehicles of class k_2 are needed in the respective homogeneous fleet VRP with customer set S, then in the heterogeneous fleet VRP with both classes of vehicles, $\kappa_{k_1}(S)$ vehicles of class k_1 are needed or $\kappa_{k_2}(S)$ vehicles of class k_2. In general, this is not true, because vehicles of both classes can be used at the same time, reducing the necessary number of vehicles of each class.

Moreover, with $\kappa_{min} := \min_{k \in F}\{\kappa_k(S)\}$ for all $S \subsetneqq RWL_C$, the following generalization is also *not* valid:

$$\sum_{k \in F} \frac{\kappa_{min}}{\kappa_k(S)} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} x_{hi}^k \geqq \kappa_{min} \quad \forall\, S \subsetneqq RWL_C. \tag{4.92}$$

A simple example where both (4.91) and (4.92) are not valid is provided by an instance without time windows and with two vehicles k_1, k_2, $q_{k_1} = 10$, $q_{k_2} = 5$, and a subset $\tilde{S}$ consisting of three customers with a supply of 5, 5 and 1, i.e., with a cumulated supply of 11. Then, $\kappa_{k_1}(S) = 2$, $\kappa_{k_2}(S) = 3$, and $\kappa_{min} = 2$ so that, for $\tilde{S}$, (4.91) becomes

$$\sum_{\{(h,i) \in A^{k_1} : loc_h \notin S \ni loc_i\}} x_{hi}^{k_1} \geqq 2 \vee \sum_{\{(h,i) \in A^{k_2} : loc_h \notin S \ni loc_i\}} x_{hi}^{k_2} \geqq 3 \tag{$*$}$$

and (4.92) becomes

$$\frac{2}{2} \sum_{\{(h,i) \in A^{k_1} : loc_h \notin S \ni loc_i\}} x_{hi}^{k_1} + \frac{2}{3} \sum_{\{(h,i) \in A^{k_2} : loc_h \notin S \ni loc_i\}} x_{hi}^{k_2} \geqq 2. \tag{$**$}$$

As far as capacity is concerned, one tour with each vehicle is sufficient to collect the complete supply of the customers in $\tilde{S}$. But setting the flow of each vehicle into $\tilde{S}$ equal to one violates both ($*$) and ($**$).

Implied Bound Cuts:

$$l_i^k + t_i^k - \max_{k \in F}\{q_k\} T^{max} \sum_{(h,i) \in A^k} x_{hi}^k \leqq 0 \quad \forall\, k \in F, i \in V \tag{4.93}$$

The collection and time variables of a vehicle at a vertex must be zero if the vehicle does not visit this vertex. See also (4.63). Implied bound cuts are not valid for the convex hull of (4.60)–(4.62), but they never cut off all optimal solutions.

Arrival Time Cuts:

$$t_o^k - t_d^k \leqq 0 \quad \forall\, k \in F \tag{4.94}$$

The LP relaxation may initially yield a very bad lower bound, indeed, a negative one, because in a fractional solution, constraints (4.61i) allow that $t_d^k < t_o^k$ for any vehicle k. The arrival time cuts prohibit this. In any feasible solution, they must be fulfilled because of constraints (4.61a) (each vehicle reaches the end depot vertex exactly once), the flow conservation constraints (4.61b), and the timing update constraints (4.61i)–(4.61l).

No attempt was made to prove whether one of the above inequalities induces a facet for the VRPTT polytope. Even for much simpler problems, polyhedral studies are extremely difficult. For example, Eglese/Letchford 2000, p. 217, call the polyhedron of the mixed rural postman problem 'bewilderingly complicated', and Naddef/Rinaldi 2002, p. 58, state that trying to prove facets for the symmetric capacitated vehicle routing problem is 'particularly hard', and that the facial structure of the latter problem 'is extremely complex and that there is still a lot to investigate in this field' (ib., p. 71).

The above inequalities were considered as additional static or dynamic cuts in the branch-and-cut algorithm to strengthen the LP relaxation of (4.59)–(4.62). The supply collection cut, the trailer and the lorry flow cut and the implied bound and arrival time cuts were directly added to the formulation (static cuts). The connectivity cuts and the generalized 1-path cuts were dynamically separated in the course of the algorithm. Connectivity cuts were only separated when no violated 1-path cuts were found. As for the κ-path cuts, the computational experiments showed that, with and without them, only very small instances could be solved. Therefore, to find potentially violated κ-path cuts, all customer subsets were enumerated, and each was tested as to whether there was a vehicle class of which at least two vehicles are necessary to collect the supply of the customers in the subset. All relevant cuts were then added to the formulation as static cuts.

4.5.2 Branching and Enumeration Strategies

The following three-stage branching strategy was used:

(i) branch on a lorry arc variable on the arc (o, d)

(ii) branch on a trailer arc variable entering a decoupling or leaving a coupling vertex

(iii) use the default CPLEX branching strategy (automatic selection of branching variable)

As enumeration strategy, the CPLEX default 'best-bound' strategy (select the vertex in the branch-and-bound tree with the best objective function value of the associated LP relaxation) was used.

4.6 Computational Experiments

The proposed branch-and-cut algorithm was implemented in C++ using the Boost Graph library and ILOG Concert Technology.

4.6.1 Test Instances

In practice, the following types of vehicle are used:

- Collection vehicles:
  - Single lorries:
    - single lorry with two axles and a capacity of 10 tons
    - single lorry with three axles and a capacity of 15 tons

  - Lorry-trailer combinations:
    - two-axle lorry with two-axle trailer, both with a capacity of 10 tons ('2/2-combination')
    - two-axle lorry with three-axle trailer, with a capacity of 10 and 15 tons respectively ('2/3-combination')
    - three-axle lorry with two-axle trailer, with a capacity of 15 and 10 tons respectively ('3/2-combination')

- Support vehicles:
  - two-axle lorry with three-axle trailer, with a capacity of zero and 25 tons respectively

The cost data used in the computational experiments are shown in Table 4.10. These data are realistic estimates, approximately reflecting absolute values of the different cost types for each vehicle type as well as ratios of costs of one vehicle type to another. Two- and three-axle single lorries have the same cost data as two- and three-axle lorries of a lorry-trailer combination.

Cost type → Vehicle type ↓	Fixed	Time-dependent	Distance-dependent
Two-axle lorry	18,000	3,600	65
Three-axle trailer	2,500	0	6
Three-axle lorry	20,000	3,600	70
Two-axle trailer	2,000	0	4

Table 4.10: Cost data for computational experiments

The customer and transshipment locations were randomly selected on a 100×100 km grid with the depot located in the centre. The resulting Euclidean distances between each pair of vertices were multiplied by a distance factor of 1.3. The customer supplies were chosen randomly from [1,000; 10,000]. The test instances were created with enough vehicles of each type such that the complete supply could be collected with one type of vehicle. For simplicity, support vehicles were not considered. As load transfer time, 2 minutes per 1,000 units of supply were assumed throughout.

The length of the planning horizon was assumed to be 12 hours or 1,320 minutes respectively. All customers and pure transshipment locations have one time window: [0; 1,320]. For a code capable of solving problems with time windows, such instances represent the worst case. With tighter time windows, larger instances could be solved.

For each $n \in \{1, \ldots, 10\}$, thirty so-called 'x_y_z' instances were created with the vehicle, cost and distance data specified above. x stands for the number of lorry customers, y stands for the number of trailer customers, and z stands for the number of pure transshipment locations, and $x = y = z = n$. This means that an x_y_z instance with $n^{TS} = 3$ has $1 + x + y + 3 \cdot num^{trailers} \cdot (y + z) + 1$ vertices: one for the start depot, x for the lorry customers, y for the trailer customers, three for each combination of trailer and transshipment location (of which there are $y + z$) to represent decoupling, transfer, and coupling, and one for the end depot.

Some further deliberations on n^{TS}, the number of transshipment processes per trailer and transshipment location, follow. With $n^{TS} = 3$, there are no transshipment subtour symmetries, and no precedence constraints for the intermediate vertices to avoid such symmetries are necessary. However, if there are only 2/3-combinations, and only lorry customers with identical supply of 5,001, then even $n^{TS} = 4$ is not sufficient to guarantee that an optimal solution to the problem can be computed with formulation (4.59)–(4.62). With $n^{TS} = 4$, the following sequence of lorry and trailer loads results:

Transshipment operation	1	2	3	4
Trailer load after operation	0	5,001	10,002	10,002
Lorry load after operation	0	0	0	5,001

With $n^{TS} = 5$, it is possible to visit one additional customer:

Transshipment operation	1	2	3	4	5
Trailer load after operation	0	5,001	10,002	15,000	15,000
Lorry load after operation	0	0	0	3	5,004

If there are only 3/2-combinations, a similar argument with lorry customer supplies of 7,501 yields that $n^{TS} = 4$ is sufficient. Hence, overall, with the above fleet data, it is sufficient to have $n^{TS} = 5$ for three-axle trailers and $n^{TS} = 4$ for two-axle trailers to guarantee that an optimal solution obtained for the formulation (4.59)–(4.62) is also an optimal solution to the underlying real-world problem instance. Nevertheless, the computational experiments were performed throughout with $n^{TS} = 3$ to keep the network sizes limited.

4.6.2 System Parameters

The system parameters used in the computational experiments are shown in the following table. All CPLEX parameters not shown in the table were at their default values. The selection of the algorithm for solving the LP relaxations was left to CPLEX, the maximum flow problems for the separation of the subtour elimination constraints were solved with the Edmonds-Karp algorithm provided by the BGL.

Parameter	Setting
CPU frequency	2 GHz
Main memory	1 GB
CPLEX version	9.1
Concert Technology version	2.1
Wall-clock time limit	7,500 seconds
Minimal violation of dynamic cuts to be considered violated	10^{-3}
`IloCplex::EpAGap`	10^{-4}
`IloCplex::EpInt`	10^{-4}
`IloCplex::CutLo`	-10^{75}
`IloCplex::EpGap`	10^{-4}
`IloCplex::ObjDif`	0
`IloCplex::RelObjDif`	0
`IloCplex::CutUp`	10^{75}

Table 4.11: System parameters VRPTT

4.6.3 Computational Results

The following tables show the computational results obtained for the core formulation (4.59)–(4.62).

Table 4.12 shows how the size of the resulting mathematical programs develops for VRPTT instances with $n^{TS} = 3$ where all locations have one time window, and for VRPTW instances with the same number of customers. To compute the data in the table, it was assumed that the VRPTW network consists of one vertex per customer, one start, and one end depot vertex, one arc from the start depot vertex to all other vertices, and one arc from each customer vertex to each other customer vertex and the end depot vertex. The number of vehicles in the VRPTW was assumed to be equal to the number of

lorries plus the number of trailers in the corresponding VRPTT. The number of constraints given in the table was computed assuming an arc variable formulation with constraints for customer covering, flow conservation, update of resource variables and maintenance of static resource windows.

Instance type	1_1_1	2_2_2	2_2_2	3_3_3	4_4_4
Vehicles	2 lorries 2 trailers	2 lorries 2 trailers	4 lorries 4 trailers	4 lorries 4 trailers	6 lorries 6 trailers
VRPTT					
Vertices	16	30	54	80	154
Arcs	131	493	1,477	3,283	11,737
Arc variables	234	876	4,816	10,736	56,412
Resource variables	92	168	496	728	1,920
Constraints	910	3,378	18,989	42,250	223,504
VRPTW					
Vertices	4	6	6	8	10
Arcs	7	21	21	43	73
Arc variables	28	84	168	344	876
Resource variables	32	48	96	128	240
Constraints	66	140	276	486	1,136

Table 4.12: VRPTT network size growth

Taking into account the number of variables and constraints as a measure for instance size, the 2_2_2 and 3_3_3 instances, i.e., even instances with only 4 customers and 4 transshipment locations, can be considered large.

Table 4.13 shows the usefulness of the cuts described in Section 4.5.1. The table indicates the gap at the root vertex of the branch-and-bound tree, i.e., the percentage by which the lower bound at the root vertex of the branch-and-bound tree (RLB) is below the best feasible solution as obtained with the branch-and-cut algorithm (BFS) for the different cut types: $(BFS - RLB)/BFS \cdot 100$. An entry of the form x / y / z in a row for cut c means that the RLB using only cut c was at least x % of BFS below BFS, was y % of BFS below BFS on average, and was at most z % of BFS below BFS. A value of 100 corresponds to an RLB of zero, and a value greater than 100 corresponds to a negative RLB (which is possible, see page 108). The trailer flow cut was not tested, because the test instances were created such that there is no customer with a supply exceeding the capacity of the largest lorry, and the number of lorries was always large enough to collect the complete supply of the customers without using a trailer, so that the number of necessary trailers is zero. Implied bound cuts were not tested, because CPLEX automatically generates cuts where binary variables imply bounds on continuous variables. As indicated in the previous section, the CPLEX cut generation routine was not turned off, so CPLEX automatically separated several types of general (as opposed to problem-specific) cut.

Instance type	1_1_1	2_2_2	3_3_3	Average
Without time-dependent costs				
No cuts	54 / 72 / 93	63 / 79 / 91	67 / 79 / 88	54 / 76 / 93
Only supply collection cut	54 / 72 / 93	63 / 79 / 91	67 / 79 / 88	54 / 76 / 93
Only lorry flow cut	8 / 21 / 41	25 / 39 / 60	30 / 33 / 38	8 / 31 / 60
Only 2-path cuts	27 / 58 / 86	48 / 65 / 79	58 / 67 / 82	27 / 62 / 86
Only arrival time cuts	54 / 72 / 93	63 / 79 / 91	67 / 79 / 88	54 / 76 / 93
Only 1-path cuts	0 / 8 / 20	9 / 24 / 50	11 / 17 / 21	0 / 16 / 50
Only connectivity cuts	32 / 44 / 59	38 / 55 / 72	41 / 49 / 52	32 / 49 / 72
All cuts	0 / 2 / 13	0 / 8 / 18	10 / 12 / 15	0 / 5 / 18

(continued on next page)

(continued from previous page)

Instance type	1_1_1	2_2_2	3_3_3	Average
With time-dependent costs				
No cuts	133 / 164 / 196	82 / 129 / 174	134 / 139 / 148	82 / 146 / 196
Only supply collection cut	133 / 164 / 196	81 / 129 / 174	134 / 139 / 148	81 / 146 / 196
Only lorry flow cut	133 / 164 / 196	66 / 126 / 173	134 / 139 / 148	66 / 145 / 196
Only 2-path cuts	124 / 160 / 193	70 / 120 / 171	122 / 130 / 143	70 / 140 / 193
Only arrival time cuts	68 / 80 / 95	77 / 86 / 94	79 / 86 / 92	68 / 83 / 95
Only 1-path cuts	126 / 161 / 196	63 / 119 / 161	122 / 129 / 136	63 / 140 / 196
Only connectivity cuts	127 / 161 / 193	76 / 122 / 166	124 / 131 / 139	76 / 141 / 193
All cuts	20 / 28 / 41	27 / 38 / 64	38 / 43 / 45	20 / 33 / 64

Table 4.13: Usefulness of cuts in VRPTT branch-and-cut algorithm (min. / avg. / max.)

The following observations can be made in Table 4.13:

- The application of all types of cut together significantly decreases the gap for both types of instance: For the instances without time-dependent costs, the average gap decreases from 76 % to 5 %, for the instances with time-dependent costs, the average gap decreases from 146 % to 33 %.
- The application of all types of cut together clearly shows a synergy effect insofar as the gap is reduced considerably more than by the best type alone.
- The average gap for the instances without time-dependent costs is much smaller than that for the instances with time-dependent costs.
- The cuts are much more useful for the instances without time-dependent costs, although the underlying polytope is the same as that for the instances with time-dependent costs. This suggests that, for the latter instances, the used inequalities cut off irrelevant parts of the feasible region of the LP relaxation, or rather that the objective function decreases in the 'wrong' direction.
- The supply collection cut is not useful.
- The arrival time cuts are not useful for the instances without time-dependent costs, but for the instances with time-dependent costs, they are the strongest type of cut.
- Each other type of cut is useful for the instances without time-dependent costs, but not for the instances with time-dependent costs unless all types of cut are applied together.

Table 4.14 shows the computational results obtained with the branch-and-cut algorithm. The meaning of the row '% Gap at end' is the same as in Chapter 3: It indicates the percentage by which the best feasible solution (BFS) exceeds the best lower bound at the end of the optimization (BLB) (it is zero if an optimal solution is found, and it is not counted if no feasible solution is found): $(BFS-BLB)/BLB \cdot 100$.

Instance type	1_1_1	2_2_2	3_3_3	Average
Without time-dependent costs				
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 28	30 / 7 / 6	30 / 22 / 21
Running time [s]	0 / 1 / 3	18 / 1,141 / 7,500	473 / 3,211 / 7,500	0 / 847 / 7,500
% Gap at end	0 / 0 / 0	0 / 1 / 18	0 / 0 / 1	0 / 0 / 18
No. of B&B vertices	1 / 3 / 11	10 / 222 / 846	124 / 1,141 / 3,085	1 / 220 / 3,085
No. of separated 1-path cuts	1 / 11 / 33	102 / 270 / 837	432 / 882 / 1,462	1 / 218 / 1,462
No. of separated conn. cuts	0 / 15 / 78	59 / 514 / 1,974	201 / 1,862 / 3,076	0 / 431 / 3,076
No. of lorry tours	1 / 1.0 / 1	1 / 1.3 / 2	1 / 1.0 / 1	1 / 1.1 / 2
No. of trailer tours	0 / 0.3 / 1	1 / 1.0 / 2	1 / 1.0 / 1	0 / 0.7 / 2

(continued on next page)

(continued from previous page)

Instance type	1_1_1	2_2_2	3_3_3	Average
With time-dependent costs				
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 0	5 / 4 / 0	22 / 21 / 10
Running time [s]	1 / 4 / 11	7,500 / 7,500 / 7,500	7,500 / 7,500 / 7,500	1 / 3,931 / 7,500
% Gap at end	0 / 0 / 0	0 / 43 / 178	45 / 58 / 62	0 / 23 / 178
No. of B&B vertices	28 / 155 / 442	172 / 6,359 / 12,230	722 / 1,445 / 1,812	28 / 3,093 / 12,230
No. of separated 1-path cuts	19 / 47 / 84	514 / 2,261 / 3,975	1,535 / 2,367 / 2,931	19 / 1,214 / 3,975
No. of separated conn. cuts	41 / 92 / 178	661 / 1,975 / 4,036	2,695 / 4,050 / 5,411	41 / 1,210 / 5,411
No. of lorry tours	1 / 1.0 / 1	1 / 1.4 / 4	1 / 1.0 / 1	1 / 1.2 / 4
No. of trailer tours	0 / 0.3 / 1	0 / 1.0 / 2	1 / 1.0 / 1	0 / 0.7 / 2

Table 4.14: Computational results for VRPTT (min. / avg. / max.)

Three 4_4_4 instances without time-dependent costs were also tried, but for none of them could a feasible solution be computed within the time limit.

The following observations can be made in Table 4.14:

- Only instances with a very small number of customers and transshipment locations can be solved. This is mainly due to the fact that the size of the resulting mathematical programs grows sharply with increasing number of (trailer) customers, transshipment locations, and vehicles. See Table 4.12.
- The difficulty of the instances, or rather, the difficulty that the algorithm has with different instances of the same type and size (as indicated by the running times), varies widely.
- The number of vertices in the branch-and-bound trees is quite high.
- It is remarkable that instances with time-dependent costs are so much harder to solve than instances without.

The largest instances that can consistently be solved to optimality are 2_2_2 instances (4 customers, 4 transshipment locations) with 8 vehicles, 4,800 binary variables, and 19,000 constraints, and 3_3_3 instances (6 customers, 6 transshipment locations) with 4 vehicles, 1,900 binary variables, and 7,400 constraints. Tests have shown that a direct solution of such instances by submitting only the formulation (4.59)–(4.62) to CPLEX is impossible. For example, the instance 2_2_2_0 (4 vehicles, 870 binary variables, 3,300 constraints) without time costs took more than 15,000 seconds to solve and had 244,000 vertices in the branch-and-bound tree with CPLEX alone; with the branch-and-cut algorithm, it took 93 seconds to solve and had 70 vertices in the branch-and-bound tree. The instance 2_2_2_1 (8 vehicles, 4,800 binary variables, 19,000 constraints) without time costs had a gap of more than 80 % after more than 15,000 seconds with CPLEX alone; with the branch-and-cut algorithm, it was solved in less than 1,500 seconds.

4.7 Conclusions

This chapter has considered the vehicle routing problem with trailers and transshipments. In this problem, the vehicle fleet consists of autonomous vehicles (lorries) able to move on their own, and of non-autonomous vehicles (trailers) which must be accompanied by an autonomous vehicle to be able to move. Moreover, both autonomous and non-autonomous vehicles are either collection or support vehicles. The decisive aspects of the problem are the consideration of trailers which can be pulled by different lorries, of support vehicles unable to visit customers, and of load transfers between arbitrary vehicles. These aspects lead to very complex interdependencies between the vehicles. Considering the complexity of the complete problem, a ‘core’ problem has been extracted capturing only central aspects of a free lorry-trailer assignment.

It has been pointed out that the concepts of autonomous/non-autonomous and collection/support vehicles allow considering real-world objects other than lorries and trailers, so that potential applications of the VRPTT go beyond this straightforward interpretation and include, for example, location-routing problems.

Three MIP formulations for the VRPTT have been presented, based on different types of variable, namely, turn, arc, and path variables. Computational experiments with a branch-and-cut algorithm for the arc variable formulation of the core model have been performed and analyzed. In addition, the difficulties arising when trying to solve the problem by branch-and-price have been discussed.

The problem is by no means artificial or contrived. All of the considered constraints appear in the real-world applications that motivated the research on this problem. When starting the work on the VRPTT, it was not foreseeable that the problem would be *that* hard, but the computational results have shown that even with the formulation for the core problem, only instances with very few customers and transshipment locations can be solved exactly. An unexpected difficulty is that the size of the resulting mathematical programs grows extremely fast with increasing numbers of trailer customers, pure transshipment locations, and necessary vehicles. Hence, the main conclusion to be drawn at the end of this chapter is that the difficulty of routing problems is dramatically increased by the consideration of autonomous and non-autonomous objects.

Chapter 5

The Truck-and-Trailer Routing Problem

The truck-and-trailer routing problem (TTRP) is a special case of the VRPTT, where there is a fixed assignment of a lorry to a trailer (and vice versa), i.e., each trailer may be pulled by only one lorry, and only this lorry may perform a load transfer into this trailer. Consequently, there are no support vehicles.

The chapter is structured as follows. In the next section, the differences between the TTRP and the core VRPTT are pointed out, and two new MIP formulations for the TTRP, based on arc and path variables respectively, are developed. In Sections 5.2 and 5.3, a branch-and-cut and a branch-and-price algorithm for the TTRP are presented. The results of computational experiments with these two algorithms are presented and analyzed in Section 5.4. The chapter ends with a short conclusion.

5.1 New MIP Formulations for the TTRP

5.1.1 Assumptions

The TTRP as considered here is a special case of the core VRPTT. The additional assumptions made in this chapter are as follows:

- Due to the fixed lorry-trailer assignment, the vehicles considered in the TTRP are either *single lorries* (as in a 'usual' VRP) or *lorry-trailer combinations* (*LTCs*). As before, the vehicles may differ with respect to (fixed, distance- and time-dependent) costs and capacity, and there may also be accessibility constraints for particular vehicles at some locations. For LTCs, two capacities are relevant: the lorry capacity and the trailer capacity. The sum of these capacities is the overall vehicle capacity of an LTC. An LTC lorry need not use its trailer and may simply leave it at the depot. If a trailer is used, it cannot be left behind at a transshipment location by its lorry, it must also be pulled back to the depot.

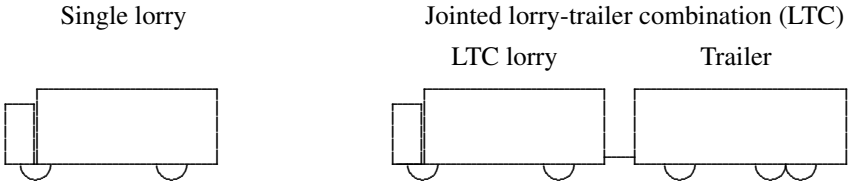


Figure 5.1: TTRP fleet

- All customers and all transshipment locations have only one time window.

Figure 5.2 depicts a possible route plan with one single lorry and one LTC. The single lorry route is evident. The LTC lorry starts at the depot with its trailer, visits a trailer customer, goes on to a transshipment location, decouples the trailer, visits two customers, returns to the transshipment location, re-couples the

trailer, pulls it to a second transshipment location, decouples it again (and perhaps performs a load transfer), visits two more customers, returns to the second transshipment location, re-couples the trailer and returns to the depot. Two transshipment locations in the centre of the figure are not used. The reader may compare this figure with Figure 4.2.

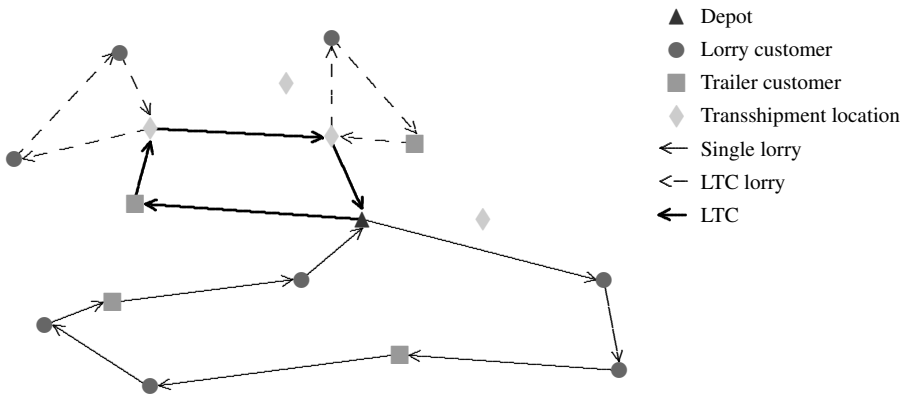


Figure 5.2: Example of a TTRP route plan

5.1.2 Underlying Network

The subsequent formulation is based on a *time-space-operation network* $D = (V, A)$ which is a simplified version of the time-space-operation-vehicle network of Section 4.3.3.2 for the VRPTT. The notation used in this chapter is the same as in the previous one unless otherwise specified. $RWL{:=}\{Depot\} \cup RWL_C \cup RWL_T$ is the set of relevant real-world locations. As before, $RWL_C := RWL_{C_L} \cup RWL_{C_{LT}}$. Each vertex in V corresponds to a location in space, an absolute and/or relative period of time, and a type of operation. For each customer, there is one vertex. The decisive difference between the two networks is that, in the TTRP network, the transshipment vertices are not trailer-specific: The set of transshipment vertices, V_{I_T}, now contains only one subset $\{v_l^{decouple}, v_l^{transfer,1}, \ldots, v_l^{transfer,n^{TS}-2}, v_l^{couple}\}$ for each transshipment location l, i.e., the number of possible transshipment operations is now assumed to be independent of the trailer, respectively, the LTC class, and of the location. This is possible because of the fixed lorry-trailer assignment: When a lorry visits a decoupling, transfer, or coupling vertex, it is clear which trailer is decoupled, to which trailer a load transfer is performed, and which trailer is coupled. If an LTC wants to use transshipment location l, the LTC must first visit $v_l^{decouple}$. The trailer then moves in time to the vertices $v_l^{transfer,1}, \ldots, v_l^{transfer,n^{TS}-2}, v_l^{couple}$, while the LTC lorry visits customers and finally re-couples the trailer at v_l^{couple}. The LTC lorry need not visit any of the transfer vertices of l. The LTC lorry must not visit any other vertex in V_{I_T} before having re-coupled its trailer at v_l^{couple}. The formulation below takes this into account.

All vehicles are initially at the start depot vertex o and end their tour at the end depot vertex d. Single lorries are allowed, in principle, to visit lorry and trailer customer vertices, and LTC lorries may, in principle, visit all vertices of D. Trailers can only reach the transshipment vertices and the trailer customers.

It is assumed that each LTC uses each transshipment location at most once. Thus, for each trailer, there are at most n^{TS} transshipment operations at each transshipment location. The deliberations on the 'correct' choice of n^{TS} in the chapter on the VRPTT apply here, too. However, in the branch-and-price algorithm, the numerical value of n^{TS} need not be fixed in advance; rather, this value is determined during the solution of the pricing problems.

The arc set consists of the following arcs:

- (o, d)
- (o, v_c) and (v_c, d) for all customer vertices $v_c \in V_C$
- $(o, v_l^{decouple})$ for all transshipment locations $l \in RWL_{C_{LT}} \cup RWL_T$
- (v_l^{couple}, d) for all transshipment locations $l \in RWL_{C_{LT}} \cup RWL_T$
- $(v_c, v_{c'})$ for all customer vertices $v_c, v_{c'} \in V_C$ with $c \neq c'$
- $(v_l^{decouple}, v_c)$ for all transshipment locations $l \in RWL_{C_{LT}} \cup RWL_T$ and all customer vertices $v_c \in V_C$
- $(v_l^{transfer,i}, v_c)$ for all transshipment locations $l \in RWL_{C_{LT}} \cup RWL_T$, all $i \in \{1, \ldots, n^{TS} - 2\}$ and all customer vertices $v_c \in V_C$
- (v_l^{couple}, j) for all transshipment locations $l \in RWL_{C_{LT}} \cup RWL_T$ and all vertices $j \in (V_{decouple} \setminus \{v_l^{decouple}\}) \cup V_{C_{LT}}$
- (i, j) for all trailer customer vertices $i \in V_{C_{LT}}$ and all transshipment vertices $j \in V_{I_T}$
- (i, j) for all lorry customer vertices $i \in V_{C_L}$ and all transfer and coupling vertices $j \in V_{transfer} \cup V_{couple}$

Figure 5.3 visualizes the subnetworks for the different vehicle types. As in the corresponding figure for the core VRPTT, there is only one arc for each arc type present in the subnetwork of the respective vehicle type. Tables 5.1, 5.2, and 5.3 also give an overview of the arcs in A.

		0 start depot	1 lorry c.	2 trailer c.	3 decoupling	4 transfer	5 coupling	6 end depot
0	start depot	–	✓	✓	–	–	–	✓
1	lorry c.	–	✓	✓	–	–	–	✓
2	trailer c.	–	✓	✓	–	–	–	✓
3	decoupling	–	–	–	–	–	–	–
4	transfer	–	–	–	–	–	–	–
5	coupling	–	–	–	–	–	–	–
6	end depot	–	–	–	–	–	–	–

Table 5.1: Single lorry arcs in TTRP

		0 start depot	1 lorry c.	2 trailer c.	3 decoupling	4 transfer	5 coupling	6 end depot
0	start depot	–	✓	✓	✓	–	–	✓
1	lorry c.	–	✓	✓	–	✓	✓	✓
2	trailer c.	–	✓	✓	✓	✓	✓	✓
3	decoupling	–	✓	✓	–	–	–	–
4	transfer	–	✓	✓	–	–	–	–
5	coupling	–	–	✓	✓	–	–	✓
6	end depot	–	–	–	–	–	–	–

Table 5.2: LTC lorry arcs in TTRP

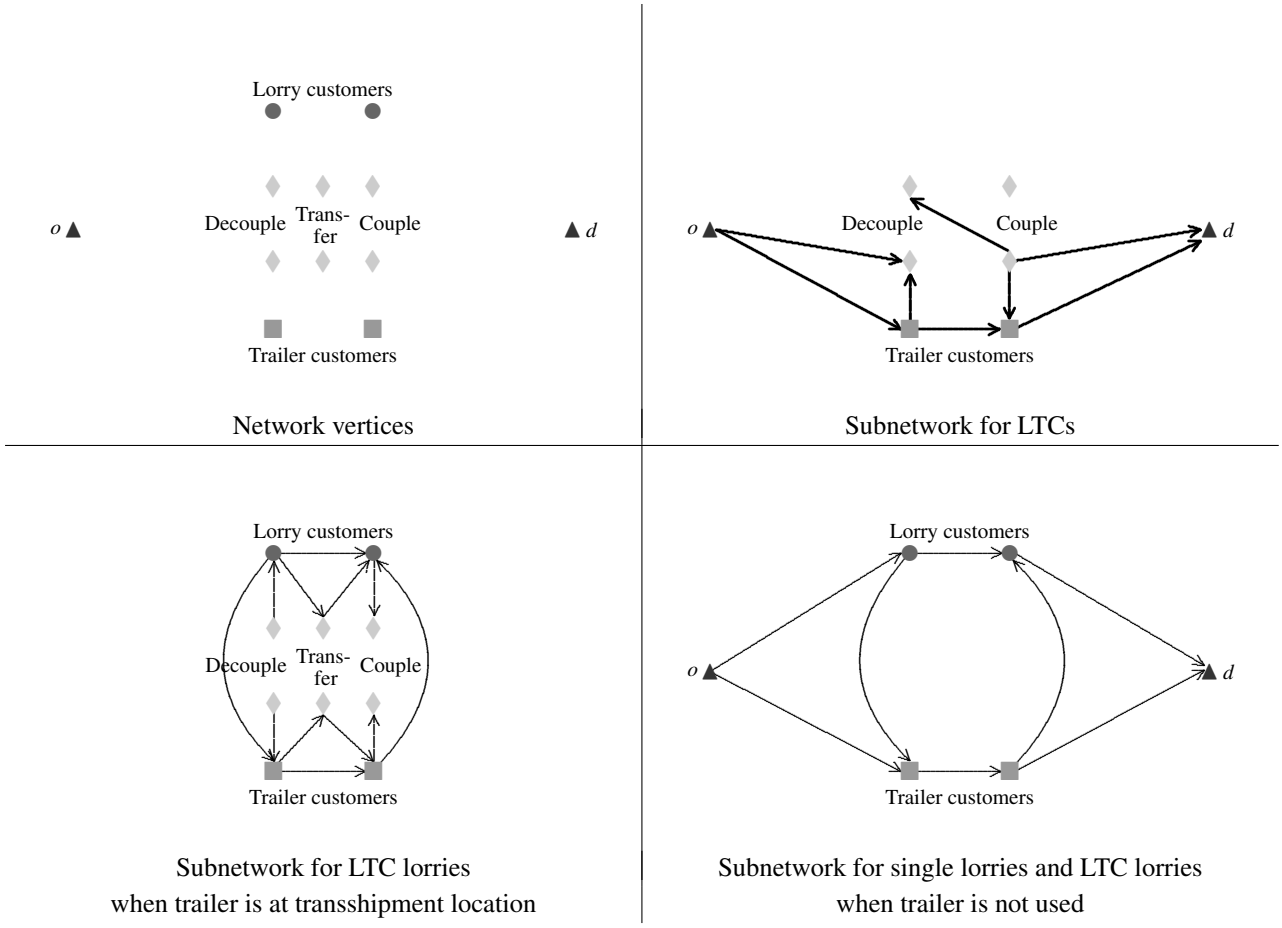


Figure 5.3: TTRP subnetworks

		0 start depot	1 lorry c.	2 trailer c.	3 decoupling	4 transfer	5 coupling	6 end depot
0	start depot	–	–	✓	✓	–	–	✓
1	lorry c.	–	–	–	–	–	–	–
2	trailer c.	–	–	✓	✓	–	–	✓
3	decoupling	–	–	–	–	(✓)	(✓)	–
4	transfer	–	–	–	–	(✓)	(✓)	–
5	coupling	–	–	✓	✓	–	–	✓
6	end depot	–	–	–	–	–	–	–

Table 5.3: Trailer arcs in TTRP

$F := F_L \cup F_{LT}$ denotes the set of vehicles. F_L is the set of single lorries, which do not have a trailer. F_{LT} is the set of lorry-trailer combinations. $A^k_{LT} := A^k \setminus \big(\{(i,j) \in A : i \in V_{C_L} \cup V_{decouple}\} \cup \{(i',j') \in A : j' \in V_{C_L} \cup V_{couple}\}\big)$ is the set of arcs LTC lorry k can traverse with its trailer attached. $c_{ij}^{lorry,k}$ denotes the costs of traversing arc (i,j) with lorry $k \in F$ (for LTC lorries, without the trailer attached), and $c_{ij}^{trailer,k}$ denotes the *additional* costs of pulling LTC lorry k's trailer over arc (i,j). On arcs (o,j) emanating from the start depot o, except for the arc (o,d), the fixed vehicle costs are included in $c_{oj}^{lorry,k}$, respectively, $c_{oj}^{trailer,k}$. For all $k \in F$, q_k^{total} is the total capacity of a vehicle, i.e., it is the capacity of single lorry k, or the capacity of LTC lorry k and its trailer respectively. For all $k \in F_{LT}$, q_k^{lorry} is the capacity of LTC lorry k, and $q_k^{trailer}$ is the capacity of LTC lorry k's trailer.

5.1.3 A Formulation Based on Arc Variables

There are the following variables:

- $x_{ij}^k \in \{0,1\}\ \forall\, k \in F, (i,j) \in A^k$.
  $x_{ij}^k = \begin{cases} 1, \text{lorry } k \text{ traverses arc } (i,j) \\ 0, \text{otherwise} \end{cases}$
- $y_{ij}^k \in \{0,1\}\ \forall\, k \in F_{LT}, (i,j) \in A_{LT}^k$.
  $y_{ij}^k = \begin{cases} 1, \text{trailer } k \text{ traverses arc } (i,j) \\ 0, \text{otherwise} \end{cases}$
- $l_i^{coll,k} \in \mathbb{R}_+^0\ \forall\, k \in F, i \in V^k$.
  The total amount of customer supplies that lorry k has collected when reaching vertex i, *before* k starts its service at i.
- $l_i^{trans,k} \in \mathbb{R}_+^0\ \forall\, k \in F_{LT}, i \in V^k$.
  The total amount of customer supplies that LTC lorry k has transferred to its trailer when reaching vertex i, or, equivalently, the load of LTC lorry k's trailer when k reaches vertex i (with or without its trailer attached), *before* k starts its service at i.
- $t_i^k \in \mathbb{R}_+^0\ \forall\, k \in F, i \in V^k$.
  The point in time when lorry k starts its service at vertex i.

The resulting formulation is:

(TTRP):

$$\sum_{k\in F}\sum_{(i,j)\in A^k} c_{ij}^{lorry,k} x_{ij}^k + \sum_{k\in F_{LT}}\sum_{(i,j)\in A_{LT}^k} c_{ij}^{trailer,k} y_{ij}^k + \sum_{k\in F} c_k^{time}(t_d^k - t_o^k) \to \min \tag{5.1}$$

subject to

$$\sum_{k\in F}\sum_{(h,i)\in A^k} x_{hi}^k = 1 \qquad \forall\, i \in V_C \tag{5.2a}$$

$$\sum_{(h,i)\in A^k} x_{hi}^k \leqq 1 \qquad \forall\, k \in F_{LT}, i \in V_{decouple} \tag{5.3a}$$

$$\sum_{(v_l^{couple},j)\in A_{LT}^k} x_{v_l^{couple}j}^k - \sum_{(h,v_l^{decouple})\in A_{LT}^k} x_{hv_l^{decouple}}^k = 0 \qquad \forall\, k \in F_{LT}, l \in RWL_{C_{LT}} \cup RWL_T \tag{5.3b}$$

$$\sum_{(h,i)\in A^k} x_{hi}^k - \sum_{(v_l^{couple},j)\in A_{LT}^k} x_{v_l^{couple}j}^k \leqq 0 \qquad \forall\, k \in F_{LT}, l \in RWL_{C_{LT}} \cup RWL_T, i \in V_{transfer},$$

$$loc_i = l \tag{5.3c}$$

$$\sum_{(i,d)\in A^k} x_{id}^k = 1 \qquad \forall\, k \in F \tag{5.3d}$$

$$\sum_{(i,d)\in A_{LT}^k} y_{id}^k = 1 \qquad \forall\, k \in F_{LT} \tag{5.3e}$$

$$\sum_{(h,i)\in A^k} x_{hi}^k - \sum_{(i,j)\in A^k} x_{ij}^k = 0 \qquad \forall\, k \in F, i \in V^k \setminus \{o,d\} \tag{5.3f}$$

$$\sum_{(h,i)\in A^k_{LT}} y^k_{hi} - \sum_{(i,j)\in A^k_{LT}} y^k_{ij} = 0 \qquad \forall\, k \in F_{LT}, i \in V^k_{C_{LT}} \tag{5.3g}$$

$$y^k_{ij} \leqq x^k_{ij} \qquad \forall\, k \in F_{LT}, (i,j) \in A^k_{LT} \setminus \{(o,d)\}, i \notin V_{couple}, j \notin V_{decouple} \tag{5.3h}$$

$$x^k_{hi} = y^k_{hi} \qquad \forall\, k \in F_{LT}, i \in V_{decouple}, (h,i) \in A^k_{LT} \tag{5.3i}$$

$$x^k_{ij} = y^k_{ij} \qquad \forall\, k \in F_{LT}, i \in V_{couple}, (i,j) \in A^k_{LT} \tag{5.3j}$$

$$l^{trans,k}_i \leqq l^{coll,k}_i \qquad \forall\, k \in F_{LT}, i \in V^k \tag{5.4a}$$

$$l^{coll,k}_i - l^{trans,k}_i \leqq q^{lorry}_k \qquad \forall\, k \in F_{LT}, i \in V^k \tag{5.4b}$$

$$x^k_{ij} = 1 \Rightarrow l^{coll,k}_i + su_i \leqq l^{coll,k}_j \qquad \forall\, k \in F, (i,j) \in A^k \tag{5.4c}$$

$$x^k_{ij} = 1 \Rightarrow l^{trans,k}_i \leqq l^{trans,k}_j \qquad \forall\, k \in F_{LT}, (i,j) \in A^k \tag{5.4d}$$

$$x^k_{ij} = 1 \Rightarrow l^{trans,k}_i \geqq l^{trans,k}_j \qquad \forall\, k \in F_{LT}, i \in V_{C_L}, (i,j) \in A^k \tag{5.4e}$$

$$x^k_{ij} = 1 \wedge y^k_{ij} = 0 \Rightarrow l^{trans,k}_j \leqq l^{trans,k}_i \qquad \forall\, k \in F_{LT}, i \in V_{C_{LT}}, (i,j) \in A^k \tag{5.4f}$$

$$y^k_{ij} = 1 \Rightarrow l^{trans,k}_j \leqq l^{trans,k}_i + su_i \qquad \forall\, k \in F_{LT}, i \in V_{C_{LT}}, (i,j) \in A^k_{LT} \tag{5.4g}$$

$$x^k_{ij} = 1 \Rightarrow t^k_i + \tau^{tr}_{ij} \leqq t^k_j \qquad \forall\, k \in F, i \in \{o\} \cup V_C, (i,j) \in A^k \tag{5.4h}$$

$$x^k_{ij} = 1 \Rightarrow t^k_i + (l^{trans,k}_j - l^{trans,k}_i)\tau^{lt} + \tau^{tr}_{ij} \leqq t^k_j \qquad \forall\, k \in F_{LT}, i \in V_{I_T}, (i,j) \in A^k \tag{5.4i}$$

$$t^k_{v^{decouple}_l} - t^k_{v^{transfer,1}_l} \leqq 0 \qquad \forall\, k \in F_{LT}, l \in RWL_{C_{LT}} \cup RWL_T \tag{5.4j}$$

$$t^k_{v^{transfer,i}_l} - t^k_{v^{transfer,i+1}_l} \leqq 0 \qquad \forall\, k \in F_{LT}, i \in \{1,\ldots,n^{TS}-3\}, l \in RWL_{C_{LT}} \cup RWL_T \tag{5.4k}$$

$$t^k_{v^{transfer,n^{TS}-2}_l} - t^k_{v^{couple}_l} \leqq 0 \qquad \forall\, k \in F_{LT}, l \in RWL_{C_{LT}} \cup RWL_T \tag{5.4l}$$

$$t^k_{v^{couple}_l} - T \sum_{(h,v^{decouple}_l)\in A^k_{LT}} x^k_{hv^{decouple}_l} \leqq 0 \qquad \forall\, k \in F_{LT}, l \in RWL_{C_{LT}} \cup RWL_T \tag{5.4m}$$

$$x^k_{ij} \in \{0,1\} \qquad \forall\, k \in F, (i,j) \in A^k \tag{5.5a}$$

$$y^k_{ij} \in \{0,1\} \qquad \forall\, k \in F_{LT}, (i,j) \in A^k_{LT} \tag{5.5b}$$

$$0 \leqq l^{coll,k}_i \leqq q^{total}_k \qquad \forall\, k \in F, i \in V^k \tag{5.5c}$$

$$0 \leqq l^{trans,k}_i \leqq q^{trailer}_k \qquad \forall\, k \in F_{LT}, i \in V^k \tag{5.5d}$$

$$twa_i \leqq t^k_i \leqq twb_i \qquad \forall\, k \in F, i \in V^k \tag{5.5e}$$

(5.1) is the objective function, which minimizes total costs. Constraints (5.2) are logical coupling constraints, (5.3) are non-coupling flow constraints, (5.4) are non-coupling constraints specifying the update of the resource variables, and (5.5) determine the ranges of the variables.

(5.2a) are the customer covering constraints. Each customer must be visited exactly once.

(5.3a)–(5.3c) make sure that each transshipment location is used at most once by each $k \in F_{LT}$, and that only intermediate and coupling vertices of those transshipment locations are used whose decoupling vertex is visited.

(5.3d) and (5.3e) require that each vehicle reach the end depot (perhaps via the arc (o, d)).

(5.3f) and (5.3g) are flow conservation constraints.

(5.3h)–(5.3j) are the constraints linking the routing of the trailer to that of its lorry. Note that it is possible that an LTC lorry does not use its trailer, in which case the latter moves directly from o to d.

(5.4a) requires that the load of the trailer be at most equal to the total amount of supply collected. Without this constraint, $l_i^{trans,k}$ can always be set to the trailer capacity. Because of constraint (5.4b), $l_i^{coll,k}$ is then bounded from above by the overall vehicle capacity, and not by the lorry capacity and the amount of load actually transferred. This effectively enlarges the lorry capacity. A lorry can then park its trailer before doing any collecting and can collect as much supply as the overall vehicle capacity permits, without having to transfer any load.

(5.4c) are the load update constraints. The total amount of load collected increases (at least) by the supply of each vertex visited.

Similarly, constraints (5.4d)–(5.4g) are for the update of the transshipment variables, respectively, the trailer load variables. (5.4d) states that the trailer load is non-decreasing. (5.4e) states that the trailer load is non-increasing at lorry customer vertices. (5.4f) states that at trailer customer vertices, the trailer load of an LTC lorry k can only increase if the vertex is visited by k *and* its trailer. (5.4g) states that, at a trailer customer vertex, the trailer load does not increase by more than the customer's supply.

Constraints (5.4h)–(5.4i) are the constraints for the update of the timing variables.

(5.4j)–(5.4m) are needed to make sure that the vertices of a transshipment location are visited in the correct order, i.e., that decoupling vertices of a transshipment location are visited *before* their corresponding coupling vertex. Without this constraint, it would be possible to visit decoupling vertex $v_{l_1}^{decouple}$, go to coupling vertex $v_{l_2}^{couple}$, then to decoupling vertex $v_{l_2}^{decouple}$ and then to $v_{l_1}^{couple}$. Moreover, these constraints avoid transshipment subtour symmetries (cf. p. 85) if there is more than one transfer vertex per location, i.e., if $n^{TS} \geqq 4$. If an LTC lorry does not visit a transshipment intermediate vertex for some l, the time variables for these vertices can be fixed appropriately; it is not required to have a visiting time of zero for non-visited vertices. In principle, it is also possible to set $n^{TS} = 2$. Then, (5.4j)–(5.4l) must be replaced by

$$t^k_{v_l^{decouple}} - t^k_{v_l^{couple}} \leqq 0 \qquad \forall\, k \in F_{LT}, l \in RWL_T \cup RWL_{C_{LT}}. \tag{5.6}$$

(5.4m) serves the same purpose as (5.3b) and (5.3c). Hence, when (5.3b) and (5.3c) are used, (5.4m) is redundant, and vice versa.

Note that there are no constraints limiting the total tour duration other than the length of the planning horizon.

An alternative choice of variables is to use a (binary) resource variable y_i^k indicating whether LTC lorry k reaches vertex i with or without its trailer attached. This leads to fewer variables and to fewer constraints (5.4g), but to more and more complicated constraints (5.3g).

Instead of the two resource variables concerning the load, it is also possible to consider variables specifying the current load of the lorry and, if applicable, its trailer. In this case, constraints (5.4a) are not necessary, but constraints (5.4b) have to be replaced by

$$l_j^{trailer,k} \leqq l_i^{trailer,k} + l_i^{lorry,k} + su_i \qquad \forall\, k \in F_{LT}, (i,j) \in A^k, \tag{5.7}$$

i.e., there are more constraints than with the above choice of variables (unless time window or capacity restrictions are extremely tight).

The above formulation differs from the TTRP as described in Chao 2002 and formulated in Scheuerer 2004 in the following aspects:

- The above papers consider no variable costs for the trailers, no fixed costs for the vehicles, and no time-dependent costs at all (Chao 2002, p. 34, Scheuerer 2004, p. 43).
- The above papers consider only the trailer customer locations and the depot as possible transshipment locations (Chao 2002, p. 34, Scheuerer 2004, p. 45). In the network used in this chapter, it is also possible to have transshipment vertices corresponding to other real-world locations. The depot is not explicitly considered as a transshipment location, but it can trivially be added to the set of possible transshipment locations. However, this does not really make sense. At the depot, no load transfer from lorry to trailer is performed, but any load a vehicle carries will be unloaded, and any vehicle leaving the depot will be empty. Hence, the load of a lorry and its trailer must be reset to zero after a 'transshipment' operation at the depot. This means that having the depot as a possible transshipment location amounts to allowing multiple use of vehicles, but this is not considered for simplicity.
- Scheuerer 2004 (p. 45) does not allow that several lorry-trailer combinations use the same transshipment location.
- The above papers do not consider time windows.
- The above papers assume that a transshipment operation takes a fixed amount of time, independent of the amount of load transferred (Scheuerer 2004, p. 51).

The only existing formulations for the TTRP, by Semet 1995 and Scheuerer 2004, are conceptually different from the above formulation. Scheuerer 2004 uses binary three-index arc variables similar to the above y_{ij}^k variables, and five-index variables indicating whether a certain lorry traverses a certain arc on the nth subtour starting at a certain trailer customer or at the depot, where n is the number of customers. (In the worst case, as many single lorry (sub-)tours starting at a trailer customer or at the depot are necessary as there are customers.) Semet 1995 uses similar variables, but requires that at most one subtour originate at each trailer customer, so that the second variable type has only four indices in his formulation. Neither author uses resource variables, but rather, both formulations are pure 0-1 IPs.

A weakness of (5.1)–(5.5) compared to the formulations presented by Semet 1995 and Scheuerer 2004 lies in the fact that, in order to linearize the implications in several constraints of (5.1)–(5.5), large constants M must be introduced, which makes for a weak LP relaxation of (5.1)–(5.5). The formulations by Semet 1995 and Scheuerer 2004 do not use such constants. A potential advantage of (5.1)–(5.5) compared to the formulation by Scheuerer 2004 lies in constraints (5.4j)–(5.4l), as Scheuerer 2004 does not include any constraints for breaking transshipment subtour symmetries.

5.1.4 A Formulation Based on Path Variables

This section presents a path variable reformulation of (5.1)–(5.5). For simplicity, and to avoid difficulties in the solution of the pricing problems as for the VRPTT, the following simplifications are made:

(i) The time-dependent costs in the objective function are disregarded.

(ii) The load transfer times at transshipment vertices are assumed to be fixed, independent of the amount of load transferred.

If fixed load transfer times for the TTRP are assumed, it is possible to consider time windows, a maximal tour duration and time costs in a labelling algorithm with the standard resource concept as described in Chapter 2 for the solution of the pricing problems. (If time-dependent costs and fixed load transfer times are considered, but no time windows, the time-dependent costs can be put entirely on the arcs, together with the distance-dependent costs, as no waiting occurs. In the presence of time windows, however, waiting may occur, and if there are time-dependent costs, the total waiting time must be minimized. This could be done as described in Desaulniers/Villeneuve 2000, cf. p. 100.) However, the TTRP, as considered in Chao 2002 and Scheuerer 2004, considers neither time costs nor time windows, and almost all papers on VRPs consider no time-dependent costs.

Fixed load transfer times are an alternative well worth considering. A little solution precision gets lost, but it is to be expected that larger instances can be solved. If this simplification is made, it is arguable that it also makes sense not to consider the trailer routing costs, because they are quite low compared to the lorry costs (in reality and in the instances used in the computational experiments). However, the consideration of the exact trailer costs does not make the solution any more difficult, so it is better to consider this aspect of the problem correctly. To consider fixed load transfer times in (5.1)–(5.5), constraints (5.4h) and (5.4i) are replaced by

$$x_{ij}^k = 1 \Rightarrow t_i^k + \tau_{ij}^{tr} \leqq t_j^k \qquad \forall\, k \in F, (i,j) \in A^k \tag{5.8}$$

The usual decomposition approach for VRPs can then also be applied to the TTRP. The logical coupling constraints, i.e., the customer covering constraints (5.2), define the master program, the non-coupling constraints (5.3)–(5.5) define the sub- or pricing problems.

5.1.4.1 The Master Program

As for VRPs without trailers and for the VRPTT, each feasible tour of a single lorry in the TTRP is an elementary path from the start depot vertex to the end depot vertex through the respective subnetwork. This is also true for the LTC lorries. However, the movements of a trailer in the network D of Section 5.1.3 do not constitute a path: There are no trailer flow variables leaving decoupling vertices or entering coupling vertices. This, though, is only to avoid unnecessary variables and constraints. In the real world, the itinerary of a single lorry or a trailer resulting from a solution to (5.1)–(5.5) is an elementary path, too, whereas the itinerary of an LTC lorry using its trailer is not necessarily elementary; it may contain cycles starting and ending at the transshipment locations where the trailer is parked.

It follows from the flow decomposition theorem (cf. Ahuja et al. 1993, p. 80 f.) that the extreme points of the convex hull of all points fulfilling the pricing problems' constraints (5.3)–(5.5) correspond to lorry paths from o to d in D, some of which may be non-elementary, because D is not acyclic. For LTCs, constraints (5.3d)–(5.3j) additionally imply that the extreme points represent a path-like structure consisting of an o-d-path for the lorry and one or more paths for its trailer. The union of these trailer paths is a subset of the lorry path.

These extreme points are described by flow and resource vectors

$$(x_p^k, y_p^k, l_p^{coll,k}, l_p^{trans,k}, t_p^k) = (x_{ijp}^k, y_{ijp}^k, l_{ip}^{coll,k}, l_{ip}^{trans,k}, t_{ip}^k)$$
$$\forall\, k \in F,\, p \in P^k,\, (i,j) \in A^k, \tag{5.9}$$

where P^k is the set of extreme points for single lorry or LTC k. For simplicity of notation, in (5.9) and in the following, it is assumed that

$$y_{ij}^k = y_{ijp}^k = y_p^k = c_{ij}^{trailer,k} := 0 \quad \forall\, k \in F_L,$$

and

$$l_i^{trans,k} = l_{ip}^{trans,k} = l_p^{trans,k} := 0 \quad \forall\, k \in F_L.$$

Any solution satisfying (5.3)–(5.5) can be expressed as a convex combination of these extreme points:

$$x_{ij}^k = \sum_{p \in P^k} x_{ijp}^k \lambda_p^k \quad \forall\, k \in F, (i,j) \in A^k \tag{5.9a}$$

$$x_{ij}^k \in \{0,1\} \quad \forall\, k \in F, (i,j) \in A^k \tag{5.9b}$$

$$y_{ij}^k = \sum_{p \in P^k} y_{ijp}^k \lambda_p^k \quad \forall\, k \in F_{LT}, (i,j) \in A_{LT}^k \tag{5.9c}$$

$$y_{ij}^k \in \{0,1\} \quad \forall\, k \in F_{LT}, (i,j) \in A_{LT}^k \tag{5.9d}$$

$$l_i^{coll,k} = \sum_{p \in P^k} l_{ip}^{coll,k} \lambda_p^k \quad \forall\, k \in F, i \in V^k \tag{5.9e}$$

$$l_i^{trans,k} = \sum_{p \in P^k} l_{ip}^{trans,k} \lambda_p^k \quad \forall\, k \in F_{LT}, i \in V^k \tag{5.9f}$$

$$t_i^k = \sum_{p \in P^k} t_{ip}^k \lambda_p^k \quad \forall\, k \in F, i \in V^k \tag{5.9g}$$

$$\sum_{p \in P^k} \lambda_p^k \leqq 1 \quad \forall\, k \in F \tag{5.9h}$$

$$\lambda_p^k \geqq 0 \quad \forall\, k \in F, p \in P^k \tag{5.9i}$$

The integer master program (IMP) is then:

$$\sum_{k \in F} \sum_{p \in P^k} \left(\sum_{(i,j) \in A^k} c_{ij}^{lorry,k} x_{ijp}^k \right) \lambda_p^k + \sum_{k \in F_{LT}} \sum_{p \in P^k} \left(\sum_{(i,j) \in A_{LT}^k} c_{ij}^{trailer,k} y_{ijp}^k \right) \lambda_p^k \to \min \tag{5.10}$$

subject to

$$\sum_{k \in F_L} \sum_{p \in P^k} \left(\sum_{i \in V_C^l} \sum_{(h,i) \in A^k} x_{hip}^k \right) \lambda_p^k = 1 \qquad \forall\, l \in RWL_C \tag{5.11a}$$

$$\sum_{p \in P^k} \lambda_p^k \leqq 1 \qquad \forall\, k \in F \tag{5.11b}$$

$$\lambda_p^k \geqq 0 \qquad \forall\, k \in F, p \in P^k \tag{5.11c}$$

$$\lambda_p^k \in \{0,1\} \qquad \forall\, k \in F, p \in P^k \tag{5.11d}$$

In general, it is not true that binary restrictions on the original x_{ij}^k arc variables can be replaced by binary restrictions on the λ_p^k path variables. However, this statement holds for the above TTRP reformulation, since the definition of the original master problem solution space, i.e., the constraint set (5.2), involves only the x_{ij}^k variables. Hence, binary requirements on the path variables are equivalent to binary requirements on the x_{ij}^k variables, cf. Desaulniers et al. 1998, p. 75. Moreover, if the x_{ij}^k variables are binary, the y_{ij}^k variables will be binary, too. This is because, for any path of an LTC lorry k represented by the values of the x_{ij}^k variables, the values of the pertinent y_{ij}^k variables are unequivocally determined: If the path visits transshipment vertices, all partial paths ending at a decoupling vertex or starting at a coupling vertex have all y_{ij}^k variables equal to one, and all partial paths between a decoupling and its corresponding coupling vertex have all y_{ij}^k variables equal to zero. Paths visiting only customers and at least one lorry customer have all y_{ij}^k variables equal to zero, and paths visiting only trailer customers have all y_{ij}^k variables equal to one, unless the total demand of the customers on the path does not exceed the lorry capacity, in which case the corresponding trailer will not be used.

5.1.4.2 The Pricing Problems

The pricing problems' constraints are (5.3)–(5.5). The objective functions for single lorries and LTCs can be derived as follows:

Introducing dual variables

$$(\alpha, \gamma) := (\alpha^1, \ldots, \alpha^{|RWL_C|}, \gamma^1, \ldots, \gamma^{|F|}) \tag{5.12}$$

for constraints (5.11a) and (5.11b), the reduced costs of a path p are

$$\tilde{c}_p^k(\alpha, \gamma) = \sum_{(i,j)\in A^k} c_{ij}^{lorry,k} x_{ijp}^k + \sum_{(i,j)\in A_{LT}^k} c_{ij}^{trailer,k} y_{ijp}^k - \sum_{l\in RWL_C} \left(\sum_{i\in V_C^l} \sum_{(h,i)\in A} x_{hip}^k \right) \alpha^l - \gamma^k. \tag{5.13}$$

The reduced costs of the arcs in the pricing problem for a vehicle k can be stated as in Table 5.4.

$\tilde{c}_{ij}^k$	for
$c_{ij}^{lorry,k} - \alpha^l$	$(i,j) \notin A_{LT}^k, j \in V_C^k, loc_j = l$
$c_{ij}^{lorry,k} + c_{ij}^{trailer,k} \delta_{ij}^{trailer,k} - \alpha^l$	$(i,j) \in A_{LT}^k, j \in V_{C_{LT}}^k, loc_j = l$
$c_{ij}^{lorry,k} + c_{ij}^{trailer,k}$	$j \in V_{decouple}$
$c_{ij}^{lorry,k}$	$j \in V_{transfer} \cup V_{couple}$
$c_{id}^{lorry,k} + c_{id}^{trailer,k} - \gamma^k$	$j = d$

Table 5.4: Reduced costs for arcs in the TTRP pricing problem for vehicle k

$\delta_{ij}^{trailer,k}$ fulfils

$$\delta_{ij}^{trailer,k} = \begin{cases} 1, k \in F_{LT} \text{ and } k \text{ traverses } (i,j) \text{ with its trailer attached} \\ 0, \text{otherwise} \end{cases} \quad \forall\, k \in F, (i,j) \in A^k.$$

The branch-and-price algorithm described in Section 5.3 uses a resource to correctly consider the $c_{ij}^{trailer,k}$ in the solution of the LTC pricing problems.

The objective function of the pricing problem for vehicle k can be written as

$$\sum_{(i,j)\in A^k} c_{ij}^{lorry,k} x_{ij}^k + \sum_{(i,j)\in A_{LT}^k} c_{ij}^{trailer,k} y_{ij}^k - \sum_{l\in RWL_C} \sum_{i\in V_C^l} \sum_{(h,i)\in A^k} \alpha^l x_{hi}^k - \gamma^k. \tag{5.14}$$

5.1.4.3 Identical Pricing Problems

The network D underlying formulation (5.1)–(5.5) contains an arc (o, d) that can be traversed by a trailer without being pulled by its lorry to model the fact that an LTC lorry may also be used without its trailer. This must be taken into account by solving a pricing problem ESPPRC for each LTC, for each single lorry, and for each LTC lorry for which there is no single lorry with identical properties. However, if several vehicles are identical, the resulting symmetry in the problem can be exploited. For simplicity, it is assumed that there is only one type of LTC, and no single lorries. Otherwise, the following steps can be performed for each subset of identical vehicles. When all vehicles are identical, let P be the common set of extreme points, let c_{ij}^{lorry} and $c_{ij}^{trailer}$ be the objective function coefficients, let x_{ij} and y_{ij} be the common integer routing variables on arc (i, j), let x_{ijp} and y_{ijp} be the common binary routing variables on arc (i, j) for path p, and define

$$\lambda_p := \sum_{k \in F} \lambda_p^k \quad \forall\, p \in P. \tag{5.15}$$

The IMP can then be formulated as follows:

$$\sum_{p \in P} \left(\sum_{(i,j) \in A} c_{ij}^{lorry} x_{ijp} \right) \lambda_p + \sum_{p \in P} \left(\sum_{(i,j) \in A_{LT}} c_{ij}^{trailer} y_{ijp} \right) \lambda_p \to \min \tag{5.16}$$

subject to

$$\sum_{p \in P} \left(\sum_{i \in V_C^l} \sum_{(h,i) \in A} x_{hip} \right) \lambda_p = 1 \qquad \forall\, l \in RWL_C \tag{5.17a}$$

$$\sum_{k \in F} \lambda_p^k = \lambda_p \qquad \forall\, p \in P \tag{5.17b}$$

$$\lambda_p \geqq 0 \qquad \forall\, p \in P \tag{5.17c}$$

$$\sum_{p \in P} \lambda_p^k \leqq 1 \qquad \forall\, k \in F \tag{5.17d}$$

$$\lambda_p^k \geqq 0 \qquad \forall\, k \in F,\, p \in P \tag{5.17e}$$

$$\lambda_p^k \in \{0, 1\} \qquad \forall\, k \in F,\, p \in P^k \tag{5.17f}$$

The dual variables in (5.12), the reduced costs for paths (5.13), the objective functions of the pricing problems (5.14), and the reduced costs for arcs given in Table 5.4 then lose their k indices.

Similar to the VRPTT pricing problem(s), the convexity constraint (5.17d) can be omitted. This is equivalent to allowing an unlimited number of vehicles of each type. In this case, the algorithm performs fleet planning and vehicle routing simultaneously. This is a considerable advantage compared to the arc variable formulation (5.1)–(5.5), where several changes must be made to the formulation and the underlying network in order to do the same. In particular, if there are time windows, it is not easy to determine the number of necessary vehicles (of each vehicle class) in advance, but these numbers must be known for fleet planning with the arc variable formulation.

5.2 A Branch-and-Cut Algorithm

5.2.1 Valid Inequalities

The following inequalities were considered as additional cuts in the branch-and-cut algorithm. It is easy to see that all of them are valid for the TTRP polytope, because for each of them, there is a correspond-

ing valid inequality for the VRPTT. Although the TTRP polytope differs from the VRPTT polytope, the arguments presented in the section on valid inequalities for the VRPTT can be applied here analogously. Hence, the inequalities are restated only because of small logical and notational differences.

Supply Collection Cut:

$$\sum_{k \in F} l_d^{coll,k} \geqq \sum_{l \in RWL_C} cs_l \tag{5.18}$$

Trailer Flow Cut:

$$\sum_{k \in F_{LT}} \sum_{\{(i,d) \in A^k : i \neq o\}} y_{id}^k \geqq num^{trailers} \tag{5.19}$$

A lower bound for $num^{trailers}$ can be computed as for the VRPTT trailer flow cut, with $su^{trailer}$ defined as

$$su^{trailer} = \sum_{l \in RWL_C} cs_l - \sum_{k \in F} q_k^{lorry}. \tag{5.20}$$

Lorry Flow Cut:

$$\sum_{k \in F} \sum_{\{(i,d) \in A^k : i \neq o\}} x_{id}^k \geqq num^{lorries} \tag{5.21}$$

The minimal number of necessary lorries, $num^{lorries}$, can be computed by sorting all vehicles by non-increasing total capacity (including trailer capacity) and summing up the total vehicle capacities, starting with a biggest vehicle, until the sum exceeds the total customer supply. For each lorry that contributes to the sum, $num^{lorries}$ must be increased by one.

Connectivity Cuts:

$$\sum_{\{a \in A^k : ta_a \notin S \cup \{i\}, he_a \in S\}} x_a^k - x_{hi}^k \geqq 0 \quad \forall\, k \in F, S \subseteqq V^k \setminus \{d\}, (h,i) \in A^k, h \in S \tag{5.22}$$

Generalized 1-Path Cuts:

$$\sum_{k \in F} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} x_{hi}^k \geqq 1 \quad \forall\, S \subseteqq RWL_C \tag{5.23}$$

Generalized 2-Path Cuts:
For the TTRP, using the information that there is a fixed lorry-trailer assignment, (4.90) can be strengthened to

$$\begin{aligned} &\sum_{k \in F_L} \frac{1}{\tilde{\kappa}_k(S)} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} x_{hi}^k \\ &+ \sum_{k \in F_{LT}} \frac{1}{\tilde{\kappa}_k(S)} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} (x_{hi}^k - y_{hi}^k) \\ &+ \sum_{k \in F_{LT}} \frac{1}{\tilde{\kappa}'_k(S)} \sum_{\{(h,i) \in A^k : loc_h \notin S \ni loc_i\}} y_{hi}^k \geqq 1 \qquad \forall\, S \subseteqq RWL_C, \end{aligned} \tag{5.24}$$

where, for each $k \in F$, $\kappa_k(S)$ is (a lower bound on) the minimal number of lorry tours necessary to collect the complete supply of the customers in S when no other vehicle is used and $\tilde{\kappa}_k(S) := \min\{2, \kappa_k(S)\}$, and where, for each $k \in F_{LT}$, $\kappa'_k(S)$ is (a lower bound on) the minimal number of LTC tours necessary to collect the complete supply of the customers in S if no other vehicle is used, and $\tilde{\kappa}'_k(S) := \min\{2, \kappa'_k(S)\}$.

Implied Bound Cuts:

$$l_i^{coll,k} + t_i^k - \max_{k \in F}\{q_k^{total}\} T \sum_{(h,i) \in A^k} x_{hi}^k \leqq 0 \quad \forall\, k \in F, i \in V \tag{5.25}$$

For transshipment vertices i, the t_i^k summand must be left out because of (5.4k)–(5.5a).

Arrival Time Cuts:

$$t_o^k - t_d^k \leqq 0 \quad \forall\, k \in F \tag{5.26}$$

5.2.2 Branching and Enumeration Strategies

Two branching strategies were tried:

(i) a five-stage strategy going from global to local branching criteria as follows:

- branch on the number of tours
- branch on an arc with fractional total flow, i.e., where the total number of lorries traversing that arc is fractional. If possible, use an arc whose tail or head is a customer vertex, whose flow value is between 0.4 and 0.6, and whose length is maximal. Otherwise, use the first arc with non-integer flow.
- branch on a lorry arc variable on the arc (o, d)
- branch on a trailer arc variable entering a decoupling or leaving a coupling vertex
- branch on a variable close to 0.5 having high absolute objective function coefficient

It did not prove useful to also use the 'close to 0.5 and high objective function coefficient' strategy for the choice of the lorry or trailer flow variable.

(ii) a three-stage strategy using only the last three, the 'local', stages of the five-stage strategy

Preliminary computational experiments showed that the three-stage strategy consistently outperformed the five-stage strategy with respect to running time. The number of subproblems and the number of solved LPs were significantly lower than with the five-stage strategy, but the highest level reached in the branch-and-bound tree was higher with the three-stage strategy. This may be partially caused by the fact that branching on the number of tours is not too useful for arc variable formulations, as there is a strong correlation between minimal total distance covered and minimal number of vehicles used. This is all the more so if fixed vehicle costs are considered, as was the case in the computational experiments performed. Consequently, Section 5.4 shows the computational results obtained with the three-stage strategy.

As enumeration strategy, 'dive and best' was used. This strategy performs depth-first search until a feasible solution is found, and then performs best-first search.

5.3 A Branch-and-Price Algorithm

5.3.1 The Pricing Problems

5.3.1.1 Literature Review

As mentioned in the previous chapter, there is a considerable amount of literature on the solution of (E)SPPRCs as pricing problems in branch-and-price approaches for VRP(TW)s and related problems. A brief review of important contributions follows. All cited papers solve the (E)SPPRCs by a labelling algorithm.

Until fairly recently, most approaches for solving VRP(TW)s by branch-and-price solved only the non-elementary SPPRC in the pricing step (see, for example, Kohl et al. 1999). Due to the negative cycles, non-elementary Pareto-optimal o-d-paths may exist. Columns corresponding to such paths may be useful during the solution process (cf. Desrosiers/Lübbecke 2005, p.12), but these columns can never be part of a feasible solution to the VRP(TW), so they must be eliminated in the branching process.

2-cycles, i.e., negative cycles consisting of two antiparallel arcs, are encountered very often. In labelling algorithms, they can be eliminated with comparatively little computational overhead. The basic idea is to keep at a vertex i only one Pareto-optimal label l_1 and one 'second-best' label l_2 with a different predecessor vertex. If a label l_3 resident at i dominates neither l_1 nor l_2, i.e., if l_3's consumption of each resource is greater than or equal to the corresponding consumption of both l_1 and l_2, l_3 can be discarded.

Irnich/Villeneuve 2006 describe κ-cycle elimination for cycles containing $\kappa \geqq 3$ arcs. (This means that all cycles containing κ *or fewer* arcs are eliminated.) Their approach has been very successfully used (see, e.g., Fukasawa et al. 2006), but it is very involved and cannot be described here.

Feillet et al. 2004 have performed computational experiments with a branch-and-price approach for the VRPTW, where they solved the ESPPRC using visitation counter resources as described in Section 2.3. These experiments show the following: There is a trade-off between solving the SPPRC and the ESPPRC. The former problem is much easier to solve (by several orders of magnitude), but the resulting lower bounds for the master problem are very weak, even when using 2-cycle elimination. The latter problem provides very good lower bounds, but direct solution of the elementary problem can be prohibitively difficult. This is mostly due to the fact that the dominance procedure is very weak when visitation counters are used.

The approaches by Chabrier 2006, Boland et al. 2006, and Salani 2005 try to find a compromise by deferring or even completely avoiding the use of all visitation counters while still solving the elementary version of the problem. To this end, the problem is solved iteratively. Chabrier 2006 solves the ESPPRC by not extending a path to an already visited vertex, while ignoring the visitation counters in the dominance procedure for partial paths comprising less than a certain number n of arcs. In the course of the algorithm, n is adjusted (decreased) until a negative reduced cost path is found or $n = 0$, in which case the visitation counters are considered for all paths. While $n > 0$, the ESPPRC is not solved exactly, because not necessarily all undominated negative reduced cost paths will be returned; it is possible that a path is dominated by another path although both cover different customers. The dominated path could possibly have been extended to a customer the dominating path has already visited, and such an extension could lead to a negative reduced cost path which now will not be discovered. Boland et al. 2006 (see also Salani 2005) use the concept of *incremental state space augmentation*. First, they solve the SPPRC without any visitation counters. (The state space of the dynamic-programming-based labelling algorithm consists only of the space defined by the 'usual' resources like cost, time, and capacity.) If non-elementary paths are returned, so-called *critical vertices* are determined. These are vertices that are visited more than once in at least one of the returned paths. For the critical vertices, visitation counters are introduced and the problem is solved again. (The state space is incremented by the space defined by the added visitation counters.) This process is repeated until only elementary paths are returned.

In the worst case, none of the last three approaches can avoid solving the ESPPRC on the complete state space if the VRP(TW) is to be solved exactly.

The above approaches perform so-called *heuristic pricing* before the exact solution of the elementary problem. There are numerous other approaches for generating negative reduced cost columns without solving the ESPPRC exactly. These include (cf. Ropke 2005, p. 202 ff.)

- solving the problem on a reduced network by considering at each vertex only the k shortest emanating arcs,

- limiting the (overall) number of labels that may exist at any point in time in the course of the algorithm,
- solving the SPPRC, examining the resulting paths for negative cycles, and removing them, and
- creating paths by a construction heuristic or by modifying existing paths via local search.

Salani 2005 successfully applies *bounded bidirectional labelling algorithms* to solve the ESPPRC. The basic idea of bidirectional labelling algorithms is to keep the number of labels as low as possible by simultaneously (forward-)extending towards d partial paths starting at o and (backward-)extending towards o partial paths 'starting' at d, and to stop the procedure as soon as forward and backward labels can be joined to yield an o-d-path. Bidirectional labelling is not a heuristic in itself; it is an alternative labelling algorithm technique. It can be applied in any of the approaches just described (cycle elimination, state space augmentation etc.). Whether it solves the underlying problem exactly or not depends on the REFs and the dominance procedure.

Bidirectional labelling algorithms have already been used for solving the SPP (Ahuja et al. 1993, p. 112 f.). Applying the principle to solve the ESPPRC is somewhat more involved. The REFs for forward extension of labels are essentially the same as in the unidirectional algorithm. The REFs for backward extension are, in principle, analogous to the forward REFs. However, their definition and implementation (at least for the TTRP with its intricate interdependencies between trailer positions, collected and transferred load) is quite complicated and laborious.

Three additional steps are necessary or at least highly recommendable in a bidirectional labelling algorithm compared to an unidirectional one (Salani 2005, p. 31 ff.):

(i) To avoid creating every path twice, once by forward and once by backward extension, *bounding* of extensions must be performed.
To this end, a cardinally scaled and constrained so-called *critical* resource is selected. This resource must have a non-negative consumption along all arcs. A forward or backward label is extended only if the consumption of the critical resource in the label resulting from the extension is not more than half of the upper resource window bound at d.

(ii) A *join* operation of a forward and a backward label is required to yield an o-d-path.
To this end, when all forward and backward extension steps are performed, the forward labels at each vertex i are joined with all backward labels at all vertices j for which an arc from i to j exists. The overall resource consumptions of the resulting o-d-paths are computed, and the feasible paths are returned (when they have negative reduced costs).

(iii) *Uniqueness of solutions* should be ensured.
Contrary to most situations in optimization, where it is sufficient to find only one (feasible or optimal) solution, in the pricing problems in column generation, it is sensible to store several or all solutions with negative reduced cost (also structurally different solutions with the same costs). This means that it is *not* sensible to keep only one negative reduced cost path. However, even when bounding is applied, paths may be generated more than once. If a negative reduced cost path contains a sequence of three vertices i, j, and k (in this order), the path may be generated by a join of the forward path from o to i and the backward path from d to j as well as by a join of the forward path from o to j and the backward path from d to k. Such duplicate paths cannot be discarded on the basis of costs alone, and checking whether two paths with the same negative reduced costs are structurally different can be computationally expensive (a comparison of the path structure of two paths cannot be done in constant time). Thus, to ensure uniqueness of solutions in constant time, a so-called *half-way test* can be performed. A feasible join of a forward path from o to i and a backward path from d to j is performed only when the following two conditions are met: (i) The consumption of the critical resource on both the forward and the backward path is

not more than half of the overall consumption of the resource along the complete path (along *this particular path*; for any feasible path, this will always be less than or equal to half of the upper resource window of the critical resource at d), and (ii) the consumption on the forward path plus the consumption on the arc (i, j) is at least half of the overall consumption of the resource along the complete path. To make this point clear, consider the following example. If the consumption of the critical resource at d must not be more than 100, and if there is a feasible, negative reduced cost path $(o, a_1, i, a_2, j, a_3, k, a_4, d)$ with an overall consumption of the critical resource of 40, then without the half-way test, this path would be generated four times, through joins via each of the four arcs. If the consumption of the critical resource is 15 at i and 25 at j (i.e., the consumption along a_2 is 10), the half-way test will join the two partial paths (o, a_1, i) and (d, a_4, k, a_3, j) via arc a_2, and will discard all other joins.

5.3.1.2 Strategies for the Solution of the Pricing Problems

Taking the results of the above authors into account, the following seven strategies for the solution of the pricing problems were tried:

(i) Solution of SPPRCs for all vehicles with two-cycle elimination.

(ii) Solution of ESPPRCs in one stage, i.e., using visitation counters for all customers in the extension and in the dominance step.

(iii) Solution of ESPPRCs in two stages. In the first stage, the visitation counters were used only in the extension step, but not in the dominance step. In the second stage, the visitation counters were used in both steps.

(iv) Solution of ESPPRCs in three stages. The first stage consisted in solving the ESPPRCs on reduced networks containing only the κ shortest arcs emanating from any vertex. For instances with up to 15 customer vertices, κ was set to 3; for larger instances, it was set to 5. As in the first stage of (iii), the visitation counters were used only in the extension step, but not in the dominance step. In the second stage, the ESPPRCs were solved on the full network, considering all visitation counters in the extension step, and considering only the visitation counters for customers whose covering constraints in the master problem had a high dual price (who were covered in the current master problem solution by an artificial variable) in the dominance step. In the third stage, the visitation counters for all customers were considered in both the extension and the dominance step. Additionally, in the second and the third stage, only labels were extended that could possibly still lead to labels with negative reduced costs at the end depot vertex. To this end, the lowest reduced costs on any arc entering a customer vertex and the lowest reduced costs on any arc entering the end depot vertex were determined. In the extension step, the former costs were multiplied with the maximum number of customers that could still be visited (considering capacity and time). Together with the latter costs, this product was added to the current label costs, and the extension was deemed infeasible if the sum was non-negative. Experiments showed that this *bounding-by-reduced-costs procedure* improved the solution speed by approximately three percent compared to a three-stage approach without bounding.

(v) Solution of ESPPRCs in three stages. In the first and second stage, the visitation counters were used only in the extension step, but not in the dominance step. In the first stage, the ESPPRCs were solved on a reduced network as in (iv). In the second stage, the ESPPRCs were solved on the full networks. The third stage consisted in solving the ESPPRCs on the full networks by incremental state space augmentation, where in each iteration, all customer vertices corresponding to customers visited more than once were added to the set of critical vertices.

(vi) Solution of ESPPRCs in three stages as in (iv), but for the second and third stage, the bounded bidirectional labelling algorithm was applied. The collected load was selected as the critical resource. At intermediate vertices, the increase in the collected load is zero. Therefore, to guarantee

uniqueness of the solutions, an additional join test was used: A join of a forward and a backward label was only performed if the end vertex of the path corresponding to the forward label was a customer vertex.

(vii) Heuristic solution of the ESPPRCs by only performing the first and second stage of (v). By skipping the third stage, the ESPPRCs are not solved exactly, and the complete branch-and-price algorithm becomes a heuristic. (This is similar to the heuristic labelling algorithm of Chapter 3.)

In strategies (iii)–(vii), a stage $s > 1$ was performed only when all stages $s' < s$ did not return any negative reduced cost paths for any vehicle (class). The pricing routine always tried to find negative reduced cost paths for all vehicles (vehicle classes) before returning, no matter whether negative reduced cost paths had already been found for one or more vehicles (vehicle classes) or not. In this way, emphasis was put on quickly returning many negative reduced cost columns. So-called *partial pricing*, i.e., not solving the (E)SPPRC for all vehicles (vehicle classes), but returning as soon as a negative reduced cost path for one vehicle (class) is found (cf. Irnich 2002, p. 97 f.) was also tried, but did not yield better results.

In addition, the discretization approach described in Section 4.3.5 was implemented for the TTRP. This yielded large acyclic networks with task cycles. (The networks themselves do not contain cycles, so that each vertex is visited at most once, but it is possible to visit more than one vertex corresponding to one and the same customer.) So, the problem of negative cycles remains. The discretization approach also did not work well. In the experiments, all computation times were much longer than with the cyclic network (by one order of magnitude).

The 2-cycle elimination strategy (i) proved unusable, because the lower bounds were much too weak, so that huge branch-and-bound trees were built up. The one- and two-stage ESPPRC strategies (ii) and (iii) were not as fast as the three-stage strategy (iv). Therefore, in Section 5.4, computational results are reported only for strategies (iv)–(vii).

5.3.1.3 Resources and Resource Extension Functions

The resources used in the labelling algorithm for vehicle k are:

- an unconstrained, cardinally scaled resource for cost
- one cardinally scaled resource for each of the three resource variables used in (5.1)–(5.5), i.e., for collected load, transferred load, and time
- a cardinally scaled visitation counter resource for each customer
- two nominally scaled auxiliary resources for LTC trailers

In the following, these resources are described in detail. The auxiliary resources are presented first, because they are needed to describe the REFs for cost and load.

The first auxiliary resource is needed for the correct modelling of the routing logic at transshipment locations. It must be considered that, on its itinerary, an LTC lorry must visit the decoupling, transfer, and coupling vertices of each transshipment location in the correct order (or not at all), and that, after an LTC lorry has visited the decoupling vertex of transshipment location l, it must not visit any decoupling, transfer, or coupling vertex of any other transshipment location before having visited the coupling vertex of l. To this end, two types of constraint are needed: so-called pairing-and-precedence constraints and follower constraints, see Irnich/Desaulniers 2005, p. 39 f. These authors also present REFs for such constraints (ib., p. 44 f.). They give a possible REF for pairing-and-precedence constraints and an REF for follower constraints. Each REF requires one extra resource. However, it is possible to use only one nominally scaled resource r^{tp} (and one REF) to model the trailer logic. r^{tp} ('trailer position') records

the current position of the trailer by means of a resource variable σ_i^{tp}, which indicates the position of the trailer when the LTC lorry reaches vertex i. σ_i^{tp} is either zero, meaning that the LTC lorry is currently pulling its trailer, or equal to the vertex number of the decoupling vertex where the trailer was parked. The corresponding REF is $f^{r^{tp}}$ with

$$f_{ij}^{r^{tp}}(\sigma_i^{tp}) = \begin{cases} \sigma_i^{tp}, j \in V_C \cup V_{transfer} \cup \{d\} \\ j, j \in V_{decouple} \\ 0, j \in V_{couple} \end{cases}, \tag{5.27}$$

and the resource windows at a vertex j are as shown in Table 5.5.

Resource window	for
$\{0\}$	$j \in \{o, d\} \cup V_{decouple}$
$\{1, \ldots, \lvert V\rvert\}$	$j \in V_{C_L}$
$\{0, \ldots, \lvert V\rvert\}$	$j \in V_{C_{LT}}$
$\{i\}$	$j \in V_{transfer} \cup V_{couple}, i \in V_{decouple}, loc_j = loc_i$

Table 5.5: Resource windows for r^{tp} at vertex j

r^{tp} is also needed to model the load transfer logic at trailer customer and transshipment locations, and for the update of the visitation counter resources.

Moreover, r^{tp} does not only determine the feasibility of an extension of a path along an arc, it is relevant for dominance considerations, too: A label can only dominate another label if their respective current trailer positions are equal.

To consider the reduced costs of the arcs in the pricing problems for LTCs, the second additional resource, r^y, is needed to express the y_{ij}^k variables. r^y is a binary resource. Its value equals one, if LTC lorry k reaches i with its trailer attached, and equals zero otherwise. The corresponding resource variable is σ_i^y, and the REF is f^{r^y} with

$$f_{ij}^{r^y}(\sigma_i^y, \sigma_i^{tp}) = \begin{cases} 1, (\sigma_i^{tp} = 0 \ \wedge \ i \notin V_{decouple}) \ \vee \ i \in V_{couple} \\ 0, \text{otherwise} \end{cases}. \tag{5.28}$$

For each of the resource variables used in formulation (5.1)–(5.5), there is one constrained resource. For the total amount of customer supplies that lorry k has collected when reaching vertex i, the resource r^{coll} with resource variable σ_i^{coll} and REF $f^{r^{coll}}$ with

$$f_{ij}^{r^{coll}}(\sigma_i^{coll}) = \sigma_i^{coll} + su_i \tag{5.29}$$

was used. $f_{ij}^{r^{coll}}$ models constraints (5.4c) and sets σ_j^{coll} to the lowest nonnegative value fulfilling these constraints. The resource window at a vertex j is $[0, q_k^{total}]$.

For the total amount of customer supplies that LTC lorry k has transferred when reaching vertex i, the resource r^{trans} with resource variable σ_i^{trans} and REF $f^{r^{trans}}$ with

$$f_{ij}^{r^{trans}}(\sigma_i^{trans}, \sigma_i^{coll}, \sigma_i^{tp}) =$$

$$\begin{cases} \sigma_i^{trans} & , j \in V_{C_L} \vee (j \in V_{C_{LT}} \wedge \sigma_i^{tp} \neq i) \\ \sigma_i^{trans} + \min\{su_i, q_k^{trailer} - \sigma_i^{trans}\} & , j \in V_{C_{LT}} \wedge \sigma_i^{tp} = i \\ \sigma_i^{trans} + \min\{\sigma_i^{coll} - \sigma_i^{trans}, q_k^{trailer} - \sigma_i^{trans}\} & , j \in V_{I_T} \end{cases} \tag{5.30}$$

was used. $f_{ij}^{r^{trans}}$ models constraints (5.4a), (5.4b), and (5.4d)–(5.4g) and sets σ_j^{trans} to the highest possible value that fulfils these constraints. The resource window at a vertex j is $[0, q_k^{trailer}]$.

This means that an LTC lorry always transfers as much load as possible to its trailer (its complete load or an amount of load equal to the residual capacity of its trailer, whichever is less). This is a sensible stipulation, as it was assumed above that the time needed for a load transfer is fixed, independent of the actual amount of load transferred.

For the point in time when lorry k begins its service at vertex i, the resource r^{time} with resource variable σ_i^{time} and REF $f^{r^{time}}$ with

$$f_{ij}^{r^{time}}(\sigma_i^{time}) = \max\left\{twa_j, \sigma_i^{time} + \tau_{ij}^{tr}\right\} \tag{5.31}$$

was used. $f_{ij}^{r^{time}}$ models constraints (5.8) and sets σ_j^{time} to the lowest nonnegative value fulfilling these constraints. The resource window at a vertex j is $[twa_j, twb_j]$.

For each $v_c \in V_C$, there is one visitation counter resource r^{visit,v_c} with resource variable $\sigma_i^{v_c}$ at each vertex $i \in V$ and REF $f^{r^{visit,v_c}}$ with

$$f_{ij}^{r^{visit,v_c}}(\sigma_i^{v_c}, \sigma_j^{coll}, \sigma_j^{trans}, \sigma_j^{time}, \sigma_j^{tp}) = \begin{cases} 0 & , v_c = i \vee \text{ not enough capacity } \vee \text{ not enough time} \\ \sigma_i^{v_c} & , \text{ otherwise} \end{cases} \tag{5.32}$$

where $\sigma_j^{v_c} = 0$ means that customer v_c cannot be visited any more, and where 'not enough capacity' means

$$\begin{aligned} & \sigma_j^{coll} + su_j > q_k^{total} \\ \vee\ & (v_c \neq j \wedge \sigma_j^{coll} + su_j + su_{v_c} > q_k^{total}) \\ \vee\ & (\sigma_j^{tp} \neq j \wedge \sigma_j^{coll} - \sigma_j^{trans} + su_j > q_k^{lorry}) \\ \vee\ & (v_c \neq j \wedge \sigma_j^{tp} \neq j \wedge \sigma_j^{coll} - \sigma_j^{trans} + su_j + su_{v_c} > q_k^{lorry}) \end{aligned}$$

and 'not enough time' means

$$\sigma_j^{time} + \tau_{jv_c}^{tr} > twb_{v_c}, \text{ where } \tau_{hh}^{tr} := 0 \; \forall\, h \in V.$$

The resource window for vertex j at vertex j is $[1, 1]$, and the resource window for a vertex $j' \neq j$ at vertex j is $[0, 1]$.

As explained in Section 5.1.3, transshipment subtour symmetries are possible if there is more than one intermediate vertex per transshipment location. Such symmetries are excluded in formulation (5.1)–(5.5) by constraints (5.4j)–(5.4l). However, when the pricing problems are solved by a labelling algorithm, it is not necessary to introduce more than one intermediate vertex in the LTC subnetworks. (It is optimal to choose $n^{TS} = 3$.) It is sufficient to have only one, and to allow multiple visits to this vertex. Visitation counters are maintained only for the customers, not for the transshipment vertices. With one intermediate vertex, no transshipment subtour symmetries are possible. Also, elementarity of paths can be ensured with visitation counters only for the customers, as every possible cycle in the subnetworks (as well as in the network corresponding to the arc variable formulation) contains a customer. It is then also possible to use a transshipment location more than once, which can be relevant if there are time windows. This is an additional degree of freedom that constitutes a considerable advantage of the path variable formulation compared to the arc variable formulation.

Indeed, when the pricing problems are solved by a labelling algorithm, it is sufficient to use only one vertex per transshipment location. Although this reduces the subnetwork size for the LTC subproblems, the number of possible labels, which determines the difficulty of an (E)SPPRC, remains the same.

Finally, there is one unconstrained resource r^{cost} measuring the costs. The corresponding resource variable is σ_i^{cost}, and the REF for single lorries and LTC lorries that do not use their trailer is $f^{r^{cost}}$ with

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}) = \sigma_i^{cost} + \tilde{c}_{ij}^k, \tag{5.33}$$

where $\tilde{c}_{ij}^k$ are the reduced costs of traversal of arc (i, j) for vehicle k. The REF for LTCs is

$$f_{ij}^{r^{cost}}(\sigma_i^{cost}, \sigma_j^{y}) = \sigma_i^{cost} + \tilde{c}_{ij}^k(\sigma_j^{y}), \tag{5.34}$$

where $\tilde{c}_{ij}^k(\sigma_j^{y})$ are the reduced costs of traversal of arc (i, j) for LTC k. σ_j^{y} is needed to carry the information whether the LTC lorry is currently pulling its trailer, so that the correct reduced costs according to Table 5.4 are taken into account. (σ_j^{y} serves the same purpose as $\delta_{ij}^{trailer,k}$.)

With these resource specifications, a feasible label l_1 dominates a feasible label l_2 if and only if

- both reside at the same vertex i,
- $\sigma_i^{tp}(l_1) = \sigma_i^{tp}(l_2)$,
- $\sigma_i^{y}(l_1) = \sigma_i^{y}(l_2)$ (which is implied by the previous item),
- $\sigma_i^{cost}(l_1) \leqq \sigma_i^{cost}(l_2)$,
- $\sigma_i^{coll}(l_1) \leqq \sigma_i^{coll}(l_2)$,
- $\sigma_i^{trans}(l_1) \leqq \sigma_i^{trans}(l_2)$,
- $\sigma_i^{time}(l_1) \leqq \sigma_i^{time}(l_2)$,
- $\sigma_i^{v_c}(l_1) \geqq \sigma_i^{v_c}(l_2)\ \forall\ v_c \in V_C$,
- and at least one of the above inequalities is strict,

where $\sigma_i^r(l)$ denotes the value of the resource variable σ_i^r for a label l resident at vertex i.

The value of the cost resource of a label l resident at the end depot vertex d indicates the reduced costs of the path represented by the label. If a path p is feasible and if the reduced costs of a path, (5.13), are negative, a new column corresponding to p can be added to the restricted master problem. The path itself is recursively reconstructed from the labels, starting with l, because each label stores its direct predecessor arc and predecessor label. As explained in Section 5.1.4.1, the trailer (sub)path(s) of a path of an LTC lorry is/are also unequivocal and can easily be reconstructed, so, the objective function coefficient of a new column in the restricted master problem can be determined efficiently, too.

5.3.1.4 Technical Issues

Some remarks concerning technical issues in the solution of the pricing problems follow.

In theory, the dominance step in the labelling algorithm is optional. To get an acceptable behaviour of the algorithm with respect to time and memory requirements, however, an 'efficient' dominance check is crucial. 'Efficient' means that, on the one hand, no dominance checks are omitted that could lead to the removal of dominated labels, and that, on the other hand, no unnecessary dominance checks are performed. A (pairwise) dominance check is unnecessary, in particular, when one of the labels involved is already known to be dominated, and when the dominance check has already been performed. In order to avoid such operations in for problems with at least one strictly increasing resource, the following is possible. At the time a label l resident at a vertex i is selected for extension, all labels resident at i that could possibly dominate l are already generated and resident at i. Hence, before extending l, a pairwise dominance check must be performed between all pairs of labels l_1, l_2 at i for which dominance has not already been checked, as long as l_1 and l_2 are undominated. If w.l.o.g. l_1 dominates l_2, l_2 is removed from the set of resident labels and not considered for any future dominance checks. Also, in the average case, it is useful to check first whether l_1 dominates l_2, if l_1 has been added to the set of resident labels

at i earlier than l_2, because it is more likely that the 'older' label dominates the 'newer' one (has used fewer resources, has been extended from a shorter partial path) than the other way round. This approach may be called *complete-pairwise strategy*.

The complete-pairwise strategy, though, is not always optimal. In some cases, different strategies lead to fewer dominance checks, because a dominance relationship is detected earlier. Experiments were performed with the strategy to simply check each label immediately before its extension against each other label currently resident at the same vertex. This yielded faster running times on some instances. Although this strategy (which may be called *check-each-before-extension strategy*) allows redundant (multiple) dominance checks between pairwisely undominated labels, it detects dominance relationships earlier in some cases, and this effect sometimes more than compensates for the redundant checks.

In the computational experiments for the TTRP, the same instances as for the VRPTT were used. These instances considered two different lorry classes. For instances where the subnetworks are identical for both lorry types, both ESPPRCs can be solved in one run of a labelling algorithm. To this end, labels for both lorry types are maintained (by storing the lorry type in an extra resource) during the course of the algorithm. However, in the dominance step at a vertex, in the worst case, a pairwise check must be performed for each pair of labels resident at that vertex. Independent of the other resource values, labels with different lorry types never dominate one another. Therefore, all calls of the dominance function with two labels with different lorry types are unnecessary but inevitable. The non-dominance relation between such labels can be checked at the very beginning of the dominance function, but the number of calls of the dominance function nevertheless increases significantly: In the worst case, if there are n labels for each lorry type, and when two subproblems are solved sequentially, there are $2 \cdot n^2$ calls. When one subproblem with both lorry types is solved, there are $1 \cdot (2n)^2$ calls, i.e., the number of calls doubles. The dominance function, along with the REF, is decisive for the run time of the algorithm. Doubling the number of calls may thus increase the overall running time by 50 %. It is therefore preferable to solve one subproblem for each lorry type.

An important point for the implementation of the bounded bidirectional labelling algorithm is the following. After the join operation, there may be an enormous number of paths (with negative reduced costs). This is because, during the join, no dominance is performed. If forward labels residing at vertex i and forward labels residing at vertex j are joined with backward labels at vertex k, the respective forward and backward labels are undominated, but the join of a forward label residing at i and a backward label residing at k may dominate the join of a forward label residing at j with a backward label residing at k. Hence, it is decisive to perform dominance over all paths resulting from the join. In the implementation used here, only paths with negative reduced costs were deemed feasible for a join; nevertheless, even for small instances, sometimes more than 100,000 paths were returned if no dominance check was performed.

An interesting note is that the solution of the pricing problems was faster when the networks for each vertex of the branch-and-bound tree were allocated and deallocated on the heap (with `new` and `delete`) than when they were constructed as local objects on the stack.

5.3.2 Adding Valid Inequalities

Although the experiments with the branch-and-cut formulation have shown that the valid inequalities from Section 5.2.1 are not very strong, the static cuts are indeed useful to raise the lower bound at the root vertex of the branch-and-bound tree. (Dynamically) adding valid inequalities to the master problem in branch-and-price algorithms leads to so-called branch-and-price-and-cut algorithms. Successful implementations of this approach are presented, e.g., in Kohl et al. 1999 and Fukasawa et al. 2006. Hence, it is worthwhile to consider which of the inequalities from Section 5.2.1 could be useful in the branch-and-price algorithm:

- The supply collection cut cannot be used, because there are no load variables in the master problem.
- The trailer flow cut cannot be used either, because, as mentioned above, the test instances for the computational experiments were created such that there is no customer with a supply exceeding the capacity of the largest lorry, and in the tests, an unlimited number of vehicles was allowed, so that the number of necessary trailers is zero.
- When solving the ESPPRC (and not the SPPRC), it can be shown that only inequalities considering more than one vehicle can be violated (cf. Kohl et al. 1999, p. 107). Hence, the connectivity cuts, the generalized 1-path cuts, the implied bound cuts, and the arrival time cuts are not helpful.
- The generalized 2-path cuts are potentially useful, but for larger instances, they have to be separated dynamically. Their separation is rather involved and could not be implemented until the deadline of this paper.

Thus, only the lorry flow cut remains. It is a simple static cut and has been added to the master problem at the root vertex of the branch-and-bound tree, thereby increasing the lower bound. For some instances, this has significantly reduced the computation time. However, the resulting overall solution algorithm cannot be called a branch-and-price-and-cut algorithm, because no valid inequalities are separated dynamically.

5.3.3 Branching and Enumeration Strategies

The following three-stage branching strategy was used:

(i) branch on the number of tours

(ii) branch on an aggregated arc variable (i.e., aggregated over all vehicles or vehicle classes) for an arc whose head and/or tail is a customer vertex

(iii) branch on an arc variable for a vehicle or vehicle class for an arc whose head and/or tail is a customer vertex

It is sufficient to consider the x_{ij}^k variables for the branching decisions. As described in Section 5.1.4.1, once these are all binary, the path variables will be binary as well. It is even sufficient that all x_{ij}^k variables where i or j are customer vertices are integral. This is because each customer is visited exactly once. This also holds if there are identical pricing problems, again because of the constraint that each customer is visited exactly once.

Computational experiments have shown that when the lorry flow cut is not added to the master, the first, 'global', branching decision, namely, to branch on the number of tours, is decisive for the whole procedure. If it is not used, the tree quickly becomes too large and too many time-consuming ESPPRCs must be solved, so that only very small instances can be solved. If the lorry flow cut is added to the master, branching on the number of tours is no longer decisive, but it is still useful on some instances.

A possible refinement that was not implemented is strong branching, i.e., the testing of several potential branching decisions. The two linear programs resulting from each candidate decision are evaluated and the decision where the minimal change of the two objective function values is maximal is taken. This is particularly interesting for problems where the branch-and-bound tree is highly unbalanced, because such unbalancedness shows that the 'standard' branching decisions are often weak and do not lead to a significant improvement of the respective lower bound. Although not all the trees of the test instances were examined, the ones that were did not exhibit an unbalanced structure, and a considerable number of instances was solved at the root vertex of the branch-and-bound tree, so that no branching decisions were necessary at all. Moreover, strong branching in branch-and-price algorithms is not as straightforward as in branch-and-cut or even pure branch-and-bound algorithms, where branching on variables is

performed. In addition, for a problem where the pricing problem solution is the most time-consuming part, not all candidate linear programs should be solved exactly, but rather, heuristically. This, however, increases the risk of taking the wrong decision. Hence, strong branching was not used.

As enumeration strategy, 'dive and best' was used as in the branch-and-cut algorithm.

5.4 Computational Experiments

The algorithms described above were implemented in C++ using the Boost Graph library and the ABACUS framework (`www.informatik.uni-koeln.de/abacus`) with CPLEX as LP solver. The maximum flow problems for the separation of the subtour elimination constraints in the branch-and-cut algorithm were solved with the Edmonds-Karp algorithm provided by the BGL. The pricing problems in the branch-and-price algorithm were solved with the `r_c_shortest_paths` framework.

5.4.1 Test Instances

The test instances described in Chapter 4 were also used here. With $n^{TS} = 3$, an x_y_z instance has $1 + x+y+3\cdot(y+z)+1$ vertices: One for the start depot, x for the lorry customers, y for the trailer customers, three for each transshipment location (of which there are $y + z$) to represent decoupling, transfer, and coupling, and one for the end depot. Table 5.6 shows how the size of the resulting mathematical programs develops for TTRP instances with $n^{TS} = 3$ where all locations have one time window. The data are compared with those of Table 4.12 on page 111.

Instance type	1_1_1	2_2_2	2_2_2	3_3_3	4_4_4
Vehicles	2 lorries 2 trailers	2 lorries 2 trailers	4 lorries 4 trailers	4 lorries 4 trailers	6 lorries 6 trailers
TTRP					
Vertices	10	18	18	26	34
Arcs	33	121	121	265	465
Arc variables	92	328	656	1,424	3,732
Resource variables	60	108	216	312	612
Constraints	348	1,134	2,264	4,746	12,206
VRPTT					
Vertices	16	30	54	80	154
Arcs	131	493	1,477	3,283	11,737
Arc variables	234	876	4,816	10,736	56,412
Resource variables	92	168	496	728	1,920
Constraints	910	3,378	18,989	42,250	223,504
VRPTW					
Vertices	4	6	6	8	10
Arcs	7	21	21	43	73
Arc variables	28	84	168	344	876
Resource variables	32	48	96	128	240
Constraints	66	140	276	486	1,136

Table 5.6: TTRP network size growth

The above formulation (5.1)–(5.5) and the branch-and-cut algorithm use load-dependent load transfer times, whereas the path variable reformulation and the branch-and-price algorithm assume fixed load transfer times (following Chao 2002 and Scheuerer 2004). What is lost in solution precision if this

simplification is also made for the test instances used here (which assume a uniformly distributed supply between 1,000 and 10,000 units and a load transfer time of two minutes per 1,000 units of supply)? The expected value of supply at a customer is 5,500. For a two-axle trailer, and for $n^{TS} = 4$, an average load transfer amount of 2,500 per vertex is reasonable. If only the decoupling and the coupling vertex of a transshipment location are visited and a total of 10,000 is transferred, an error of -10 minutes results, i.e., the tour is erroneously made 10 minutes too short. If all four transshipment vertices are visited, but only an infinitely small amount of load is transferred, an error of $+20$ minutes results. Similarly, for a three-axle trailer, $n^{TS} = 5$, and an average load transfer amount of 3,000 per vertex, if only the decoupling and the coupling vertex are visited and 15,000 units of load are transferred, the tour is made 18 minutes too short. If all five transshipment vertices are visited and only an infinitely small amount of load is transferred, the resulting tour is 30 minutes too long. In the worst case, this means that, for every vehicle, the tour length is overestimated by half an hour. For the assumed cost structure and a rough estimate for tour length and duration of 250 kilometers and 10 hours in practice, this means a deviation from the optimal costs of 3.3 % in the worst case. This may or may not be acceptable.

5.4.2 System Parameters

The method selected for the solution of the master problem in the branch-and-price algorithm was the CPLEX barrier method with crossover (`ABA_LP::BarrierAndCrossover`). Preliminary experiments were also conducted using primal or dual simplex algorithms. None of the three methods was consistently better than the others. For the branch-and-cut algorithm, the selection of the method for the solution of the LP-relaxations was left to CPLEX.

The system parameters used in the computational experiments are shown in the following table. All ABACUS parameters not shown in the table were at their default values.

Parameter	Setting
CPU frequency	2 GHz
Main memory	1 GB
ABACUS version	2.3
CPLEX version	9.1
Wall-clock time limit	11,100 seconds
Min. violation of subtour elimination constraints to be considered violated	10^{-3}
`Guarantee`	0.0
`ObjInteger`	false
`TailOffNLps`	3
`TailOffPercent`	10^{-4}
`FixSetByRedCost`	false
`MaxConAdd`	10^5
`MaxConBuffered`	10^5
`MaxVarAdd`	10^5
`MaxVarBuffered`	10^5
`EliminateFixedSet`	false
`ConstraintEliminationMode`	none
`ConElimEps`	10^{-3}
`VariableEliminationMode`	none
`VarElimEps`	10^{-3}

Table 5.7: System parameters TTRP

Again, all reported running times are wall-clock times. As the check for the elapsed time was not implemented in a different thread, the running times sometimes exceeded the specified wall-clock time limit, in particular for the branch-and-price algorithm.

5.4.3 Computational Results

As before, table entries of the form x / y / z indicate the minimal, average, and maximal value respectively.

5.4.3.1 Results for the Branch-and-Cut Algorithm

In the experiments with branch-and-cut, taking into account the results for the VRPTT, no time costs were considered, and $n^{TS} = 3$ was assumed. Setting $n^{TS} = 3$ means, strictly speaking, that the algorithm is only a heuristic. However, comparing the objective function values of the branch-and-cut and the branch-and-price algorithm showed that all instances that were completely solved by the branch-and-cut algorithm within the time limit were indeed solved to optimality.

The results obtained are shown in the following table. The line headings in the table have the following meanings:

- % Gap at root: percentage by which best feasible solution (BFS) exceeds lower bound at the root vertex of the branch-and-bound tree when all cuts are used ($RLB\text{-}All$): $(BFS - RLB\text{-}All)/RLB\text{-}All \cdot 100$
- % Gap at root without cuts: percentage by which best feasible solution (BFS) exceeds lower bound at the root vertex of the branch-and-bound tree when no cuts are used ($RLB\text{-}No$): $(BFS - RLB\text{-}No)/RLB\text{-}No \cdot 100$
- % LB increase by cuts: percentage by which lower bound at the root vertex of the branch-and-bound tree when all cuts are used ($RLB\text{-}All$) exceeds lower bound at the root vertex of the branch-and-bound tree when no cuts are used ($RLB\text{-}No$): $(RLB\text{-}All - RLB\text{-}No)/RLB\text{-}No \cdot 100$
- % Gap at end: percentage by which best feasible solution (BFS) exceeds best lower bound at the end of the optimization (BLB) (zero if optimal solution is found, not counted if no feasible solution is found): $(BFS - BLB)/BLB \cdot 100$

Instance type	1_1_1	2_2_2	3_3_3	4_4_4
No. of flow variables	92 / 92 / 92	328 / 437 / 656	712 / 1,258 / 1,424	2,488 / 2,799 / 3,732
No. of resource variables	60 / 60 / 60	108 / 144 / 216	156 / 275 / 312	408 / 459 / 612
No. of constraints	348 / 348 / 348	1,134 / 1,511 / 2,264	2,376 / 4,193 / 4,746	8,140 / 9,157 / 12,206
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 29	30 / 29 / 8	16 / 6 / 0
Running time [s]	0 / 0 / 1	8 / 607 / 11,100	357 / 10,152 / 15,100	11,100 / 12,267 / 14,700
% Gap at root	0 / 8 / 30	6 / 13 / 29	5 / 33 / 88	5 / 39 / 88
% Gap at root without cuts	69 / 216 / 613	108 / 310 / 946	135 / 426 / 1,024	275 / 531 / 851
% LB increase by cuts	69 / 193 / 523	96 / 266 / 884	105 / 291 / 602	167 / 287 / 581
% Gap at end	0 / 0 / 0	0 / 0 / 14	0 / 21 / 79	32 / 38 / 53
No. of B&B vertices	3 / 15 / 29	61 / 548 / 5,671	501 / 3,740 / 7,065	785 / 1,363 / 1,581
Highest level in tree	2 / 6 / 10	13 / 29 / 64	43 / 73 / 106	29 / 157 / 668
No. of separated 1-path cuts	1 / 12 / 20	61 / 138 / 236	257 / 718 / 1,780	173 / 776 / 1,290
No. of separated conn. cuts	0 / 37 / 77	201 / 679 / 1,970	1,731 / 3,838 / 6,604	290 / 3,836 / 13,944

Table 5.8: Computational results for branch-and-cut algorithm

The most important observations to be made in Table 5.8 are:

- As was to be expected, only very small instances can be solved. However, as for the VRPTT, 'small' refers only to the number of customers and transshipment locations, not to the size of the mathematical programs, i.e., the number of variables and constraints.

- Again, the difficulty of the instances within the same instance type varies widely.
- The number of vertices in the branch-and-bound tree and the tree levels reached are both very high on average.
- The use of the static and dynamic cuts increases the lower bound at the root vertex of the branch-and-bound tree by a factor of between 2 and 3 on average. This means that the use of the cuts is the decisive point which enables the algorithm to solve any instances at all.
- Overall, the results are comparable to those obtained with the branch-and-cut algorithm for the VRPTT.

5.4.3.2 Results for the Branch-and-Price Algorithm

The results obtained for the branch-and-price algorithm with the different pricing problem solution strategies are shown in the following tables. Results for the smallest instance types (1_1_1 and 2_2_2) are not reported. The instances of these types were solved in fractions of a second, and only two of them were not solved at the root vertex of the branch-and-bound tree.

Even when the entries in the line 'No. of tried / feasible / optimal' are the same, the different strategies sometimes differ with respect to the number of subproblems, of (E)SPPRCs per subproblem, of generated variables, and with respect to the highest level reached in the branch-and-bound tree. Such differences are due to the fact that the paths are returned to the master problem in a different sequence by the strategies. This leads to different branching decisions and, hence, to different branch-and-bound trees.

Instance Type	3_3_3	4_4_4	5_5_5	6_6_6
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30	30 / 30 / 30
No. solved optimally at root	25	20	20	12
Running time [s]	1 / 4 / 75	1 / 33 / 258	3 / 344 / 8,230	10 / 869 / 6,120
Pricing time [% Running time]	77 / 85 / 92	78 / 92 / 98	93 / 98 / 100	93 / 99 / 100
No. of subproblems	1 / 19.7 / 403	1 / 32.5 / 243	1 / 27.1 / 545	1 / 95.3 / 1,533
No. of (E)SPPRCs per subpr.	25 / 63 / 96	24 / 62 / 112	28 / 82 / 160	21 / 69 / 200
No. of labels per (E)SPPRC	268 / 508 / 982	561 / 1,469 / 2,333	1,788 / 4,644 / 13,012	3,057 / 9,268 / 23,945
No. of generated variables	46 / 184 / 2,222	91 / 524 / 3,287	145 / 869 / 15,414	236 / 1,788 / 17,421
Highest level in tree	1 / 3.0 / 27	1 / 5.5 / 24	1 / 3.9 / 31	1 / 7.8 / 34
No. of tours	1 / 1.77 / 2	2 / 2.37 / 3	2 / 2.63 / 3	2 / 3.1 / 4
No. of LTC Tours	0 / 1.13 / 2	0 / 1.73 / 2	1 / 1.87 / 3	2 / 2.5 / 3
Longest Tour [No. of arcs]	4 / 7.4 / 11	5 / 8.27 / 11	6 / 8.87 / 14	6 / 8.97 / 12

Table 5.9: Computational results for unidirectional strategy

Instance Type	7_7_7	8_8_8	9_9_9	10_10_10
No. of tried / feasible / optimal	30 / 30 / 26	30 / 28 / 16	30 / 20 / 4	30 / 21 / 2
No. solved optimally at root	12	8	1	0
Running time [s]	22 / 2,659 / 12,300	49 / 5,965 / 14,600	115 / 11,894 / 36,200	1,860 / 15,782 / 45,900
Pricing time [% Running time]	97 / 99 / 100	98 / 100 / 100	98 / 100 / 100	100 / 100 / 100
No. of subproblems	1 / 94.2 / 1,077	1 / 211.2 / 1,365	1 / 63.6 / 535	1 / 40.2 / 145
No. of (E)SPPRCs per subpr.	21 / 65 / 164	14 / 56 / 160	15 / 37 / 116	14 / 44 / 184
No. of labels per (E)SPPRC	5,802 / 16,851 / 83,877	7,411 / 25,572 / 81,969	14,469 / 65,572 / 125,673	42,288 / 94,858 / 160,782
No. of generated variables	265 / 1,723 / 12,171	306 / 2,718 / 23,879	426 / 1,184 / 3,799	624 / 1,023 / 1,987
Highest level in tree	1 / 8.4 / 36	1 / 16.2 / 131	1 / 12.9 / 94	1 / 9.6 / 60
No. of tours	3 / 3.8 / 4	3 / 4.14 / 6	4 / 4.35 / 5	4 / 5.38 / 9
No. of LTC tours	2 / 3.07 / 4	2 / 3.68 / 5	3 / 3.95 / 4	2 / 4.19 / 6
Longest Tour [No. of arcs]	7 / 8.97 / 11	7 / 9.93 / 13	8 / 10.65 / 14	8 / 10.62 / 18

Table 5.10: Computational results for unidirectional strategy (cont.)

Instance Type	3_3_3	4_4_4	5_5_5	6_6_6
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 29	30 / 30 / 30
No. solved optimally at root	25	20	20	12
Running time [s]	1 / 5 / 86	1 / 46 / 358	4 / 516 / 11,100	11 / 1,193 / 6,080
Pricing time [% Running time]	79 / 87 / 98	82 / 93 / 99	95 / 98 / 100	94 / 99 / 100
No. of subproblems	1 / 19.7 / 403	1 / 33.9 / 279	1 / 16.3 / 233	1 / 92.7 / 1,533
No. of (E)SPPRCs per subpr.	25 / 63 / 96	24 / 62 / 112	28 / 82 / 160	21 / 69 / 200
No. of labels per (E)SPPRC	304 / 532 / 1,159	608 / 1,332 / 2,100	1,741 / 3,852 / 10,268	2,563 / 6,183 / 12,909
No. of generated variables	46 / 184 / 2,229	91 / 546 / 3,843	145 / 525 / 5,223	236 / 1,755 / 17,522
Highest level in tree	1 / 2.9 / 27	1 / 5.5 / 24	1 / 3.6 / 22	1 / 7.7 / 31
No. of tours	1 / 1.77 / 2	2 / 2.37 / 3	2 / 2.63 / 3	2 / 3.1 / 4
No. of LTC tours	0 / 1.13 / 2	0 / 1.73 / 2	1 / 1.87 / 3	2 / 2.53 / 3
Longest Tour [No. of arcs]	4 / 7.4 / 11	5 / 8.37 / 12	6 / 8.9 / 14	6 / 9.03 / 12

Table 5.11: Computational results for bidirectional strategy

Instance Type	7_7_7	8_8_8	9_9_9	10_10_10
No. of tried / feasible / optimal	30 / 30 / 25	30 / 29 / 15	30 / 23 / 9	30 / 25 / 7
No. solved optimally at root	12	8	4	4
Running time [s]	27 / 3,420 / 14,900	93 / 6,687 / 16,200	148 / 11,116 / 52,900	2,140 / 11,200 / 15,400
Pricing time [% Running time]	99 / 99 / 100	98 / 100 / 100	99 / 100 / 100	100 / 100 / 100
No. of subproblems	1 / 83.2 / 775	1 / 139.2 / 1,313	1 / 26.7 / 153	1 / 18.8 / 87
No. of (E)SPPRCs per subpr.	21 / 66 / 164	14 / 54 / 160	14 / 54 / 156	12 / 56 / 188
No. of labels per (E)SPPRC	5,228 / 9,739 / 41,418	6,431 / 13,953 / 31,110	7,454 / 16,294 / 36,591	11,351 / 36,151 / 291,166
No. of generated variables	265 / 1,570 / 9,873	306 / 1,756 / 11,013	426 / 7,566 / 28,342	625 / 4,958 / 16,458
Highest level in tree	1 / 8.6 / 43	1 / 13.6 / 78	1 / 7.2 / 59	1 / 4.8 / 13
No. of tours	3 / 3.8 / 4	3 / 4.14 / 6	3 / 4.27 / 5	4 / 4.76 / 6
No. of LTC tours	2 / 3.13 / 4	2 / 3.66 / 5	3 / 3.86 / 5	3 / 4.48 / 6
Longest Tour [No. of arcs]	7 / 9.07 / 12	7 / 10.24 / 13	8 / 10.23 / 13	8 / 10.12 / 14

Table 5.12: Computational results for bidirectional strategy (cont.)

Instance Type	3_3_3	4_4_4	5_5_5	6_6_6
No. of tried / feasible / optimal	30 / 30 / 30	30 / 30 / 30	30 / 30 / 28	30 / 30 / 27
No. solved optimally at root	25	20	20	12
Running time [s]	1 / 11 / 130	1 / 139 / 1,060	8 / 1,149 / 12,900	21 / 2,799 / 14,500
Pricing time [% Running time]	83 / 92 / 100	85 / 96 / 100	96 / 99 / 100	97 / 100 / 100
No. of subproblems	1 / 19.6 / 403	1 / 33.4 / 261	1 / 12.3 / 89	1 / 90.3 / 1,535
No. of (E)SPPRCs per subpr.	31 / 70 / 100	31 / 72 / 128	36 / 93 / 200	31 / 80 / 196
No. of labels per (E)SPPRC	365 / 1,200 / 4,604	799 / 2,974 / 7,084	2,929 / 11,595 / 64,898	5,194 / 19,019 / 43,595
No. of generated variables	46 / 187 / 2,325	91 / 544 / 3,666	145 / 422 / 1,891	236 / 1,626 / 17,510
Highest level in tree	1 / 3.0 / 27	1 / 5.5 / 24	1 / 3.6 / 22	1 / 7.7 / 32
No. of tours	1 / 1.77 / 2	2 / 2.37 / 3	2 / 2.67 / 4	2 / 3.1 / 4
No. of LTC tours	0 / 1.13 / 2	0 / 1.73 / 2	1 / 1.9 / 3	2 / 2.53 / 3
Longest Tour [No. of arcs]	4 / 7.4 / 11	5 / 8.37 / 12	6 / 8.97 / 14	6 / 9.03 / 12

Table 5.13: Computational results for incremental state space augmentation strategy

Instance Type	7_7_7	8_8_8	9_9_9	10_10_10
No. of tried / feasible / optimal	30 / 30 / 21	30 / 26 / 14	7 / 4 / 1	0 / 0 / 0
No. solved optimally at root	12	8	1	–
Running time [s]	48 / 5,156 / 25,200	186 / 7,620 / 24,400	283 / 9,671 / 15,500	–
Pricing time [% Running time]	99 / 100 / 100	99 / 100 / 100	100 / 100 / 100	–
No. of subproblems	1 / 56.1 / 617	1 / 67.6 / 641	1 / 20.0 / 33	–
No. of (E)SPPRCs per subpr.	20 / 75 / 176	23 / 70 / 208	22 / 47 / 128	–
No. of labels per (E)SPPRC	8,604 / 31,091 / 127,581	13,118 / 59,687 / 163,704	26,119 / 115,161 / 158,989	–
No. of generated variables	265 / 1,039 / 6,396	306 / 1,157 / 9,056	499 / 733 / 1,300	–
Highest level in tree	1 / 7.1 / 24	1 / 7.2 / 20	1 / 5.3 / 8	–
No. of tours	3 / 3.87 / 4	0 / 3.67 / 6	0 / 2.57 / 5	–
No. of LTC tours	2 / 3.1 / 4	0 / 3.07 / 5	0 / 2.29 / 4	–
Longest Tour [No. of arcs]	7 / 9.07 / 11	0 / 8.6 / 13	0 / 5.71 / 11	–

Table 5.14: Computational results for incremental state space augmentation strategy (cont.)

Instance Type	3_3_3	4_4_4	5_5_5	6_6_6
No. of tried / feasible / optimal	30 / 30 / 29	30 / 30 / 30	30 / 30 / 30	30 / 30 / 29
No. solved optimally at root	24	20	20	13
Running time [s]	1 / 3 / 41	1 / 12 / 99	2 / 31 / 460	7 / 162 / 2,090
Pricing time [% Running time]	70 / 80 / 92	63 / 87 / 96	84 / 95 / 100	84 / 96 / 99
No. of subproblems	1 / 16.5 / 325	1 / 27.3 / 225	1 / 13.3 / 241	1 / 91.8 / 2,009
No. of (E)SPPRCs per subpr.	19 / 56 / 84	19 / 56 / 104	20 / 72 / 136	20 / 62 / 164
No. of labels per (E)SPPRC	242 / 421 / 739	510 / 1,097 / 1,867	1,410 / 2,960 / 7,625	1,716 / 4,703 / 9,821
No. of generated variables	46 / 170 / 1,977	91 / 438 / 2,991	145 / 514 / 6,770	232 / 1,601 / 24,809
Highest level in tree	1 / 2.9 / 29	1 / 4.9 / 24	1 / 2.8 / 21	1 / 6.9 / 34
No. of tours	1 / 1.77 / 2	2 / 2.37 / 3	2 / 2.63 / 3	2 / 3.1 / 4
No. of LTC tours	0 / 1.13 / 2	0 / 1.73 / 2	1 / 1.87 / 3	2 / 2.53 / 3
Longest Tour [No. of arcs]	4 / 7.3 / 11	5 / 8.27 / 11	6 / 8.9 / 14	6 / 8.97 / 12

Table 5.15: Computational results for heuristic algorithm

Instance Type	7_7_7	8_8_8	9_9_9	10_10_10
No. of tried / feasible / optimal	30 / 30 / 25	30 / 30 / 15	30 / 30 / 4	30 / 30 / 2
No. solved optimally at root	12	7	1	1
Running time [s]	15 / 1,000 / 8,540	33 / 4,400 / 14,500	68 / 5,939 / 14,400	128 / 11,818 / 111,000
Pricing time [% Running time]	91 / 97 / 100	92 / 97 / 100	93 / 98 / 100	94 / 99 / 100
No. of subproblems	1 / 122.9 / 1,403	1 / 882.6 / 6,015	1 / 446.1 / 2,359	1 / 361.3 / 1,549
No. of (E)SPPRCs per subpr.	21 / 60 / 160	12 / 51 / 156	12 / 46 / 156	10 / 50 / 188
No. of labels per (E)SPPRC	3,591 / 8,387 / 48,737	4,118 / 10,180 / 45,200	7,454 / 16,107 / 36,591	11,351 / 48,625 / 461,214
No. of generated variables	265 / 2,712 / 28,870	306 / 9,555 / 32,111	426 / 8,775 / 28,342	625 / 5,340 / 16,458
Highest level in tree	1 / 8.3 / 31	1 / 16.7 / 80	1 / 25.8 / 202	1 / 20.6 / 107
No. of tours	3 / 3.8 / 4	3 / 4.03 / 5	3 / 4.2 / 5	4 / 4.73 / 6
No. of LTC tours	2 / 3.13 / 4	3 / 3.67 / 5	3 / 3.87 / 5	3 / 4.47 / 6
Longest Tour [No. of arcs]	7 / 8.93 / 11	7 / 9.9 / 13	8 / 10.37 / 13	8 / 10.23 / 14

Table 5.16: Computational results for heuristic algorithm (cont.)

Pricing problem strategy	Unidirectional	Bidirectional	ISSA	Heuristic
No. of tried / feasible / optimal	240 / 219 / 168	240 / 229 / 175	187 / 180 / 151	240 / 240 / 164
No. solved optimally at root	98	105	98	98
Running time [s]	1 / 3,898 / 45,900	1 / 3,899 / 52,900	1 / 2,858 / 25,200	1 / 2,921 / 111,000
Pricing time [% Running time]	77 / 96 / 100	79 / 97 / 100	83 / 98 / 100	63 / 94 / 100
No. of subproblems	1 / 73.5 / 1,533	1 / 55.1 / 1,533	1 / 45.5 / 1,535	1 / 245.2 / 6,015
No. of (E)SPPRCs per subpr.	14 / 61 / 200	12 / 64 / 200	20 / 75 / 208	10 / 57 / 188
No. of labels per (E)SPPRC	268 / 22,839 / 160,782	304 / 10,248 / 291,166	365 / 24,455 / 163,704	242 / 11,560 / 461,214
No. of generated variables	46 / 1,251 / 23,879	46 / 2,118 / 28,342	46 / 825 / 17,510	46 / 3,638 / 32,111
Highest level in tree	1 / 8.1 / 131	1 / 6.7 / 78	1 / 5.6 / 32	1 / 11.1 / 202

Table 5.17: Comparison of computational results for branch-and-price algorithm

The most important observations to be made in Tables 5.9–5.17 are:

- The difficulty of the instances varies widely. For example, the shortest running time for the unidirectional strategy for a 7_7_7 instance is 22 seconds, the longest is 12,300 seconds (which is more than 500 times longer).
- About 40 % of the instances were solved optimally without branching.
- The pricing step is by far the most time-consuming part in all strategies.
- The number of (E)SPPRCs per subproblem, i.e., the number of column generation iterations, is rather high. The numbers indicated in the lines 'No. of labels per (E)SPPRC' must be divided by four, because at each iteration, there are four different (E)SPPRCs that have to be solved, one for each vehicle (class). Still, the average number of (E)SPPRCs per subproblem ranges between 15 and 19.

- The unidirectional strategy is the best exact strategy for the instance types up to 7_7_7. It is the fastest and computes optimal solutions to more instances than the other two exact strategies.
- The bidirectional strategy is superior for the large instance types. It is faster, computes feasible solutions to more instances, and solves more instances to optimality than the other two exact strategies. With the exception of the 3_3_3 instances, it always creates fewer labels than the other two exact strategies; for the 10_10_10 instances, it even creates fewer labels than the heuristic strategy. Profiling of the code has shown that the reason why the bidirectional strategy is slower for the small instances is the time-consuming join operation. (Without the additional join test described on page 132, the join operation would have taken much longer still.) For the large instances, this effect is overcompensated by the lower number of created labels. These results are in accordance with the findings of Salani 2005.
- The incremental state space augmentation strategy is clearly the worst strategy. The number of (E)SPPRCs that are solved per subproblem and the number of created labels are much higher than with the other strategies. These results are contrary to the findings of Salani 2005 and Boland et al. 2006. A possible explanation is that the absence of time windows in the test instances leads to excessive cycling.
- The performance of the heuristic strategy is very good. Feasible solutions are computed for all instances, and the largest gap between the heuristic and the optimal solution is less than 0.25 %. A pairwise comparison of running times shows that the heuristic strategy is about 1.8 times as fast as the unidirectional strategy and twice as fast as the bidirectional strategy, but more than 95 % of the instances are solved to optimality. Due to the time limits, there are no fewer than 50 instances where the heuristic obtained a better solution than the unidirectional strategy.

5.5 Conclusions

The chapter has presented an arc-variable-based and a path-variable-based formulation for the TTRP including some extensions not yet covered in the literature, most notably, optional parking and transshipment locations. Moreover, the chapter has described the first two exact solution procedures for the problem, a branch-and-cut and a branch-and-price algorithm.

The decisive advantages of the branch-and-price algorithm over the branch-and-cut algorithm are:

- The pricing problems can be solved exactly on a smaller network than the one on which the arc variable formulation is based.
- Fleet planning (deciding on the number of vehicles of each available type that should be used) and vehicle routing can be performed simultaneously without additional modelling or computational effort.

Extensive computational experiments with implementations of both algorithms have been performed. The tests have been executed on randomly generated instances structured to resemble real-world situations. The experiments have shown that the branch-and-price algorithm is clearly superior to the branch-and-cut algorithm. The former is able to solve instances which are one order of magnitude larger than those solvable by the latter. However, even with a heuristic algorithm based on branch-and-price, only instances considerably smaller than typical real-world instances can be solved.

The largest instances that could be solved to optimality comprised 20 customers, 20 transshipment locations, four types of vehicle, and no time windows. The following quote from Fukasawa et al. 2006, p. 492, on the VRPTW with homogeneous fleet, puts these computational results into perspective: ‘It should be noted that . . . column generation has been the dominant approach for the Vehicle Routing Problem with Time Windows (VRPTW). Current branch-and-price algorithms can consistently solve tightly

constrained instances (those with narrow time windows) with up to 100 clients. However, they often fail on less constrained instances with only 50 clients.' This shows that the developed implementation is on a par with existing branch-and-price codes.

This chapter has provided an answer to the question posed in Section 4.3.2.6, 'which aspects of the VRPTT can be considered in a model that can still be solved exactly?': When comparing the arc variable formulations of the VRPTT and the TTRP, the reader cannot fail to notice that the VRPTT formulation is more elegant and easier to understand. However, comparing the path variable formulations of the problems and the growth rates of the underlying networks with increasing number of customers, transshipment locations, and vehicles, it becomes evident that the step from a fixed to a free lorry-trailer assignment constitutes the decisive leap in complexity. The synchronization constraints for lorries and trailers are apparently a point where matters become really difficult. Conversely, the TTRP is on a level of complexity that may still be considered tractable by exact algorithms.

Chapter 6

Possible Lines of Research

None of the three considered generalized routing problems could be treated exhaustively; much remains to be investigated. This chapter describes interesting lines of research that could not be pursued in the time available for this paper.

6.1 The GDRPP

6.1.1 Alternative Formulations

In the computational experiments for the GDRPP, only formulation (3.6) was solved by branch-and-cut. The other two formulations, (3.3) and (3.5), could be tested as well.

6.1.2 Improvements of Branch-and-Cut

The branch-and-cut algorithm for formulation (3.6) still leaves much room for improvement. Most importantly, a thorough polyhedral investigation should be made to identify further valid inequalities, and for the GDRPP (contrary to the VRPTT and the TTRP), it is still promising to try to prove which of them induce facets. Additional valid inequalities could be derived (generalized), for example, from the known cuts for the GATSP, the WRPP (taking into account that an arc is a special kind of windy link), or the so-called road travelling salesman problem (RTSP) (Fleischmann 1985). Recent theoretical papers on the WRPP and the windy general routing problem (WGRP) are Corberán et al. 2003, Corberán et al. 2004, Corberán et al. 2005a, Corberán et al. 2005b. The branch-and-cut algorithm could also benefit from a tailored branching strategy (e.g., branch on the δ_i variables first).

6.1.3 Further Possibilities

There is not much literature on the DRPP (Campos/Savall 1995 present a cutting plane algorithm), so that it would be worthwhile to make experiments with formulation (3.1), too.

In addition, the two direct formulations for the WRPPTP should be compared theoretically and empirically. Moreover, of the three possibilities to transform a graph with turn penalties into one without, only two were used in the computational experiments. The 'dual graph' approach should also be tried.

For instances with many zigzag links, it would be worthwhile to try to exploit the special structure of the resulting two non-disjoint r-groups in a solution algorithm.

The computational results showed that the solution quality of the heuristics is not very good. Additional local search improvement procedures and/or the use of metaheuristics should be considered. This could be helpful for the branch-and-cut algorithm, too.

When solving the GDRPP exactly or heuristically by a labelling algorithm, it is easily possible to consider resource constraints, most notably time windows. ARPs with time windows are rarely addressed in the literature (see, for example, Dror/Langevin 2000), despite the fact that time windows are just as relevant in arc routing applications as they are relevant in TSP or VRP applications. The exact labelling algorithm should be able to solve much larger instances when tight time windows are given.

As shown above, it is easy to transform the GDRPP into the GATSP. This means that all considered problems can also be solved as GATSPs. It would be interesting to examine for which problem which transformation (into GDRPP or into GATSP) is more easily solved by branch-and-cut and to compare the branch-and-cut algorithm for the GDRPP to existing exact GATSP procedures.

6.2 The VRPTT

6.2.1 Alternative Formulations

The turn variable formulation should also be restricted to the core problem, solved by branch-and-cut, and compared to the arc variable formulation. In addition, the discretized formulations should be implemented and tested empirically. In particular, experiments should be made with different load transfer amounts (cf. Del Pia/Filippi 2006).

Two difficulties with the MIP formulations for the VRPTT presented in Chapter 4 are that the networks on which the formulations are based quickly become intractably large with increasing instance size and that the formulations contain a lot of logical constraints that must be linearized by using large constants M. (This is the main reason why the LP relaxation of (4.59)–(4.62) is so weak.) The branch-and-price approach offers only a partial remedy: Several implications are pricing problem constraints in the path variable formulation. When the pricing problems are solved by a labelling algorithm, these implications are represented in the REFs, and the Ms are essentially removed. However, there are still many implications in the master problem. A technique that takes a different approach to problem representation in general and in particular to logical constraints is constraint programming (Lustig/Puget 2001, Hooker 2002, Van Hentenryck 2002). Constraint programming is able to directly represent and solve logical implications without having to take the indirection of M constants, and it may also offer ways of representing the load transfer logic more efficiently. So, constraint programming is an alternative for the (exact as well as heuristic) solution of the VRPTT that should be seriously explored.

In recent years, there have been efforts to integrate mixed integer programming and constraint programming in order to combine the strengths of both methods. A paper that demonstrates the viability of the constraint programming approach for the solution of problems with many logical constraints is Codato /Fischetti 2006. The authors use so-called *combinatorial Benders' cuts* to get rid of the M constants in MIP formulations. They report very promising computational results for problems known from the literature and show that their approach clearly outperforms CPLEX for such problems.

6.2.2 Improvements of Branch-and-Cut

The lower bounds in the branch-and-cut algorithm are still very weak and should be improved. Several possibilities exist for finding additional cuts. Many different classes of valid inequalities for the symmetric VRP without time windows are known (cf. Lysgaard et al. 2004, Fukasawa et al. 2006). In principle, such inequalities should also be applicable to the VRPTT (Grünert/Irnich 2005b, p. 427). For the asymmetric VRPTW, Kallehauge et al. 2005, p. 85 ff., describe different valid inequalities and mention references where these inequalities were used in branch-and-cut or branch-and-price-and-cut algorithms. It would be very interesting to investigate further how all these inequalities, in particular, the κ-path cuts, can effectively and efficiently be generalized for the VRPTT, or, at least, for VRPs with heterogeneous fleet.

Upper bounds are useful, too. Such bounds could be computed by solving the corresponding TTRP exactly or heuristically, or by a heuristic for the VRPTT.

The number of variables and constraints in the arc variable formulation, although polynomial in the number of customers, transshipment locations, and vehicles, increases sharply with increasing number of these objects. Therefore, it would be worthwhile to try to use the concept of *lazy constraints* to speed up the solution process. Contrary to the 'real' cuts described in Section 4.5.1 (inequalities such as the supply collection and the lorry flow cut), lazy constraints are irredundant but unlikely to be violated by an optimal solution to the LP relaxation of the formulation without them. Hence, they may be omitted when the LP relaxation is solved, and they must be checked for violation only afterwards. All violated lazy constraints must then be added to the LP relaxation, and the latter must be re-solved. This procedure must be repeated until no more lazy constraints are violated, in which case the solution to this 'restricted' LP relaxation is an optimal solution to the 'complete' LP relaxation. Constraints such as (4.60e) and (4.60h)–(4.60j) account for a considerable number of all constraints in the VRPTT arc variable formulation. Such constraints are promising lazy constraint candidates, as their omission might significantly speed up the solution of the LP relaxation.

6.2.3 Improvements of Branch-and-Price

The first and decisive 'improvement' on the branch-and-price approach would be to actually implement it. At first, the pricing problems could be solved by branch-and-cut. To keep the subproblem networks as small as possible, the problems should at first be solved with $n^{TS} = 2$ or $n^{TS} = 3$ until no more negative reduced cost paths are found. Then, n^{TS} could be increased in the respective subnetworks for those transshipment locations which are actually used. If the results are promising, tackling the pricing problems by a labelling algorithm could be tried.

Generally, solving ESPPRCs by branch-and-cut is an interesting approach in itself. There are practically no publications in this field. Starting with the ESPPTW with negative cycles, valid inequalities for ESPPRCs should be investigated.

The (E)SPPRC is similar to a multi-criteria optimization problem. Research in this field is becoming more and more intense (cf. the survey Ehrgott/Gandibleux 2000). A thorough survey of the pertinent literature might lead to new insights for solving negative-cycle (E)SPPRCs.

6.2.4 Further Possibilities

Taking into account the results of the computational experiments in Chapter 4 and the practical relevance of the problem, what is most urgently needed is a good heuristic capable of solving realistic instances with hundreds of customers and dozens of transshipment locations.

In airline crew scheduling problems, the issue of *robustness* of solutions to modifications of the data is becoming increasingly important (cf. Lan et al. 2006). It is desirable to compute solutions that are relatively insensitive to unexpected events happening in practice (delays etc.). The flight plans should be such that as few other flights, respectively, passengers, as possible are affected when one flight is late. The solutions to VRPTTs are also susceptible to such effects. When trailers or support vehicles arrive late at transshipment locations, other vehicles are affected, and infeasibilities due to time window violations may occur. Hence, solutions to VRPTTs should be evaluated in this respect, and ideally, (heuristic) algorithms for solving practical problems should be constructed so as to favour 'robust' solutions. (In the TTRP, robustness of solutions (with respect to time) is not an issue. Different vehicles do not influence one another.)

The idea of discretizing the load transfer amounts maintains exactness of the formulation in cases where the goods to be collected at (or delivered to) customers are not homogeneous. For example, there are

many applications where the goods consist in swap-body platforms or containers. The lorries and trailers used for such transports each have capacity one, i.e., they are able to transport one container at a time, which makes load synchronization between lorry and trailer trivial. Such problems constitute a very promising application area for VRPTT models and algorithms.

Another application area for VRPTT models or, at least, for models considering support vehicles, is bimodal traffic road/rail. Trains cannot usually visit customers and can therefore be modelled as support vehicles. Trains normally have fixed schedules, so at least the temporal synchronization between collection vehicles (lorries and trailers) and support vehicles (trains) is easy.

A potential extension of the VRPTT is the pickup-and-delivery problem with trailers and transshipments (PDPTT). This is particularly interesting for networks of less-than-truckload forwarders, where it is common to use transshipment locations to transfer consignments between feeder, long-haul, and distribution vehicles.

It is possible to consider more complex synchronization requirements. For example, when drivers are also considered, a trailer needs two other objects to be able to move in space, a lorry and a driver, so that three types of object have to be synchronized. An even more complex situation is the transport of consignments which must be transported in swap-body platforms. These platforms may only be transported by trailers, which in turn must be pulled by lorries, which must be operated by drivers. In this case, five different types of object must be synchronized.

The concept of a *turn* in a graph was originally used in arc routing applications modelling turn restrictions and prohibitions in real road networks. As shown in Chapter 4, turn variables can also be sensible if there are no such restrictions. It would be very interesting to evaluate the benefits of formulations using turn variables for other types of routing problem.

The above formulations for the VRPTT consider the underlying real-world problem from a routing perspective. It may also be interesting to view the problem from a scheduling perspective, where there are jobs to be performed (customers to be served) and the assignment of these jobs to machines (lorries) and the sequence in which each machine performs its assigned jobs are to be determined. To perform a job, a machine needs resources (loading capacity). If a machine does not have enough resources to perform a job, it must use auxiliary resources (trailers).

The VRPTT is a concrete case of an abstract 'time-constrained vehicle routing and scheduling problem with multiple synchronization (or multiple logically coupling) constraints'. These synchronization constraints contain resource variables, thus making a branch-and-price approach difficult. It would be interesting to study this class of problem on a more abstract level and to examine more closely how it fits into the unified model of Desaulniers et al. 1998.

6.3 The TTRP

6.3.1 Alternative Formulations

In the arc variable formulation, it would be possible to use x_{ij}^k variables that may take the three values 0, 1, and 2, where $x_{ij}^k = 2$ means that LTC lorry k traverses arc (i, j) with its trailer attached. This would make the y_{ij}^k variables obsolete. However, it is not immediately clear whether this would have been a better approach (for example, the consideration of the trailer costs in the objective function is not so straightforward then).

As explained above, the formulation given by Scheuerer 2004 does not use logical implications and therefore does not need any M constants. It would be interesting to implement this formulation and to perform computational experiments to see how it compares with the arc variable formulation presented here.

The networks for TTRPs do not grow as fast as those for the VRPTT. Moreover, all logical constraints in the TTRP are pricing problem constraints and are represented in the REFs, so that no M issue arises in the branch-and-price approach. Therefore, a constraint programming approach is not so interesting here, but if such an approach proves successful for the VRPTT, it could also be tried for the TTRP.

6.3.2 Improvements of Branch-and-Cut

The comments in the corresponding section on the VRPTT apply here analogously. Additionally, it must be mentioned that, contrary to CPLEX, the ABACUS framework does not include general cuts like Gomory cuts or flow cover cuts. Such cuts were automatically added by CPLEX in the computational experiments for the VRPTT, and they would have increased the solution speed of the branch-and-cut algorithm for the TTRP, too.

6.3.3 Improvements of Branch-and-Price

The implementation of the branch-and-price algorithm presented in the previous chapter is more than a bare proof of concept. However, given the current state of research on column generation and branch-and-price, it is simply impossible for an individual to include all currently known acceleration techniques and refinements in a code within a reasonable amount of time. Therefore, the implementation can still be improved significantly. The most important refinements are:

- The inclusion of stabilization in the column generation process (Lübbecke/Desrosiers 2005, p. 1017 ff.).
  One problem with column generation algorithms in general is the so-called *tailing-off effect*, which means a slow convergence at the end of the process. This can be due to several reasons; the effect, however, is not yet completely understood theoretically. Nevertheless, there are several possibilities for speeding up the procedure. The simplest method is to add more than one column to the master problem at a time. This has been done in the algorithm of Chapter 5. Another option is to use interior point methods for the solution of the master problem. This has been tried for the TTRP algorithm, but with little effect. More sophisticated techniques are described in Lübbecke/Desrosiers 2005 (ib.). The theory as well as the implementation of such techniques is non-trivial. The number of (E)SPPRCs solved at each vertex of the branch-and-price tree is an indicator for the potential usefulness of stabilization. As the computational experiments showed, this number was rather high, so that a powerful stabilization algorithm should be the first improvement made in the TTRP branch-and-price algorithm.

- The use of a primal heuristic to quickly obtain good upper bounds.
  The procedures by Chao 2002 and Scheuerer 2004 could be used for this purpose.

- The development of better lower bounding procedures.
  Any progress in the research on valid inequalities for the VRPTT and the TTRP could possibly also be put to good use in the TTRP branch-and-price algorithm, which might then be augmented to become a branch-and-price-and-cut procedure.

A possible improvement on the dominance check in the labelling algorithm for the solution of the pricing problems is multidimensional divide-and-conquer (cf. Kung et al. 1975, Bentley 1980). This algorithm, respectively, algorithmic paradigm, is particularly hard to implement if there are non-integer resources, as is the case in branch-and-price algorithms because of the dual prices. On 64-bit computers and with compilers with 64-bit integer types, it would be interesting to transform the dual prices from `double` to `int`, so that a numerically stable implementation is possible. The complexity of multidimensional divide-and-conquer increases logarithmically with the number of resources, so if there are visitation counters, it is not clear whether the procedure will lead to performance improvements.

Besides, the solution of the pricing problems may still be accelerated by using further techniques for heuristic pricing. It may also be interesting to test the heuristic approach with the bounded bidirectional dynamic programming algorithm instead of the unidirectional version. Moreover, the pricing problems could be solved by branch-and-cut just to see how this approach compares with the labelling algorithm and to see whether it is sensible to try to solve the VRPTT pricing problems by branch-and-cut.

6.3.4 Further Possibilities

For both free and fixed lorry-trailer assignments, i.e., for both the VRPTT (or rather, the VRP with trailers and without transshipments, VRPT~~T~~ or VRPT) and the TTRP, it would also be interesting to consider the case where transshipments are impossible or forbidden. It may be technically impossible because the technical equipment for loading and unloading a good may not be available outside plants and warehouses. The customer may forbid it, because the goods to be transported may be very fragile. The law may forbid it (e.g., customs-sealed vehicles in international traffic, livestock haulage, transport of hazardous materials). Without transshipments, support vehicles are not useful, but trailers still may be.

Several possibilities for the collection of the supplies are possible. It could be required that the supply of a trailer customer be entirely loaded onto a trailer, or that it be entirely loaded on either lorry or trailer, or it could still be allowed to split it arbitrarily between lorry and trailer. In any case, capacity constraints for the lorries as well as for the trailers are necessary. Depending on the requirements for the collection of trailer customer supplies, also constraints restricting the total vehicle load may be needed.

When trailer customer supplies must be entirely loaded onto trailers, in both the VRPT and the TTRP, there are no more load synchronization constraints. However, in the VRPT, the constraints for temporal synchronization at parking locations/vertices and for routing synchronization remain relevant.

6.4 Location-Routing Problems

As already mentioned in Chapter 4, *location-routing problems* (LRPs, cf. Engele 1980, Laporte 1988, Daskin 1995, p. 339 ff., Nagy/Salhi 1996, Albareda-Sambola et al. 2005) combine facility location with vehicle routing. LRPs simultaneously address the following five questions (Daskin 1995, p. 339 f.):

(i) How many facilities (depots, warehouses, factories) should be located (opened, built, rented)?

(ii) Where should these facilities be located?

(iii) Which customers (regional warehouses, retailers) should be assigned to which facility?

(iv) Which customers should be visited on one and the same vehicle route starting and ending at a certain facility?

(v) In what sequence should the customers on one route be visited?

LRPs come in several variants. Deterministic, single period LRPs can be categorized by the following four criteria (cf. Albareda-Sambola et al. 2005, p. 408):

(i) the type of the facilities to be located (are they allowed to be origins and destinations of vehicle routes or are they only intermediate depots?),

(ii) whether the facilities are capacitated or not,

(iii) whether facilities and/or customers have time windows or not, and

(iv) whether the vehicles are capacitated (or, more generally, homogeneous) or not.

There are also dynamic and stochastic types of LRP (cf. ib.).

In Chapters 4 and 5, network representations for the VRPTT and the TTRP were presented. Interestingly, a similar network was used in Laporte et al. 1988 for the solution of a location-routing problem with capacitated facilities, no time windows, and capacitated vehicles, where the facilities are allowed to be origins and destinations of vehicle routes. (The facility capacities are given indirectly by specifying the maximal number of vehicle tours that may start and end at a facility.)

All of the sixteen types of LRP that can be distinguished by the above four criteria can be modelled as VRPTTs or TTRPs, and there are several options how to do this.

The main idea is to consider the trailers as abstract, virtual objects for which there are no corresponding objects in the real world. Trailers are used as modelling tools that serve as the connection between the vehicle(s) (the lorry/lorries) and the potential facilities (the transshipment locations). The visited transshipment locations correspond to the facilities to be opened. All customers are lorry customers.

The four criteria of the above taxonomy can be considered as follows in a VRPTT or a TTRP:

(i) One central depot is created where all single lorries and all LTCs start and end their tours. In the case where the facilities to be located are allowed to be origins and destinations of vehicle routes, this depot is a virtual location with a distance of zero to and from any other location. The fixed costs of opening a facility are added to the trailer costs on arcs entering the corresponding decoupling vertex. The costs of returning to the virtual depot are set to zero. Moreover, when the lorries must be empty when they return to the central depot, i.e., when the complete supply of the customers must be transferred into a trailer at a transshipment location, the load transfer variables of the lorries are fixed accordingly.

(ii) The (finite or infinite) capacities of the potential facilities are represented by the capacities of the trailer(s). When the capacities of the potential facilities are finite, there is one LTC for each transshipment location (and, hence, for each potential facility), and the capacity of the LTC trailer is equal to the capacity of the corresponding potential facility. If all potential facilities have unlimited capacity (and if there are no time windows), there is only one lorry-trailer combination with a trailer with unlimited capacity. Symmetries concerning the sequence of the visited transshipment locations can then be avoided by assigning artificial pairwise disjoint time windows to the transshipment locations, thus fixing the sequence in which they may be visited.

(iii) Time windows for potential facilities as well as for customers carry over naturally to the transshipment locations and customers. However, if there are time windows, there must be a sufficient number of single lorries which may transfer load to the trailers corresponding to the potential facilities. Hence, to consider time windows, a VRPTT model must be used. Accessibility constraints must then make sure that each single lorry is used only at one facility/transshipment location. To model the case where the potential facilities have time windows and are the origins and destinations of vehicle routes, the costs of a single lorry on an arc from the virtual depot to a customer in the VRPTT network is set to the costs of this lorry from its assigned potential facility to the respective customer.

(iv) Capacities for the vehicles in the LRP are represented by the lorry capacities in the VRPTT or TTRP. Again, accessibility constraints are used to ensure that a certain vehicle is used only at the desired facility.

The above description also applies to the VRPTT when the LTC is replaced by a support vehicle or even by a single trailer/when the LTCs are replaced by support vehicles or even by single trailers without fixed lorry-trailer assignment.

The presented transformations of LRPs constitute a promising application area for TTRPs and VRPTTs. If only one LTC is necessary, the LRP can be solved with a labelling algorithm. The difficulty with the solution of the resulting TTRPs or VRPTTs by branch-and-cut is that n^{TS} then determines the maximal number of tours/vehicles for each potential facility in the LRP. When the capacities of the potential facilities are finite, and when solving the resulting TTRPs or VRPTTs by branch-and-price, there is one subproblem per potential facility. For instances where the number of potential facilities is similar to the number of customers, an undesirably large number of subproblems must be solved. Therefore, judging by the results of the computational experiments, LRPs with uncapacitated potential facilities, uncapacitated, homogeneous vehicles, and no time windows should be well solvable as TTRPs or VRPTTs.

Final Remark

On his first day as a university student, the author attended an introductory lecture given by the then Dean of the Economics Faculty at the University of Augsburg. The Dean told his audience that a good student is not only one who has learned to answer a lot of questions, but also one who has learned to ask a lot of (interesting) questions.

It is the author's conviction that both the problems treated in this paper, and the treatment itself, have left open and thrown up more questions than could possibly be answered here, and it is his hope that this paper will be helpful in sparking off interest among its readers to concern themselves further with turns, trailers, and transshipments.

Acknowledgements

Thanks are due to Jacques Desrosiers for helpful hints on the discretized VRPTT, to Christine Stibbe and Jens Wollenweber for proofreading the manuscript, to all people at the Deutsche Post Endowed Chair of Optimization of Distribution Networks for being colleagues in the best sense of the word and providing the best possible working atmosphere, and in particular to Stefan Irnich for many lively discussions and invaluable advice.

Bibliography

Aho, A.; Hopcroft, J.; Ullman, J. (1983):
Data Structures and Algorithms
Addison-Wesley, Reading

Ahuja, R.; Magnanti, T.; Orlin, J. (1993):
Network Flows
Prentice-Hall, Upper Saddle River

Albareda-Sambola, M.; Díaz, J.; Fernández, E. (2005):
A Compact Model and Tight Bounds for a Combined Location-Routing Problem
Computers & Operations Research, vol. 32, pp. 407–428

Añez, J.; de la Barra, T.; Pérez, B. (1996):
Dual Graph Representation of Transport Networks
Transportation Research Part B, vol. 30, pp. 209–216

Arkin, E.; Bender, M.; Demaine, E.; Fekete, S.; Mitchell, J.; Sethia, S. (2005):
Optimal Covering Tours with Turn Costs
SIAM Journal on Computing, vol. 35, pp. 531–566

Assad, A.; Golden, B. (1995):
Arc Routing Methods and Applications
in:
Ball, M.; Magnanti, T.; Monma, C.; Nemhauser, G. (eds.):
Network Routing
Handbooks in Operations Research and Management Science, vol. 8
Elsevier, Amsterdam
pp. 375–483

Bamberg, G.; Baur, F. (1989):
Statistik
Oldenbourg, München

Barnhart, C.; Johnson, E.; Nemhauser, G.; Savelsbergh, M.; Vance, P. (1998):
Branch-and-Price: Column Generation for Solving Huge Integer Programs
Operations Research, vol. 46, pp. 316–329

Beasley, J.; Christofides, N. (1989):
An Algorithm for the Resource Constrained Shortest Path Problem
Networks, vol. 19, pp. 379–394

Benavent, E.; Corberán, A.; Sanchis, J. (2000):
Linear Programming Based Methods for Solving Arc Routing Problems
in:
Dror, M. (ed.):
Arc Routing: Theory, Solutions, and Applications

Kluwer, Boston
pp. 231–275

Benavent, E.; Soler, D. (1999):
The Directed Rural Postman Problem with Turn Penalties
Transportation Science, vol. 33, pp. 408–418

Bentley, J. (1980):
Multidimensional Divide-and-Conquer
Communications of the ACM, vol. 23, pp. 214–229

Blais, M.; Laporte, G. (2003):
Exact Solution of the Generalized Routing Problem Through Graph Transformations
Journal of the Operational Research Society, vol. 54, pp. 906–910

Bodin, L.; Kursh, S. (1978):
A Computer-Assisted System for the Routing and Scheduling of Street Sweepers
Operations Research, vol. 26, pp. 525–537

Bodin, L.; Kursh, S. (1979):
A Detailed Description of a Computer System for the Routing and Scheduling of Street Sweepers
Computers & Operations Research, vol. 6, pp. 181–198

Bodin, L.; Levy, L. (2000):
Scheduling of Local Delivery Carrier Routes for the United States Postal Service
in:
Dror, M. (ed.):
Arc Routing: Theory, Solutions, and Applications
Kluwer, Boston
pp. 419–442

Boland, N.; Dethridge, J.; Dumitrescu, I. (2006):
Accelerated Label Setting Algorithms for the Elementary Resource Constrained Shortest Path Problem
Operations Research Letters, vol. 34, pp. 58–68

Bousonville, T.; Kopfer, H. (2000):
A Hybrid Evolutionary Algorithm for the Mixed Rural Postman Problem with Turn Penalties
Technical Report, Bremer Institut für Betriebstechnik und angewandte Arbeitswissenschaft, Universität Bremen

Bramel, J.; Simchi-Levi, D. (1997):
The Logic of Logistics
Springer, New York

Bramel, J.; Simchi-Levi, D. (2002):
Set-Covering-Based Algorithms for the Capacitated VRP
in:
Toth, P.; Vigo, D. (eds.):
The Vehicle Routing Problem
SIAM Monographs on Discrete Mathematics and Applications, Philadelphia
pp. 85–108

Brunswicker, J. (1986):
Optimale Standort- und Tourenplanung für die Rohmilcherfassung eines Molkereibetriebes
Lit, Münster

Cabral, E.; Gendreau, M.; Ghiani, G.; Laporte, G. (2004):
Solving the Hierarchical Chinese Postman Problem as a Rural Postman Problem
European Journal of Operational Research, vol. 155, pp. 44–50

Caldwell, T. (1961):
On Finding Minimum Routes in a Network with Turn Penalties
Communications of the ACM, vol. 4, pp. 107–108

Campos, V.; Savall, J. (1995):
A Computational Study of Several Heuristics for the DRPP
Computational Optimization and Applications, vol. 4, pp. 67–77

Chabrier, A. (2006):
Vehicle Routing Problem with Elementary Shortest Path Based Column Generation
Computers & Operations Research, vol. 33, pp. 2972–2990

Chao, I. (2002):
A Tabu Search Method for the Truck and Trailer Routing Problem
Computers & Operations Research, vol. 29, pp. 33–51

Chisman, J. (1975):
The Clustered Traveling Salesman Problem
Computers & Operations Research, vol. 2, pp. 115–119

Chvátal, V. (1983):
Linear Programming
Freeman, New York

Clossey, J.; Laporte, G.; Soriano, P. (2001):
Solving Arc Routing Problems with Turn Penalties
Journal of the Operational Research Society, vol. 52, pp. 433–439

Codato, G.; Fischetti, M. (2006):
Combinatorial Benders' Cuts for Mixed-Integer Linear Programming
Operations Research, vol. 54, pp. 756–766

Corberán, A.; Martí, R.; Martínez, E.; Soler, D. (2002):
The Rural Postman Problem on Mixed Graphs with Turn Penalties
Computers & Operations Research, vol. 29, pp. 887–903

Corberán, A.; Plana, I.; Sanchis, J. (2004):
Zigzag Inequalities: A New Class of Facet-Inducing Inequalities for Arc Routing Problems
Technical Report TR04-2004, Departamento de Estadística e Investigación Operativa, Universidad de Valencia

Corberán, A.; Plana, I.; Sanchis, J. (2005a):
A Branch & Cut Algorithm for the Windy General Routing Problem
Technical Report TR04-2005, Departamento de Estadística e Investigación Operativa, Universidad de Valencia

Corberán, A.; Plana, I.; Sanchis, J. (2005b):
The Windy General Routing Polyhedron: A Global View of Many Known Arc Routing Polyhedra
Technical Report TR06-2005, Departamento de Estadística e Investigación Operativa, Universidad de Valencia

Corberán, A.; Romero, A.; Sanchis, J. (2003):
The Mixed General Routing Polyhedron
Mathematical Programming, Series A, vol. 96, pp. 103–137

Cordeau, J.; Desaulniers, G.; Lingaya, N.; Soumis, F.; Desrosiers, J. (2001a):
Simultaneous Locomotive and Car Assignment at VIA Rail Canada
Transportation Research Part B, vol. 35, pp. 767–787

Cordeau, J.; Soumis, F.; Desrosiers, J. (2000):
A Benders Decomposition Approach for the Locomotive and Car Assignment Problem
Transportation Science, vol. 34, pp. 133–149

Cordeau, J.; Soumis, F.; Desrosiers, J. (2001b):
Simultaneous Assignment of Locomotives and Cars to Passenger Trains
Operations Research, vol. 49, pp. 531–548

Cordeau, J.; Toth, P.; Vigo, D. (1998):
A Survey of Optimization Models for Train Routing and Scheduling
Transportation Science, vol. 32, pp. 380–404

Dantzig, G. (1963):
Linear Programming and Extensions
Princeton University, Princeton

Daskin, M. (1995):
Network and Discrete Location
Wiley, New York

Del Pia, A.; Filippi, C. (2006):
A Variable Neighborhood Descent Algorithm for a Real Waste Collection Problem with Mobile Depots
International Transactions in Operational Research, vol. 13, pp. 125–141

Desaulniers, G.; Desrosiers, J.; Ioachim, I.; Solomon, M.; Soumis, F.; Villeneuve, D. (1998):
A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems
in:
Crainic, T.; Laporte, G. (eds.):
Fleet Management and Logistics
Kluwer, Boston
pp. 57–93

Desaulniers, G.; Villeneuve, D. (2000):
The Shortest Path Problem with Time Windows and Linear Waiting Costs
Transportation Science, vol. 34, pp. 312–319

Desrosiers, J.; Lübbecke, M. (2005):
A Primer in Column Generation
in:
Desaulniers, G.; Desrosiers, J.; Solomon, M. (eds.):
Column Generation
Springer, New York
pp. 1–32

Domschke, W. (1995):
Logistik: Transport
Oldenbourg, München

Domschke, W. (1997):
Logistik: Rundreisen und Touren
Oldenbourg, München

Drexl, M. (2005a):
IP Formulations for Postman Problems
Technical Report 08/2005, Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Rheinisch-Westfälische Technische Hochschule Aachen

Drexl, M. (2005b):
On the Generalized Directed Rural Postman Problem
Technical Report 10/2005, Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Rheinisch-Westfälische Technische Hochschule Aachen

Drexl, M. (2005c):
The Vehicle Routing Problem with Trailers and Transshipments
Technical Report 09/2005, Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Rheinisch-Westfälische Technische Hochschule Aachen

Drexl, M. (2006):
A Branch-and-Price Algorithm for the Truck-and-Trailer Routing Problem
Technical Report 27/2006, Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Rheinisch-Westfälische Technische Hochschule Aachenp

Dror, M. (1994):
Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW
Operations Research, vol. 42, pp. 977–978

Dror, M. (2000a):
Arc Routing: Complexity and Approximability
in:
Dror, M. (ed.):
Arc Routing: Theory, Solutions, and Applications
Kluwer, Boston
pp. 133–169

Dror, M. (ed.) (2000b):
Arc Routing: Theory, Solutions, and Applications
Kluwer, Boston

Dror, M.; Langevin, A. (1997):
A Generalized Traveling Salesman Problem Approach to the Directed Clustered Rural Postman Problem
Transportation Science, vol. 31, pp. 187–192

Dror, M.; Langevin, A. (2000):
Transformations and Exact Node Routing Solutions by Column Generation
in:
Dror, M. (ed.):
Arc Routing: Theory, Solutions, and Applications
Kluwer, Boston
pp. 277–326

Dror, M.; Trudeau, P. (1989):
Savings by Split Delivery Routing
Transportation Science, vol. 23, pp. 141–145

Edmonds, J.; Johnson, E. (1973):
Matching, Euler Tours and the Chinese Postman
Mathematical Programming, vol. 5, pp. 88–124

Eglese, R.; Letchford, A. (2000):
Polyhedral Theory for Arc Routing Problems
in:
Dror, M. (ed.):
Arc Routing: Theory, Solutions, and Applications
Kluwer, Boston
pp. 199–230

Ehrgott, M.; Gandibleux, X. (2000):
A Survey and Annotated Bibliography of multiobjective combinatorial optimization
OR Spektrum, vol. 22, pp. 425–460

Eiselt, H.; Gendreau, M.; Laporte, G. (1995a):
Arc Routing Problems, Part I: The Chinese Postman Problem
Operations Research, vol. 43, pp. 231–242

Eiselt, H.; Gendreau, M.; Laporte, G. (1995b):
Arc Routing Problems, Part II: The Rural Postman Problem
Operations Research, vol. 43, pp. 399–414

Engele, G. (1980):
Simultane Standort- und Tourenplanung
Heymanns, Köln

Feillet, D.; Dejax, P.; Gendreau, M.; Gueguen, C. (2004):
An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems
Networks, vol. 44, pp. 216–229

Fisher, M.; Jaikumar, R. (1981):
A Generalized Assignment Heuristic for Vehicle Routing
Networks, vol. 11, pp. 109–124

Fleischmann, B. (1985):
A Cutting Plane Procedure for the Travelling Salesman Problem on Road Networks
European Journal of Operational Research, vol. 21, pp. 307–317

Freling, R.; Huisman, D.; Wagelmans, A. (2003):
Models and Algorithms for Integration of Vehicle and Crew Scheduling
Journal of Scheduling, vol. 6, pp. 63–85

Fukasawa, R.; Longo, H.; Lysgaard, J.; Poggi de Aragão, M.; Reis, M.; Uchoa, E.; Werneck, R. (2006):
Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem
Mathematical Programming, Series A, vol. 106, pp. 491–511

Garey, M.; Johnson, D. (1979):
Computers and Intractability
Freeman, New York

Gendreau, M.; Dejax, P.; Feillet, D.; Gueguen, C. (2005):
Vehicle Routing with Time Windows and Split Deliveries
Draft Technical Report

Gerdessen, J. (1996):
Vehicle Routing Problem with Trailers
European Journal of Operational Research, vol. 93, pp. 135–147

Ghiani, G.; Improta, G. (2000):
An Algorithm for the Hierarchical Chinese Postman Problem
Operations Research Letters, vol. 26, pp. 27–32

Ghiani, G.; Laporte, G. (2001):
Location-Arc Routing Problems
Opsearch, vol. 38, pp. 151–159

Gillett, B.; Miller, L. (1974):
A Heuristic Algorithm for the Vehicle-Dispatch Problem
Operations Research, vol. 22, pp. 340–349

Golden, B.; Stewart, W. (1985):
Empirical Analysis of Heuristics
in:
Lawler, E.; Lenstra, J.; Rinnooy, A.; Shmoys, D. (eds.):
The Traveling Salesman Problem
Wiley, Chichester
pp. 207–249

Grünert, T.; Irnich, S. (2005a):
Optimierung im Transport
Band I: Grundlagen
Shaker, Aachen

Grünert, T.; Irnich, S. (2005b):
Optimierung im Transport
Band II: Wege und Touren
Shaker, Aachen

Guan, M. (1962):
Graphic Programming Using Odd or Even Points
Chinese Mathematics, vol. 1, pp. 273–277

Gutin, G.; Punnen, A. (eds.) (2002):
The Traveling Salesman Problem and Its Variations
Kluwer, Dordrecht

Haase, K.; Desaulniers, G.; Desrosiers, J. (2001):
Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems
Transportation Science, vol. 35, pp. 286–303

Hertz, A.; Mittaz, M. (2001):
A Variable Neighborhood Descent Algorithm for the Undirected Capacitated Arc Routing Problem
Transportation Science, vol. 35, pp. 425–434

Hoff, A.; Løkketangen, A. (2006):
A Tabu Search Approach for Milk Collection in Western Norway Using Trucks and Trailers
Technical Report, Molde University College

Hooker, J. (2002):
Logic, Optimization, and Constraint Programming
INFORMS Journal on Computing, vol. 14, pp. 295–321

Ioachim, I.; Desrosiers, J.; Soumis, F.; Bélanger, N. (1999):
Fleet Assignment and Routing with Schedule Synchronization Constraints
European Journal of Operational Research, vol. 119, pp. 75–90

Ioachim, I.; Gélinas, S.; Soumis, F.; Desrosiers, J. (1998):
A Dynamic Programming Algorithm for the Shortest Path Problem with Time Windows and Linear Node Costs
Networks, vol. 31, pp. 193–204

Irnich, S. (2002):
Netzwerk-Design für zweistufige Transportsysteme und ein Branch-and-Price-Verfahren für das gemischte Direkt- und Hubflugproblem
Ph.D. Thesis, Fakultät für Wirtschaftswissenschaften, Rheinisch-Westfälische Technische Hochschule Aachen

Irnich, S. (2005):
A Note on Postman Problems with Zigzag Service
INFOR, vol. 43, pp. 33–39

Irnich, S.; Desaulniers, G. (2005):
Shortest Path Problems with Resource Constraints
in:
Desaulniers, G.; Desrosiers, J.; Solomon, M. (eds.):
Column Generation
Springer, New York
pp. 33–65

Irnich, S.; Villeneuve, D. (2006):
The Shortest Path Problem with Resource Constraints and k-Cycle Elimination for $k \geq 3$
INFORMS Journal on Computing, vol. 18, pp. 391–406

Johnson, D.; Papadimitriou, C. (1985):
Computational Complexity
in:
Lawler, L.; Lenstra, J.; Rinnooy Kan, A.; Shmoys, D. (eds.):
The Traveling Salesman Problem
Wiley, Chichester
pp. 37–85

Kallehauge, B.; Larsen, J.; Madsen, O.; Solomon, M. (2005):
Vehicle Routing Problem with Time Windows
in:
Desaulniers, G.; Desrosiers, J.; Solomon, M. (eds.):
Column Generation
Springer, New York
pp. 67–98

Kirby, R. (1966):
A Minimum Path Algorithm for a Road Network with Turn Penalties
ARRB Proceedings, vol. 3, pp. 434–442

Kirby, R.; Potts, R. (1969):
The Minimum Route Problem for Networks with Turn Penalties and Prohibitions
Transportation Research, vol. 3, pp. 397–408

Kohl, N.; Desrosiers, J.; Madsen, O.; Solomon, M.; Soumis, F. (1999):
2-Path Cuts for the Vehicle Routing Problem with Time Windows
Transportation Science, vol. 33, pp. 101–116

Kung, H.; Luccio, F.; Preparata, F. (1975):
On Finding the Maxima of a Set of Vectors
Journal of the Association for Computing Machinery, vol. 23, pp. 469–476

Lacomme, P.; Prins, C.; Ramdane-Chérif, W. (2004):
Competitive Memetic Algorithms for Arc Routing Problems
Annals of Operations Research, vol. 131, pp. 159–185

Lan, S.; Clarke, J.; Barnhart, C. (2006):
Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions
Transportation Science, vol. 40, pp. 15–28

Laporte, G. (1988):
Location-Routing Problems
in:
Assad, A.; Golden, B. (eds.):
Vehicle Routing: Methods and Studies
Elsevier, Amsterdam
pp. 163–197

Laporte, G. (1997):
Modeling and Solving Several Classes of Arc Routing Problems as Traveling Salesman Problems
Computers & Operations Research, vol. 24, pp. 1057–1061

Laporte, G.; Nobert, Y.; Taillefer, S. (1988):
Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems
Transportation Science, vol. 22, pp. 161–172

Lawler, E.; Lenstra, J.; Rinnooy, A.; Shmoys, D. (eds.) (1985):
The Traveling Salesman Problem
Wiley, Chichester

Lübbecke, M.; Desrosiers, J. (2005):
Selected Topics in Column Generation
Operations Research, vol. 53, pp. 1007–1023

Lustig, I.; Puget, J. (2001):
Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming
Interfaces, vol. 31, pp. 29–53

Lysgaard, J.; Letchford, A.; Eglese, R. (2004):
A New Branch-and-Cut Algorithm for the Capacitated Vehicle Routing Problem
Mathematical Programming, Series A, vol. 100, pp. 423–445

Martin, R. (1999):
Large Scale Linear and Integer Optimization: A Unified Approach
Kluwer, Boston

McBride, R. (1982):
Controlling Left- and U-Turns in the Routing of Refuse Collection Vehicles
Computers & Operations Research, vol. 9, pp. 145–152

Murty, K. (1983):
Linear Programming
Wiley, New York

Naddef, D.; Rinaldi, G. (2002):
Branch-and-Cut Algorithms for the Capacitated VRP
in:
Toth, P.; Vigo, D. (eds.):
The Vehicle Routing Problem
SIAM Monographs on Discrete Mathematics and Applications, Philadelphia
pp. 53–84

Nagy, G.; Salhi, S. (1996):
Nested Heuristic Methods for the Location-Routeing Problem
Journal of the Operational Research Society, vol. 47, pp. 1166–1174

Noon, C.; Bean, J. (1991):
A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem
Operations Research, vol. 39, pp. 623–632

Orloff, C. (1974):
A Fundamental Problem in Vehicle Routing
Networks, vol. 4, pp. 35–64

O'Rourke, J. (1998):
Computational Geometry in C
Cambridge University, Cambridge

Ropke, S. (2005):
Heuristic and Exact Algorithms for Vehicle Routing Problems
Ph.D. Thesis, Datalogisk Institut, Københavns Universitet

Roy, S.; Rousseau, J. (1989):
The Capacitated Canadian Postman Problem
INFOR, vol. 27, pp. 58–73

Salani, M. (2005):
Branch-and-Price Algorithms for Vehicle Routing Problems
Ph.D. Thesis, Facoltà di Scienze Matematiche, Fisiche e Naturali, Università degli Studi di Milano

Scheuerer, S. (2004):
Neue Tabusuche-Heuristiken für die logistische Tourenplanung bei restringierendem Anhängereinsatz, mehreren Depots und Planungsperioden
Ph.D. Thesis, Wirtschaftswissenschaftliche Fakultät, Universität Regensburg

Scheuerer, S. (2006):
A Tabu Search Heuristic for the Truck and Trailer Routing Problem
Computers & Operations Research, vol. 33, pp. 894–909

Sedgewick, R. (1992):
Algorithmen in C
Addison-Wesley, Bonn

Semet, F. (1995):
A Two-Phase Algorithm for the Partial Accessibility Constrained Vehicle Routing Problem
Annals of Operations Research, vol. 61, pp. 45–65

Semet, F.; Taillard, E. (1993):
Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search
Annals of Operations Research, vol. 41, pp. 469–488

Siek, J.; Lee, L.; Lumsdaine, A. (2002):
The Boost Graph Library: User Guide and Reference Manual
Addison-Wesley, Boston

Sierksma, G. (1996):
Linear and Integer Programming
Dekker, New York

Stroustrup, B. (1998):
Die C++-Programmiersprache
Addison-Wesley, Bonn

Tan, K.; Chew, Y.; Lee, L. (2006):
A Hybrid Multi-Objective Evolutionary Algorithm for Solving Truck and Trailer Vehicle Routing Problems
European Journal of Operational Research, vol. 172, pp. 855–885

Toth, P.; Vigo, D. (eds.) (2002):
The Vehicle Routing Problem
SIAM Monographs on Discrete Mathematics and Applications, Philadelphia

Vahrenkamp, R. (1989):
Transportation Logistic in Rural Setting—The Case of Milk Collection
Technical Report 5/1989, Fachbereich Wirtschaftswissenschaften, Gesamthochschule Kassel

Van Hentenryck, P. (2002):
Constraint and Integer Programming in OPL
INFORMS Journal on Computing, vol. 14, pp. 345–372

Vanderbeck, F. (2000):
On Dantzig-Wolfe-Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithm
Operations Research, vol. 48, pp. 111–128

Volkmann, L. (1996):
Fundamente der Graphentheorie
Springer, Wien

Wattleworth, J.; Shuldiner, P. (1963):
Analytical Methods in Transportation: Left-Turn Penalties in Traffic Assignment Models
Journal of the Engineering Mechanics Division, Proceedings of the American Society of Civil Engineers, vol. 89, pp. 97–126

West, D. (1996):
Introduction to Graph Theory
Prentice-Hall, Upper Saddle River

Williams, H. (1999):
Model Building in Mathematical Programming
Wiley, Chichester

Win, Z. (1987):
Contributions to Routing Problems
Ph.D. Thesis, Naturwissenschaftliche Fakultät, Universität Augsburg

Winter, S. (2002):
Modeling Costs of Turns in Route Planning
GeoInformatica, vol. 6, pp. 345–361

Wolsey, L. (1998):
Integer Programming
Wiley, New York

Wyskida, R.; Gupta, J. (1972):
IE's Improve City's Solid Waste Collection
Industrial Engineering, vol. 46, pp. 12–15

Lebenslauf

Persönliche Daten:
Michael Drexl
Geburtsdatum: 26. Mai 1969
Geburtsort: Schwabmünchen

Schulausbildung:

1975–1979	Sankt-Ulrich-Volksschule Schwabmünchen
1979–1988	Leonhard-Wagner-Schule Schwabmünchen, Gymnasialzug Abschluß: Abitur

Wehrdienst:

1988–1989	Gebirgsnachschubbataillon 8, Mittenwald/München

Studium:

1989–1994	Betriebswirtschaftslehre, Universität Augsburg Abschluß: Diplomkaufmann

Berufstätigkeit:

1994–1998	Kaufmännischer Angestellter, DACHSER GmbH & Co., Augsburg

Zusatzstudium:

1998–2001	Zusatzstudium Operations Research und Wirtschaftsinformatik, Rheinisch-Westfälische Technische Hochschule Aachen Abschluß: Magister des Operations Research

Berufstätigkeit:

2001–2003	Softwareentwickler und Unternehmensberater, GTS Systems & Consulting GmbH, Herzogenrath
2003–2007	Wissenschaftlicher Angestellter und Doktorand, Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken, Rheinisch-Westfälische Technische Hochschule Aachen
2007–	Wissenschaftlicher Mitarbeiter, Fraunhofer-Arbeitsgruppe für Technologien der Logistik-Dienstleistungswirtschaft, Nürnberg