Contributions to Branch-and-Price-and-Cut Algorithms for Routing Problems

Dissertation
zur Erlangung des Grades eines Doktors der
wirtschaftlichen Staatswissenschaften
(Dr. rer. pol.)
des Fachbereichs Recht- und Wirtschaftswissenschaften
der Johannes Gutenberg-Universität Mainz

vorgelegt von
M. Sc. Ann-Kathrin Rothenbächer
in Mainz

im Jahre 2017

Tag der mündlichen Prüfung: 19.10.2017

Danksagung

Zuallererst möchte ich mich ganz herzlich bei meinem Doktorvater bedanken, der fachlich und menschlich ein großartiger Lehrstuhlleiter ist und mich bei der Erstellung dieser Doktorarbeit stets unterstützt hat. Ein großer Dank geht auch an die Kolleginnen und Kollegen am Lehrstuhl, die alle in den vergangenen drei Jahren zu richtig guten Freunden geworden sind. Die Zusammenarbeit mit ihnen war stets sehr angenehm.

Darüber hinaus möchte ich meinem Freund, meinen Eltern und meinen Geschwistern von ganzem Herzen dafür danken, dass sie immer zu mir halten und mir viel Freude schenken.

Auch allen anderen Menschen, die mich bereits auf meinem Lebensweg begleitet haben, gilt mein Dank.

Contents

List of Papers

- Christian Tilk[1], Ann-Kathrin Rothenbächer[1], Timo Gschwind[1], Stefan Irnich[1] (2017). Asymmetry matters: Dynamic Half-Way Points in Bidirectional Labeling for Solving Shortest Path Problems with Resource Constraints Faster. *European Journal of Operational Research* 261 (2). pp. 630–539.
- Timo Gschwind, Stefan Irnich, Ann-Kathrin Rothenbächer, Christian Tilk (2017). Bidirectional Labeling in Column-Generation Algorithms for Pickup-and-Delivery Problems. *Forthcoming in European Journal of Operational Research.*
- Ann-Kathrin Rothenbächer, Michael Drexl[2], Stefan Irnich (2016). Branch-and-Price-and-Cut for the Truck-and-Trailer Routing Problem with Time Windows. *Forthcoming in Transportation Science.*
- Ann-Kathrin Rothenbächer (2017). Branch-and-Price-and-Cut for the Periodic Vehicle Routing Problem with Flexible Schedule Structures. *Submitted to Transportation Science.*
- Ann-Kathrin Rothenbächer, Michael Drexl, Stefan Irnich (2016). Branch-and-Price-and-Cut for a Service Network Design and Hub Location Problem. *European Journal of Operational Research* 255 (3). pp. 935–947.

[1] Johannes Gutenberg-Universität Mainz, Lehrstuhl für BWL insb. Logistikmanagement, Jakob-Welder-Weg 9, 55128 Mainz, Germany

[2] Technische Hochschule Deggendorf, Statistics and Logistics, Dieter-Görlitz-Platz 1, 94469 Deggendorf, Germany

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Branch-and-price-and-cut algorithms are powerful approaches for solving combinatorial optimization problems to optimality (Desrosiers and Lübbecke, 2011). In general, combinatorial optimization problems aim to find an optimal solution over a finite set of discrete points that fulfill certain conditions (Schrijver, 2003). They can usually be represented by a graph and can often be formulated as an integer linear program (IP). A compact formulation (typically arc-based) can be transformed to an extensive formulation (typically path-based) by redefining the variables (Dantzig and Wolfe, 1960). On the one hand, this increases the number of variables significantly. On the other hand, this reformulation has a sharper linear relaxation and reduces the symmetry of some problems. Routing problems are an important representative of combinatorial optimization problems. In this thesis, applications of and improvements on branch-and-price-and-cut algorithms for routing problems will be discussed. Section 1.1 explains the basic principles of branch-and-price-and-cut algorithms, Section 1.2 introduces the routing problems studied in this thesis, and Section 1.3 describes the outline and the contributions of this thesis.

1.1 Branch-and-price-and-cut methodology

Branch-and-price-and-cut algorithms comprise the techniques branch-and-bound, column generation, and addition of valid inequalities, which are interdependent. The branch-and-bound method was developed by Dakin (1965) and ensures feasible integer solutions through a systematic and usefully restricted enumeration. Column generation is applied to formulations with a huge number of variables to find and add improving columns dynamically (Barnhart *et al.*, 1996; Lübbecke and Desrosiers, 2005). Valid inequalities (cuts) can be identified and added to a formulation to strengthen its linear relaxation. The details of all three parts depend on the problem that is tackled.

Branching decisions may be taken on individual variable values, but also on subsets of variables or on properties the variables have. However, deciding on an individual variable that was generated should be avoided, because it is hard to ensure that the column is not generated again. The order of branching decisions should comply with a decreasing order of influence on the whole problem. Finally, the branching rules have to be complete, i.e. they have to guarantee that no fractional solutions remain.

Cuts are constraints that are not necessary to define the solution space while inte-

ger variables are considered, but cut off fractional points in the linear relaxation of an IP. There are generic cuts that are applicable to all IPs such as the Chvatal-Gomory cuts (Chvátal, 1973; Gomory, 1963). Furthermore, there are specialized cuts for certain problem structures such as capacity cuts or subset-row inequalities (Ropke and Cordeau, 2009; Jepsen *et al.*, 2008). The detection of violated cuts is called separation and may be costly. Additionally, adding cuts to a linear program (LP) makes the problem more difficult and induces higher run times in the LP optimization and sometimes in the column generation. Thus, the benefit of tightening the linear relaxation has to be weighed up against these drawbacks and only helpful cuts should be added.

Column generation corresponds to the decomposition of a problem to a master problem and a subproblem (or pricing problem). The master problem usually contains a huge number of variables that is exponential in the problem size. Thus, only a small subset of the variables is given at the start, forming the restricted master problem. The dual information of the solution of a restricted master problem is then used in one or several subproblems to find new columns (variables). More precisely, a subproblem has to find at least one column with negative reduced costs or has to determine that no such column exists. Thereby, the reduced costs of a column are influenced by branching decisions and added cuts.

In routing problems, the subproblem is usually an elementary shortest path problem with resource constraints (ESPPRC, Irnich and Desaulniers, 2005). Here, the task is to find the best path (with respect to the reduced costs) from a given source to a given sink that fulfills a set of constraints defined over a set of resources. Although the ESPPRC is known to be NP-hard (Dror, 1994), it can be solved effectively with a dynamic-programming based labeling algorithm (Dijkstra, 1959; Ahuja *et al.*, 1993). Labeling algorithms entail the benefit that it is easy to find more than one path with negative reduced costs, which is especially helpful in a column generation approach. The ingredients for the solution of an ESPPRC with a labeling algorithm are the knowledge of the underlying network, of the label propagation functions (resource extension functions, REF), and of a dominance relation between labels. Obviously, all of them depend on the considered problem. Moreover, there are several acceleration strategies for the labeling algorithm such as a bidirectional search (Righini and Salani, 2006) and relaxations of the elementarity constraint, e.g. with the *ng*-neighborhood (Baldacci *et al.*, 2011d).

All in all, column-generation based algorithms have become predominant for the exact solution of many types of routing problems (Desaulniers *et al.*, 2005).

1.2 Considered routing problems

Routing refers to the task of selecting a path through a network. A very famous routing problem is the vehicle routing problem (VRP) where a fleet of vehicles has to perform a set of tasks with minimal costs (Toth and Vigo, 2014). There are numerous variants of the vehicle routing problem. In the following, the variants relevant for this dissertation will be introduced in the order of their appearance in the thesis.

In most applications, the vehicles deliver goods to customers and the capacity of the

vehicles is limited, which leads to the capacitated VRP (often just named VRP). When the customers specify certain time intervals for their visits, the problem becomes a VRP with time windows (VRPTW, Cordeau *et al.*, 2002; Desaulniers *et al.*, 2010; Baldacci *et al.*, 2012b; Desaulniers *et al.*, 2014). Several branch-and-price-and-cut algorithms were proposed for the VRPTW in the literature (e.g., Desrochers *et al.*, 1992; Kohl *et al.*, 1999; Irnich and Villeneuve, 2006; Desaulniers *et al.*, 2008).

In the electric VRP (EVRP), limited driving ranges of electric vehicles and the possibility to recharge at certain stations are considered. This problem was first mentioned by Schneider *et al.* (2014) who proposed an effective hybrid metaheuristic. The first exact approach was a branch-and-price algorithm and developed by Desaulniers *et al.* (2016a).

In the vehicle routing and truck driver scheduling problem (VRTDSP), the hours of service regulations for truck drivers are incorporated, including allowed driving times and required breaks. Goel and Irnich (2017) presented a branch-and-price algorithm, which was the first exact method for the VRTDSP complying with the new U.S. regulations. Their approach was refined by Tilk (2016) who presented, among other enhancements, a bidirectional labeling algorithm for the VRTDSP-specific subproblem.

The pickup-and-delivery problem (PDP) deals with requests of transporting goods or passengers from specific pickup locations to corresponding delivery locations (Parragh *et al.*, 2008; Battarra *et al.*, 2014). Thus, there are additional pairing and precedence constraints on the locations that need to be visited. A successful exact approach based on specialized bounding procedures and a branch-and-price-and-cut algorithm can be found in (Baldacci *et al.*, 2011c).

The truck-and-trailer routing problem (TTRP) concerns situations where trucks and attached trailers are used to visit customers with the restriction that some customers are not reachable by trailers (Chao, 2002; Cuda *et al.*, 2015). Hence, trailers can be parked at some locations and have to be recoupled later. Parragh and Cordeau (2017) present both an adaptive large neighborhood heuristic and an exact branch-and-price method to tackle the TTRP.

In the periodic VRP (PVRP), the planning horizon is extended to multiple periods and customers need to be visited several times according to one of their offered visiting patterns (Francis *et al.*, 2008; Campbell and Wilson, 2014; Irnich *et al.*, 2014b). In the standard variant, the customers need to be visited regularly with a preset visit frequency. Francis *et al.* (2006) and Francis *et al.* (2007) presented exact and heuristic methods dealing with more flexible visiting patterns.

The service network design and hub location problem (SNDHLP) does not belong to VRPs, but it also includes routing decisions. Its task is to select locations and connections between them to establish a network for the routing of a given set of transport requests. Surveys on service network design can be found in (Wieberneit, 2008) and (Yaghini and Akhavan, 2012) whereas an overview over hub location problems is given by Alumur and Kara (2008). Combinations of these two aspects are manifold and differ in several problem aspects.

1.3 Outline and contributions

This thesis by publication consists of five papers that have either been published in or submitted to scientific journals. All papers contribute to exact branch-and-price-and-cut algorithms for routing problems with a focus on the solution of the ESPPRC pricing problem. New techniques improve the performance of state-of-the-art methods and new problem variants can be tackled. The presented algorithms include general acceleration techniques and make use of the individual problem characteristics. In the following, the structure of the thesis and the contributions of the individual papers are described.

Chapter 2 introduces a new search strategy for labeling algorithms to solve ESPPRCs. The idea is based on the bidirectional search that starts simultaneously (forward) at the start and (backward) at the end of a path and merges these two sub-paths at a fixed half-way point (Righini and Salani, 2006). This bidirectional search performs better than the monodirectional counterpart because the number of labels typically grows exponential in the path length. However, the workload in the two directions is often unbalanced, either because of asymmetric input data or because of different propagation and dominance functions in the two directions. Thus, instead of choosing the half-way point a priori, our contribution is a procedure to determine this point dynamically by estimating the workload in both directions. The effectiveness of this idea is demonstrated by computational studies with the VRPTW, the EVRPTW and the VRTDSP.

In Chapter 3, the first bidirectional labeling approach for subproblems of PDPs is presented. A strong dominance relation can only be achieved when it is never beneficial to visit the second location of a request (the delivery point in forward direction and the pickup point in backward direction). Because the costs can not be distributed on the arcs such that this is true in both directions, bidirectional search seemed to be unattractive for ESPPRCs with a pickup and delivery structure. We could show that it is possible to use different cost matrices for the two directions when the merge procedure is adapted accordingly. Thus, the strong dominance relation can be used in both directions making the bidirectional search attractive again. Extensive experiments on PDPTW benchmark instances confirm the superiority of our approach compared to monodirectional labelings.

Chapter 4 deals with a branch-and-price-and-cut algorithm for the TTRP with time windows and two new extensions. The algorithm combines several state-of-the-art techniques for VRPs, such as a bidirectional labeling algorithm, the usage of the *ng*-neighborhood, and the incorporation of subset-row cuts. The first problem extension is the enlargement of the planning horizon from one to two days. Thereby, a symmetry arises, which is tackled by a tailored column-generation stabilization method. The second problem extension is the consideration of quantity-dependent transfer times for the load transfers from the truck to the trailer. We propose an adapted labeling algorithm that can handle the emerging tradeoff between the consumed time and the free space in the truck. Computational results provide insights on the new extensions and demonstrate that our algorithm outperforms existing approaches on TTRP benchmark instances.

In Chapter 5, a branch-and-price-and-cut algorithm for the classical PVRP with time windows and an extension to more different visiting patterns is presented. Whereas the

classical PVRP expects visiting patterns with regular intervals between the visits and a fixed visit frequency for every customer during the planning horizon, this paper enables all kinds of visiting patterns. Therefor, two new ways to model the underlying pricing network are introduced, on which a labeling algorithm with many known acceleration techniques can find new columns. Moreover, for instances that fulfill a special symmetry, we show that constraint aggregation significantly improves the solution process. Different PVRP-specific settings are evalutated by computational tests and the optimality of two PVRPTW benchmark instances can be proven for the first time.

Chapter 6 tackles the SNDHLP in context of combined transport with a branch-and-price-and-cut algorithm. Requests have to be transported via road and rail through a network whose hub nodes and connections between them have to be selected. In contrast to most other related works, the decision on a connection is not only about its existence but about the corresponding frequency of services. Furthermore, several other real-world conditions such as restricted number of transshipments, transport time limits and splittable requests are considered in this combination for the first time. The branch-and-price-and-cut algorithm is adapted to the problem by a specific layered pricing network, an hierarchical branching scheme, the incorporation of three types of valid inequalities, and further acceleration techniques. Computational tests on real world instances of the major German rail company could prove the practical applicability of the algorithm.

Finally, Chapter 7 summarizes and concludes the thesis.

Chapter 2

Asymmetry matters: Dynamic Half-Way Points in Bidirectional Labeling for Solving Shortest Path Problems with Resource Constraints Faster

Christian Tilk, Ann-Kathrin Rothenbächer, Timo Gschwind, Stefan Irnich

Abstract

With their paper "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints" [Discrete Optimization 3, 2006, pp. 255–273] Righini and Salani introduced bounded bidirectional dynamic programming (DP) as an acceleration technique for solving variants of the shortest path problem with resource constraints (SPPRC). SPPRCs must be solved iteratively when vehicle routing and scheduling problems are tackled via Lagrangian relaxation or column-generation techniques. Righini and Salani and several subsequent works have shown that bounded bidirectional DP algorithms are often superior to their monodirectional counterparts, since the former can mitigate the fact that the number of labels increases strongly with the path length. Bidirectional DP has become a quasi-standard for solving SPPRCs with general resource extension functions. In computational experiments, however, one can still observe that the number of forward and backward label extensions is very unbalanced despite a symmetric bounding of a critical resource in the middle of its feasible domain. We exploit this asymmetry in forward and backward label extensions to reduce the overall workload by introducing a so-called dynamic half-way point, which is a dynamic bounding criterion based on the current state of the simultaneously solved forward and backward DPs. Experiments with the standard and the electric vehicle routing problem with time windows as well as the vehicle routing and truck driver scheduling problem confirm that dynamic half-way points better balance forward and backward labeling and reduce the overall runtime.

2.1 Introduction

Vehicle routing and scheduling problems have been the subject of intensive study for more than half of a century now. Standard variants of the *vehicle routing problem* (VRP), such as the capacitated VRP and the *vehicle routing problem with time windows* (VRPTW), are well studied and effective solution algorithms can be found in the literature. Even more, a plethora of variants of the VRP has been investigated in thousands of scientific papers and powerful commercial vehicle routing software is available. Research on these more complex VRP variants is constantly ongoing, motivated by unsolved theoretical problems as well as continuous input from logistics practice (Drexl, 2012; Irnich *et al.*, 2014a).

For solving VRPs exactly, algorithms based on column generation and Lagrangian relaxation are the state-of-the-art (see, e.g., Poggi and Uchoa, 2014; Desaulniers *et al.*, 2014). Herein, the master program typically is a (possibly extended) set-partitioning formulation and the subproblem a variant of the *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). Any feasible solution in the SPPRC describes a feasible route in the respective VRP. The task of the pricing subproblem is to find at least one negative reduced-cost route, or to prove that no such route exists. In most applications, the solution of the SPPRC subproblem requires by far the largest portion of the overall computation time. Therefore, a lot of research has been spent in enhancing SPPRC algorithms. This includes the handling of elementarity constraints (Feillet *et al.*, 2004; Boland *et al.*, 2006) and relaxing elementarity (Irnich and Villeneuve, 2006; Baldacci *et al.*, 2011d; Bode and Irnich, 2014), handling of non-robust cuts (Jepsen *et al.*, 2008; Contardo *et al.*, 2015), the use of completion bounds (Baldacci *et al.*, 2011d), and SPPRC heuristics either dynamic-programming (DP) based (Irnich and Desaulniers, 2005; Feillet *et al.*, 2007) or based on metaheuristics (e.g., Desaulniers *et al.*, 2008).

The variants of SPPRC we focus on in this paper are those with constrained resources that are propagated in a general, not necessarily purely additive fashion. This includes the handling of time windows, resource-dependent cost functions, and interdependent resources as needed in many rich VRPs. In contrast, the special case of paths constrained by completely additive resources is often referred to as the *(resource) constrained shortest path problem (CSPP)*. Several highly effective algorithms have been developed for the CSPP (e.g., Garcia, 2009; Lozano and Medaglia, 2013; Pugliese and Guerriero, 2013; Horváth and Kis, 2016).

For the exact solution of SPPRCs with general resource constraints, DP based labeling algorithms are predominant (Irnich and Desaulniers, 2005). Let $G = (V, A)$ be the digraph of the SPPRC with vertex set V and arc set A. In labeling algorithms, a label refers to a partial path from the given source vertex $s \in V$ to some vertex $i \in V$. It holds the information about the resource consumption along this path and refers to its predecessor, the label of the partial path from which the actual label was extended. Starting with the initial partial path $P = (s)$, labeling algorithms

(1) select an unprocessed partial path $P = (s, \ldots, i)$,

(2) extend the partial path P along all arcs (i, j) of the forward star of vertex i,

(3) check whether the new partial paths $(P, j) = (s, \ldots, i, j)$ are feasible, and if so,
(4) store them so that they can be extended at a later point.

Labeling algorithms allow the steps (1)–(4) to be performed implicitly with the help of specific attributes, the resource values of the label. In particular, the extension step (2) propagates labels directly using so-called *resource extension functions* (REFs, Desaulniers *et al.*, 1998). Repeating the steps (1)–(4) creates all possible paths starting at s. Dominance procedures are therefore invoked to identify and discard those partial paths and their labels that cannot lead to an optimal SPPRC solution.

The above labeling scheme is known as a monodirectional forward labeling algorithm because it extends partial paths only in the forward direction. For many types of SPPRCs, the direction of propagation can be reversed. It means that the first partial path is $P = (t)$, where t is the sink vertex, and partial paths $P = (j, \ldots, t)$ are then extended against the orientation of an arc (i, j) creating new partial paths $P' = (i, j, \ldots, t)$. This requires that REFs can be inverted. A systematic way to do this is described in (Irnich, 2008). However, reversion may be complicated or create a different set of resources needed to describe backward partial paths. Whether forward or backward labeling is then beneficial still depends on many factors, and is often not clear in advance. Even worse, both monodirectional forward and monodirectional backward labeling typically suffer from an *explosion of labels*. This effect, also known as *combinatorial explosion*, describes that progressively more labels are created, are extended, and need to be stored when the length (=number of arcs) of partial paths increases.

Righini and Salani (2006) have presented bounded bidirectional labeling in order to mitigate the explosion of labels. Both forward partial paths and backward partial paths are created, but processed only up to a so-called *half-way point*. The half-way point is defined with the help of a monotone resource (e.g., the time), often as the midpoint of its feasible domain. When labeling terminates, suitable forward and backward labels must be merged to obtain complete feasible s-t-paths.

The paper by Righini and Salani (2006) and several subsequent works have shown that bounded bidirectional labeling algorithms are usually superior to their monodirectional counterparts. Hence, bidirectional labeling has become a quasi-standard for solving SPPRCs with general resource extension functions. It works particularly well if forward and backward labeling are similar, i.e., the same number of resources can be used and fathoming criteria of comparable strength can be applied in both directions. However, in applications in which forward and backward labeling differs significantly, it is unclear how a reasonable half-way point should be defined. For example, in VRPs with renewable resources such as the electric vehicle routing problem with time windows (Desaulniers *et al.*, 2016a), forward and backward labeling can require a different number and different types of resources (see Section 2.3.2). In such a situation, the computation time spent in one labeling direction can strongly exceed the time spent in the other direction. But even in cases with the same number of resources and similar fathoming criteria for both directions, it can still be observed in computational experiments that the number of forward and backward label extensions is very unbalanced despite of a symmetric bounding with the critical resource in the middle of its feasible domain. This can be caused by asym-

metric VRP instance data, e.g., time windows being unevenly spread over the planning horizon. In addition, oscillation of dual prices over the column-generation iterations can cause further asymmetry in SPPRC instances. Note also that in many VRPs, the dual price of one vertex may either be assigned to the ingoing or outgoing arcs of this vertex, producing another form of asymmetry. With the goal of balancing the workload between the forward and backward labeling, one may define a different half-way point in each column-generation iteration. However, it is unclear how to a priori set a computationally convenient half-way point so that the overall workload is minimized.

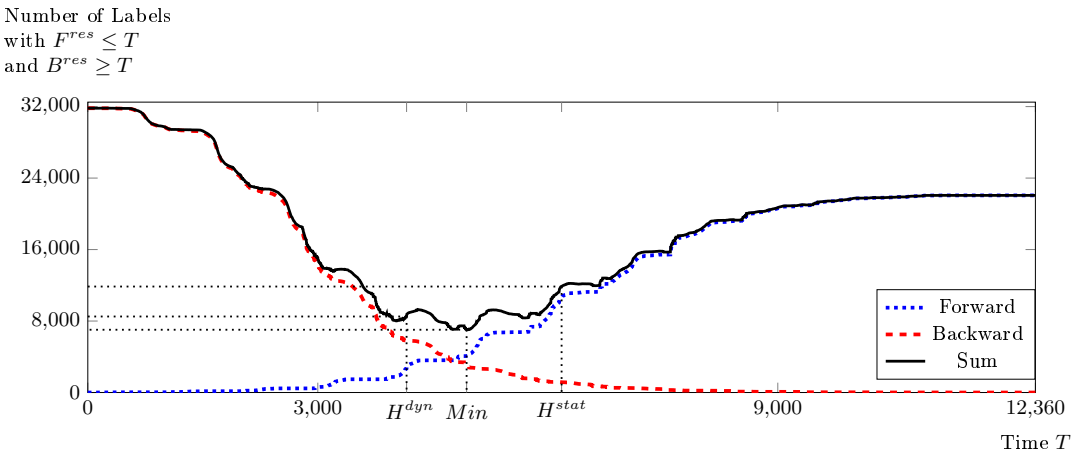


Figure 2.1: Example of an SPPRC instance with unbalanced forward and backward labeling; static half-way point $H^{stat} = 6{,}180$ in the middle of the time horizon and dynamic half-way point $H^{dyn} = 4{,}158$

In Figure 2.1, an example of an SPPRC instance with unbalanced forward and backward labeling is depicted. Herein, time is chosen as the monotone resource. The functions show, for any value T of the monotone resource, the number of processed forward labels with time $\leq T$, the number of processed backward labels with time $\geq T$, and the sum of both values. Obviously, forward labeling creates less labels than backward labeling, and bidirectional labeling even less whenever a half-way point is chosen somewhere in the middle (here between 2,523 and 11,118). Note that with a half-way point in the midpoint 6,180 of the time horizon $[0; 12{,}360]$, bidirectional labeling produces more forward and backward labels in total than with appropriately chosen smaller half-way points.

The contribution of the paper at hand is the introduction of a *dynamic half-way point*, which is a dynamic bounding criterion based on the current state of the simultaneously solved forward and backward SPPRC. The dynamic half-way point exploits the asymmetry in forward and backward label extensions. It can be used in all labeling algorithms that have the same monotone resource available in both directions. The idea of a dynamic bounding criterion has been sketched in an earlier work of Pecin (2014) and was used in (Pecin *et al.*, 2017) for successfully solving large-scale capacitated VRPs using a tailored arc-load indexed formulation that creates multiple start-labels on one side and

herewith a strong imbalance between forward and backward labeling. As far as we know, however, the idea has not been thoroughly analyzed for VRPs with general resources. Our experiments with three types of VRPs confirm that dynamic half-way points better balance forward and backward workload. We have selected the VRPTW as an example of a standard VRP with relatively few resources that have similar forward and backward REFs. The *electric vehicle routing problem with time windows* (EVRPTW), on the contrary, has a different number of forward and backward resources. Finally, the *vehicle routing and truck driver scheduling problem* (VRTDSP) has a larger number of resources resulting in generally more difficult SPPRC instances.

The paper is organized as follows. In Section 2.2, we formally describe bidirectional labeling for SPPRCs while emphasizing the differences of using a static or a dynamic half-way point. The definition of the three VRP variants and their forward and backward labels is given in Section 2.3. The impact of using a dynamic half-way point is computationally evaluated in Section 2.4. Finally, concluding remarks are given in Section 2.5.

2.2 Bidirectional Labeling

We start with a formal description of a generic bidirectional labeling algorithm. An instance of the SPPRC is given by the SPPRC network $G = (V, A)$, the source vertex $s \in V$, the sink vertex $t \in V$, and forward and backward REFs f_{ij} and b_{ij}, respectively, for each arc $(i, j) \in A$. For convenience, we assume that G is simple so that arcs (i, j) and their REFs can be uniquely described by tail and head vertex. Forward labels are denoted by F and backward labels by B.

The following assumptions are made for the remainder of the paper: Both forward and backward labels have a specific resource res. Let $F_i = F(P)$ denote the forward label associated with a forward partial path $P = (s, \ldots, i)$ and define $v(F_i) = i$. Then, for any arc $(i, j) \in A$ and the extension $F_j := f_{ij}(F_i)$ the inequality $F_i^{res} \leq F_j^{res}$ holds. Similarly, let $B_j = B(P)$ denote the backward label associated with a backward partial path $P = (j, \ldots, t)$ and define $v(B_j) = j$. Then, for any arc $(i, j) \in A$ and the extension $B_i := b_{ij}(B_j)$ the inequality $B_i^{res} \leq B_j^{res}$ holds. We call this resource res the *monotone resource* and assume that its domain is the interval $[L, U]$. Note that in most applications at least one monotone resource is naturally included in the problem, since time and load/residual capacity usually fulfill the requirements. If this is not the case, a monotone resource can always be created artificially, e.g., by counting the number of arcs in a path. Different strategies to implement bidirectional labeling algorithms were presented in the literature and these strategies are very important for the effectiveness of the overall algorithm (Righini and Salani, 2006). Algorithm 1 shows a generic version of a labeling algorithm for solving an SPPRC. The generic parts are the label selection strategy (label setting, label correcting) determined by the functions getNextForwardLabel() and getNextBackwardLabel(), the dominance strategy determined by the function checkDominance(), and the function getNextDirection() that controls whether

the next label to extend is a forward or a backward label.

Algorithm 1: Generic Labeling Algorithm for SPPRCs

Input: Forward half-way point H_F and backward half-way point H_B (required $H_F \geq H_B$)

```
1  F := F(s), B := B(t);
2  while F ≠ null or B ≠ null do
3      direction := getNextDirection();
4      if direction = forward then
5          if F^res ≤ H_F then
6              for (i, j) ∈ A with i = v(F) do
7                  Propagate F to vertex j with REF f_ij;
8          H_B := max{H_B, min{F^res, H_F}};
9          F := getNextForwardLabel();
10     else
11         if B^res > H_B then
12             for (i, j) ∈ A with j = v(B) do
13                 Propagate B to vertex i with REF b_ij;
14         H_F := min{H_F, max{B^res, H_B}};
15         B := getNextBackwardLabel();
16     if checkDominance() then
17         Apply dominance rules;
```

We comment on Algorithm 1 in more detail. The algorithm requires the forward half-way point H_F and the backward half-way point H_B as an input. Only for $H_F \geq H_B$ optimality is guaranteed. Initial labels F and B are created for the trivial forward and backward partial paths in Step 1. As long as there are unprocessed labels, the Steps 3–17 are repeated. First, in Step 3, the function getNextDirection() controls the direction of the next extension step. One must ensure that it returns *forward* if $F \neq null$ and $B = null$, and *backward* if $F = null$ and $B \neq null$, while in case of $F, B \neq null$ both directions are possible. In Steps 5 and 11, it is ensured that only labels respecting the half-way criterion are extended. The forward propagation in Step 7 includes the creation of the tentative new label $F_j := f_{ij}(F)$, the feasibility check of the path associated with F_j, and if feasible, the storage of the new label F_j so that it can later be retrieved via getNextForwardLabel(). The backward propagation in Step 13 performs the respective operations for the tentative label $B_i := b_{ij}(B)$ and the corresponding partial backward path. The half-way points are updated in Steps 8 and 14.

In Steps 9 and 15, the function getNextForwardLabel()/getNextBackwardLabel() returns an unprocessed forward/backward label that is used for extension the next time the algorithm chooses the respective direction. If no unprocessed forward/backward label exists, the value *null* is returned. To allow the efficient retrieval of the next label to process, the unprocessed labels should be managed with an appropriate

data structure. For example, labels may be stored in hash tables according to the value of the critical resource so that a label-setting approach can be implemented by retrieving forward/backward labels in ascending/descending order of the critical resource.

In Step 17, a dominance algorithm is called in order to identify and discard provably non-optimal labels. Note that the invocation of a dominance algorithm can always be delayed to some convenient point in time. In particular, the determination of dominated labels, every time a new feasible label is created, is generally not efficient because it requires a quadratic number of comparisons of labels in the worst case. More efficient algorithms, e.g., based on the multidimensional divide-and-conquer principle exist (see Bentley, 1980) but require that batches of labels are tested for dominance. The function CHECKDOMINANCE() in Step 16 controlling the dominance algorithm should therefore call it only when there is the chance that the next label to process is dominated.

The two half-way points H_F and H_B are upper and lower bounds respectively for the final half-way point H. The update Steps 8 and 14 guarantee that $H_F \geq H_B$. Initially setting H_F and H_B to the same value, therefore, leaves these parameters fixed, and $H = H_F = H_B$ results. Moreover, the backward half-way point H_B can only be increased. This happens when the processed forward label has a larger value of the monotone resource (Step 8). Conversely, the forward half-way point H_F can only be decreased when the processed backward label has a smaller value of the monotone resource (Step 14). In the course of Algorithm 1, H_F and H_B approach each other and it terminates with $H = H_F = H_B$. Thus, depending on the initial values of H_F and H_B, the resulting algorithm behaves in the following manner:

$H_F = H_B > U$: monodirectional forward labeling algorithm;

$H_F = H_B < L$: monodirectional backward labeling algorithm;

$H_F = H_B \in (L, U)$: bidirectional labeling algorithm with static half-way point;

$U = H_F > H_B = L$: bidirectional labeling algorithm with dynamic half-way point.

In the two monodirectional cases $H_F = H_B > U$ and $H_F = H_B < L$ either of the conditions in Step 11 or Step 5 is always false. Consequently, extensions occur only in one direction. There is no specific requirement on the ordering in which labels are retrieved in Steps 9 and 15.

In the version with a static half-way point, H_F and H_B are initialized to the same value inside the domain of the monotone resource, e.g., the middle $(U + L)/2$. By defining GETNEXTDIRECTION() as *forward* whenever $F \neq null$, and *backward* otherwise, Algorithm 1 performs all forward extensions before the backward extensions. Any other strategy, however, still produces the same set of forward and backward labels. Again, in the forward and in the backward part, labels can be processed in any order.

On the contrary, in the version with a dynamic half-way point ($H_F > H_B$), the forward processing should be performed in ascending order while the backward processing should be done in descending order of the value of the monotone resource. More precisely, in subsequent iterations $F :=$ GETNEXTFORWARDLABEL() should return non-decreasing values F^{res} and $B :=$ GETNEXTBACKWARDLABEL() should return non-increasing values B^{res}. Otherwise, e.g., prematurely processing a forward label with a large value

of the monotone resource would force the (backward) half-way point towards U resulting in almost pure forward labeling. Anyway, Algorithm 1 maintains $H_B \leq H_F$ so that even a non-monotone label extension does not produce incorrect results. Furthermore, non-decreasing values F^{res} and non-increasing values B^{res} guarantee that all forward/backward labels with a value of the monotone resource smaller/greater than H_B/H_F have already been extended.

In the course of Algorithm 1, the two half-way points approach each other due to the update Steps 8 and 14. Clearly, as soon as $H_F = H_B$, the labeling behaves like the bidirectional labeling with this static half-way point. Note that choosing initial values $H_F < U$ and/or $H_B > L$ restricts the value of the dynamic half-way point to lie in the interval $[H_F, H_B]$. However, the difference to a classical bidirectional labeling algorithm is that the half-way point is implicitly chosen with the function GETNEXTDIRECTION(). Therefore, the criterion for alternating between forward and backward labeling is of crucial importance for choosing a good half-way point and the overall performance of Algorithm 1.

An ideal strategy to choose a direction would be one that minimizes the overall computation time. It is not clear in advance how this can be achieved, since the computational effort is influenced by many different factors. A good indicator may be the overall number of labels that have to be considered in the course of Algorithm 1. However, this overall number is not known at the time when a direction must be determined. Reasonable choices try to estimate the remaining effort, e.g., by choosing

(1) the direction with the smaller number of *generated* labels,
(2) the direction with the smaller number of *processed* labels,
(3) the direction with the smaller number of *unprocessed* labels.

All these measures are estimates only because the overall computational effort is also influenced, e.g., by the portion of the time spent in dominance tests. This computation time is however not directly proportional to the overall number of labels.

When Algorithm 1 terminates, compatible forward and backward labels must be merged to obtain complete s-t-paths. Righini and Salani (2006) elaborate a half-way point test that avoids the creation of the same s-t-path from different pairs of forward and backward labels.

We now consider again the example shown in Figure 2.1. Recall that bidirectional labeling with a static half-way point $H^{stat} = 6{,}180$ in the middle of the time horizon reduces the number of labels by approximately 50 % compared to monodirectional forward labeling (even more compared to backward). Although the overall number of backward labels is larger than the number of forward labels, our dynamic half-way point is smaller than the static half-way point. This is justified by the course of the forward and backward labeling, since the strong increase of backward labels occurs at small time values T that are not relevant for the bidirectional labeling if the half-way point ist chosen larger. The dynamic half-way point shown in Figure 2.1 results from the strategy of choosing the direction with the smaller number of unprocessed labels. For this SPPRC instance, the dynamic half-way point $H^{dyn} = 4{,}158$ misses the global minimum $Min = 4{,}941$ of the overall number of generated labels. Nevertheless, compared to the static half-way

point, the number of generated labels is further reduced by around 30 %.

2.3 Problem Descriptions

In this section, we sketch the three considered VRP variants, give references for the most effective column-generation based algorithms, and describe the resources that have been used in the DP labeling algorithms to solve the associated SPPRC pricing problems. In particular, we point out those cases in which forward and backward labeling differ significantly.

2.3.1 Vehicle Routing Problem with Time Windows (VRPTW)

The VRPTW is an extension of the capacitated VRP in which additionally travel times between locations are given and it is required that the service at a customer starts within a pre-specified hard time window. Surveys on the VRPTW are (Cordeau *et al.*, 2002; Desaulniers *et al.*, 2010; Baldacci *et al.*, 2012b; Desaulniers *et al.*, 2014). Different branch-and-price-and-cut algorithms were proposed for the VRPTW in the literature (e.g., Desrochers *et al.*, 1992; Kohl *et al.*, 1999; Irnich and Villeneuve, 2006; Desaulniers *et al.*, 2008).

When solving the SPPRC of the VRPTW pricing problem, the following resources describe the state of a vehicle at the end of a forward partial path P from the source vertex s to a vertex $i \in V$, represented by a label $F_i = \left(F_i^{cost}, F_i^{load}, F_i^{time}, (F_i^{cust_n})_{n \in N}\right)$, where N is the set of all customers. The components of a label are:

F_i^{cost}: reduced cost of path P;

F_i^{load}: total load collected along path P;

F_i^{time}: earliest service start time at vertex i;

$F_i^{cust_n}$: number of times that customer $n \in N$ is visited along path P. The attribute is also set to 1 if customer n is unreachable from P (for details see Feillet *et al.*, 2004).

A backward label B_i describes the state of a vehicle at the start of a backward partial path P from a vertex $i \in V$ to the sink vertex t. Its components B_i^{cost} and $(B_i^{cust_n})_{n \in N}$ are defined as in the forward case, whereas the time-related resource B_i^{time} and the load-related resource B_i^{load} have a slightly different meaning:

B_i^{load}: residual vehicle capacity with respect to the collection along path P;

B_i^{time}: latest service start time at vertex i.

Details about the initial labels, the forward and backward propagation of the resources, the dominance between two forward or two backward labels, and the conditions for a concatenation of a forward and a backward partial path can be found in (Righini and Salani, 2006; Irnich, 2008). Note that it is generally not necessary to solve the described elementary version of the SPPRC pricing problem which is $\mathcal{NP}$-hard in the strong sense. Indeed, state-of-the-art approaches to most VRP variants rely on the so-called *ng*-path

relaxation that allows certain non-elementarities. For the corresponding SPPRCs, the labeling algorithms have pseudo-polynomial complexity. Infeasibility with respect to the elementarity condition of routes is then eliminated by branching or by dynamically enlarging the *ng*-neighborhood size (see Boland *et al.*, 2006; Righini and Salani, 2008; Baldacci *et al.*, 2012b). For necessary modifications of the resources $(F_i^{cust_n})_{n\in N}$ and $(B_i^{cust_n})_{n\in N}$ when using the *ng*-path relaxation see (Baldacci *et al.*, 2011d).

Note that along a partial path the time-related and load-related resources are monotone. Along a forward path, earliest service start times and collected quantities are non-decreasing. Along a backward path, now considered in the direction from the sink vertex t to the source vertex s, latest service start times and residual capacities are non-increasing. Hence, both resources are suitable to define the monotone resource. If the time resource is chosen, the static half-way point is often set to the middle of the time horizon, i.e., $H^{time} = (a_s + b_t)/2$, where a_s is the earliest start time at the source vertex s and b_t is the latest arrival time at the sink vertex t. If the load resource is chosen, the static half-way point is typically set to $H^{load} = Q/2$, where Q is the vehicle capacity. The decision of using H^{time} or H^{load} should take into account whether time-window or capacity constraints are more restrictive in a given VRPTW instance. The more constrained the monotone resource is, the more effective is bidirectional labeling.

2.3.2 Electric Vehicle Routing Problem with Time Windows (EVRPTW)

The EVRPTW is a special variant of the VRPTW taking into account the limited driving range of electric vehicles and the possibility to recharge at recharging stations. As in the VRPTW, a set of customers must be visited exactly once during their service time windows by vehicles with a limited capacity. Because of rather short cruising ranges of electric vehicles and the restricted number of recharging stations, the refueling has to be considered explicitly with a non-negligible recharging time.

The problem was first mentioned by Schneider *et al.* (2014) who proposed an effective hybrid metaheuristic. The first exact branch-and-price algorithm for the EVRPTW was developed by Desaulniers *et al.* (2016a). They consider four problem variants by distinguishing between a maximum of one recharge per route and multiple possible recharges as well as between the full and the partial recharge policies. The full recharge policy requires that the battery is always filled up completely at every recharge while the partial recharge policy allows any amount of electrical energy to be recharged. In the following, we focus on the problem variant with multiple full recharges (MF) that requires a difficult and the most asymmetric SPPRC pricing problem among these four variants.

In addition to the above mentioned SPPRC resources (F_i^{cost}, F_i^{load}, F_i^{time}, and $F_i^{cust_n}$) of the VRPTW, the following information has to be stored in a forward label F_i to determine the status of the vehicle routed along a forward partial path P from the source vertex s up to vertex $i \in V$:

F_i^{rch}: number of recharges performed along path P;

F_i^{rt}: cumulated required recharging time since the last recharge along path P

(or since the beginning of P if P contains no recharge).

There are four types of vertices in the EVRTPW: the source vertex s, the sink vertex t, the customers $n \in N$, and the recharging stations $r \in R$. Moreover, there are two types of REFs f_{ij} depending on the type of vertex i from which the label is extended. Leaving a customer or depot vertex requires the same resource update as in the VRPTW and an increase of the required recharging time. However, when a recharging station is left, the required recharging time F_i^{rt} is added to the time resource F_j^{time} to represent the full recharge, F_j^{rt} is reset, and the recharge counter is increased by 1.

The backward propagation is more complicated, since the necessary recharging amount at a recharging station is not exactly known. It depends on the energy consumed chronologically before the current position, which is not yet determined when a recharging station is reached in a backward extension step. Thus, for describing a backward partial path P from a vertex $i \in V$ to the sink vertex t represented by a label B_i, the standard resources ($B_i^{cost}, B_i^{load}, B_i^{time}$, and $B_i^{cust_n}$) of the VRPTW are complemented by four other resources; two similar to the ones of the forward labeling and two new resources:

B_i^{nrch}: negative number of recharges performed along path P except at the last vertex i;

B_i^{rt}: cumulated required recharging time needed to recover the energy consumed up to the next recharging station r (or consumed along P if no recharge is performed);

B_i^{sl}: if a recharging station is visited along path P, this is the cumulated slack time at vertex i that can be used for recharging at the next recharging station without changing the latest service start time at i;

B_i^{avrt}: if a recharging station is visited along path P, this is the maximum additionally available recharging time at the next recharging station that ensures time window feasibility along P.

Backward REFs b_{ij} depend on whether a recharging station was visited on the path and on the type of vertex j. Further details such as the initial labels, the definition of all REFs, the feasibility checks, the dominance rules, and the merging conditions can be found in (Desaulniers *et al.*, 2016a). Summarizing, the backward labeling uses two additional resources, has a weaker dominance than the forward labeling, and requires more computational effort.

As in the VRPTW, both the time and the load resource are non-decreasing along a forward path and non-increasing along a backward path. Therefore, they are suitable monotone resources for the half-way point definition.

2.3.3 Vehicle Routing and Truck Driver Scheduling Problem (VRTDSP)

The VRTDSP is the variant of the VRPTW that schedules the vehicles according to customer service time windows and hours of service regulations. Hours of service regulations for truck drivers have been imposed by many governments worldwide to ensure

that break and rest periods are regularly taken. Transport companies have to take these into account and plan the routes and schedules of their truck drivers simultaneously. Recently, Goel and Irnich (2017) presented the first exact approach to the VRTDSP complying with the new U.S. hours of service regulations. They use a branch-and-price algorithm to solve the resulting VRTDSP. According to the current U.S. regulations, a driver may take break and rest periods at any time and with any duration larger than 30 minutes and ten hours, respectively. However, driving periods must be scheduled in accordance with the current state of the driver. On the contrary, no limitations regarding service activities are imposed by the U.S. regulations. The relevant parameters are summarized in Table 2.1.

Symbol	Value	Description
t^{drive}	11 hours	The maximum accumulated driving time between two consecutive rest periods
t^{break}	$\frac{1}{2}$ hours	The minimum duration of a break period
t^{rest}	10 hours	The minimum duration of a rest period
$t^{\text{el\|B}}$	8 hours	The maximum time after the end of the last break or rest period until which a driver may drive
$t^{\text{el\|R}}$	14 hours	The maximum time after the end of the last rest period until which a driver may drive

Table 2.1: Parameters imposed by the new U.S. hours of service regulations (Table 1 in Goel and Irnich, 2017)

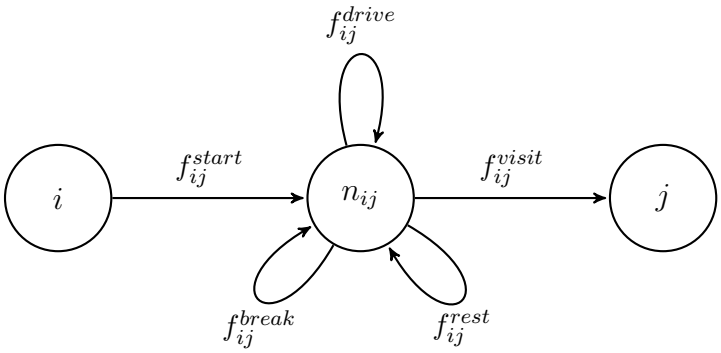


Figure 2.2: Subnetwork for arc $(i, j) \in A$ (similar to Figure 1 in Tilk, 2016)

The column-generation subproblem of the VRTDSP is an SPPRC that takes into account capacity and time-window constraints as well as hours of service regulations. In (Goel and Irnich, 2017), it is solved with a forward labeling algorithm in an auxiliary network where an intermediate vertex n_{ij} is created for every feasible arc $(i, j) \in A$. Five different REFs are then needed to model the activities of a vehicle and its driver on the

trip from i to j, see Figure 2.2. The feasibility of a partial path strongly depends on the driver's current state. Therefore, the standard VRPTW resources ($F_i^{cost}, F_i^{load}, F_i^{time}$ and $(F_i^{cust_n})_{n \in N}$) of a forward label F_i associated with the partial path P from the source vertex s to a vertex $i \in V$ are supplemented by six additional resources:

- F_i^{dist}: remaining driving time to reach the next customer;
- F_i^{drive}: accumulated driving time since the last rest;
- $F_i^{el|B}$: time elapsed since the last break;
- $F_i^{el|R}$: time elapsed since the last rest;
- $F_i^{la|B}$: latest possible point in time to which the end of the last break period can be extended without violating any resource constraints;
- $F_i^{la|R}$: latest possible point in time to which the end of the last rest period can be extended without violating any resource constraints.

The five REFs can be summarized as follows: First, the REF f_{ij}^{start} models the start of the trip from i to j immediately after i was serviced. Second, the REF f_{ij}^{drive} implements a driving activity of Δ_{ij} time units within the trip from i to j. The amount of driving time can be computed as $\Delta_{ij} = \min\{F_i^{dist}, t^{\text{drive}} - F_i^{drive}, t^{\text{el|B}} - F_i^{el|B}, t^{\text{el|R}} - F_i^{el|R}\}$. It is the maximal driving time that does not violate any time-window or hours of service constraints. Third, taking a 30-minute break or a ten-hour rest on the trip from i to j is modeled with the REFs f_{ij}^{break} and f_{ij}^{rest}, respectively. Finally, performing the service at customer j is modeled with the help of the REF f_{ij}^{visit}. This REF also extends the length of the last rest and break period by any unavoidable waiting time.

Tilk (2016) refined this approach and presented, among other enhancements, a bidirectional labeling algorithm for the VRTDSP-specific SPPRC. The backward labeling is defined on a time-reversed network in which all time windows are inverted and arcs are reversed. This allows to use the same resources with the same meaning as in the forward labeling. However, the REFs need to be slightly altered in order to take into account that waiting and service times do not count as driving times and are, therefore, not restricted by the hours of service regulations. As waiting occurs only before and service after the start of a service time window, the forward and backward labeling are asymmetric due to this fixed chronological order. Details about the initial labels, the forward and backward propagation of the resources, the dominance rules, the feasibility conditions for concatenating a forward and a backward label, the adaption of the ng-path relaxation, and other acceleration techniques can be found in (Tilk, 2016).

Since the backward times are negative due to the inversion of the network, the time-related resource B^{time} is non-decreasing in the backward labeling, i.e., $B_i^{time} \geq B_j^{time}$ for a backward extension along arc $(i, j) \in A$. To nevertheless apply Algorithm 1, the following small modification is necessary: B^{time} has to be replaced by its negative value $B^{res} = -B^{time}$ in Steps 11 and 14. The load-related resource behaves as in the VRPTW, thus, both resources are suitable to define a half-way point.

2.4 Computational Results

This section reports the computational results on the three aforementioned VRP variants. All results were obtained using a standard PC with an Intel(R) Core(TM) i7-5930k 3.5 GHz processor and 64 GB of main memory. The algorithms were coded in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2013. The callable library of CPLEX 12.6.0 was used for solving the restricted master programs (RMPs) in the column-generation algorithms. No branching is performed, since in most cases different branch-and-bound trees result when columns are generated by different SPPRC algorithms. This makes computation time for the same problem instance scatter more and the impact of using a dynamic half-way point cannot be determined accurately with the (relatively small) benchmark sets used in the literature. However, the general behaviour of a branch-and-price(-and-cut) does not differ much, on average, from the results we present here.

Pre-tests revealed that the time resource is better suited to define the half-way point than the load resource in all considered problem variants. We set $H^{time} = (a_s + b_t)/2$ to define the static half-way point and initialize the dynamic labeling algorithm by setting $H_F = b_t$ and $H_B = a_s$. Labels are processed in order of the monotone resource and ties are broken arbitrarily. The dominance algorithm is invoked every time the value of H_F or H_B changes. Moreover, we decide whether the next label to extend is a forward or a backward label by choosing the direction with the smaller number of *unprocessed* labels (forward if the numbers are identical). In pre-tests on reduced instance sets, this strategy has proven to be superior on average to the other strategies mentioned in Section 2.2 for the three considered VRP variants.

We run two different types of tests. In the first type of test, we solve every instance of the SPPRC pricing problem using both the static and the dynamic half-way point in each iteration of the linear relaxation of the master program. Since both versions are solved with identical reduced costs in each iteration, the comparison is not impacted by an otherwise different trajectory of the dual prices. Thus, we expect the dynamic-to-static ratios to be more stable leading to more reliable results. We compute the following values for each instance: the ratios dynamic to static of the number of generated labels, the number of processed labels, and the time spent in the pricing as well as the coefficient of variation (COV) of the dynamic half-way point. All ratios are computed as geometric means over all pricing iterations. We provide minimum, geometric mean, and maximum of these values, aggregated over the different instance groups. For the computation of the time ratio, we take into account only those iterations of the pricing problem that take more than 0.1 seconds in the static and the dynamic version. Herewith, the impact of inaccuracies in time measurement is reduced. Whenever the linear relaxation is not completely solved within the time limit, the value is taken over the iterations solved by both versions up to this point.

In the second type of test, we compare the solution of the RMPs when either the dynamic or the static version of the half-way point is used in the pricing problems. For each instance group, the following values are reported: the number of successfully solved

linear relaxations and the average solution time in seconds needed in the static and dynamic version, respectively. The average is taken only over the instances that were solved by both versions.

The following subsections report the results for the different VRP variants along with the settings and instances they were obtained with.

2.4.1 VRPTW Results

We test our column-generation algorithm for the VRPTW on the 56 benchmark instances of Solomon (1987). Additionally, we use the 60 instances with 200 customers proposed by Gehring and Homberger (2002). A CPU time limit of one hour is used for all computations. Furthermore, the *ng*-path relaxation with a neighborhood size of ten is applied, and networks with a reduced arc set are used as pricing heuristics.

Instance group	Generated labels			Processed labels			Time			COV		
	min	avg	max	min	avg	max	min	avg	max	min	avg	max
Solomon	0.76	0.95	1.08	0.68	0.92	1.08	0.68	0.96	1.15	0.02	0.15	0.52
Gehring & Homberger	0.72	0.91	1.08	0.58	0.84	1.06	0.32	0.77	1.16	0.08	0.22	1.19
all		0.93			0.88			0.86			0.19	
Solomon (SR)	0.76	0.94	1.08	0.68	0.92	1.09	0.47	0.89	1.28	0.02	0.14	0.52
Gehring & Homberger (SR)	0.69	0.91	1.08	0.59	0.83	1.07	0.31	0.73	1.10	0.09	0.22	1.19
all (SR)		0.92			0.87			0.81			0.18	

Table 2.2: VRPTW: Ratios for using the dynamic and static half-way point with identical reduced cost

The results of the first type of test, where labeling with static and dynamic half-way points is applied with identical reduced cost, are summarized in Table 2.2. The values are aggregated according to the instance group. In addition to the results for solving the linear relaxation, we present results when subset-row inequalities (SR) are added (Jepsen *et al.*, 2008). We restrict ourselves to those SR inequalities defined on subsets with three vertices as proposed by Jepsen *et al.* (2008). To limit the number of violated inequalities added to the RMP, we use the same cut-generation strategy with identical parameters as proposed by Desaulniers *et al.* (2008).

The results for solving the linear relaxation without cuts already indicate that the dynamic half-way point is superior. Only 93 % of the labels generated in the static version are generated in the dynamic version, and only 88 % of the labels processed in the static version are processed in the dynamic version. The average computation time of the dynamic version is reduced to 86 % in relation to the static. The reported coefficient of variation shows that the values of the dynamic half-way points vary by 19 % on average among the pricing iterations of each VRP instance. Our interpretation of this result is that a computationally convenient half-way point differs from iteration to iteration of the pricing problem. Thus, any cleverly chosen but fixed static half-way point is nonetheless inferior in most iterations.

When SR inequalities are added to the RMP, the pricing problem becomes harder because more resources are involved and labels are less comparable in terms of dominance. The ratios for generated and processed labels decrease slightly, but the computation time ratio decreases by another five percent on average.

Table 2.3 shows the results for the second type of test, i.e., the comparison between static and dynamic half-way points for solving the linear relaxation. Herein, the results are also aggregated per instance group. No significant difference can be observed when solving the Solomon instances. However, the algorithm with the dynamic half-way point is able to solve three more linear relaxations of the Gehring & Homberger instances and the solution time decreases by around 30 %. This implies that the benefit of using a dynamic half-way point increases with the size and difficulty of the SPPRC instances.

Instance group	No. solved		Avg time [s]	
	static	dynamic	static	dynamic
Solomon	56/56	56/56	118.64	111.24
Gehring & Homberger	46/60	49/60	674.32	466.51
all	102/116	105/116	369.24	271.46

Table 2.3: VRPTW: Comparison of computation times using static vs. dynamic half-way points

2.4.2 EVRPTW Results

The experiments for the EVRPTW are performed on the 56 instances with 100 customers introduced by Schneider *et al.* (2014). We incorporated the small modifications suggested by Desaulniers *et al.* (2016a). Smaller instances are created by considering only the first 25 and 50 customers. The CPU time limit is set to one hour per instance. As for the VRPTW, the *ng*-path relaxation with a neighborhood size of ten is applied, and heuristic pricing is implemented by using reduced networks.

The values are aggregated according to the characteristics of the instances. Characteristics of the instances are the customer distribution (C=clustered, R=random, and RC=both), the number of customers per instance (25, 50, and 100), and the length of the planning horizon (Series 1 and Series 2). *Series 1* contains the instance groups $C1$, $R1$, and $RC1$ with shorter planning horizon, whereas *Series 2* comprises the remaining instances of type $C2$, $R2$, and $RC2$ with longer planning horizon.

Table 2.4 summarizes the results of the first type of test, i.e., the direct comparison per pricing problem with identical reduced costs. Overall, the pricing times are reduced through the use of the dynamic half-way point by 24 %, the number of generated and processed labels is decreased by more than ten percent on average. The instances of the second series with longer routes profit significantly more from the use of the dynamic half-way point. Surprisingly, the average COV shows no direct correlation with the

Instance group	Generated labels			Processed labels			Time			COV		
	min	avg	max	min	avg	max	min	avg	max	min	avg	max
25	0.59	0.87	1.00	0.55	0.89	1.12	0.11	0.80	1.09	0.08	0.25	0.60
50	0.65	0.89	1.00	0.71	0.90	1.11	0.15	0.76	1.13	0.09	0.23	0.61
100	0.40	0.84	1.02	0.46	0.87	1.06	0.05	0.72	1.43	0.06	0.25	0.54
Series 1	0.59	0.90	1.00	0.64	0.93	1.12	0.29	0.91	1.13	0.06	0.27	0.53
Series 2	0.40	0.83	1.02	0.46	0.84	1.03	0.05	0.62	1.43	0.07	0.22	0.61
all		0.87			0.89			0.76			0.25	

Table 2.4: EVRPTW: Ratios for using the dynamic and static half-way point with identical reduced cost

effectiveness of the dynamic half-way point. However, an average COV of 25 % indicates that the dynamic adaptation of the half-way points is beneficial.

We have also run the first type of test for the linear relaxation of the master program with SR inequalities. As the ratios decrease only slightly, we do not report details. It seems that, unlike for the VRPTW, the SPPRC pricing problem of the EVRPTW is already so hard that adding SR inequalities does not significantly increase its difficulty.

Instance group	No. solved		Avg time [s]	
	static	dynamic	static	dynamic
25	55/56	56/56	5.36	2.68
50	54/56	55/56	324.50	240.46
100	41/56	42/56	692.63	517.26
Series 1	87/87	87/87	137.14	116.48
Series 2	63/81	66/81	541.80	382.08
all	150/168	153/168	305.53	227.00

Table 2.5: EVRPTW: Comparison of computation times using static vs. dynamic half-way points

Table 2.5 shows the results for the second type of test. With the dynamic half-way point, the linear relaxation of three more instances is solved within the time limit. The additional instances are in the *Series 2* showing again that EVRPTW instances with longer routes benefit more from the dynamic half-way point determination. Moreover, the solution times are reduced by 25 % on average.

2.4.3 VRTDSP Results

The VRTDSP experiments use the 56 benchmark instances proposed by Goel (2009), which can be obtained at `http://www.telematique.eu/research/downloads`. These

100-customer instances are derived from the VRPTW benchmark of Solomon (1987). Smaller instances are created by considering only the first 25 or 50 customers. We set a CPU time limit of two hours and use the *ng*-path relaxation with a neighborhood size of ten. Limited discrepancy search (Feillet *et al.*, 2007) and a heuristic dominance procedure are applied as heuristic pricing strategies (see also Goel and Irnich, 2017). Moreover, we stop Algorithm 1 as soon as the sum of negative reduced-cost labels at the sink in the forward labeling and the source in the backward labeling reaches 500.

The results of the first type of test are summarized in Table 2.6. The average dynamic-

Instance group	Generated labels			Processed labels			Time			COV		
	min	avg	max	min	avg	max	min	avg	max	min	avg	max
25	0.48	0.86	1.07	0.45	0.73	0.90	0.11	0.64	1.68	0.02	0.10	0.72
50	0.75	0.87	1.02	0.59	0.75	0.95	0.32	0.67	1.02	0.02	0.14	0.50
100	0.65	0.87	0.96	0.56	0.75	0.87	0.21	0.59	0.86	0.02	0.11	0.55
all		0.87			0.74			0.63			0.12	

Table 2.6: VRTDSP: Ratios for using the dynamic and static half-way point with identical reduced cost

to-static ratio of the generated and processed labels is nearly identical around 0.87 and 0.74, respectively, independent of the instance sizes. The average computation time reduces to 63 % for the dynamic version in relation to the static. The value of the dynamic half-way point varies by 12 % on average over the solution of a single instance. A detailed analysis shows that the pricing problem is solved faster with the static half-way point in only eleven of the 168 instances, where none of these master programs takes longer than 20 seconds. Moreover, in these eleven instances, either the static is very close to the dynamic half-way point in all pricing iterations or the solution of each single pricing problem takes less than 0.2 seconds. As in the case of the EVRPTW, adding SR inequalities to the linear relaxation has no significant impact on the dynamic-to-static ratios.

Instance group	No. solved		Avg time [s]	
	static	dynamic	static	dynamic
25	56/56	56/56	96.80	39.11
50	51/56	54/56	702.08	388.32
100	19/56	26/56	2123.98	1023.38
all	126/168	136/168	646.60	327.92

Table 2.7: VRTDSP: Comparison of computation times using static vs. dynamic half-way points

Table 2.7 reports the result of the second type of test showing that again labeling with a

dynamic half-way point is superior to the static version. Ten additional linear relaxations are solved, and the solution of a single instance takes on average approximately half of the time.

2.5 Conclusions

In this paper, we addressed the solution of different variants of the SPPRC with the help of bidirectional dynamic programming labeling algorithms. This type of algorithm is based on the definition of a so-called half-way point separating the forward from the backward labeling. In previous bidirectional labeling algorithms, the half-way point had to be specified a priori, before the labeling procedure starts. We exploit asymmetry in input data (constraints and reduced costs) as well as asymmetry in the forward and backward labeling process (different number of labels, bounding procedures, and dominance rules of different strength) by defining forward and backward half-way points that are dynamically adjusted according to an expected workload to be spent in forward and backward labeling. These forward and backward half-way points can be interpreted as upper and lower bounds of a possible static half-way point.

We have conducted extensive computational experiments with three types of VRPs where we compared the solutions of the column-generation SPPRC pricing problems using bidirectional labeling algorithms. For the VRPTW, replacing the static by the new dynamic half-way point reduces the solution times of SPPRCs by approximately 15 to 20 %. Comparing results with and without incorporating subset-row inequalities as well as Solomon's 100 customers and Gehring & Homberger's 200 customer instances, we found that the more difficult and larger VRPTW instances benefit more from a dynamic half-way point. For the EVRPTW, which has more SPPRC resources and is asymmetric in forward and backward labeling, the dynamic half-way point implementation on average reduced computation times by approximately 25 %. Also here, more difficult and larger SPPRC instances such as the second series with 100 customers profit more (38 % speedup on average). Finally, the VRTDSP is an example of a VRP with a large number of SPPRC resources and complex REFs. The experiments have revealed that on average the computation times are reduced by 37 % (41 % for the 100 customer instances).

The general insight of the experiments is the following: the harder the SPPRC instance, the more effective is the dynamic half-way point version. The necessary modifications needed to alter a bidirectional SPPRC labeling algorithm with static into one with dynamic half-way point are very minor. Thus, a significant effect can be achieved with little implementation effort.

Acknowledgement

We would like to thank Eduardo Uchoa from Pontifícia Universidade Católica (PUC) in Rio de Janeiro for pointing us to the works (Pecin, 2014; Pecin *et al.*, 2017).

Bibliography

Baldacci, R., Mingozzi, A., and Roberti, R. (2011c). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012b). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, **218**, 1–6.

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, **23**(4), 214–229.

Bode, C. and Irnich, S. (2014). The shortest-path problem with resource constraints with $(k, 2)$-loop elimination and its application to the capacitated arc-routing problem. *European Journal of Operational Research*, **238**(2), 415–426.

Boland, N., Dethridge, J., and Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, **34**(1), 58–68.

Contardo, C., Desaulniers, G., and Lessard, F. (2015). Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks*, **65**(1), 88–99.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M.M., and Soumis, F. (2002). VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 7, pages 155–194. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M.M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57–93. Springer, Boston, MA.

Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, **42**(3), 387–404.

Desaulniers, G., Desrosiers, J., and Spoorendonk, S. (2010). The vehicle routing problem with time windows: State-of-the-art exact solution methods. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*, volume 8, pages 5742–5749. John Wiley & Sons, Inc.

Desaulniers, G., Madsen, O.B.G., and Ropke, S. (2014). The vehicle routing problem with time windows. In P. Toth and D. Vigo, editors, *Vehicle Routing*, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016a). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*. Forthcoming.

Desrochers, M., Desrosiers, J., and Solomon, M.M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, **40**(2), 342–354.

Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research*, **5**(1), 47–63.

Feillet, D., Dejax, P., Gendreau, M., and Guéguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, **44**(3), 216–229.

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, **45**(4), 239–256.

Garcia, R. (2009). *Resource Constrained Shortest Paths and Extensions*. Ph.D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, U.S.A.

Gehring, H. and Homberger, J. (2002). Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics*, **8**(3), 251–276.

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, **43**(1), 17–26.

Goel, A. and Irnich, S. (2016). An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*. DOI: 10.1287/trsc.2016.0678.

Horváth, M. and Kis, T. (2016). Solving resource constrained shortest path problems with LP-based methods. *Computers & Operations Research*, **73**, 150–164.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, New York, NY.

Irnich, S. and Villeneuve, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, **18**(3), 391–406.

Irnich, S., Toth, P., and Vigo, D. (2014a). The Family of Vehicle Routing Problems. In P. Toth and D. Vigo, editors, *Vehicle Routing*, chapter 1, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Kohl, N., Desrosiers, J., Madsen, O.B.G., Solomon, M.M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, **33**(1), 101–116.

Lozano, L. and Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. *Computers & Operations Research*, **40**(1), 378 – 384.

Pecin, D. (2014). *Exact Algorithms for the Capacitated Vehicle Routing Problem*. Ph.D. thesis, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation.* **9**(1), 61 – 100.

Poggi, M. and Uchoa, E. (2014). New Exact Algorithms for the Capacitated Vehicle Routing. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 3, pages 59–86. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Pugliese, L.D.P. and Guerriero, F. (2013). A reference point approach for the resource constrained shortest path problems. *Transportation Science*, **47**(2), 247–265.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, **51**(3), 155–170.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, **48**(4), 500–520.

Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, **35**(2), 254–265.

Tilk, C. (2016). Branch-and-price-and-cut for the vehicle routing and truck driver scheduling problem. Technical Report LM-2016-04, Chair of Logistics Management, Johannes Gutenberg University, Mainz, Germany.

Chapter 3

Bidirectional Labeling in Column-Generation Algorithms for Pickup-and-Delivery Problems

Timo Gschwind, Stefan Irnich, Ann-Kathrin Rothenbächer, Christian Tilk

Abstract

For the exact solution of many types of vehicle-routing problems, column-generation based algorithms have become predominant. The column-generation subproblems are then variants of the shortest-path problem with resource constraints which can be solved well with dynamic-programming labeling algorithms. For vehicle-routing problems with a pickup-and-delivery structure, the strongest known dominance between two labels requires the delivery triangle inequality (DTI) for reduced costs to hold. When the direction of labeling is altered from forward labeling to backward labeling, the DTI requirement becomes the pickup triangle inequality (PTI). DTI and PTI cannot be guaranteed at the same time. The consequence seemed to be that bidirectional labeling, one of the most successful acceleration techniques developed over the last years, is not applicable with the strong dominance on both directions for problems with a pickup-and-delivery structure. In this paper, we show that bidirectional labeling with the strong dominance rules in forward as well as backward direction is possible by using different cost matrices in the two directions. We adopt the both-sided strong bidirectional labeling approach to integrate the standard robust and non-robust cuts. Moreover, a recent acceleration technique that dynamically adjusts the bidirectional half-way point is implemented. Full-fledged branch-cut-and-price algorithms are tested on the pickup-and-delivery problem with time windows (PDPTW). In particular, an in-depth analysis of different mono- and bidirectional labeling algorithms is presented. Overall, compared to the standard pure forward labeling-based algorithm, we obtain average reductions in computation time of more than 40 percent when solving PDPTW instances exactly.

3.1 Introduction

The classical *pickup-and-delivery problem* (PDP) is concerned with the transportation of passengers or goods from request-specific pickup points to their delivery points (Parragh *et al.*, 2008). The case of passenger transportation is also known as the *dial-a-ride problem* (DARP, Doerner and Salazar-González, 2014). Pickup-and-delivery for goods transportation has been recently surveyed by Battarra *et al.* (2014). Variants of the PDP include time windows (PDPTW), loading constraints such as last-in-first-out (LIFO) loading and multiple compartments (Cherkesly *et al.*, 2016), and handling operations (Veenstra *et al.*, 2017). Thus, the PDP and its variants are an important family of vehicle-routing problems (VRPs, Toth and Vigo, 2014).

For the exact solution of many types of VRPs, column-generation based algorithms have become predominant (Desaulniers *et al.*, 2005; Poggi and Uchoa, 2014). The column-generation subproblems are then variants of the *elementary shortest-path problem with resource constraints* (ESPPRC) which can be solved well with dynamic-programming labeling algorithms (Irnich and Desaulniers, 2005). For ESPPRCs with a pickup-and-delivery structure, the strongest known dominance between two labels (see the *strong dominance forward* and *backward* Rules 2 and 4, respecively) requires that the reduced cost matrix fulfills specific properties. In forward labeling, it is known that the *delivery triangle inequality* (DTI) for reduced costs must hold (Ropke and Cordeau, 2009). This means that a detour over a delivery point is always at least as costly in terms of reduced costs as a direct connection between two points. When the direction of labeling is altered from forward labeling to backward labeling, the DTI requirement becomes the *pickup triangle inequality* (PTI). Analogously, this means that a detour over a pickup location is always at least as costly in terms of reduced costs as a direct connection. DTI and PTI cannot be guaranteed at the same time. The consequence seemed to be that bidirectional labeling (Righini and Salani, 2006), one of the most successful acceleration techniques developed over the last years, is not applicable with strong dominance in both directions for problems with a pickup-and-delivery structure. Clearly, relying on a weaker dominance in one direction is feasible but makes the bidirectional approach less powerful.

The main theoretical contribution of this paper is to show that bidirectional labeling with strong dominance in forward as well as backward direction is possible. This is achieved by constructing different cost matrices fulfilling DTI and PTI, respectively. We use the PDPTW as a test case to empirically prove the effectiveness of that dominance principle. Our new labeling algorithm is compared against pure forward and backward labeling as well as bidirectional approaches that use strong dominance only in one direction. Furthermore, we integrate one of the most recent developments in bidirectional labeling which is the use of dynamic half-way points as suggested independently in Chapter 2 and by Pecin *et al.* (2017). We implement a full-fledged branch-cut-and-price algorithm that uses state-of-the-art cutting planes and a PDP-specific branching scheme. We then test all labeling algorithms on thousands of PDPTW pricing problems that need to be solved in column-generation iterations. The new bidirectional labeling with strong dominance

in both directions turns out superior to all these alternative labeling algorithms.

The remainder of this paper is structured as follows: Section 3.2 formally defines the PDPTW and sketches the column-generation algorithm by discussing its extended set-partitioning master problem and ESPPRC subproblem. Moreover, the role of the DTI/PTI for strong dominance in ESPPRC labeling algorithms is clarified. The section also derives the main theoretical result of the paper. Computational results are provided in Section 3.3. Final conclusions are drawn in Section 3.4.

3.2 Branch-Cut-and-Price for PDPTW

The PDPTW can be formally defined as follows: Let n be the number of requests and $R = \{(i, i+n) : i = 1, \ldots, n\}$ be the set of pickup-and-delivery requests, where i represents the pickup operation and $i+n$ the corresponding delivery operation. We define the PDPTW on a directed graph G with vertex set V and arc set A. The set of vertices $V = \{0, 1, \ldots, 2n{+}1\}$ comprises two copies of the depot with origin depot 0 and destination depot $2n{+}1$, the set of pickup vertices $P = \{1, \ldots, n\}$, and the set of delivery vertices $D = \{n+1, \ldots, 2n\}$. For each vertex $i \in V$, a time window $[a_i, b_i]$ representing the time interval in which service must start is given. For each request $(i, i+n) \in R$, the demand at its pickup and delivery vertices satisfies $q_i = -q_{i+n} > 0$. For convenience, we define $q_0 = q_{2n+1} = 0$. Furthermore, a fleet of K homogeneous vehicles with capacity Q is available at the depot. With each arc $(i, j) \in A$ are associated a travel time t_{ij} and a routing cost c_{ij}. We assume that the travel time t_{ij} of arc $(i, j) \in A$ already includes a service duration at vertex i. We also assume that the triangle inequality holds for both routing costs (c_{ij}) and travel times (t_{ij}).

3.2.1 Master Problem

The branch-cut-and-price for the PDPTW uses an extended set-partitioning formulation that we describe now. Let Ω be the set of all feasible PDPTW routes. A route $r \in \Omega$ is defined as an elementary 0-$(2n+1)$-path. It is feasible if it fulfills time-window, capacity, and pairing and precedence constraints (see Dumas *et al.*, 1991, for details). The cost c_r of a route $r \in \Omega$ is defined as the sum of the travel costs of the traversed arcs. The goal of the PDPTW is to find a set of feasible routes serving each request exactly once minimizing the total routing costs. The set-partitioning formulation uses binary variables λ_r indicating whether route $r \in \Omega$ is used or not. It can be stated as follows:

$$\min \quad \sum_{r\in\Omega} c_r\lambda_r \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{r\in\Omega} a_{ir}\lambda_r = 1 \qquad \forall(i,i+n)\in R \tag{3.1b}$$

$$\sum_{r\in\Omega} \lambda_r \leq K \tag{3.1c}$$

$$\lambda_r \in \{0,1\} \qquad \forall r\in\Omega \tag{3.1d}$$

The objective (3.1a) minimizes the total travel costs. Partitioning constraints (3.1b) ensure that each request is served exactly once. Herein, the binary coefficients a_{ir} are equal to 1 if and only if request $(i,i+n)\in R$ is served by route r. We replace (3.1b) by equivalent covering constraints (≥ 1), which is allowed as the routing costs and travel times fulfill the triangle equality. The number of routes is limited by (3.1c), and the variable domains are given in (3.1d).

For solving the linear relaxation of formulation (3.1), a column-generation algorithm (Desaulniers *et al.*, 2005) is employed. Starting with a subset $\bar{\Omega}\subset\Omega$ of the feasible routes, the linear relaxation of formulation (3.1) defined over $\bar{\Omega}$ is denoted as the *restricted master program* (RMP). The column-generation algorithm alternates between the re-optimization of the RMP and the solution of the column-generation pricing problem that adds negative reduced-cost variables (columns) to the RMP, if one exists. If no reduced-cost variables exist, the current linear relaxation is solved to optimality.

The linear relaxation can be strengthened by adding valid inequalities (see Section 3.2.3), and branching is required to finally ensure integer solutions (see Section 3.2.4). The most time-consuming part of this branch-cut-and-price algorithm is the solution of the pricing subproblem, described in the following.

3.2.2 Pricing Subproblem

Let π_i for $(i,i+n)\in R$ be the dual prices of the partitioning constraints (3.1b) and μ the dual price of constraint (3.1c). The pricing problem must compute at least one feasible route $r\in\Omega$ with negative reduced cost $\bar{c}_r := c_r - \mu - \sum_{(i,i+n)\in R} a_{ir}\pi_i$ (or guarantee that no such route exists). This problem is an ESPPRC which can be solved by means of a dynamic-programming labeling algorithm (Irnich and Desaulniers, 2005). Herein, for defining reduced costs of arcs, note that the dual price π_i for serving request $(i,i+n)\in R$ can be associated to the pickup vertex i, the delivery vertex $i+n$, or be arbitrarily split among them. Accordingly, for any given and fixed $\alpha\in\mathbb{R}$, define

$$\bar{\pi}_i(\alpha) := \alpha\pi_i \qquad \text{and} \qquad \bar{\pi}_{i+n}(\alpha) := (1-\alpha)\pi_i$$

and $\bar{\pi}_0(\alpha) := \bar{\pi}_{2n+1}(\alpha) := \mu$. The associated reduced cost of an arc $(i,j) \in A$ is then defined as

$$\bar{c}_{ij}(\alpha) := c_{ij} - \frac{1}{2}\bar{\pi}_i(\alpha) - \frac{1}{2}\bar{\pi}_j(\alpha).$$

With this definition, $\bar{c}_r = \sum_{(i,j)\in r} \bar{c}_{ij}(\alpha)$ holds for all routes $r \in \Omega$. In the following, we will use different values for α depending on the direction in which the subproblem is solved.

Labeling Algorithms A forward labeling algorithm starts with an initial label at the origin depot 0. It propagates forward labels over arcs toward the destination depot with the help of so-called *resource extension functions* (REFs, see Desaulniers *et al.*, 1998). Each forward label stores the resource consumption of the corresponding partial path $(0,\ldots,i)$ starting at the origin depot 0 and ending at some vertex $i \in V$. To avoid the enumeration of all feasible paths, provably redundant labels are eliminated through a dominance criterion.

Righini and Salani (2006) and several subsequent works have shown that bounded bidirectional labeling algorithms are usually superior to their monodirectional counterparts. A requirement for bidirectional labeling is that REFs can be inverted, as discussed by Irnich (2008). The respective backward labeling component starts with an initial label at the destination depot and propagates labels in backward direction, i.e., against the orientation of the arcs, toward the origin depot. Each backward label stores the resource consumption of a partial path $(i,\ldots,2n+1)$ starting at some vertex $i \in V$ and ending at the destination depot $2n+1$. A major design decision for bidirectional labeling is the definition of a so-called *half-way point*. Its purpose is to alleviate combinatorial explosion, since both forward and backward labels are extended only up to the half-way point. When labeling terminates, suitable forward and backward labels are merged to obtain complete feasible 0-$(2n+1)$-paths. For VRPs with time windows, traditional (static) approaches often choose the middle $(a_0 + b_{2n+1})/2$ of the planning horizon as the half-way point.

Up to now, bidirectional labeling has not been used for solving ESPPRC with a pickup-and-delivery structure, because it was not possible to apply strong dominance rules in both forward and backward labeling. By introducing the PTI, we show at the end of this section how forward and backward paths can be merged even if their labels are built on the basis of different cost matrices. We start however with the description of monodirectional forward and backward labeling.

Forward Labeling A forward partial path $(0,\ldots,i)$ in G is represented by a forward label $L_f = (i, \bar{c}_f, t_f, l_f, S_f, O_f)$. Its attributes are the last visited vertex i, the accumulated reduced cost $\bar{c}_f$, the earliest feasible start of service t_f at vertex i, the accumulated load l_f, the set of completed requests S_f, and the set of open requests O_f at vertex i. Requests are *open* if they have been picked up but not yet delivered. The initial forward label is given by $(0, 0, a_0, 0, \varnothing, \varnothing)$.

Propagating a forward label L_f over an arc $(i,j) \in A$ only results in a feasible extension

L'_f at vertex j if either $j \in P$ and $(j, j+n) \notin O_f \cup S_f$, or $j \in D$ and $(j-n, j) \in O_f$, or $j = 2n+1$ and $O_f = \varnothing$. Otherwise, pairing and precedence constraints are violated resulting in an infeasible label. Furthermore, consistency with respect to time-window and capacity constraints is ensured by requiring $t_f + t_{ij} \leq b_j$ and $l_f + q_j \leq Q$, respectively.

If the extension of L_f along arc $(i,j) \in A$ is feasible, a new forward label $L'_f = (j, \bar{c}'_f, t'_f, l'_f, S'_f, O'_f)$ is created by the following REFs:

$$\bar{c}'_f = \bar{c}_f + \bar{c}_{ij}(\alpha) \tag{3.2a}$$

$$t'_f = \max\{t_f + t_{ij}, a_j\} \tag{3.2b}$$

$$l'_f = l_f + q_j \tag{3.2c}$$

$$S'_f = \begin{cases} S_f \cup \{(j-n, j)\} & \text{if } j \in D \\ S_f & \textit{otherwise} \end{cases} \tag{3.2d}$$

$$O'_f = \begin{cases} O_f \cup \{(j, j+n)\} & \text{if } j \in P \\ O_f \setminus \{(j-n, j)\} & \text{if } j \in D \\ O_f & \textit{otherwise} \end{cases} \tag{3.2e}$$

Note that the load resource (l_f and l'_f) is redundant as $l_f = \sum_{(o,o+n) \in O_f} q_o$ and $l'_f = \sum_{(o,o+n) \in O'_f} q_o$. It is however convenient to use the attributes for a quick test of the feasibility of extensions.

Since the forward REFs are non-decreasing in the resources $\bar{c}_f$, t_f, and S_f, the following dominance rule is directly applicable (Dumas *et al.*, 1991):

Rule 1. (Weak Dominance Forward) A forward label $L^1_f = (i, \bar{c}^1_f, t^1_f, l^1_f, S^1_f, O^1_f)$ dominates another forward label $L^2_f = (i, \bar{c}^2_f, t^2_f, l^2_f, S^2_f, O^2_f)$ with identical last vertex i if $\bar{c}^1_f \leq \bar{c}^2_f$, $t^1_f \leq t^2_f$, $S^1_f \subseteq S^2_f$, and $O^1_f = O^2_f$ hold.

Using $\alpha = 1$ in the forward labeling, the dual prices of covering requests are completely assigned to the pickup vertices as already suggested by Dumas *et al.* (1991). Formally, $\alpha = 1$ gives $\bar{\pi}_i(\alpha) = \pi_i$ for all pickup vertices $i \in P$ and $\bar{\pi}_{i+n}(\alpha) = 0$ for all delivery vertices $i+n \in D$. With this definition, the forward reduced-cost matrix $(\bar{c}^f_{ij}) := (\bar{c}_{ij}(1))$ satisfies $\bar{c}^f_{ij} \leq \bar{c}^f_{ik} + \bar{c}^f_{kj}$ for all $(i,j) \in A$ and $k \in D$. Ropke and Cordeau (2009) call this property the DTI. Roughly speaking, the DTI ensures that visiting an additional delivery vertex is never beneficial. This property enables the use of the following stronger dominance rule (Dumas *et al.*, 1991):

Rule 2. (Strong Dominance Forward) Let the forward reduced-cost matrix fulfill the DTI.

A forward label $L^1_f = (i, \bar{c}^1_f, t^1_f, l^1_f, S^1_f, O^1_f)$ dominates another forward label $L^2_f = (i, \bar{c}^2_f, t^2_f, l^2_f, S^2_f, O^2_f)$ with identical last vertex i if $\bar{c}^1_f \leq \bar{c}^2_f$, $t^1_f \leq t^2_f$, $S^1_f \subseteq S^2_f$, and $O^1_f \subseteq O^2_f$ hold.

Backward Labeling In analogy to the forward labeling, a backward path $(j, \dots, 2n{+}1)$ in G defines a backward label $L_b = (j, \bar{c}_b, t_b, l_b, S_b, O_b)$. Its attributes are the first visited vertex j, the accumulated reduced cost $\bar{c}_b$, the latest feasible start of service t_b at vertex j, the accumulated load l_b, the set of completed requests S_b, and the set of open requests O_b at vertex j. Requests are *open* if they have been delivered but not yet picked up. The initial backward label is given by $L_b = (2n{+}1, 0, b_{2n+1}, 0, \varnothing, \varnothing)$.

Propagating a backward label L_b against the orientation of arc $(i, j) \in A$ only results in a feasible extension L'_b at vertex i if either $i \in P$ and $(i, i+n) \in O_b$, or $i \in D$ and $(i-n, i) \notin O_b \cup S_b$, or $i = 0$ and $O_b = \varnothing$. Furthermore, consistency with respect to time-window and capacity constraints is ensured by requiring $t_b - t_{ij} \geq a_i$ and $l_b + q_i \leq Q$ (recall that $q_i < 0$ for $i \in D$), respectively.

If the backward extension of L_b against the orientation of arc (i, j) is feasible, the new backward label $L'_b = (i, \bar{c}'_b, t'_b, l'_b, S'_b, O'_b)$ is created by the following REFs:

$$\bar{c}'_b = \bar{c}_b + \bar{c}_{ij}(\alpha) \tag{3.3a}$$

$$t'_b = \min\{t_b - t_{ij}, b_i\} \tag{3.3b}$$

$$l'_b = l_b - q_i \tag{3.3c}$$

$$S'_b = \begin{cases} S_b \cup \{(i, i+n)\} & \text{if } i \in P \\ S_b & \textit{otherwise} \end{cases} \tag{3.3d}$$

$$O'_b = \begin{cases} O_b \cup \{(i-n, i)\} & \text{if } i \in D \\ O_b \setminus \{(i, i+n)\} & \text{if } i \in P \\ O_b & \textit{otherwise} \end{cases} \tag{3.3e}$$

As in the forward case, also backward REFs are non-decreasing in the resources $\bar{c}_b$ and S_b and t_b so that the following dominance rule is valid:

Rule 3. (Weak Dominance Backward) A backward label $L_b^1 = (j, \bar{c}_b^1, t_b^1, l_b^1, S_b^1, O_b^1)$ dominates another backward label $L_b^2 = (j, \bar{c}_b^2, t_b^2, l_b^2, S_b^2, O_b^2)$ with identical first vertex j if $\bar{c}_b^1 \leq \bar{c}_b^2$, $t_b^1 \geq t_b^2$, $S_b^1 \subseteq S_b^2$, and $O_b^1 = O_b^2$ hold.

Using $\alpha = 0$ in the backward labeling, the dual prices of covering requests are completely assigned to the delivery vertices, i.e., $\bar{\pi}_i(\alpha) = 0$ for all pickup vertices $i \in P$ and $\bar{\pi}_{i+n}(\alpha) = \pi_i$ for all delivery vertices $i + n \in D$. The backward reduced-cost matrix $(\bar{c}_{ij}^b) := (\bar{c}_{ij}(0))$ satisfies $\bar{c}_{ij}^b \leq \bar{c}_{ik}^b + \bar{c}_{kj}^b$ for all $(i, j) \in A$ and $k \in P$. We call this property the PTI. It ensures that, for any backward partial path, visiting an additional pickup vertex is never beneficial. As with DTI in the forward labeling, PTI enables the use of a strong dominance rule in the backward labeling:

Rule 4. (Strong Dominance Backward) Let the backward reduced-cost matrix fulfill the PTI.

A backward label $L_b^1 = (j, \bar{c}_b^1, t_b^1, l_b^1, S_b^1, O_b^1)$ dominates another backward label $L_b^2 = (j, \bar{c}_b^2, t_b^2, l_b^2, S_b^2, O_b^2)$ with identical first vertex j if $\bar{c}_b^1 \leq \bar{c}_b^2$, $t_b^1 \geq t_b^2$, $S_b^1 \subseteq S_b^2$, and $O_b^1 \subseteq O_b^2$ hold.

Bidirectional Labeling For the bidirectional labeling algorithm, we assume that the forward reduced-cost matrix is $(\bar{c}^f_{ij}) := (\bar{c}_{ij}(1))$ and the backward reduced-cost matrix is $(\bar{c}^b_{ij}) := (\bar{c}_{ij}(0))$ so that the DTI and PTI are fulfilled, respectively. This enables the use of the strong dominance rules in both directions (Rule 2 and Rule 4).

The bidirectional labeling limits the effort in both directions, since not all forward (backward) labels are propagated to the destination (origin) depot. Instead, labels are propagated only up to a so-called *half-way point* h that is defined on a monotone resource of the labels. This limits the overall number of created labels. In their seminal work, Righini and Salani (2006) suggest to set h equal to the middle of the feasible domain of the chosen resource. In the PDPTW, the half-way point h is best defined on the time resource so that forward labels L_f are propagated only if $t_f \leq h$ and backward labels L_b only if $t_b > h$. Compatible forward and backward labels must then be merged to obtain complete 0-$(2n+1)$-paths.

Recently, Chapter 2 demonstrated that it is often beneficial to choose a half-way point h that differs from the middle of the feasible domain of the chosen resource. The reason is that due to asymmetries, e.g., coming from the dual variables, labeling in one of the directions (forward or backward) might be significantly harder than the opposite direction for a specific ESPPRC instance. Even more, this relation of the hardness of the two directions can be different in each column-generation iteration. To generally better balance the workload between the forward and backward labeling, Chapter 2 introduced a so-called *dynamic* half-way point which is a dynamic bounding criterion that takes into account the remaining efforts in the simultaneously solved forward and backward labeling. They estimate the remaining workload in forward and backward direction by the number of unprocessed forward and backward labels, respectively. We use the same heuristic criterion in our implementation of bidirectional labeling with a dynamic half-way point.

To avoid creating the same path multiple times from different pairs of forward and backward labels, Righini and Salani (2006) introduced a *half-way point test*. A forward label L_f at a vertex $i \in V$ is a candidate for merging if $i = 2n+1$ or $t_f > h$. All backward labels L_b at vertex i are then checked whether or not they can be merged with this candidate forward label L_f. The resulting path is feasible if the two labels L_f and L_b fulfill the following three merge conditions:

$$S_f \cap S_b = \varnothing \tag{3.4a}$$

$$t_f \leq t_b \tag{3.4b}$$

$$\begin{cases} O_b = O_f \setminus \{(i, i+n)\} & \text{if } i \in P \\ O_f = O_b \setminus \{(i-n, i)\} & \text{if } i \in D \\ O_b = O_f & \text{otherwise} \end{cases} \tag{3.4c}$$

where (3.4a) ensures that each request is served at most once, (3.4b) guarantees that the path is feasible with respect to the time resource, and (3.4c) requires that the concatenation completes all open requests.

Recall that reduced costs of partial paths in the forward and backward labeling rely on different cost matrices $(\bar{c}^f_{ij})$ and $(\bar{c}^b_{ij})$, respectively. Hence, the reduced cost of a path resulting from a merge is generally not the sum of the reduced costs of its forward and backward labels. The next proposition shows how reduced costs can, however, be computed.

Proposition 3.1. The reduced cost of a route r generated by a feasible merge of a forward label L_f and a backward label L_b can be obtained as

$$\bar{c}_r = \bar{c}_f + \bar{c}_b + \sum_{\substack{(k,k+n)\in \\ \{O_b \cap O_f\}}} \pi_k + \sum_{\substack{(k,k+n)\in \\ \{O_b \cup O_f\}\setminus\{O_b \cap O_f\}}} \frac{\pi_k}{2}. \tag{3.5}$$

Proof. Obviously, for all requests $(k, k+n) \in R$ that are completed (picked up and delivered) either in L_f or L_b, the dual price π_k is correctly incorporated. Let i be the vertex at which the merge is performed. For all requests $(k, k+n) \in R$ with $k \neq i$ and $k+n \neq i$ that are open in both directions, i.e., $(k, k+n) \in O_f \cap O_b$, the dual price π_k was subtracted twice (once in $\bar{c}_f$ and once in $\bar{c}_b$). This is corrected by adding it once to $\bar{c}_f + \bar{c}_b$. Finally, if the merge vertex i is a pick-up or a delivery vertex, the corresponding request is open either in L_f or L_b and its dual price π_i (π_{i-n}) for $i \in P$ ($i \in D$) was subtracted exactly 1.5 times, which is corrected by adding $\pi_i/2$ in (3.5). □

The following proposition shows that our bidirectional labeling algorithm is exact, although we (i) perform forward and backward labeling on different cost matrices and (ii) use a subset relation for open requests in the dominance while equality is required in the merge procedure..

Proposition 3.2. A bidirectional labeling algorithm that uses the strong dominance (Rule 2 and Rule 4) in both directions and the described merge procedure with (3.4) and (3.5) finds an optimal solution to the pricing subproblem of the PDPTW.

Proof. Let $\mathcal{P}$ be an optimal path, i.e., an elementary, feasible 0-$(2n+1)$-path with minimum reduced cost. We have to show that the bidirectional labeling algorithm finds this path or one with the same reduced cost.

Note that the pure forward labeling algorithm using the strong dominance Rule 2 is clearly valid because the reduced cost matrix $(\bar{c}^f_{ij})$ satisfies the DTI (see, e.g., Dumas *et al.*, 1991). This has two consequences. First, all paths that arrive at the destination depot $2n+1$ with feasible start of service not later than h are found in the forward part of the bidirectional labeling. Hence, we can restrict ourselves to cases where all optimal $\mathcal{P}$ result from a merge of a forward and a backward label. Second, it can be assumed that $\mathcal{P}$ is a path that is generated by the complete forward labeling algorithm.

In addition, we assume that $\mathcal{P}$ has a minimum number of served requests.

Then, $\mathcal{P}$ can be represented as $\mathcal{P} = (F, B)$, where F is its forward partial path ending at the merge vertex i and B is the backward partial path starting at the merge vertex i.

Denote by $L_f = (i, \bar{c}_f, t_f, l_f, S_f, O_f)$ the label representing F that is generated in the forward part of the bidirectional labeling algorithm.

Note first that backward labeling with Rule 4 is also valid because $(\bar{c}^b_{ij})$ fulfills the PTI. If the backward part of the bidirectional labeling produces the label $L^1_b = (i, \bar{c}^1_b, t^1_b, l^1_b, S^1_b, O^1_b)$ associated with B, then the merge step generates $\mathcal{P} = (F, B)$ and nothing remains to show. Otherwise, the label L^1_b does *not* exist in the backward part. However, there must exist a backward label $L^2_b = (i, \bar{c}^2_b, t^2_b, l^2_b, S^2_b, O^2_b)$ that dominates L^1_b, i.e., $t^2_b \geq t^1_b$, $S^2_b \subseteq S^1_b$, $O^2_b \subseteq O^1_b$, and $\bar{c}^2_b \leq \bar{c}^1_b$, where the latter inequality refers to the reduced costs given by the backward reduced cost matrix $(\bar{c}^b_{ij})$. We distinguish two cases.

Assume first that $O^2_b = O^1_b$. Then L_f and L^2_b fulfill the merge conditions (3.4), since L_f and L^1_b fulfill it due to the feasibility of $\mathcal{P}= (F, B)$. Moreover, the reduced cost of the path $\mathcal{P}' = (F, B')$ resulting from the merge of L_f and L^2_b has reduced cost not greater than $\mathcal{P} = (F, B)$, because the only difference in (3.5) is that $\bar{c}^1_b$ is replaced by $\bar{c}^2_b \leq \bar{c}^1_b$. Thus, $\mathcal{P}' = (F, B')$ is another feasible 0-$(2n+1)$-path with minimum reduced cost which is found in the bidirectional labeling algorithm.

Second, assume that $O^2_b \subsetneq O^1_b$ holds. We now show by contradiction that such a backward label L^2_b cannot exist. Consider the forward partial path F' which results from removing all pickup vertices of requests $(k, k+n) \in O^1_b \setminus O^2_b$ from F. Then, $\mathcal{P}^* = (F', B')$ is a feasible 0-$(2n + 1)$-path. Moreover, it serves a smaller number of requests than $\mathcal{P}$ because $S^2_b \subseteq S^1_b$ due to the dominance between L^2_b and L^1_b. Note that this implication relies on elementarity of feasible paths. The reduced cost of $\mathcal{P}^* = (F', B')$ can be estimated as follows. Consider $\mathcal{P}^*$ as a backward path. Due to the PTI, the backward reduced costs of F' and F fulfill $\bar{c}_b(F') \leq \bar{c}_b(F)$. Moreover, the dominance between L^2_b and L^1_b implies $\bar{c}^2_b \leq \bar{c}^1_b$. Hence, the reduced cost $\bar{c}_b(F') + \bar{c}^2_b$ of $\mathcal{P}^*$ is not greater than the reduced cost $\bar{c}_b(F) + \bar{c}^1_b$ of $\mathcal{P}$. This means that $\mathcal{P}^*$ is an optimal path with fewer served requests as $\mathcal{P}$ which is a contradiction to the request-minimality of $\mathcal{P}$. □

3.2.3 Cutting Planes

We now present details on the cutting planes that are used in our branch-cut-and-price algorithm with a focus on their implication for the pricing subproblem.

Robust Cuts We use two families of *robust* cuts, i.e., inequalities on the aggregated flows of arcs $(i, j) \in A$, which can be incorporated into the master problem using expressions $x(\delta^+(S)) \leq rhs$ or $x(\delta^+(S)) \geq rhs$. Here, $\delta^+(S) = \{(i, j) \in A : i \in S, j \in V \setminus S\}$ denotes the set of arcs leaving the vertex subset $S \subset V$ and $x(\delta^+(S)) := \sum_{r\in\Omega} \sum_{(i,j)\in\delta^+(S)} b_{ijr}\lambda_r$ gives the flow out of the corresponding set S. Parameter b_{ijr} denotes the number of times route $r \in \Omega$ traverses arc (i, j). The two families of cuts that we use are rounded capacity cuts and 2-path cuts (Kohl *et al.*, 1999). Details on these cuts and the corresponding separation strategies can be found in the work of Ropke and Cordeau (2009).

The implication of both families of cuts are additional dual prices on arcs (i, j) that have to be included in the reduced costs $\bar{c}^f_{ij}$ and $\bar{c}^b_{ij}$ in the pricing subproblem. As a result, the forward (backward) reduced cost matrix does generally not fulfill the DTI (PTI). In

order to use the strong dominance rule in forward (backward) labeling, however, the DTI (PTI) is indispensable. Ropke and Cordeau (2009) have shown how to transform an arbitrary cost matrix into one that that fulfills the DTI while at the same time keeping the reduced costs of all feasible routes unchanged. They compute for each request $(j, j+n) \in R$ the maximum violation $\theta_j^f = \max_{i,k \in V}\{\bar{c}_{ik}^f - (\bar{c}_{i,j+n}^f + \bar{c}_{j+n,k}^f)\}^+$ of the DTI. Then, $\theta_j^f/2$ is subtracted from the reduced cost $\bar{c}_{ij}^f$ and $\bar{c}_{ji}^f$ of all in- and outgoing arcs of the corresponding pickup vertex $j \in P$. For the delivery vertex $j+n \in D$, the same amount $\theta_j^f/2$ is added to the reduced costs $\bar{c}_{i,j+n}^f$ and $\bar{c}_{j+n,i}^f$ of all in- and outgoing arcs. Since each feasible route either visits both the pickup and the delivery vertex of a request or none of the two, its reduced cost remains unchanged with this transformation. Clearly, an analog transformation can be performed to ensure the PTI for the backward reduced cost matrix $\bar{c}_{ij}^b, i, j \in V$ using the terms $\theta_j^b = \max_{i,k \in V}\{\bar{c}_{ik}^b - (\bar{c}_{ij}^b + \bar{c}_{jk}^b)\}^+$ for all $(j, j+n) \in R$.

The merge procedure has to take into account the modified cost matrices in forward and backward labeling. When merging suitable forward and backward labels L_f and L_b the reduced cost of the resulting route r can be computed as:

$$\bar{c}_r = \bar{c}_f + \bar{c}_b + \sum_{\substack{(k,k+n)\in \\ \{O_b \cap O_f\}}} \left(\pi_k + \theta_k^b + \theta_k^f\right) + \sum_{\substack{(k,k+n)\in \\ \{O_b \cup O_f\}\setminus\{O_b \cap O_f\}}} \frac{\pi_k + \theta_k^b + \theta_k^f}{2}.$$

The correctness follows straightforwardly by analog considerations as in the proof of Proposition 3.1. Note also that the proof of Proposition 3.2 is not affected by the altered reduced-cost formula.

Subset-Row Cuts Subset-row cuts were first introduced by Jepsen *et al.* (2008) for the *vehicle-routing problem with time windows* (VRPTW). They are Chvátal-Gomory rank-1 cuts based on a subset of the constraints in the master program. Because they change the structure of the ESPPRC pricing subproblem (each additional cut requires an additional resource), the subset-row cuts are *non-robust*. For the PDPTW, a subset-row cut is defined on a subset of requests. The cut for a request set $U_k \subset P$ and a parameter $1 \leq l \leq |U_k|$, in the following denoted by $SR(U_k, l)$, is given by $\sum_{r\in\Omega} \lfloor (\sum_{i \in U_k} a_{ir})/l \rfloor \lambda_r \leq \lfloor |U_k|/l \rfloor$. We restrict ourselves to those cuts defined for $l = 2$ and $|U_k| = 3$ as proposed by Jepsen *et al.* (2008) because they can be separated by straightforward enumeration. In the following, we write $SR(U_k)$ instead of $SR(U_k, 2)$.

The addition of subset-row cuts in the master problem requires the following adjustments to the pricing problem: Let $\sigma_k \leq 0$ be the dual price of the subset-row cut $SR(U_k)$. In forward labeling, the value σ_k must be subtracted from the reduced cost for every second visit to a request $(i, i+n) \in U_k$. Thereby, it is beneficial in terms of dominance to incorporate this penalty as early as possible into the reduced cost of a respective path. Thus, we subtract σ_k at the second visit to pickup vertices i with $(i, i+n) \in U_k$. Contrary, in backward labeling, the value σ_k is subtracted from the reduced cost for every second visit to delivery vertices i with $(i-n, i) \in U_k$. Therefore, additional binary resources

$sr_f(k)$ and $sr_b(k)$, one for each cut $SR(U_k)$, are necessary in forward and backward labeling, respectively, for indicating the parity of the number of times a pickup or delivery vertex of a request in U_k is visited.

Jepsen *et al.* (2008) have proposed a tailored dominance rule that avoids a pointwise comparison of all resources $sr_f(k)$ and $sr_b(k)$ reducing the number of incomparable labels significantly. In forward labeling, the dominance condition regarding the resource $\bar{c}_f$ changes as follows: Let L_f and L'_f be two labels with identical last vertex i and let $H := \{k : sr_f(k) = 0 \; and \; sr'_f(k) = 1\}$ be the index set of the new resources on which L_f is inferior compared to L'_f. Then, $\bar{c}_f - \sum_{k \in H} \sigma_k \leq \bar{c}'_f$ has to hold if L_f dominates L'_f. The tailored dominance rule in backward labeling regarding the resource $\bar{c}_b$ can be defined analogously.

The computation of the reduced cost of a merged path in the bidirectional labeling algorithm also has to account for the new resources and dual prices associated with each cut $SR(U_k)$. Let L_f and L_b be a pair of forward and backward labels fulfilling the merge conditions (3.4). If both L_f and L_b have visited an odd number of requests in U_k, i.e., $sr_f(k) = sr_b(k) = 1$, then the dual price σ_k of the subset-row cut $SR(U_k)$ has to be subtracted once more in the reduced cost of the merged path. This is similar to the merge procedure in labeling algorithms for many other ESPPRC variants. Additional care has to be taken in the proposed bidirectional labeling for the PDPTW because visited requests $(i, i+n) \in U_k$ are counted at the pickups in forward and at the deliveries in backward direction. Thus, if some requests $(i, i+n) \in U_k$ are open in the forward label L_f or in the backward label L_b, they have been counted twice (once at the pickup vertex in L_f and once at the delivery vertex in L_b) and, therefore, the dual price σ_k might have been subtracted too often. To repair this defect, we compute the number of requests that have been counted twice as $cnt_k := |U_k \cap (O_f \cup O_b)|$. Furthermore, let $\Theta_k := 1 - |sr_f(k) - sr_b(k)|$ be the indicator telling whether the sum of counted visits in both directions is even. If this sum is even, we have to add the dual price for every odd number of double-counts. Vice versa, if the sum of counted visits is odd, we have to add the dual price for every even number of double-counts. Consequently, we have to add $\lfloor \frac{cnt_k + \Theta_k}{2} \rfloor$ times σ_k to the reduced cost of the merged path to adjust for possible double-inclusions. Again, these changes in the computation of the reduced cost of the merge of labels L_f and L_b do not influence the validity of the arguments of the proof of Proposition 3.2.

3.2.4 Branching

Note first that branching on arcs is not possible in PDPs if one wants to preserve the DTI or PTI (see Ropke and Cordeau, 2009, for details). We therefore use the following hierarchical branching scheme. First, we try to branch on the number of vehicles. If the number of vehicles is integer, we branch on the outflow of a set of vertices as proposed for the CVRP by Naddef and Rinaldi (2002) and applied to the PDPTW by Ropke and Cordeau (2009). Both branching decisions can be implemented by including a single inequality of the form $x(\delta^+(S)) \leq rhs$ or $x(\delta^+(S)) \geq rhs$ (see Section 3.2.3) in the master problem.

3.3 Computational Results

In this section, we report our computational results on the comparison of monodirectional and bidirectional labeling for the solution of PDPTW pricing problems. All results were obtained using a standard PC with an Intel(R) Core(TM) i7-5930k processor clocked at 3.5 GHz, 64 GB RAM, and Windows 7 Enterprise. The algorithms were implemented in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2010. The callable library of CPLEX 12.6.0 was used for solving the RMPs.

Benchmark Instances The computational studies use two groups of instances from the literature, the 40 instances of Ropke and Cordeau (2009) called `RC`, and 30 instances of Li and Lim (2003) denoted by `LL`. The `RC` instances comprise 30 to 75 requests and have high vehicle fixed costs such that the minimization of the number of vehicles is the primary objective. Moreover, they are characterized by small vehicle capacities, narrow time windows, and time windows of corresponding pickup and delivery locations that are very close to each other. Consequently, these instances rarely allow multiple open requests at the same time. To be precise, on average only 22 percent of the generated labels during a forward labeling have additional open requests (not considering the request associated with the current vertex). As a result, most routes visit the pickup and delivery vertices of each request in direct sequence. Thus, the `RC` instances resemble more VRPTW instances without capacity restrictions rather than being true PDPTW instances. As a consequence, there is no considerable difference between weak and strong dominance in the labeling algorithm on such instances. We therefore modified the original set of RC instances by increasing the right hand side of all time windows by 25 units and the vehicle capacity by 10 units.

With this modification, 65 percent of the generated labels in a forward labeling have additional open requests.

These instances are denoted `RC+`.

The `LL` instances comprise around 100 requests and are based on the Solomon benchmark set for the VRPTW. No fixed costs for the vehicles are assumed. Furthermore, the lengths of the time windows vary strongly between the instances so that multiple open requests at the same time are possible.

In preliminary tests, we also detected that in almost all instances the time windows were distributed asymmetrically over the time horizon. As a result, the instances are much harder when solving them in backward direction. To achieve a balanced ratio of forward and backward labeling, we additionally built reversed instances by inverting the time windows, i.e., swapping pickup and delivery and taking new time windows $[b_{2n+1} - b_i, b_{2n+1} - a_i]$ for each vertex $i \in V$. To obtain instances that are really different from the original ones, we also permuted the demands of the requests randomly. The inverse instance sets is denoted by $\texttt{RC}^{\leftarrow}$, $\texttt{RC+}^{\leftarrow}$, and $\texttt{LL}^{\leftarrow}$ for `RC`, `RC+`, and `LL` instances, respectively.

Computational Setup To compare different labeling strategies for solving the PDPTW, they are all embedded into the same basic branch-cut-and-price algorithm. It has the following components. To speed up the column-generation process, the heuristic pricer uses arc-reduced networks (Irnich and Desaulniers, 2005, p. 58). When the solution of the RMP is fractional, we first separate robust cuts (rounded capacity cuts and 2-path cuts) followed by the separation of the non-robust SR cuts. If no more cuts are found, branching is performed. As done in the literature (Ropke and Cordeau, 2009), we branch on the number of vehicles before separating cuts for the `RC`, `RC+`, $\texttt{RC}^{\leftarrow}$, and $\texttt{RC+}^{\leftarrow}$ instances. Cuts are separated at all branch-and-bound nodes. For the SR cuts, however, we set a limit of 60 cuts in total.

Overall, we compare six different *labeling strategies* for the solution of the ESPPRC pricing problems: pure (monodirectional) forward and backward labeling (`Fw-S` and `Bw-S`), bidirectional labeling with a static half-way point using strong dominance in the forward part and weak or strong dominance in the backward part (`Bi-St-SW` or `Bi-St-SS`), and the two analog strategies `Bi-Dy-SW` and `Bi-Dy-SS` for bidirectional labeling with a dynamic half-way point as described in Chapter 2. The different strategies are summarized in Table 3.1. Note that we refrain from presenting results of the bidirectional labeling strategies with weak dominance in the forward part and strong dominance in the backward part (`WS`), because the overall instance set is relatively symmetric and no significant difference to the results with dominance `SW` occurred in preliminary tests.

	Direction(s)		Half-way Point		Dominance Fw		Dominance Bw	
Strategy	Forw.	Backw.	Static	Dynamic	Strong	Weak	Strong	Weak
`Fw-S`	×				×			
`Bw-S`		×					×	
`Bi-St-SW`	×	×	×		×			×
`Bi-St-SS`	×	×	×		×		×	
`Bi-Dy-SW`	×	×		×	×			×
`Bi-Dy-SS`	×	×		×	×		×	

Table 3.1: Labeling strategies

Experimental Results We conduct two types of tests to measure the impact of the different labeling strategies. In the first type of test, every instance of the ESPPRC pricing problem is solved using all considered labeling strategies in each iteration of the linear relaxation of the master program. In this way, it is possible to directly compare the different labeling strategies because they all solve the exact same ESPPRC instances (identical dual prices) in the course of the branch-cut-and-price algorithm. Because each pricing problem has to be solved with all six labeling strategies, the time limit is set to two hours per instance in these tests.

For each PDPTW instance and labeling strategy, we compute the average ratio of the pricing time compared to the baseline strategy `Fw-S` as the geometric mean over all pricing

iterations. In order to reduce inaccuracies in time measurement, we take into account only those pricing iterations that consumed more than 0.1 seconds of computation time for all six labeling strategies. Whenever the linear relaxation is not completely solved within the time limit, the average is taken over the iterations solved by all strategies up to this point.

Table 3.2 summarizes the results for this first type of experiment. The first column indicates whether only the pricing iterations for the linear relaxation of (3.1) (*LP*) or those of the entire branch-and-bound tree (*B&B*) are considered. The second column (*Group*) gives the instance group and the third column (*#inst*) gives the number of instances for which there was at least one pricing iteration requiring at least the minimal time of 0.1 seconds. The fourth column (*#BB*) shows the average number of solved branch-and-bound nodes, the fifth column (*#SR*) shows the average number of added subset-row cuts, and the sixth column (*#RO*) shows the average number of added robust cuts.

The remaining columns present the geometric means over the instances of the pricing time ratios of the different labeling strategies compared to forward labeling.

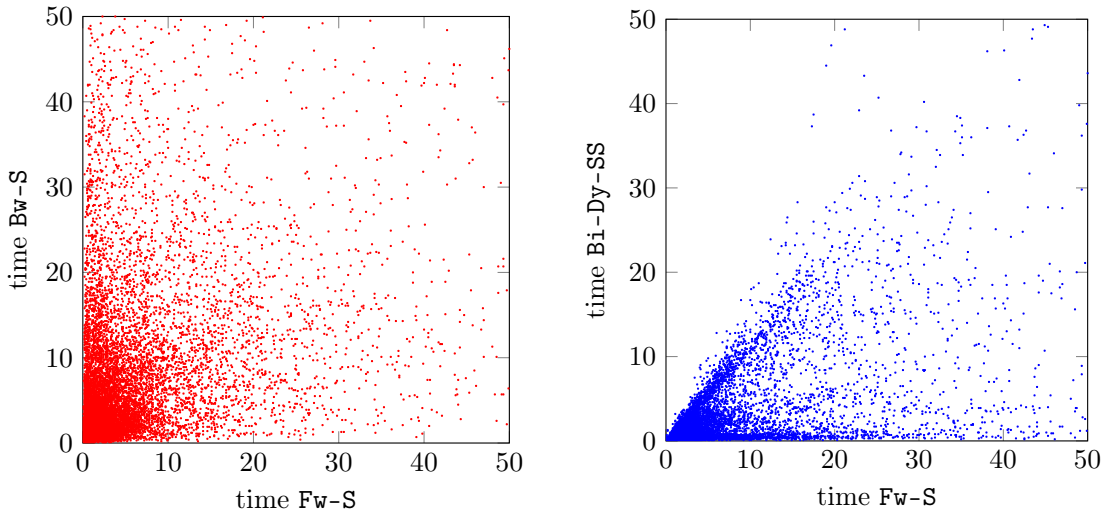
						Geometric mean of ratio of pricing times relative to `Fw-S`				
	Group	#inst	#BB	#SR	#RO	`Bw-S`	`Bi-St-SW`	`Bi-St-SS`	`Bi-Dy-SW`	`Bi-Dy-SS`
LP	`RC`	13/40				1.23	1.34	1.23	1.11	1.08
	`RC+`	33/40				1.41	1.85	1.44	1.04	0.93
	`LL`	21/30				1.21	2.67	1.02	1.10	0.90
	`RC`$^{\leftarrow}$	13/40				0.96	1.11	0.99	1.06	1.01
	`RC+`$^{\leftarrow}$	32/40				0.87	1.06	0.86	0.98	0.83
	`LL`$^{\leftarrow}$	21/30				0.86	4.12	0.95	1.05	0.77
	Total	133/220				1.07	1.78	1.06	1.06	0.92
B&B	`RC`	17/40	21.3	40.2	24.4	1.05	0.48	0.45	0.40	0.38
	`RC+`	35/40	7.4	25.0	12.4	1.61	1.11	0.89	0.67	0.58
	`LL`	21/30	0.4	3.2	1.0	1.21	2.63	1.01	1.09	0.90
	`RC`$^{\leftarrow}$	14/40	22.2	38.8	27.8	1.15	0.63	0.58	0.50	0.48
	`RC+`$^{\leftarrow}$	35/40	11.5	29.0	21.4	1.04	0.83	0.68	0.64	0.55
	`LL`$^{\leftarrow}$	21/30	0.4	2.4	1.3	0.86	4.03	0.94	1.05	0.77
	Total	143/220				1.13	1.20	0.73	0.68	0.58

Table 3.2: Comparison of different labeling strategies on same ESPPRC instances (identical dual prices)

The results reveal that employing the strong dominance in both directions of a bidirectional labeling algorithm is beneficial: Both strategies `Bi-St-SS` and `Bi-Dy-SS` dominate their respective counterpart `Bi-St-SW` or `Bi-Dy-SW` that uses the weak dominance in the backward part. Moreover, strategy `Bi-Dy-SS` clearly outperforms the other strategies.

Especially when comparing results for the entire branch-and-bound tree `Bi-Dy-SS` can on average decrease the pricing time by more than 40 percent when compared to the forward labeling which constitutes the state of the art. It is also more than 15 percent faster than the runner-up strategy `Bi-Dy-SW`. Noticeable is that the advantage of the bidirectional labeling strategies compared to their monodirectional counterparts becomes significant when going beyond the linear relaxation. Apparently, they can better handle the increasing difficulty of the pricing problems that stems from the addition of the non-robust SR cuts. The only marginal difference between the linear relaxation and the entire branch-and-bound tree for the 21 considered `LL` instances can be explained by the fact that only for three of these instances additional pricing iterations were performed after solving the root node (either because the LP solution was already integer or the run time limit was reached).

Table 3.2 also shows that bidirectional labeling with a static half-way point and weak dominance in the backward part performs rather disappointing: `Bi-St-SW` is clearly inferior to strategy `Fw-S`. These findings are especially true for the original instance sets (recall that they are much harder to solve in backward than in forward direction, see also column `Bw-S` of Table 3.2) and are in line with the findings of Ropke and Cordeau (2009) for their prototype implementation of strategy `Bi-St-SW`.



(a) `Bw-S` against `Fw-S` **(b)** `Bi-Dy-SS` against `Fw-S`

Figure 3.1: Comparison of individual pricing times (in seconds) of three labeling strategies

Figure 3.1 displays two scatter plots of the solution times of individual pricing iterations of `Bw-S` and `Bi-Dy-SS` compared to `Fw-S`. Figure 3.1a (on the left-hand side) shows that solution times of the two pure monodirectional strategies `Bw-S` to `Fw-S` behave almost randomly. Indeed, it seems impossible to predict whether forward or backward labeling requires more computational effort for a given ESPPRC instance. Figure 3.1b (on the

right-hand side) shows the same plot for the new and best-performing labeling strategy `Bi-Dy-SS` in comparison to `Fw-S`. The majority of the data points lay under the diagonal showing that in almost all cases `Bi-Dy-SS` is performing at least as well as `Fw-S`. The many data points close to the abscissa stand for those cases in which `Bi-Dy-SS` is strongly superior to `Fw-S`.

In a second type of experiments, we compare the six full-fledged branch-cut-and-price algorithms originating from the use of the six different labeling strategies. Clearly, the generation of different columns when using different labeling algorithms in the pricing problems leads to a different course of the algorithms resulting in different ESPPRC instances, different results of the RMPs, and different branch-and-bound trees for the six algorithms. The test aims at showing the impact of the labeling strategies on the solution of the overall problem and at the analysis whether the structure of the generated columns has an influence on the overall solution time. To further accelerate pricing and to obtain reasonable overall results, an additional pricing heuristic that uses a heuristic dominance relation ignoring the open requests was included in the algorithms in these experiments. Note that the heuristic dominance reduces the difference between branch-cut-and-price algorithms using labeling strategies with strong and weak dominance in the backward part. The time limit per PDPTW instance was set to one hour for these experiments.

	Fw-S		Bw-S		Bi-St-SW		Bi-St-SS		Bi-Dy-SW		Bi-Dy-SS	
Group	#opt	time	#opt	time	#opt	time	#opt	time	#opt	time	#opt	time
RC	32	372.2	33	483.3	32	215.6	32	215.6	33	147.2	33	181.6
RC+	13	1607.7	12	1949.0	18	896.5	18	824.7	18	783.6	18	804.9
LL	17	502.0	18	521.7	12	1404.6	17	474.6	17	553.5	18	251.4
RC$^{\leftarrow}$	34	557.9	31	640.4	35	336.2	35	330.9	36	306.8	35	338.9
RC+$^{\leftarrow}$	16	1441.4	13	1667.2	19	1201.4	19	1036.2	19	746.5	20	629.6
LL$^{\leftarrow}$	16	662.5	16	650.6	12	1549.1	17	450.4	18	338.1	18	323.2
Total	128	734.2	123	818.4	128	727.1	138	498.1	141	423.3	142	389.3

Table 3.3: Comparison of different labeling strategies on separate runs: number of instances solved to proven optimality (*#opt*) and average run times (in seconds)

The results of the second type of experiment can be found in Table 3.3. It reports for each instance group (*Group*) and each of the six labeling strategies the following values: the number of instances solved to proven optimality within the time limit (*#opt*) and the average solution *time* in seconds. The average is taken only over those instances that were solved to optimality by at least one of the six branch-cut-and-price algorithms (counting the time limit for labeling strategies that did not finish). Overall, Table 3.3 confirms the result of the first type of test: The branch-cut-and-price algorithm with labeling strategy `Bi-Dy-SS` produces the best results concerning both number of solved instances and total solution times. However, over the subsets `RC`, `RC+`, and RC$^{\leftarrow}$, the

strategy `Bi-Dy-SW` performs slightly better than `Bi-Dy-SS` and solves two instances that are unsolved by `Bi-Dy-SS`, while three instances can only be solved by `Bi-Dy-SS`. As there are almost no additional open requests in the subsets `RC` and `RC`$^{\leftarrow}$, the difference in performance cannot be attributed to the two dominance strategies (`SS` vs. `SW`) but to different branching decisions. Note also that for `RC+` the relative difference in runtime between `Bi-Dy-SS` and `Bi-Dy-SW` is marginal.

Moreover, all algorithms using bidirectional labeling outperform those using monodirectional labeling.

These observations are also confirmed by the *performance profiles* of the six branch-cut-and-price algorithms depicted in Figure 3.2. According to Dolan and Moré (2002), given a set $\mathcal{A} = \{A_1, A_2, \ldots, A_p\}$ of algorithms, the performance profile $\rho_A(\tau)$ of an algorithm $A \in \mathcal{A}$ is the fraction of instances that algorithm A can solve within a factor τ of the fastest algorithm of $\mathcal{A}$. Any unsolved instances are taken into account with infinite run time. In particular, $\rho_A(1)$ is the percentage of instances on which A wins. For the same algorithm A, $100 - \rho_A(\infty)$ is the percentage of unsolved instances.

Overall, the performance profiles in Figure 3.2 confirm that the branch-cut-and-price with labeling strategy `Bi-Dy-SS` clearly dominates all other algorithms on the 220 PDPTW test instances of which 65.5 percent are solved by at least one algorithm. Indeed, `Bi-Dy-SS` is at most 1.5 times slower than the fastest algorithm for more than 90 percent of the instances solved by at least one algorithm. Moreover, the branch-cut-and-price algorithms with bidirectional labeling and strong dominance in both directions are consistently and significantly superior to their counterparts with weak dominance in the backward labeling.

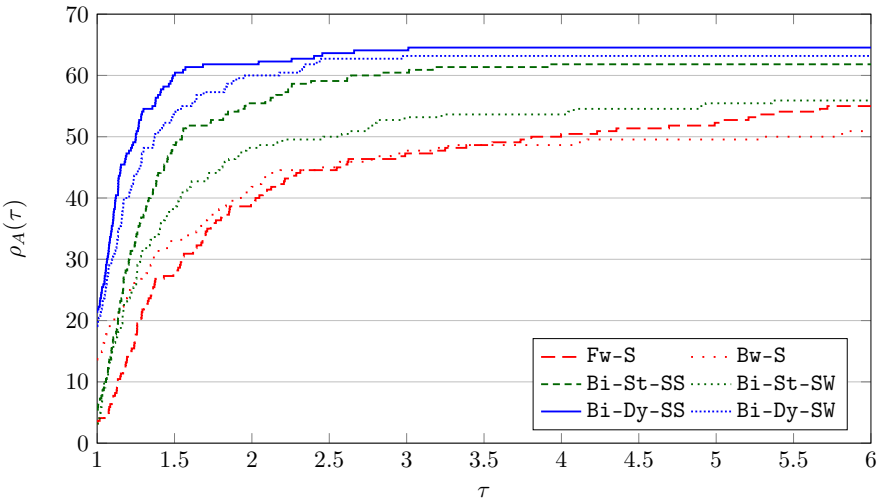


Figure 3.2: Performance profiles of branch-cut-and-price algorithms using the six different labeling strategies

Comparison with Baldacci *et al.* (2011c) A dedicated exact approach for the PDPTW based on a set-partitioning formulation, bounding procedures, cut-and-column generation procedures, and route enumeration was suggested by Baldacci *et al.* (2011c). The integer programming solver CPLEX is finally used to obtain integer solutions. The authors use two different algorithmic setups for the `RC` and `LL` instances exploiting their specific characteristics. Although this approach is based on a different column-generation principle, the overall results are comparable. For the `RC` benchmark, they solved 32 of the 40 instances to proven optimality within one hour (`Bi-Dy-SS` produces 33 optima, see Table 3.3). Seven of the `LL` benchmark instances remain completely untractable for both approaches, since the algorithm of Baldacci *et al.* runs out of memory and ours fails to solve the linear relaxation (root node). Within one hour of computation time, the approach of Baldacci *et al.* cannot close the integrality gap for one other instance while ours fails on another five instances. As the enumeration approach is very much tailored to the PDPTW, it seems hard to generalize it to other vehicle-routing problems with pickup-and-delivery structure. An instance-wise comparison can be found in the Appendix.

3.4 Conclusions

In this paper, we introduced a bidirectional labeling algorithm to be used in column-generation based algorithms for the PDPTW. We construct different cost matrices in forward and backward direction that fullfill the DTI and the PTI, respectively. This enables, for the first time, the use of strong dominance rules in both directions. Extensive computational tests showed that the new bidirectional labeling outperforms former labeling strategies.

Moreover, the new bidirectional labeling can be generalized to other variants of vehicle-routing problems with a pickup-and-delivery structure. It can be expected that bidirectional labeling with strong dominance performs even better for more difficult PDPs, since the advantage of bidirectional over monodirectional approaches typically increases with problem difficulty, e.g., more ESPPRC resources or weaker dominance.

A promising avenue for future research is the extension of the bidirectional labeling to the *ng*-path relaxations of the ESPPRC (Baldacci *et al.*, 2012a). Such an extension is, however, not straightforward. When using an *ng*-path relaxation in the PDPTW with LIFO loading, Cherkesly *et al.* (2015) have shown that non-elementary paths can be incorrectly dominated. Note that similar anomalies result from the widely-used preprocessing and arc-elimination techniques in non-elementary pricing for PDPs. As a consequence, the lower bounds computed with these relaxations are not unique. However, bounds are valid as exploited in the selective pricing paradigm of Desaulniers *et al.* (2016b). The incorrect dominance is particularly critical in the bidirectional labeling because the existence of compatible forward and backward pairs of labels can no longer be ensured with arguments similar to those used in the proof of Proposition 3.2.

Bibliography

Baldacci, R., Bartolini, E., and Mingozzi, A. (2011b). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, **59**(2), 414–426.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012a). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, **24**(3), 356–371.

Battarra, M., Cordeau, J.-F., and Iori, M. (2014). Pickup-and-delivery problems for goods transportation. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 6, pages 161–191. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Cherkesly, M., Desaulniers, G., and Laporte, G. (2015). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, **49**(4), 752–766.

Cherkesly, M., Desaulniers, G., Irnich, S., and Laporte, G. (2016). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. *European Journal of Operational Research*, **250**(3), 782–793.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57–93. Springer, Boston, MA.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Desaulniers, G., Contardo, C., and Pecin, D. (2016). Selective pricing in branch-price-and-cut algorithms for vehicle routing. Les Cahiers du GERAD G-2016-110, GERAD, Montréal, Canada.

Doerner, K. F. and Salazar-González, J.-J. (2014). Pickup-and-delivery problems for people transportation. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 7, pages 193–212. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pick-up and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7–22.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, New York, NY.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, **33**(1), 101–116.

Li, H. and Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools* **12**(2), 173–186.

Naddef, D. and Rinaldi, G. (2002). Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 3, pages 53–84. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Parragh, S., Doerner, K., and Hartl, R. (2008). A survey on pickup and delivery problems, Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, **58**(2), 81–117.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, **43**(3), 267–286.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Veenstra, M., Cherkesly, M., Desaulniers, G., and Laporte, G. (2017). The pickup and delivery problem with time windows and handling operations. *Computers & Operations Research*, **77**, 127–140.

Appendix

In the following tables, we provide instance-wise information about the solution process of our best labeling strategy `Bi-Dy-SS`. We show our lower bound (LB), the optimal solution (opt) whenever we solved the instance to proven optimality, the run time ($time$), the number of branch-and-bound nodes ($\#BB$), the number of added subset-row cuts ($\#SR$) and the number of added robust cuts ($\#RO$). Additionally, for the instances tackled by Baldacci *et al.* (2011c), we provide their run time ($time$) and the optimal solution (opt) whenever they solved the instance.

While our maximal computation time (indicated by TL) was 3600 seconds (1 hour), Baldacci *et al.* (2011c) allowed 10 times longer computation times of 36,000 seconds (10 hours). An entry *MEM* indicates that the algorithm of Baldacci *et al.* (2011c) caused an out-of-memory exception.

	Our results						Baldacci *et al.* (2011c)	
Instance	LB	opt	time	#BB	#SR	#RO	time	opt
*AA*30	31119.1	31119.1	11.8	3	3	0	3.2	31119.1
*AA*35	31299.8	31299.8	23.7	5	3	63	24.2	31299.8
*AA*40	31515.9	31515.9	17.8	1	0	96	26.1	31515.9
*AA*45	31759.8	31759.8	8.0	1	0	0	10.2	31759.8
*AA*50	41775.0	41775.0	48.4	9	32	28	97.6	41775.0
*AA*55	41907.8	41907.8	30.7	3	0	33	9.7	41907.8
*AA*60	42140.7	42140.7	86.6	10	60	22	91.1	42140.7
*AA*65	42250.2	42250.2	80.3	3	50	149	72.5	42250.2
*AA*70	42452.3	42452.3	250.9	37	60	28	438.2	42452.3
*AA*75	52461.6	52461.6	117.3	15	60	49	6442.4	52461.6
*BB*30	31086.3	31086.3	5.5	3	0	0	2.5	31086.3
*BB*35	31281.2	31281.2	10.7	5	9	0	5.1	31281.2
*BB*40	31493.4	31493.4	13.8	3	10	4	4.8	31493.4
*BB*45	41555.1	41555.1	18.5	3	30	1	7.2	41555.1
*BB*50	41701.0	41701.0	31.8	7	60	10	9.6	41701.0
*BB*55	41885.7	41885.7	15.2	3	0	0	8.2	41885.7
*BB*60	62420.1	62420.1	51.3	15	60	0	17.6	62420.1
*BB*65	62639.1	62639.1	74.6	19	60	0	44.0	62639.1
*BB*70	62951.0	62951.0	133.3	23	60	5	418.1	62951.0
*BB*75	63127.5	63127.5	3089.3	391	60	0	MEM	
*CC*30	31087.7	31087.7	9.3	3	40	0	3.9	31087.7
*CC*35	31230.6	31230.6	14.8	3	43	7	5.8	31230.6
*CC*40	31358.5	31358.5	39.2	5	60	3	26.4	31358.5
*CC*45	31509.1	31509.1	59.3	3	50	29	17.0	31509.1
*CC*50	41685.3	41685.3	99.8	23	60	12	38.9	41685.3
*CC*55	41836.3	41836.3	263.3	53	60	5	185.2	41836.3
*CC*60	42007.9		TL	298	60	48	2128.2	42009.3
*CC*65	42157.6		TL	217	60	54	7111.2	42164.0
*CC*70	42386.9		TL	17	60	10	5565.7	52201.7
*CC*75	52359.0	52359.0	914.5	103	60	5	259.6	52359.0
*DD*30	21133.3	21133.3	13.6	3	0	86	7.1	21133.3
*DD*35	31210.9	31210.9	19.5	3	10	26	5.4	31210.9
*DD*40	31352.2	31352.2	33.7	3	30	28	13.7	31352.2
*DD*45	31483.9	31483.9	43.2	3	30	15	62.9	31483.9
*DD*50	31600.9	31600.9	174.7	7	60	15	96.7	31600.9
*DD*55	31743.3	31743.3	189.5	3	60	17	36.5	31743.3
*DD*60	32004.4		TL	28	60	7	13048.3	32069.2
*DD*65	42103.5		TL	186	60	38	25929.1	42107.3
*DD*70	42208.2		TL	113	60	17	20737.5	42214.2
*DD*75	42350.4		TL	134	60	24	34718.6	42359.9

	Our results					
Instance	LB	opt	time	#BB	#SR	#RO
$AA30+$	20893.8	20893.8	25.1	3	20	5
$AA35+$	21008.6	21008.6	44.1	3	20	19
$AA40+$	31132.7	31132.7	49.7	3	30	46
$AA45+$	31278.4	31278.4	191.6	5	60	44
$AA50+$	31361.1	31361.1	1499.5	49	60	55
$AA55+$	31471.1	31471.1	309.8	7	60	60
$AA60+$	31597.3		TL	82	60	27
$AA65+$	31676.7	31676.7	328.7	7	60	44
$AA70+$	31816.8		TL	25	60	33
$AA75+$	31901.4		TL	7	60	30
$BB30+$	20941.5	20941.5	8.1	3	0	0
$BB35+$	21089.2	21089.2	11.8	3	0	0
$BB40+$	31200.5	31200.5	35.6	3	26	9
$BB45+$	31283.4	31283.4	165.6	19	60	38
$BB50+$	31388.7	31388.7	60.1	3	10	0
$BB55+$	31576.7	31576.7	838.9	5	60	18
$BB60+$	41971.6		TL	206	60	8
$BB65+$	42170.0	42170.0	2668.3	101	60	1
$BB70+$	52223.2		TL	277	60	11
$BB75+$	52278.2		TL	177	60	20
$CC30+$	20901.6	20901.6	94.2	3	30	2
$CC35+$	21000.2	21000.2	272.6	3	10	0
$CC40+$	21102.8		TL	4	60	10
$CC45+$	21052.9		TL	2	0	0
$CC50+$	24141.5		TL	1	0	0
$CC55+$	6642.02		TL	1	0	0
$CC60+$	0		TL	0	0	0
$CC65+$	0		TL	0	0	0
$CC70+$	0		TL	0	0	0
$CC75+$	0		TL	0	0	0
$DD30+$	20919.6	20919.6	1866.1	11	60	6
$DD35+$	21014.6		TL	6	60	0
$DD40+$	21078.2	21078.2	427.0	3	20	2
$DD45+$	21218.1		TL	2	40	1
$DD50+$	21328.6		TL	2	10	4
$DD55+$	22156.1		TL	1	0	0
$DD60+$	24079.9		TL	1	0	0
$DD65+$	27275.1		TL	1	0	0
$DD70+$	28336.7		TL	1	0	0
$DD75+$	0		TL	0	0	0

	Our results						Baldacci *et al.* (2011c)	
Instance	LB	opt	time	#BB	#SR	#RO	time	opt
*LC*1_2_1	2704.57	2704.57	3.5	1	0	0	3.3	2704.57
*LC*1_2_2	2764.56	2764.56	16.1	1	0	0	21.5	2764.56
*LC*1_2_3	2772.18	2772.18	231.0	1	0	0	114.9	2772.18
*LC*1_2_4	0		TL	0	0	0	MEM	
*LC*1_2_5	2702.05	2702.05	5.2	1	0	0	4.8	2702.05
*LC*1_2_6	2701.04	2701.04	7.1	1	0	0	7.4	2701.04
*LC*1_2_7	2701.04	2701.04	6.9	1	0	0	7.7	2701.04
*LC*1_2_8	2689.83	2689.83	12.1	1	0	0	16.0	2689.83
*LC*1_2_9	2724.24	2724.24	352.4	1	15	0	55.3	2724.24
*LC*1_2_10	2739.49		TL	1	26	12	137.1	2741.60
*LR*1_2_1	4819.12	4819.12	2.9	1	0	0	1.6	4819.12
*LR*1_2_2	4093.05	4093.05	51.1	1	0	0	20.6	4093.05
*LR*1_2_3	0		TL	0	0	0	3690.8	3486.80
*LR*1_2_4	0		TL	0	0	0	MEM	
*LR*1_2_5	4221.62	4221.62	12.3	1	3	0	2.6	4221.62
*LR*1_2_6	3763.00	3763.00	222.6	1	0	0	180.9	3763.00
*LR*1_2_7	0		TL	0	0	0	MEM	
*LR*1_2_8	0		TL	0	0	0	MEM	
*LR*1_2_9	3953.47	3953.47	275.3	1	49	2	15.4	3953.47
*LR*1_2_10	3375.85		TL	1	0	0	1376.7	3386.30
*LRC*1_2_1	3606.06	3606.06	12.0	1	0	12	3.1	3606.06
*LRC*1_2_2	3292.43	3292.43	366.6	1	10	9	322.3	3292.43
*LRC*1_2_3	0		TL	0	0	0	MEM	
*LRC*1_2_4	0		TL	0	0	0	MEM	
*LRC*1_2_5	3715.81	3715.81	238.5	1	18	8	42.1	3715.81
*LRC*1_2_6	3360.86	3360.86	23.7	1	0	0	7.0	3360.86
*LRC*1_2_7	3317.74	3317.74	2686.0	7	60	10	408.2	3317.74
*LRC*1_2_8	3024.26		TL	1	0	0	1562.7	3086.50
*LRC*1_2_9	2982.58		TL	1	0	2	1757.2	3053.80
*LRC*1_2_10	0		TL	0	0	0	MEM	

	Our results					
Instance	LB	opt	time	#BB	#SR	#RO
$AA30^{\leftarrow}$	31179.1	31179.1	35.4	11	50	6
$AA35^{\leftarrow}$	31349.9	31349.9	31.3	5	10	7
$AA40^{\leftarrow}$	41521.3	41521.3	20.4	3	0	10
$AA45^{\leftarrow}$	41749.0	41749.0	30.8	3	0	9
$AA50^{\leftarrow}$	41912.8	41912.8	26.0	3	0	0
$AA55^{\leftarrow}$	42069.1	42069.1	68.8	11	60	3
$AA60^{\leftarrow}$	42254.5	42254.5	70.7	9	60	5
$AA65^{\leftarrow}$	42362.9	42362.9	61.7	3	60	12
$AA70^{\leftarrow}$	42555.3	42555.3	42.8	1	30	11
$AA75^{\leftarrow}$	52527.3	52527.3	589.2	75	60	20
$BB30^{\leftarrow}$	31298.4	31298.4	8.5	1	0	8
$BB35^{\leftarrow}$	41610.7	41610.7	24.9	3	0	180
$BB40^{\leftarrow}$	41787.6	41787.6	15.9	1	0	124
$BB45^{\leftarrow}$	51903.2	51903.2	43.1	7	60	88
$BB50^{\leftarrow}$	51998.3	51998.3	26.6	3	0	81
$BB55^{\leftarrow}$	52215.0	52215.0	51.3	5	40	125
$BB60^{\leftarrow}$	52469.7	52469.7	21.1	3	0	0
$BB65^{\leftarrow}$	62654.0	62654.0	321.7	51	60	82
$BB70^{\leftarrow}$	62893.6	62893.6	66.4	3	40	2
$BB75^{\leftarrow}$	62969.7	62969.7	295.0	49	60	11
$CC30^{\leftarrow}$	21131.9	21131.9	7.9	3	0	0
$CC35^{\leftarrow}$	31214.5	31214.5	21.6	3	20	137
$CC40^{\leftarrow}$	31347.1	31347.1	72.3	13	60	13
$CC45^{\leftarrow}$	31463.4	31463.4	382.9	53	60	1
$CC50^{\leftarrow}$	31711.0		TL	21	60	3
$CC55^{\leftarrow}$	41753.5	41753.5	150.1	13	60	26
$CC60^{\leftarrow}$	41901.8	41901.8	579.5	49	60	91
$CC65^{\leftarrow}$	42098.2		TL	163	60	9
$CC70^{\leftarrow}$	42308.5		TL	77	60	12
$CC75^{\leftarrow}$	42479.5		TL	6	60	16
$DD30^{\leftarrow}$	21167.7	21167.7	6.9	3	0	0
$DD35^{\leftarrow}$	31295.4	31295.4	16.6	3	16	0
$DD40^{\leftarrow}$	31511.9	31511.9	87.7	27	60	0
$DD45^{\leftarrow}$	31680.9	31680.9	47.2	7	60	3
$DD50^{\leftarrow}$	31800.6	31800.6	65.6	3	50	3
$DD55^{\leftarrow}$	31991.9		TL	35	60	0
$DD60^{\leftarrow}$	42012.3	42012.3	448.4	39	60	0
$DD65^{\leftarrow}$	42131.7	42131.7	3265.3	251	60	0
$DD70^{\leftarrow}$	42249.5	42249.5	558.7	31	60	2
$DD75^{\leftarrow}$	42389.5	42389.5	1038.5	13	60	8

	Our results					
Instance	LB	opt	time	#BB	#SR	#RO
$AA30+^{\leftarrow}$	20918.6	20918.6	38.3	3	39	20
$AA35+^{\leftarrow}$	21037.7	21037.7	67.6	3	39	1
$AA40+^{\leftarrow}$	31133.6	31133.6	72.9	3	50	41
$AA45+^{\leftarrow}$	31286.4	31286.4	94.1	3	30	137
$AA50+^{\leftarrow}$	31397.8	31397.8	596.8	29	60	74
$AA55+^{\leftarrow}$	31481.8	31481.8	174.2	7	60	51
$AA60+^{\leftarrow}$	31581.2	31581.2	848.8	39	60	50
$AA65+^{\leftarrow}$	31689.2		TL	48	60	82
$AA70+^{\leftarrow}$	31827.5		TL	19	60	46
$AA75+^{\leftarrow}$	31914.3		TL	6	60	38
$BB30+^{\leftarrow}$	31084.2	31084.2	14.0	3	0	0
$BB35+^{\leftarrow}$	31320.8	31320.8	40.9	3	28	27
$BB40+^{\leftarrow}$	31433.9	31433.9	29.3	1	6	0
$BB45+^{\leftarrow}$	41543.6	41543.6	440.3	89	60	78
$BB50+^{\leftarrow}$	41605.7	41605.7	413.9	71	60	34
$BB55+^{\leftarrow}$	41772.7	41772.7	1664.8	135	60	26
$BB60+^{\leftarrow}$	41936.0	41936.0	272.5	7	60	65
$BB65+^{\leftarrow}$	42162.9		TL	109	60	9
$BB70+^{\leftarrow}$	52261.1		TL	239	60	36
$BB75+^{\leftarrow}$	52313.6		TL	191	60	33
$CC30+^{\leftarrow}$	20850.6	20850.6	1041.9	7	60	14
$CC35+^{\leftarrow}$	20953.8	20953.8	1266.7	3	50	15
$CC40+^{\leftarrow}$	21021.8	21021.8	2049.4	3	20	4
$CC45+^{\leftarrow}$	17720.4		TL	1	0	0
$CC50+^{\leftarrow}$	0		TL	0	0	0
$CC55+^{\leftarrow}$	0		TL	0	0	0
$CC60+^{\leftarrow}$	0		TL	0	0	0
$CC65+^{\leftarrow}$	0		TL	0	0	0
$CC70+^{\leftarrow}$	0		TL	0	0	0
$CC75+^{\leftarrow}$	0		TL	0	0	0
$DD30+^{\leftarrow}$	20892.6	20892.6	141.4	3	60	10
$DD35+^{\leftarrow}$	21052.4		TL	5	60	10
$DD40+^{\leftarrow}$	21177.3	21177.3	468.2	3	20	7
$DD45+^{\leftarrow}$	21317.4		TL	2	40	7
$DD50+^{\leftarrow}$	31381.0	31381.0	2855.8	63	60	6
$DD55+^{\leftarrow}$	31453.7		TL	2	60	14
$DD60+^{\leftarrow}$	25009.7		TL	1	0	0
$DD65+^{\leftarrow}$	22190.9		TL	1	0	0
$DD70+^{\leftarrow}$	0		TL	0	0	0
$DD75+^{\leftarrow}$	0		TL	0	0	0

	Our results					
Instance	LB	opt	time	#BB	#SR	#RO
$LC1\_2\_1^{\leftarrow}$	2704.57	2704.57	3.8	1	0	0
$LC1\_2\_2^{\leftarrow}$	2764.56	2764.56	11.0	1	0	0
$LC1\_2\_3^{\leftarrow}$	2772.18	2772.18	1338.4	1	0	0
$LC1\_2\_4^{\leftarrow}$	0		TL	0	0	0
$LC1\_2\_5^{\leftarrow}$	2702.05	2702.05	6.0	1	0	0
$LC1\_2\_6^{\leftarrow}$	2701.04	2701.04	4.9	1	0	0
$LC1\_2\_7^{\leftarrow}$	2701.04	2701.04	8.8	1	0	0
$LC1\_2\_8^{\leftarrow}$	2689.83	2689.83	16.1	1	0	0
$LC1\_2\_9^{\leftarrow}$	2724.24	2724.24	528.8	1	15	0
$LC1\_2\_10^{\leftarrow}$	2740.49		TL	1	25	6
$LR1\_2\_1^{\leftarrow}$	4819.12	4819.12	3.5	1	0	0
$LR1\_2\_2^{\leftarrow}$	4093.05	4093.05	26.1	1	0	0
$LR1\_2\_3^{\leftarrow}$	0		TL	0	0	0
$LR1\_2\_4^{\leftarrow}$	0		TL	0	0	0
$LR1\_2\_5^{\leftarrow}$	4221.62	4221.62	12.1	1	3	0
$LR1\_2\_6^{\leftarrow}$	3763.00	3763.00	373.5	1	0	0
$LR1\_2\_7^{\leftarrow}$	0		TL	0	0	0
$LR1\_2\_8^{\leftarrow}$	0		TL	0	0	0
$LR1\_2\_9^{\leftarrow}$	3953.47	3953.47	293.3	1	51	4
$LR1\_2\_10^{\leftarrow}$	0		TL	0	0	0
$LRC1\_2\_1^{\leftarrow}$	3606.06	3606.06	12.7	1	0	21
$LRC1\_2\_2^{\leftarrow}$	3292.43	3292.43	792.6	1	13	20
$LRC1\_2\_3^{\leftarrow}$	0		TL	0	0	0
$LRC1\_2\_4^{\leftarrow}$	0		TL	0	0	0
$LRC1\_2\_5^{\leftarrow}$	3715.81	3715.81	273.5	1	27	5
$LRC1\_2\_6^{\leftarrow}$	3360.86	3360.86	21.0	1	0	0
$LRC1\_2\_7^{\leftarrow}$	3317.74	3317.74	2092.1	2	60	10
$LRC1\_2\_8^{\leftarrow}$	0		TL	0	0	0
$LRC1\_2\_9^{\leftarrow}$	2982.58		TL	1	0	0
$LRC1\_2\_10^{\leftarrow}$	0		TL	0	0	0

Chapter 4

Branch-and-Price-and-Cut for the Truck-and-Trailer Routing Problem with Time Windows

Ann-Kathrin Rothenbächer, Michael Drexl, Stefan Irnich

Abstract

In this paper, we present a new branch-and-price-and-cut algorithm to solve the truck-and-trailer routing problem with time windows (TTRPTW) and two real-world extensions. In all TTRPTW variants, the fleet consists of one or more trucks that may attach a trailer. Some customers are not accessible with a truck-and-trailer combination, but can however be serviced by one if the trailer is previously detached and parked at a suitable location. In the first extension, the planning horizon comprises two days and customers may be visited either on both days or only once, in which case twice the daily supply must be collected. The second extension incorporates load transfer times depending on the quantity moved from a truck to its trailer. The exact branch-and-price-and-cut algorithm for the standard variant and the two new extensions is based on a set-partitioning formulation in which columns are routes describing the movement of a truck and its associated trailer. Linear relaxations of this formulation are solved by column generation where new routes are generated with a dynamic programming labeling algorithm. The effectiveness of this pricing procedure can be attributed to the adaptation of techniques such as bidirectional labeling, the *ng*-neighbourhood, and heuristic pricing using dynamically reduced networks and relaxed dominance. The cutting component of the branch-and-price-and-cut adds violated subset-row inequalities to strengthen the linear relaxation. Computational studies show that our algorithm outperforms existing approaches on TTRP and TTRPTW benchmark instances used in the literature.

4.1 Introduction

We consider variants of the truck-and-trailer routing problem with time windows (TTRPTW) motivated by an application in raw milk collection in Bavaria. Vehicles of a dairy need to pick up milk from farms. Because of usually large distances between dairy and farms and high milk production rates, trucks with attached trailers are used to increase the total vehicle capacity. However, some farms are not accessible by truck-and-trailer combinations (henceforth referred to as *complete vehicles*) because of small yards or restricted access roads. These farms, called *truck customers*, can though be serviced by a complete vehicle if the trailer is previously detached and parked at a suitable location. The other farms are referred to as *trailer customers*; they can be serviced by either a complete vehicle or a single truck. At trailer customers and at a set of optional *transshipment locations*, a trailer can be decoupled from its truck and parked. The truck can then visit truck and/or trailer customers without the trailer. This is referred to as a *subtour*. A truck can terminate a subtour, i.e., return to its parked trailer, at any time. It can then transfer load to the trailer. After a subtour, the truck can start another one or recouple the trailer and pull it away. We assume that collected supply is directly loaded into the trailer if the trailer is attached and has residual capacity. At truck customers and whenever supply exceeds the trailer's residual capacity, supply goes directly into the truck. Before going back to the depot, the truck must return to the parking location of its trailer and recouple it.

Given a vehicle fleet and a set of customers, the task is to find a cost-minimal set of routes to collect the whole supply of all customers while respecting vehicle capacities, time window constraints, and accessibility restrictions. The vehicle fleet is heterogeneous, containing several types of trucks and trailers with different costs and capacities. Trucks can either perform a route alone, or they can pull a single, uniquely assigned trailer, which can be detached temporarily as described above. Optimization includes deciding whether to use a given truck as a single vehicle or as the motorized part of a complete vehicle.

An example of the TTRP (for simplicity without time windows and load amounts) with a possible solution is depicted in Figure 4.1. The fleet consists of two trucks and two trailers that can be combined as indicated in the figure. All vehicles are based at a central depot where all routes start and end. There are four truck and four trailer customers to serve. Additionally, two transshipment locations are offered whose visit is optional. The depicted solution comprises two routes: the orange route is performed by truck 1, and the light blue/purple route, which contains two subtours, is executed by a complete vehicle consisting of truck 2 and trailer 2. Trailer 1 and transshipment location 10 are not used.

The two challenging extensions that we address are of high practical relevance and, to the best of our knowledge, they have not yet been discussed in the literature. First, an important trend in the dairy business is *two-day collection*: many modern farms dispose of sufficient storage capacity to keep their milk for at least one day before it must be collected. Such customers can then be visited either daily or only every other day. The

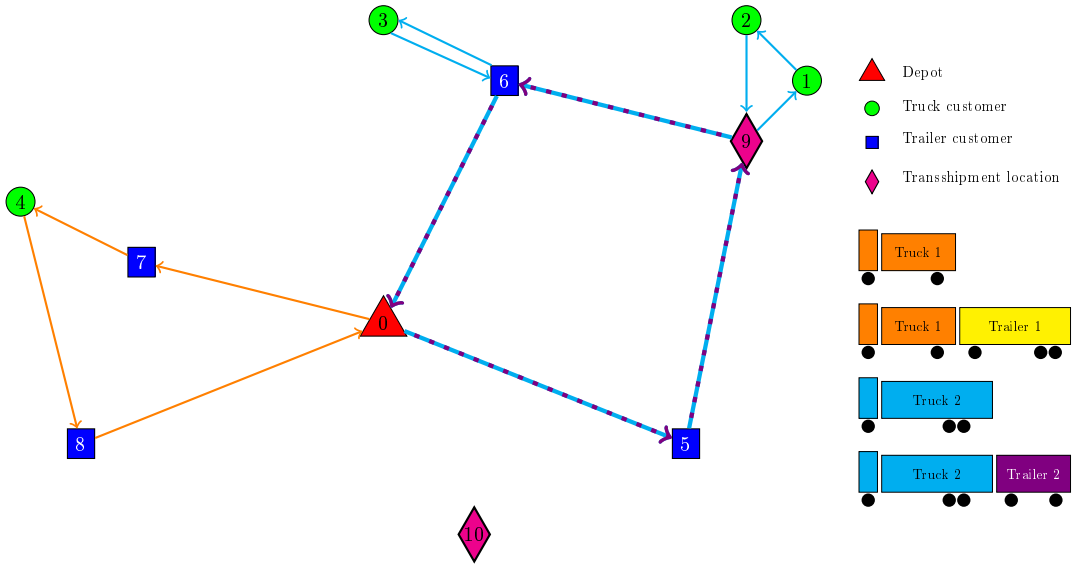


Figure 4.1: Example of the TTRP

decisions to take in the resulting two-day planning problem include the determination of those customers that are serviced only once by collecting their entire two-day supply, and, if so, on which day the visit occurs (day 1 or day 2). The problem is perfectly symmetric with respect to the two days. We exploit this symmetry by applying a tailored column-generation stabilization method, which allows to halve the number of pricing problems to be solved, at least before day-specific branching is performed.

Second, an aspect up to now neglected in the TTRP literature is that in practice, the time for transferring load from truck to trailer often depends on the transferred quantity. Notably, this is the case in milk collection. The literature either assumes an immediate transfer or a fixed positive time independent of the quantity. Consequently, our second extension is the introduction of a quantity-dependent load transfer time, which imposes a tradeoff between the duration of a transfer operation and the resulting free capacity in the truck. Such a tradeoff heavily complicates the labeling algorithm.

Overall, our contribution is the presentation of effective branch-and-price-and-cut algorithms for the TTRPTW and the two extensions. For the classical TTRPTW, our new algorithm outperforms alternative exact approaches from the literature. The success of the new approach can be attributed to a new bidirectional labeling algorithm for the pricing subproblem and the use of the subset-row cuts to strengthen the linear relaxation of the master problem (see Jepsen *et al.*, 2008).

The remainder of this paper is organized as follows. A review of the relevant literature is provided in the next section. In Section 4.3, our branch-and-price-and-cut algorithm is explained with emphasis on the solution of the pricing subproblem and branching strategy. Section 4.4 establishes the extension of the planning horizon from one to two days. Section 4.5 presents the necessary adaptations when a positive load transfer time is assumed. Computational results are discussed in Section 4.6, and conclusions are drawn

in Section 4.7.

4.2 Literature Review

Although the consideration of trailers in the literature on vehicle routing problems (VRPs) dates back several decades, the TTRP has attracted increased attention of the research community only in the last few years. This is surprising, given the large number of practical applications described in the existing publications: in addition to the already mentioned raw milk collection, TTRP applications include the distribution of goods to grocery stores (Semet and Taillard, 1993), of finished dairy products to customers (Gerdessen, 1996), of compound animal food to farmers (Gerdessen, 1996), postal mail delivery (Bodin and Levy, 2000), the movement of empty and loaded containers for a logistics company (Tan *et al.*, 2006), and fuel oil delivery to private households (Drexl, 2011).

The term "truck-and-trailer routing problem" was coined by Chao (2002). Since then, numerous advanced metaheuristics for different TTRP variants were presented by, e.g., Scheuerer (2006), Villegas *et al.* (2010, 2011, 2013), Caramia and Guerriero (2010a,b), Lin *et al.* (2011), Derigs *et al.* (2013), Pasha *et al.* (2014), and Batsyn and Ponomarenko (2014). All of these papers deal with deterministic problem settings. By contrast, Torres *et al.* (2015) consider a TTRP with fuzzy constraints, and Mirmohammadsadeghi and Ahmed (2015) handle stochastic demands. Only Lin *et al.* (2011) and Derigs *et al.* (2013) take into account time windows. The paper by Cuda *et al.* (2015) contains a detailed overview of the TTRP literature, discussing modeling aspects and details of solution approaches.

Exact procedures for TTRPs are still scarce. We are aware of only three works on this topic: Belenguer *et al.* (2016) propose a branch-and-cut algorithm for a single-vehicle variant of the TTRP without time windows. On a test bed with 32 random Euclidean instances, the largest instance solved to optimality has 100 customers and 10 transshipment locations. To the best of our knowledge, the only published exact algorithms tackling a multi-vehicle TTRP are the branch-and-price algorithms by Drexl (2011) and Parragh and Cordeau (2017). A comparison of these two papers with the present work, concerning the problem aspects considered and the solution methods used, is shown in Table 4.1.

4.3 Branch-and-Price-and-Cut

We start with a partial formalization of the problem and present the path-based formulation of the TTRPTW in Section 4.3.1. The path-based formulation is solved with a branch-and-price-and-cut algorithm. The column-generation subproblem and its resolution by a dynamic-programming labeling algorithm are discussed in detail in Section 4.3.2. Section 4.3.3 describes the branching strategy, Section 4.3.4 the integration of subset-row cuts, and Section 4.3.5 acceleration techniques.

	Our paper	Parragh and Cordeau (2017)	Drexl (2011)
Time windows	✓	✓	✓
Heterogeneous fleet	✓		✓
Dedicated transshipment locations	✓		✓
Two-day planning horizon	✓		
Quantity-dependent load transfer times	✓		
Heuristic and exact labeling algorithm	✓	✓	✓
Bidirectional labeling	✓		
Metaheuristic for pricing		✓	
ng-route relaxation	✓	✓	
Valid inequalities	✓		
Stabilization	✓	✓	

Table 4.1: Comparison of branch-and-price algorithms for different TTRP variants

4.3.1 Path-Based Formulation

Let N be the set of customers, L be the set of truck types, and H be the set of trailer types. The set of vehicle classes K contains both single trucks and certain combinations of a truck and a trailer and is defined as a subset of $L \times (H \cup \{0\})$. Elements $k = (l, 0)$ represent the single trucks of type $l \in L$. TTRPTW variants may assume a limited fleet so that the number of available vehicles of type $f \in L \cup H$ is constrained by n_f. For convenience, we define the vehicle indicator b_f^k to be 1 if vehicle class $k \in K$ uses vehicle type $f \in L \cup H$, and 0 otherwise. Note that several vehicles of one vehicle class may be available.

The path-based formulation is a set-partitioning model with one variable for each possible TTRPTW route describing the scheduled movement of a truck and, if present, its associated trailer. For a vehicle of class k, the set of all valid routes is R^k. Each route $r \in R^k$ has a corresponding decision variable λ_r^k, which equals 1 if the route is chosen, and 0 otherwise. Depending on the type of TTRPTW variant, route costs c_r can include variable travel costs, depending on traveled distance and on whether the truck is moving alone or towing the trailer, and handling costs. Handling costs occur whenever trailers are coupled or decoupled, or load is transferred from truck to trailer. We define a_{rn} as the number of times route r *serves* customer n. The coefficient a_{rn} does not include visits to the location of n that are only used for parking or coupling the trailer or for transferring load.

The path-based model is:

$$\min \quad \sum_{k \in K} \sum_{r \in R^k} c_r \lambda_r^k \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{r \in R^k} a_{rn} \lambda_r^k = 1 \qquad (\pi_n) \qquad \forall\, n \in N \tag{4.1b}$$

$$\sum_{k \in K} \sum_{r \in R^k} b_f^k \lambda_r^k \leq n_f \qquad (\mu_f) \qquad \forall\, f \in L \cup H \tag{4.1c}$$

$$\lambda_r^k \geq 0, \quad \text{integer} \qquad \forall\, k \in K, r \in R^k \tag{4.1d}$$

The objective function (4.1a) aims at minimizing the total transportation costs. Then, the partitioning constraints (4.1b) ensure that every customer is served exactly once. Constraints (4.1c) guarantee that the fleet size is not exceeded by the selected routes. Finally, the variable domains are given by (4.1d).

4.3.2 Column Generation

Model (4.1) cannot be solved directly due to the huge number of route variables. Instead we can solve it with a branch-and-price algorithm which uses a column-generation subproblem, the pricing problem, to dynamically generate route variables as they are needed (Lübbecke and Desrosiers, 2005). Similar to many other vehicle-routing problems, the pricing problems of the TTRPTW are variants of a shortest-path problem with resource constraints (SPPRC) on graphs with negative cost cycles (Irnich and Desaulniers, 2005). For each vehicle class $k \in K$, there is a specific variant of SPPRC that has to be solved. Here, the goal is to find at least one route with negative reduced costs or to prove that no such route exists.

We start with the description of the underlying pricing network in Section 4.3.2. The resources and their resource extension functions are introduced in Section 4.3.2, while the TTRPTW-specific dominance is explained in Section 4.3.2. Details concerning bidirectional labeling are discussed in Section 4.3.2.

Network

The vehicle-class specific pricing network $D^k = (V, A^k)$ for vehicle class $k \in K$ is constructed as in (Drexl, 2011) so that each location is associated with one or several vertices. The four types of relevant locations are the depot, truck customers, trailer customers, and dedicated transshipment locations. The depot is modeled by a start vertex d^+ and a destination vertex d^-. Let $N^L \subset N$ be the set of truck customers and let $N^H \subset N$ be the set of trailer customers. Each truck customer is represented by a single vertex $n \in N^L$, while trailer customers are represented by four vertices, i.e., n for the service, $d(n)$ for detaching the trailer, $\tau(n)$ for load transfer, and $c(n)$ for coupling the trailer at the customer's location. Similarly, let T be the set of optional transshipment locations. There are three vertices for each transshipment location $t \in T$, namely, $d(t)$ for detaching the trailer, $\tau(t)$ for load transfer, and $c(t)$ for coupling the trailer at the transshipment location. As both trailer customers and transshipment locations offer the parking and load transfer possibility, we define the set of *parking locations* $P = N^H \cup T$. Summarizing, the vertices V comprise the depots $\{d^+, d^-\}$, vertices $N^L \cup N^H$ for service at customers, decoupling vertices $D = \{d(p) : p \in P\}$, transfer vertices $\mathcal{T} = \{\tau(p) : p \in P\}$, and coupling vertices $C = \{c(p) : p \in P\}$. Table 4.2(a) lists the vertices for the network D^k.

Location	Associated vertices	For all
Depot	d^+, d^-	
Truck customers	n	$n \in N^L$
Trailer customers	$n, d(n), \tau(n), c(n)$	$n \in N^H$
Transfer locations	$d(t), \tau(t), c(t)$	$t \in T$

(a) Vertices $i \in V$

	Arcs (i,j)	to $j \in$ N^L	N^H	D	$\mathcal{T}$	C	d^-
	d^+	ST	✓	CV			
	N^L	✓	✓		CV	CV	ST
from	N^H	✓	✓	CV	CV	CV	✓
$i \in$	D	CV	CV				
	$\mathcal{T}$	CV	CV				
	C		CV	CV			CV

(b) Arcs $(i,j) \in A^k$ depend on the vehicle class k (CV=complete vehicle only, ST=single truck only, and ✓=both)

Table 4.2: Network $D^k = (V, A^k)$

The type of arcs present in the pricing network depends on the vehicle class k. For single trucks $k = (l, 0)$, the arc set A^k contains only connections between the customer vertices and from/to the depot. Table 4.2(b) reports these arcs with the symbols ST (single truck only) and ✓. For complete vehicles $k = (l, h)$ with $h \neq 0$, there exist many more possible connections. They are indicated in Table 4.2(b) with the symbols CV (complete vehicle only) and ✓.

Decoupling vertices can be reached only with the trailer attached; transfer and coupling vertices can be reached only if the trailer is waiting at the corresponding location. Decoupling and transfer vertices can be left only by the truck; coupling vertices can be left only with the trailer attached. In other words, a trailer parked at a parking location implicitly 'moves' from the corresponding decoupling vertex to the transfer vertex (may be skipped) and from there to the coupling vertex. After parking a trailer at a location and before attaching it again, the truck may perform an arbitrary number of subtours and visits of the corresponding transfer vertex. Visiting a transfer or a coupling vertex includes the possibility of load transfer at zero cost and time. Indeed, our transfer strategy requires to always transfer as much load as possible from the truck to the trailer at these vertices (see next section). By contrast, when reaching a decoupling vertex, a transfer is not needed, because, as mentioned in the Introduction, the supply collected since the last visit to a transshipment location is loaded in the trailer as far as possible. This logic is guaranteed by the network structure and the feasibility checks of the resource extension functions (REFs) described in the following.

Labeling Algorithm

We start with the description of the cost of a TTRPTW route for a fixed vehicle $k = (l, h) \in K$ and introduce all relevant parameters needed to describe the resources and their consumption. The costs of an arc $(i, j) \in A^k$ consist of travel costs and handling costs occurring when a certain operation (service, decoupling, transfer, or coupling) is performed at a location. Travel costs can depend on the specific vehicle class k and

on whether the trailer is towed. We incorporate the handling costs associated with the tail vertex i into the costs of (i,j). Handling costs can be location-specific, e.g., service costs can depend on the supply that a customer provides. In any case, the cost of arc $(i,j) \in A^k$ for $k = (l,h)$ can be expressed as

$$c_{ij}^k(\delta) = c_{ij}^l + \delta \cdot c_{ij}^h,$$

where c_{ij}^l are truck-dependent and c_{ij}^h are trailer-dependent costs, and $\delta \in \{0,1\}$ indicates whether trailer h is actually towed from i to j. Similarly, the time t_{ij} consumed by traversing arc (i,j) comprises the travel time and the handling time at the vertex i. We assume that t_{ij} does not depend on the vehicle k (but this could easily be modeled, too). While the service time may depend on the customer's supply, the coupling, the transfer, and the decoupling operations each consume a fixed time independent of the location. In Section 4.5 we will relax the constant-time assumption for load transfer.

For any feasible route r in $D^k = (V, A^k)$ from the start d^+ to the destination d^-, the sequence of vertices uniquely determines when and where the trailer is attached to the truck. Hence, we can define an indicator $\delta_{r,ij}^{trailer} \in \{0,1\}$ being equal to 1 if the trailer is attached to the truck when traveling along arc $(i,j) \in A^k$. Moreover, let the set of customer vertices and arcs covered by r be defined as $N(r)$ and $A(r)$ respectively. Let (π_n) be the dual prices associated with the partitioning constraints (4.1b) and (μ_f) be the dual prices associated with the fleet-size constraints (4.1c). Now, the reduced costs of a route r performed by a vehicle $k = (l,h)$ can be expressed as

$$\tilde{c}_r^k = \sum_{(i,j)\in A(r)} c_{ij}^k(\delta_{r,ij}^{trailer}) - \sum_{n\in N(r)} \pi_n - \sum_{f\in L\cup H} b_f^k \mu_f = \sum_{(i,j)\in A(r)} \tilde{c}_{ij}^k(\delta_{r,ij}^{trailer}),$$

where we define

$$\tilde{c}_{ij}^k(\delta) = c_{ij}^k(\delta) - \begin{cases} \pi_j, & j \in N \\ 0, & \text{otherwise} \end{cases} - \begin{cases} \mu_l + \mu_h, & i = d^+ \\ 0, & \text{otherwise} \end{cases}$$

with $\mu_h = 0$ if $h = 0$, i.e., if no trailer is used.

A TTRPTW route is time-feasible, if service at a visited customer n starts within its service time window $[e_n, \ell_n]$. The time window $[e_{d^+}, \ell_{d^+}] = [e_{d^-}, \ell_{d^-}]$ models the planning horizon. It is assumed that all other vertices $v \in D \cup \mathcal{T} \cup C$ have a non-restrictive time window. This means that trailer customers may have a restricted service time window, but that their location can be used for parking and load transfer at any time.

Each customer $n \in N$ has a positive supply q_n. The capacity of a vehicle $k = (l,h)$ is given by its truck and its trailer capacity Q_L^k and Q_H^k respectively. Single trucks, i.e., vehicles with $h = 0$, have $Q_H^k = 0$. A TTRPTW route is load-feasible, if at no point truck or trailer carry more collected supply than their given capacity.

Our labeling approach for the classical TTRPTW uses six types of resources to ensure feasibility of the route of a complete vehicle. A partial path r from d^+ to a vertex $i \in V$ is represented by a label $E_i = (E_i^{cost}, E_i^{time}, E_i^{loadL}, E_i^{loadH}, E_i^{posH}, (E_i^{cust_n})_{n\in N})$, where

the label components are:

E_i^{cost}: reduced cost of path r;
E_i^{time}: earliest service/operation start time at vertex i;
E_i^{loadL}: collected supply loaded into the truck at vertex i;
E_i^{loadH}: collected supply loaded into the trailer at vertex i;
E_i^{posH}: current position $p \in P$ of the trailer if detached from the truck, otherwise $\perp$;
$E_i^{cust_n}$: number of times that customer $n \in N$ is visited along path r. Also set to 1 if customer n is not visited but is unreachable from r. A customer n is unreachable if $E_i^{loadL} + E_i^{loadH} + q_n > Q_L^k + Q_H^k$ or $E_i^{time} + t_{in} > \ell_n$ in which case it cannot be part of any feasible extension of path r.

In the initial label at vertex d^+, all components are set to 0 except $E_i^{time} = e_{d^+}$ and $E_i^{posH} = \perp$. The extension of a label E_i along an arc $(i,j) \in A^k$ is performed using the following REFs:

$$E_j^{cost} = E_i^{cost} + \tilde{c}_{ij}^k(\delta_{E_i^{posH},\perp}) \tag{4.2a}$$

$$E_j^{time} = \max\{e_j, E_i^{time} + t_{ij}\} \tag{4.2b}$$

$$E_j^{loadL} = \begin{cases} E_i^{loadL} + q_j, & j \in N \text{ and } E_i^{posH} \neq \perp \\ E_i^{loadL} + \min\{E_i^{loadH} + q_j - Q_H^k, 0\}, & j \in N^H \text{ and } E_i^{posH} = \perp \\ E_i^{loadL}, & j \in D \cup \{d^-\} \\ E_i^{loadL} - \min\{E_i^{loadL}, Q_H^k - E_i^{loadH}\}, & j \in \mathcal{T} \cup C \end{cases} \tag{4.2c}$$

$$E_j^{loadH} = \begin{cases} E_i^{loadH}, & (j \in N \text{ and } E_i^{posH} \neq \perp) \text{ or} \\ & j \in D \cup \{d^-\} \\ \min\{E_i^{loadH} + q_j, Q_H^k\}, & j \in N^H \text{ and } E_i^{posH} = \perp \\ E_i^{loadH} + \min\{E_i^{loadL}, Q_H^k - E_i^{loadH}\}, & j \in \mathcal{T} \cup C \end{cases} \tag{4.2d}$$

$$E_j^{posH} = \begin{cases} E_i^{posH}, & j \in N \cup \mathcal{T} \\ p, & j \in D \text{ where } j = d(p), p \in P \\ \perp, & j \in C \cup \{d^-\} \end{cases} \tag{4.2e}$$

$$E_j^{cust_n} = \begin{cases} E_i^{cust_n} + 1, & j \in N \text{ and } j = n \\ \max\{E_i^{cust_n}, U_n(E_j)\}, & \text{otherwise.} \end{cases} \tag{4.2f}$$

We briefly explain the non-trivial parts of the REFs. In (4.2a), the Kronecker delta $\delta_{E_i^{posH},\perp}$ indicates whether the trailer is currently attached, i.e., whether $E_i^{posH} = \perp$. In (4.2c) and (4.2d), we distribute the collected supply among the truck and the trailer. As the truck can reach customers that the trailer cannot, the best strategy is to keep the truck as free as possible. Thus, supply is always directly loaded into the trailer whenever it is attached using the residual capacity of the trailer. Furthermore, during each coupling and transfer operation, as much load as possible is transferred from the truck to the trailer. As mentioned, no load transfer takes place at decoupling vertices. In (4.2e), the decoupling operation is modeled in the second case in which we record

the physical location where the trailer is parked. This is either a trailer customer or a transshipment location. Conversely, the recoupling operation is modeled in the third case of (4.2e). Finally, in (4.2f), the visited and unreachable customers are updated, where $U_n(E_j) \in \{0,1\}$ indicates whether customer n has become unreachable either due to the time E_j^{time} or the total load $E_j^{loadL} + E_j^{loadH}$.

The label E_j resulting from the extension along arc $(i,j) \in A^k$ is feasible if $E_j^{time} \leq \ell_j$, $E_j^{loadL} \leq Q_L^k$, $E_j^{loadH} \leq Q_H^k$, $E_j^{cust_n} \leq 1$ for all $n \in N$, and the trailer position allows the visit of vertex j. The latter is true if $j \in N^H$ or one of the following conditions is fulfilled:

$$\begin{aligned} E_i^{posH} &\neq \perp \quad \text{for} \quad j \in N^L \\ E_i^{posH} &= \perp \quad \text{for} \quad j \in D \cup \{d^-\} \\ E_i^{posH} &= p \quad \text{for} \quad j \in \mathcal{T} \cup C \text{ with } j = t(p) \text{ or } j = c(p) \end{aligned}$$

The last condition means that transfer and coupling vertices can only be reached if the trailer is not attached and waiting at this location.

For the routes of single trucks $k = (l, 0)$, the resources E_i^{loadH} and E_i^{posH} are not needed. The REFs for cost, time, truck load, and unreachable customers are defined as in the standard vehicle-routing problem with time windows (VRPTW).

Note that we consider locations of trailer customers as potential transshipment locations without additional constraints. Any number of trailers may be parked there. Transfer is possible within and outside the customer's service time window. Moreover, parking the trailer and serving the customer are independent so that this customer may be served by another vehicle or by the vehicle under consideration but at another stop earlier or later in the route. Parragh and Cordeau (2017), however, require that locations of trailer customers are only used as parking places if their supply is collected during the stop (called *strict parking rule* in the following). For this setting, we introduce an additional binary label component $E_i^{collPosH}$ indicating whether the trailer is currently parked (in this case, it is at the parking location E_i^{posH}) and the supply of that customer still needs to be collected. Therefore, the attribute is set to 1 whenever a decoupling vertex $d(n)$ of a customer n is reached from a vertex $v \neq n$. Its value is only changed to 0 if the supply vertex stored in E_i^{posH} is visited.

Initially, $E_{d^+}^{collPosH}$ is set to 0 and the REF for any arc $(i,j) \in A^k$ is

$$E_j^{collPosH} = \begin{cases} 1, & j = d(n) \in D, n \in N^H, \text{ and } i \neq n \\ 0, & j \in N^H \text{ and } (j = E_i^{posH} \text{ or } c(j) = i) \\ E_i^{collPosH}, & \text{otherwise} \end{cases}.$$

For testing the feasibility of the resulting label E_j after the extension along arc (i,j), we must check the additional condition

$$E_i^{collPosH} = 0 \quad \text{or} \quad j \in N^H \text{ with } c(j) = i$$

whenever the partial path is extended from a coupling vertex $i \in C$.

Dominance

Labeling algorithms use dominance to discard labels that cannot lead to an improved d^+-d^--path w.r.t. another label (or set of labels). A label E_j dominates another label E'_j associated to the same vertex j if all the following conditions are fulfilled:

$$E_j^{res} \leq E_j'^{res} \qquad res \in \{cost, time, loadL, cust_n(n \in N)\} \tag{4.4a}$$

$$E_j^{loadL} + E_j^{loadH} \leq E_j'^{loadL} + E_j'^{loadH} \tag{4.4b}$$

$$E_j^{posH} = E_j'^{posH} \tag{4.4c}$$

and

$$E_j^{collPosH} = E_j'^{collPosH} \tag{4.4d}$$

The last condition (4.4d) is only relevant in the case of the strict parking rule of Parragh and Cordeau (2017).

The above dominance can be strengthened using three additional arguments.

Different Trailer Positions $E_j^{posH} \neq E_j'^{posH}$. Condition (4.4c) can only be violated at a customer vertex j. Then, there is no clear preference between having the trailer attached or parked, because in the first case the truck does not need to return to the parking place, whereas in the second case truck customers can be approached immediately. Moreover, two labels with different current parking positions of the trailers (with $E_j^{posH}, E_j'^{posH} \neq \perp$) are not directly comparable, as no completion of one path is a feasible completion of the other path.

However, label E_j may still dominate E'_j if the first vehicle can be transferred into the same trailer status as the second without violating the dominance constraints (4.4). To check this, we hypothetically move the first vehicle in up to three steps: (1) If $E_j^{posH} \neq \perp$, the first truck must pick up its trailer at location E_j^{posH}. (2) If $E_j'^{posH} \neq \perp$, the first truck must park its trailer at location $E_j'^{posH}$. (3) The first truck must move back to the current customer j. This creates a detour imposing higher values in the cost and time resources of E_j, but if the resulting label still fulfills condition (4.4), label E'_j can be discarded. The return to vertex j conflicts with the elementarity constraints. We eliminate this conflict by removing j from the hypothetical detour. If the triangle inequality holds, such an operation creates a path that is not longer than the non-elementary path. Then, the above check is a sufficient condition to show that all extensions of E'_j are dominated.

Higher Truck Load $E_j^{loadL} \geq E_j'^{loadL}$. Fulfilling condition (4.4b) and at the same time $E_j^{loadL} > E_j'^{loadL}$ is only possible during a subtour, i.e., for $E_j^{posH} \neq \perp$. Again,

label E_j may dominate E'_j fulfilling all dominance conditions except $E_j^{loadL} \leq E_j'^{loadL}$ if the first vehicle could perform a detour to its trailer position in order to transfer load from truck to trailer. If the resulting label has smaller or equal cost and time resources, it dominates the second label.

Different Values of Collection at the Trailer Position $E_j^{collPosH} < E_j'^{collPosH}$. A strengthened dominance rule can be formulated for the case that $E_j^{collPosH} = 0$ and $E_j'^{collPosH} = 1$ hold. Both vehicles have to return to the location of customer $n = E_j^{posH}$, but the first cannot collect the reward π_n anymore. The second, however, needs to serve the customer n, finally imposing higher costs and additional load and service time. It is also less flexible on the current subtour due to the customer's service time window $[e_n, \ell_n]$. The modified dominance rule for $E_j^{collPosH} = 0$ and $E_j'^{collPosH} = 1$ is:

$$E_j^{res} \leq E_j'^{res} \qquad res \in \{time, loadL\} \tag{4.5a}$$

$$E_j^{cost} \leq E_j'^{cost} - \pi_{E_j^{posH}} \tag{4.5b}$$

$$E_j^{cust_n} \leq E_j'^{cust_n} \qquad n \in N, n \neq E_j^{posH} \tag{4.5c}$$

$$E_j^{loadL} + E_j^{loadH} \leq E_j'^{loadL} + E_j'^{loadH} + q_{E_j^{posH}} \tag{4.5d}$$

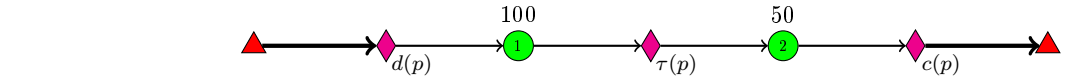
$$E_j^{posH} = E_j'^{posH} \tag{4.5e}$$

Bidirectional Labeling

When solving hard SPPRC instances, the number of labels often increases strongly with the length of the generated partial paths. Bidirectional labeling has been successfully applied to mitigate this type of combinatorial explosion. Forward and backward labels are only extended up to a so-called half-way point that splits the domain of a monotone resource into two intervals, one for the forward and one for the backward labeling. For the TTRPTW, we use the time resource as the monotone resource and define the half-way point as the middle $t^{hwp} = (e_{d^+} + \ell_{d^-})/2$ of the planning horizon. After labeling in both directions has terminated, compatible forward and backward partial paths are merged into complete d^+-d^--paths. For details we refer to (Righini and Salani, 2006).

As the forward labeling for the TTRPTW is already non-trivial to describe, we would like to avoid giving a long-winded and complicated description of the backward case. Therefore, we rely on a formal argument and design the backward labeling algorithm for the TTRPTW completely symmetric to the forward labeling algorithm. To this end, we define a reversed SPPRC instance in which the underlying network $D^k = (V, A^k)$ is replaced by the inverse network (V, B^k) with $(i, j) \in B^k$ if and only if $(j, i) \in A^k$, all time windows $[e_i, \ell_i]$ replaced by $[-\ell_i, -e_i]$, and all other coefficient are kept. Moreover, start and destination vertices d^+ and d^- as well as the role of coupling and decoupling vertices are swapped. We claim that for every feasible route of the given SPPRC instance, the reversed route is feasible in the reversed SPPRC instance, and vice versa. Note first that the statement is certainly true for single trucks $k = (l, 0)$ and routes of complete

vehicles $k = (l, h)$ that collect less supply than the truck's capacity. In these cases, the reversed SPPRC is equivalent to what has been suggested for the VRPTW in (Righini and Salani, 2006) and several subsequent publications. However, if more supply is collected, our loading policy that transfers as much as possible from the truck to the trailer is not directly interchangeable between forward and backward solutions because feasible transfer amounts for one direction may be infeasible for the opposite direction. To see this, consider Figure 4.2. At the transfer vertex in the middle, the optimal strategy to transfer as much as possible leads to a transfer of 100 units in the forward direction. However, in the backward direction, only 50 units have been collected up to this vertex, so that a transfer of 100 units is impossible there. To prove our claim in this case, we



Forward	*loadL*	0	0	100	0	50	50	50
	loadH	0	0	0	100	100	100	100
Backward	*loadL*	50	50	100	0	50	0	0
	loadH	100	100	50	50	0	0	0

Figure 4.2: Example for transfer amounts that are not feasible in the opposite direction ($Q_L^k = Q_H^k = 100$

argue similar to Desaulniers *et al.* (2016a) and consider a feasible forward route $r = (v_0, v_1, v_2, \ldots, v_m)$ for which the collected supply exceeds the truck's capacity Q_L^k. One can obviously define transfer quantities in such a way that the truck arrives fully loaded at the destination. Moreover, let the transfer quantities at the vertices be $(t_0, t_1, t_2, \ldots, t_m)$ (with $t_i = 0$ for all non-parking vertices i) and the supply loaded into the truck at the vertices be $(q_0^L, q_1^L, q_2^L, \ldots, q_m^L)$ (with $q_i^L = 0$ for all parking vertices i). Then, the feasibility of the route implies

$$Q_L^k \geq \sum_{j=0}^{i} q_j^L - \sum_{j=0}^{i-1} t_j \geq \sum_{j=0}^{i} q_j^L - \sum_{j=0}^{i} t_j \geq 0,$$

for all $i \in \{0, 1, \ldots, m\}$, meaning that before (left term) and after (right term) the visit at vertex v_i the current load in the truck is non-negative and does not exceed the truck capacity. The fact that the truck arrives at d^- fully loaded implies $Q_L^k = \sum_{j=0}^{m} q_j^L - \sum_{j=0}^{m} t_j$. By subtracting this term from the above inequalities and multiplying with -1, we get

$$0 \leq \sum_{j=i+1}^{m} q_j^L - \sum_{j=i}^{m} t_j \leq \sum_{j=i+1}^{m} q_j^L - \sum_{j=i+1}^{m} t_j \leq Q_L^k,$$

showing that the route with same the transfer quantities $(t_0, t_1, t_2, \ldots, t_m)$ considered in the reversed direction is also feasible.

In order to simplify the merge procedure, we restrict ourselves to customer vertices as merge points and mimic a kind of merge over arcs. First, the restriction to customer vertices is feasible because any useful route contains at least one customer vertex. The only necessary modification is on the forward half-way test: it requires that forward labels E_i are extended along arcs $(i,j) \in A^k$ up to the first customer vertex $j \in N$ that fulfills $E_j^{time} > t^{hwp}$. Second, we merge forward labels E_i and backward labels E'_i associated with the same vertex $i \in N$. However, it is more convenient to consider the predecessor label $E'_j = pred(E'_i)$ instead of the backward label E'_i itself (this trick was first described by Tilk *et al.*, 2016). Now, the concatenation of the forward partial path given by E_i and the backward partial path given by E'_i with $E'_j = pred(E'_i)$ is a feasible d^+-d^--path if and only if

$$E_i^{time} + t_{ij} \leq -{E'_j}^{time} \tag{4.6a}$$

$$E_i^{loadL} + {E'_j}^{loadL} \leq Q_L^k \tag{4.6b}$$

$$E_i^{loadL} + E_i^{loadH} + {E'_j}^{loadL} + {E'_j}^{loadH} \leq Q_L^k + Q_H^k \tag{4.6c}$$

$$E_i^{posH} = {E'_j}^{posH} \tag{4.6d}$$

$$E_i^{cust_n} + {E'_j}^{cust_n} \leq 1 \qquad \forall\, n \in N \tag{4.6e}$$

$$E_i^{posH} = \perp \text{ or } E_i^{collPosH} \neq {E'_j}^{collPosH}. \tag{4.6f}$$

The resulting route has reduced costs $E_i^{cost} + \tilde{c}_{ij}(\delta_{E_i^{posH},\perp}) + {E'_j}^{cost}$. Note that condition (4.6a) uses the time resource ${E'_j}^{time}$ defined in the reversed SPPRC, which is negative when used in the original context. As the lorry load is already as small as possible in both directions, its sum must not exceed the lorry capacity (4.6b). However, the realized trailer load could be reduced by choosing smaller transfer quantities before. Therefore, we only have to require that the total collected supply be less than or equal to the total vehicle capacity in (4.6c). Condition (4.6d) is equivalent to $E_i^{posH} = {E'_i}^{posH}$, because at customer vertices i we have ${E'_i}^{posH} = {E'_j}^{posH}$ due to the definition of the backward REF. Moreover, condition (4.6e) is only valid if in at least one direction the customer-service related resources $E_i^{cust_n}$ or ${E'_j}^{cust_n}$ respectively describe exactly the subset of customers that are served. If unreachable customers are included in both directions, condition (4.6e) is too strict and thus incorrect. Therefore, we calculate (on the fly) the set of visited customers of backward labels in the merge step to compare it against $E_i^{cust_n}$. Finally, condition (4.6f) is not applicable if the extended dominance related to the resource *collPosH* is used in both forward and backward direction.

4.3.3 Branching

Let $\tilde{\lambda}_r^k$ be the values of variables λ_r^k in a solution of the linear relaxation to model (4.1). Moreover, we define the flow value $\tilde{x}_{ij}^k = \sum_{r \in R^k} a_{r,ij} \tilde{\lambda}_r^k$ for all $k \in K$ and $(i,j) \in A^k$, where $a_{r,ij}$ is the number of times that route r traverses arc (i,j). Let the aggregated flow value of arc (i,j) be $\tilde{x}_{ij} = \sum_{k \in K} \tilde{x}_{ij}^k$. Note that for an arc (i,j) with at least one

customer vertex as an endpoint, the values $\tilde{x}_{ij}$ and $\tilde{x}^k_{ij}$ are binary in a feasible integer solution.

As long as there are fractional values $\tilde{\lambda}^k_r$, we apply the following four-level branching scheme. First, we branch on the overall number of vehicles. Second, we branch on the aggregated flow value for an arc (i, j). Third, we branch on the vehicle-specific flow value for an arc (i, j). In the two latter cases, we always select arcs with at least one customer vertex as an endpoint first, as the branching decision can be implemented by network modifications. If all arcs with fractional value have no customer vertices as endpoints, we branch on the integer flow value by adding an inequality to the master program.

Note that branching on arcs disables the extended dominance rules including fictional detours. The existence of forbidden arcs would require a check of all arcs in the detour and even all arcs from the last visited parking location vertex in this detour to all possible successors.

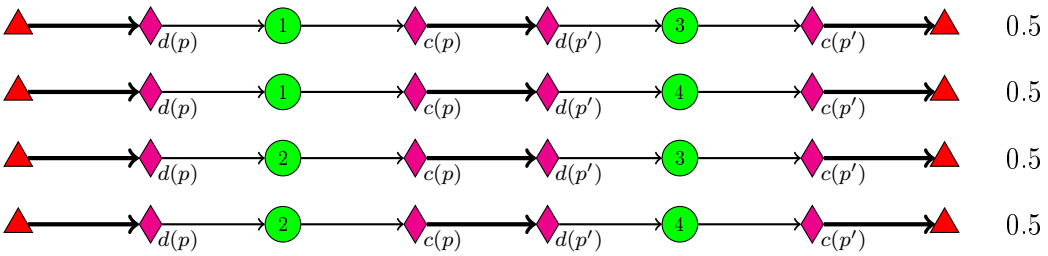


Figure 4.3: Example of four fractional paths of same vehicle class despite integral arc flows

There can still exist fractional solutions with integer aggregated and vehicle-specific flows after applying the above branching rules. An example is depicted in Figure 4.3. Four routes performed with the same vehicle class $k \in K$ each with a value $\tilde{\lambda}^k_r$ of 0.5 impose integer flows on the depicted arcs. The arcs $(d^+, d(p))$, $(c(p), d(p'))$, and $(c(p'), d^-)$ have a flow of 2, while all others have a flow of 1.

The rule for the fourth level is based on the following property: if for each customer the predecessor and successor customer (if any) are uniquely determined, a basic solution to the linear relaxation of (4.1) is integer. This is true because the predecessors and successors unambiguously define sets of customers served by the same vehicle. The branching decisions on the third level have already determined the vehicle class $k \in K$ that visits the customer set. All routes with a vehicle of class k serving the same set of customers have the same coefficients in the constraints of (4.1). Therefore, in a basic solution to the linear relaxation of (4.1) only one of the corresponding route variables can be positive so that one with the lowest cost is selected exactly once.

The fourth level of branching decisions finally guarantees integer path-variables in (4.1) without explicitly branching on the associated variables. If two customer vertices i and j are served consecutively, i before j, the two are either connected directly via arc (i, j) (handled by the rules above) or with a subpath Q of length 2 or 3, i.e., $Q = (i, v_1, j)$ or $Q = (i, v_1, v_2, j)$ where $v_1, v_2 \in C \cup \mathcal{T} \cup D$. We branch on such subpaths connecting two

customers i and j. More precisely, to exclude a subpath Q in the network D^k this subpath is declared forbidden, and several such decisions are managed effectively by the network modification procedure of Villeneuve and Desaulniers (2005). To enforce a subpath Q in the network D^k, another type of network modification can be performed. One must create copies of the intermediate vertices v_1 and v_2 (if present), remove all arcs from the forward/backward star of i/j, and connect i with j via the copies. Such branching on sequences of customers has been applied in different contexts but with similar network modification procedures in (Desaulniers, 2010) and (Bode and Irnich, 2012).

4.3.4 Cuts

To strengthen the linear relaxation, we incorporate subset-row (SR) inequalities into model (4.1) (see Jepsen *et al.*, 2008). A valid SR inequality is defined on a subset of tasks. For the standard TTRPTW, tasks are the visits to the customers. For the TTRPTW with two-day planning horizon (see the later Section 4.4) tasks are the visits to the customers at a particular day, i.e., pairs of customer and day. We restrict ourselves to SR inequalities defined on three tasks as proposed by Jepsen *et al.* (2008) because they can be separated by straightforward enumeration. Given a set S_f of three tasks, the SR inequality $SR(S_f)$ is given by

$$\sum_{k \in K} \sum_{r \in R^k} \left\lfloor \frac{g_{fr}}{2} \right\rfloor \lambda_r^k \leq 1 \qquad (\sigma_f),$$

where g_{fr} is the number of times route r serves tasks in S_f.

The incorporation of SR inequalities into the master problem complicates the TTRPTW pricing problems: let $\sigma_f \leq 0$ be the dual price of the SR inequality $SR(S_f)$. Then, σ_f has to be subtracted from the reduced costs of a route for every second service to tasks in S_f. In order to keep track of such services, one additional binary resource E^{SR_f} for each inequality $SR(S_f)$ must be defined. It indicates the parity of the number of times tasks in S_f are served.

Jepsen *et al.* (2008) suggested an improved dominance, in which otherwise incomparable labels can be considered. For two labels E_j and E'_j fulfilling the dominance conditions (4.4), the reduced-cost comparison in (4.4a) has to be replaced by the following generally tighter condition

$$E_j^{cost} - \sum_{f : {E_j}^{SR_f} > {E'_j}^{SR_f}} \sigma_f \leq {E'_j}^{cost}.$$

In the merge step of the bidirectional labeling algorithm, the concatenation of the forward partial path given by E_i and the backward partial path given by E'_i with $E'_j = pred(E'_i)$ has reduced costs of

$$E_i^{cost} + \tilde{c}_{ij}(\delta_{E_i^{posH},\perp}) + {E'_j}^{cost} - \sum_{f : {E_i}^{SR_f} + {E'_j}^{SR_f} = 2} \sigma_f.$$

4.3.5 Acceleration Techniques

In this section, we sketch some further techniques that we use to accelerate our algorithm. First, we start the algorithm with a valid lower bound for the number of vehicles necessary to serve all customers with respect to their supply. This bound is calculated by solving a bin-packing problem with a bin size equal to the maximal total vehicle capacity Q^{max} among all vehicle classes. If the run time exceeds one minute, the trivial lower bound $\lceil \sum_{n \in N} q_n / Q^{max} \rceil$ is used. Second, we relax the elementary requirement and use the ng-neighborhood relaxation presented by Baldacci *et al.* (2011d) in order to accelerate the solution of the pricing problem. Third, in the first pricing iterations, we solve the pricing problem only for single-truck vehicle classes as long as new columns are generated. This significantly reduces the number of time-consuming pricings of routes of complete vehicles. Fourth, the pricing problems for complete vehicles are solved heuristically until no new columns are found. To this end, the dominance criterion is relaxed by ignoring conditions (4.4a) for $res = cust_n$, (4.4c), and (4.4d). Moreover, we use reduced networks by restricting, for each vertex, the number of outgoing arcs to both customer vertices and vertices of parking locations, where those with lower reduced costs are preferred. Fifth and finally, we apply the MIP solver of CPLEX for a maximum of one minute using the columns generated up to this point in order to increase the chance of finding good upper bounds. The MIP solver is called after solving the branch-and-bound root node and when reaching the time limit.

4.4 Two-Day Planning Horizon

Caused by the nowadays widespread use of larger storage tanks with improved cooling technology, many modern milk-producing farms no longer need to be visited every day. Instead, a visit every second day suffices, in which case the accumulated supply of two days must be collected. We model the case of a two-day planning horizon with two types of customers as follows. The first subset of customers (N^{every}) still needs a visit every day. The other customers (N^{option}) can be visited either every day for collection of their 'normal', daily supply, or every second day for collection of twice that amount.

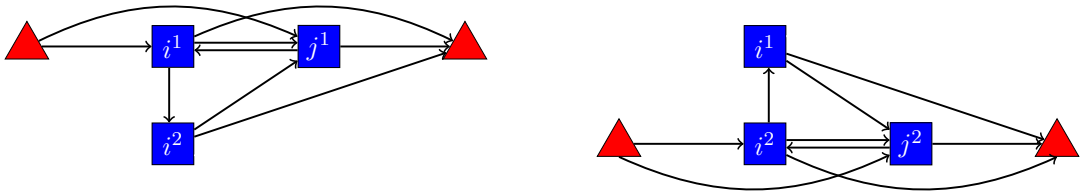
We solve the two-day problem by creating two tasks n^1, n^2 for each customer $n \in N$, representing the collection of the supply of the first and the second day respectively. Now there exist for each pair $(k, \theta) \in K \times \{1, 2\}$ of vehicle class and day an associated network $D^{k\theta}$, a set $R^{k\theta}$ of routes, and of routing variables $\lambda_r^{k\theta}$. The partitioning conditions in the master problem (4.1b) are replaced by

$$\sum_{k \in K} \sum_{\theta \in \{1,2\}} \sum_{r \in R^{k\theta}} a_{rn^\vartheta} \lambda_r^{k\theta} = 1 \qquad (\pi_{n^\vartheta}) \qquad \forall\, n \in N, \vartheta \in \{1, 2\}, \tag{4.7}$$

where a_{rn^ϑ} indicates whether route r covers task n^ϑ.

The distinction between the two types of customers is realized in the pricing networks. The network $D^{k\theta}$ is structured as the network D^k described in Section 4.3.2 and contains depot, decoupling, transfer, and coupling vertices. In addition, each customer vertex is

replaced by one or two task vertices. Precisely, customers $n \in N^{\text{every}}$ who need to be visited every day have one vertex n^θ for the task of day θ in the network $D^{k\theta}$, while customers $n \in N^{\text{option}}$ with service option have two vertices n^1 and n^2 for both tasks, see Figure 4.4. Vertices for tasks of the same day are connected as in the standard TTRPTW network D^k. Moreover, each task vertex representing a customer $n \in N^{\text{option}}$ is connected to its corresponding task vertex of the other day. In addition, each such vertex can be left towards all vertices in the network of day θ that can be reached from customer n.



(a) Network for collection on the first day ($\theta = 1$)

(b) Network for collection on the second day ($\theta = 2$)

Figure 4.4: Example of the two pricing networks for a customer $i \in N^{\text{option}}$ (collection every other day allowed) and a customer $j \in N^{\text{every}}$ (must be visited every day)

Symmetry Considerations. A drawback of the two-day modeling and solution approach presented above is that the number of partitioning constraints (4.7) of the path-based formulation and the number of column-generation subproblems that need to be solved is doubled. We now show that the perfect symmetry with respect to the two planning days can be exploited so that the computational effort is approximately halved again.

We claim that for solving the linear relaxation of the two-day horizon master program, i.e., (4.1) with (4.1b) replaced by (4.7), nothing is lost with respect to the strength of the dual bound when partitioning constraints are aggregated. More precisely, every two constraints (4.7) for $\vartheta \in \{1, 2\}$ of a customer $n \in N$ can be replaced by aggregated constraints of the form

$$\sum_{k \in K} \sum_{r \in R^{k1}} (a_{rn^1} + a_{rn^2}) \lambda_r^{k1} = 2 \qquad \forall\, n \in N. \tag{4.8}$$

Note that (4.8) is equivalent to $\sum_{k \in K} \sum_{\theta \in \{1,2\}} \sum_{r \in R^{k\theta}} (a_{rn^1} + a_{rn^2}) \lambda_r^{k\theta} = 2$, because pairs of corresponding routes $r \in R^{k1}$ and $r' \in R^{k2}$ have identical coefficients ($a_{rn^1} + a_{rn^2} = a_{r'n^1} + a_{r'n^2}$).

The proof relies on the concept of deep dual-optimal inequalities (DDOIs, Ben Amor *et al.*, 2006), originally introduced in order to stabilize column-generation algorithms: pairwise aggregation of two constraints as suggested by (4.8) is equivalent to imposing

constraints

$$\pi_{n^1} = \pi_{n^2} \qquad \forall\, n \in N,$$

to the dual formulation as shown by Gschwind and Irnich (2016, Proposition 6). This is again equivalent to the introduction of additional unconstrained variables η_n (=additional columns), one per customer, to the primal formulation. The new variable η_n has coefficient $+1$ in the partitioning constraint of the day-one-task and -1 in the constraint of the day-two-task so that the resulting partitioning constraints become

$$\sum_{k\in K}\sum_{\theta\in\{1,2\}}\sum_{r\in R^{k\theta}} a_{rn^1}\lambda_r^{k\theta} + \eta_n = 1 \qquad \forall\, n \in N \tag{4.9a}$$

$$\sum_{k\in K}\sum_{\theta\in\{1,2\}}\sum_{r\in R^{k\theta}} a_{rn^2}\lambda_r^{k\theta} - \eta_n = 1 \qquad \forall\, n \in N. \tag{4.9b}$$

Thus, the *extended primal formulation* has additional variables η_n and the task-partitioning constraints (4.7) are replaced by (4.9). Any feasible solution to the linear relaxation of the extended primal formulation can be transformed into a feasible solution to the linear relaxation of the original formulation having constraints (4.7) with identical costs. The procedure works as follows. Let $\tilde{\lambda}_r^{k\theta}$ be a feasible solution to the extended primal formulation. Regardless of the values of η_n, define for each route $r \in R^{k\theta}$ operated on day θ by vehicle $k \in K$ the corresponding route r' operated on the other day θ'. Further define a new solution by $\widehat{\lambda}_r^{k\theta} = (\tilde{\lambda}_r^{k\theta} + \tilde{\lambda}_{r'}^{k\theta'})/2$ for all vehicles $k \in K$, days $\theta \in \{1,2\}$, and routes $r \in R^{k\theta}$, which yields the equivalent feasible solution for formulation (4.1) with task-partitioning constraints (4.7). Now, Proposition 1 in (Gschwind and Irnich, 2016) ensures that under these conditions, the dual bound of the extended primal formulation and the original formulation coincide.

In summary, the use of the aggregated partitioning constraints (4.8) accomplishes that the number of constraints is the same as in the one-day problem. Moreover, it induces identical subproblems for both days $\theta = 1$ and $\theta = 2$, so that only one pricing subproblem, for example for day $\theta = 1$, needs to be solved. Admittedly, this subproblem is larger than the one-day subproblem. However, a by-product of aggregation is that the dual variables are stabilized, leading to a generally faster termination of the column-generation procedure.

Branching and cutting can destroy the perfect symmetry. In order to maintain symmetry as long as possible, we always generate symmetric pairs of SR inequalities. Hence, if $SR(S_f)$ is separated for tasks S_f, the inequality $SR(S_{f'})$ for the other-day tasks $S_{f'}$ is also generated. We enforce the same dual price for these paired SR inequalities as described above for the partitioning constraint by adding the aggregated inequality, i.e., $\sum_{k\in K}\sum_{r\in R^{k1}}(\lfloor g_{fr}/2\rfloor + \lfloor g_{f'r}/2\rfloor)\lambda_r^{k1} \le 2$.

While branching on the number of vehicles preserves symmetry, branching on individual arcs of the two day-specific networks (see Figure 4.4) requires that the column-generation master program be disaggregated. We do so by replacing the aggregated constraints (4.8) with the disaggregated ones (4.7), and by adding the other-day route

variables $\lambda^k_{r'}$ for all active route variables λ^k_r.

Overnight Return of Trailers. As pointed out by Tricoire (2016), a two-day planning horizon allows further savings when the requirement that all vehicles return to the depot at the end of each day is relaxed and it is allowed to park an empty trailer overnight at a transshipment location. It depends on the application whether this can be done in practice. The consideration of the overnight parking option requires considerable modifications to our model and the solution approach described in this section. Moreover, we expect the additional savings to be much smaller than those obtained by switching from a one-day to a two-day horizon, so we leave this as a topic for further research.

4.5 Quantity-Dependent Transfer Time

Up to now, the TTRP literature either assumes an immediate transfer of load from truck to trailer or a fixed positive time independent of the quantity. In reality, however, the time for transferring load from truck to trailer increases with the transferred quantity. In this section, we discuss the non-trivial modifications that are needed to exactly model quantity-dependent load transfer times in the column-generation subproblem. The modified SPPRC in this case must consider the tradeoff between the visit time at parking locations and the free capacity in the truck. A similar issue arises in the context of ship routing and scheduling when port visits are time-constrained and the port stay times are influenced by the amount of cargo loaded or unloaded. Brønmo *et al.* (2007) and Hennig *et al.* (2012) study such problems and respectively apply column generation and branch-and-price. Brønmo *et al.* solve an LP to determine an optimal schedule and load quantities for given port visit sequences. Hennig *et al.* use nested branch-and-price: they solve the subproblem of determining a time- and load-feasible ship route (port visit sequence) by decomposing it into a master problem that computes schedules and load patterns and a subproblem for computing ship routes (a shortest path problem with time windows solved by dynamic programming). However, as far as we know, we herewith present the first labeling algorithm for this type of tradeoff, i.e., the first that handles the determination of route, schedule, and load transfer quantity simultaneously.

In the following, let $\rho > 0$ be the transfer rate for the quantity transferrable from the truck to the trailer during one time unit. Because the technical equipment of the vehicles usually determines the transfer rate, the latter can be assumed independent of the transfer location. Note that with quantity-dependent transfer times, the strategy to always transfer as much as possible from the truck to the trailer is no longer optimal. If more load than necessary is transferred, the additional delay may cause a time window of a customer visited later to be missed. Therefore, the load-time tradeoff needs to be taken into account explicitly.

Figure 4.5 depicts the route $(d^+, d(p), 1, \tau(p), 2, c(p), d^-)$ visiting customers 1 and 2 and the decoupling vertex $d(p)$, the transfer vertex $\tau(p)$, and the coupling vertex $c(p)$ of a parking location p, where load can be transferred. Below each parking and customer vertex, a diagram of truck load vs. time is presented. The time axis is marked with

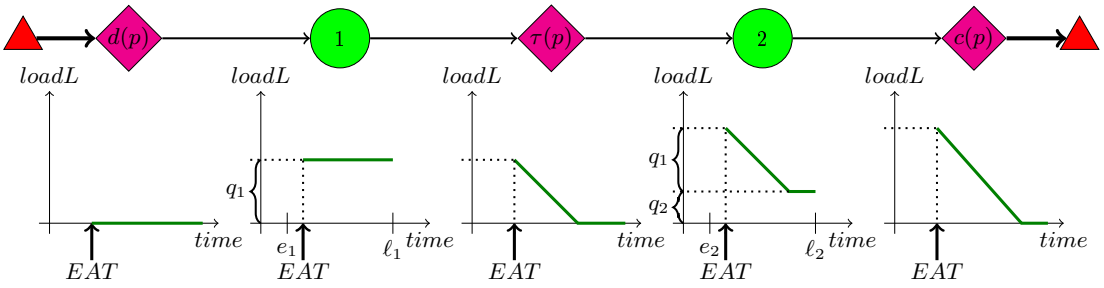
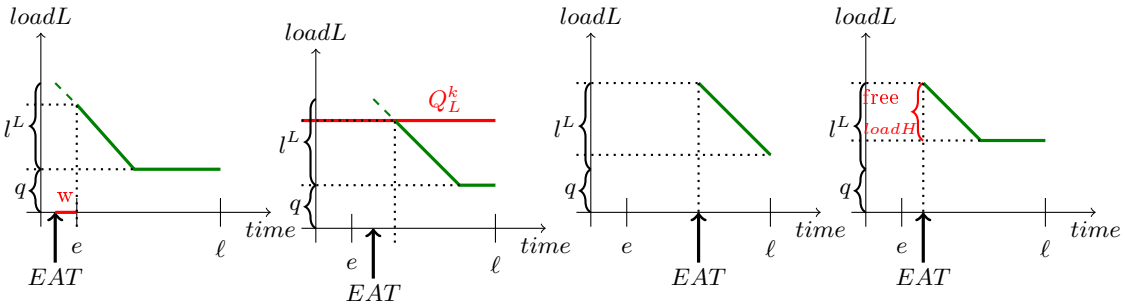


Figure 4.5: Example for tradeoff propagation with quantity-dependent transfer times

EAT representing the *earliest arrival time* at the vertex (this time can be before the service time window opens). Moreover, at customers 1 and 2, the time windows $[e_1, \ell_1]$ and $[e_2, \ell_2]$ are shown. We discuss the five tradeoff curves (depicted in green) for the vertices $d(p), 1, \tau(p), 2$, and $c(p)$. Each tradeoff curve displays to which extent a later start of service (starting not earlier than E_i^{time}) allows to reduce the load inside the truck (resource E_i^{loadL}). When the decoupling vertex $d(p)$ is reached, no load has been collected yet. Thus, no transfer possibility arises at $d(p)$, reflected by the 0-function. During the first subtour to customer 1, load has to be collected to the truck so that still no tradeoff is visible. When the truck meets its trailer again at $\tau(p)$, the option is to either continue the route immediately (with q_1 loaded in the truck) or to spend time to transfer load from truck to trailer. The non-zero slope of the tradeoff curve is $-\rho$. In our example, we assume that the truck as well as the trailer each have sufficient capacity to accommodate the complete supply of both customers, $q_1 + q_2$. Hence, the tradeoff curves in the diagrams below $\tau(p)$ and below customer 2 indicate that the supply of customer 1 can be partly or completely transferred to the trailer at $\tau(p)$. Likewise, the curve in the diagram below $c(p)$ shows that the entire customer supply can either remain in the truck or be transferred partly or completely.



(a) Waiting time used for transfer

(b) Transfer necessary because of truck capacity

(c) Time window limits transfer time

(d) Trailer capacity limits transferrable load

Figure 4.6: Special cases for arrival at customer concerning the tradeoff

In general, a non-zero time-load tradeoff can or must be consumed or reduced. Figure 4.6 illustrates four cases when this occurs. All variants depict a possible situation for the visit of a customer with service time window $[e, \ell]$, in which l^L refers to the maximum potentially transferable load on board the truck before the visit and q is the supply of the currently visited customer. First, if a customer is reached before the time window opens, it is an optimal policy to use as much as possible of the unavoidable waiting time for transfer, see Figure 4.6(a). Second, if the available capacity of the truck does not suffice to collect the complete supply, the surplus must be transferred lowering the disposable amount of load, see Figure 4.6(b). Third, if a customer is reached within its service time window then the remaining time limits the transfer time, see Figure 4.6(c). Fourth and finally, the available capacity of the trailer limits the transferrable amount, see Figure 4.6(d).

4.5.1 Time-Load Tradeoff, Resources, and their Propagation

The general form of the time-load tradeoff curve is shown in Figure 4.7. It can be defined by three parameters: the earliest possible starting time of the service at the vertex (E_i^{time}), the highest possible truck load at this time ($E_i^{\overline{loadL}}$), and the minimal achievable truck load ($E_i^{\underline{loadL}}$). The latter two values define two new resources and replace the resource E_i^{loadL} of the case with constant transfer times handled in Section 4.3.2. Accordingly, new resource extension functions for the resources E_i^{time}, $E_i^{\overline{loadL}}$, $E_i^{\underline{loadL}}$, and E_i^{loadH} must be defined whereas the propagation and feasibility checks of all other resources ($E_i^{cost}, E_i^{posH}, E_i^{cust_n}$, and $E_i^{collPosH}$) are not affected.

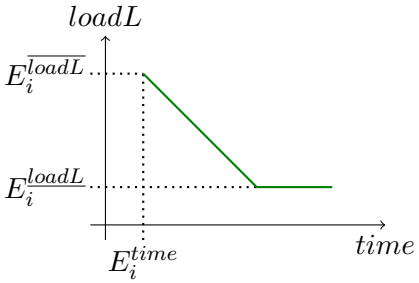


Figure 4.7: Shape of the tradeoff curve

Let $(i, j) \in A^k$ be any arc in the subnetwork of a complete vehicle $k \in K$. We define some auxiliary values useful for defining the new REFs for propagating the new resources along (i, j). First, the load that can be transferred in the waiting time ($LTWT$) depends on the unavoidable waiting time, which is the time difference between the earliest arrival time and the start of the time window, and it is also limited by the maximum potentially transferable load at previous transshipment locations. Hence, this value can be defined as

$$R_{ij}^{LTWT}(E_i) = \min\{\rho \cdot \max\{e_j - (E_i^{time} + t_{ij}), 0\}, E_i^{\overline{loadL}} - E_i^{\underline{loadL}}\}.$$

Second, in order to serve the following customer j, there may be some load that must be

transferred to the trailer before ($NLTB$). This amount is

$$R_{ij}^{NLTB}(E_i) = \max\{E_i^{\overline{loadL}} + q_j - Q_L^k, 0\}.$$

Third, the minimal load that has to remain in the truck ($MinLL$) is restricted by the trailer's capacity and the time window of the current vertex j, i.e.,

$$R_{ij}^{MinLL}(E_j) = \max\{0, E_j^{\overline{loadL}} - (Q_H^k - E_j^{loadH}), E_j^{\overline{loadL}} - \rho \cdot (\ell_j - E_j^{time})\}.$$

The REFs for extending a label E_i along an arc (i, j) primarily depend on the type of the head vertex j. Therefore, we present the REFs by distinguishing the following three cases:
(1) the arrival at a customer without the trailer,
(2) the arrival at a customer with the trailer attached, and
(3) the arrival at a coupling, transfer, decoupling, or the depot vertex d^-.
For convenience, we also present all REFs for the inclusion of quantity-dependent transfer times in the standard form, i.e., grouped by resources, in the Appendix, Section 4.A.

Arrival at Customer without Trailer. All load has to be collected to the truck, but sometimes the truck load can ($LTWT$) or must ($NLTB$) be reduced.

$$\begin{aligned}
E_j^{time} &= \max\{e_j,\ E_i^{time} + t_{ij} + R_{ij}^{NLTB}(E_i)/\rho\} \\
E_j^{\overline{loadL}} &= E_i^{\overline{loadL}} + q_j - \max\{R_{ij}^{NLTB}(E_i),\ R_{ij}^{LTWT}(E_i)\} \\
E_j^{loadH} &= E_i^{loadH} + \max\{R_{ij}^{NLTB}(E_i),\ R_{ij}^{LTWT}(E_i)\} \\
E_j^{\underline{loadL}} &= \max\{E_i^{\underline{loadL}} + q_j, R_{ij}^{MinLL}(E_j)\}
\end{aligned}$$

The label E_j resulting from the extension along arc $(i, j) \in A^k$ is feasible if $E_j^{time} \leq \ell_j$, $E_j^{loadH} \leq Q_H^k$, and $R_{ij}^{NLTB} \leq E_i^{\overline{loadL}} - E_i^{\underline{loadL}}$ holds in addition to the conditions for the remaining resources (see Section 4.3.2).

Arrival at Customer with Trailer attached. The supply is collected as much as possible directly to the trailer to keep the truck flexible. Thus, the maximal truck load $E_j^{\overline{loadL}}$ either remains unchanged or is reduced during the unavoidable waiting time ($LTWT$) or is increased if the residual trailer capacity does not suffice to collect the whole supply. The trailer load E_j^{loadH} is adapted accordingly. The minimal truck load $E_j^{\underline{loadL}}$ is increased by the supply collected to the truck and limited as above by the residual trailer capacity and the remaining time slack ($MinLL$).

$$\begin{aligned}
E_j^{time} &= \max\{e_j,\ E_i^{time} + t_{ij}\} \\
E_j^{\overline{loadL}} &= E_i^{\overline{loadL}} - \min\{R_{ij}^{LTWT}(E_i), Q_H^k - E_i^{loadH} - q_j\} \\
E_j^{loadH} &= E_i^{loadH} + q_j + \min\{R_{ij}^{LTWT}(E_i), Q_H^k - E_i^{loadH} - q_j\}
\end{aligned}$$

$$E_j^{\overline{loadL}} = \max\{E_i^{\overline{loadL}} + \max\{E_i^{loadH} + R_{ij}^{LTWT}(E_i) + q_j - Q_H^k, 0\}, R_{ij}^{MinLL}(E_j)\}$$

Similarly to the first case, the label E_j is feasible w.r.t. these resources if $E_j^{time} \leq \ell_j$ and the minimal truck load does not exceed the truck capacity, i.e., $E_j^{\overline{loadL}} \leq Q_L^k$.

Arrival at Coupling, Transfer, Decoupling, or Depot. No change in the loads is realized, only the potential transfer is limited by the free capacity in the trailer.

$$\begin{aligned} E_j^{time} &= E_i^{time} + t_{ij} \\ E_j^{\overline{loadL}} &= E_i^{\overline{loadL}} \\ E_j^{loadH} &= E_i^{loadH} \\ E_j^{\underline{loadL}} &= R_{ij}^{MinLL}(E_j) \end{aligned}$$

The only feasibility condition for these resources is $E_j^{time} \leq \ell_j$ when arriving at the depot $j = d^-$.

4.5.2 Dominance

The original dominance has to be extended to model the tradeoff correctly. A label E_j is better than another label E'_j belonging to the same vertex j, if the dominance criteria (4.4) concerning the unchanged resources are fulfilled and if furthermore the tradeoff curve of E_j lies completely left below the curve of E'_j. This means that, at every feasible service start time of label E'_j, the truck load of E_j is less than or equal to the truck load of E'_j. Because the slope of the decreasing part of all tradeoff curves is equal, this can be expressed by requiring a smaller or equal truck load of label E_j at the starting time of E'_j and a not greater minimal truck load in addition to the not greater time. So E_j dominates E'_j, if the following conditions hold:

$$\begin{aligned} &\text{(4.4a) with } res \in \{cost, time, cust_n(n \in N)\},\ \text{(4.4c)},\ \text{(4.4d)} \\ &E_j^{\overline{loadL}} + E_j^{loadH} \leq E_j'^{\overline{loadL}} + E_j'^{loadH} \\ &E_j^{\overline{loadL}} - \rho \cdot (E_j'^{time} - E_j^{time}) \leq E_j'^{\overline{loadL}} \\ &E_j^{\underline{loadL}} \leq E_j'^{\underline{loadL}} \end{aligned}$$

4.5.3 Bidirectional Labeling

As described in Section 4.3.2, backward propagation can be done on a reversed network. The merge conditions need to be adapted slightly to consider the time-load tradeoff. The total vehicle capacity limits the sum of the maximal truck loads and the trailer loads. The combined truck load has to respect the capacity but can use the transfer potentials from both partial paths. The time difference between the forward and the backward label can be used for transfer, but is restricted by the sum of the transfer potentials

($\overline{loadL} - \underline{loadL}$). Thus, a forward label E_i can be concatenated with a backward label E'_i whose predecessor is given by $E'_j = pred(E'_i)$, if and only if the following conditions are respected:

$$\text{(4.6a), (4.6d)–(4.6f)}$$

$$E_i^{\overline{loadL}} + E_i^{loadH} + E'^{\overline{loadL}}_j + E'^{loadH}_j \leq Q_L^k + Q_H^k$$

$$E_i^{\overline{loadL}} + E'^{\overline{loadL}}_j$$
$$- \min\{\rho \cdot (E'^{time}_j - t_{ij} - E_i^{time}), E_i^{\overline{loadL}} - E_i^{\underline{loadL}} + E'^{\overline{loadL}}_j - E'^{\underline{loadL}}_j\} \leq Q_L^k$$

4.6 Computational Study

The algorithms were implemented in C++ and use the LP and MIP solver of CPLEX 12.6.0. All computations were performed in single thread mode on a PC with an Intel Core i7-4790K CPU clocked at 3.60 GHz, with 8 GB RAM, running Windows 7 Enterprise. Apart from the single-thread mode, default settings were used for the CPLEX MIP solver. All tests were run with a CPU time limit of one hour.

4.6.1 Instances

To evaluate the performance of our algorithm, we used two different test sets.

The first instance set was developed by Drexl (2011). The instances are structured so as to resemble the situation of raw milk collection in Bavaria, Germany. Each instance has the same number n of truck customers, trailer customers, and dedicated transshipment locations, with $n \in \{6, \ldots, 10, 25\}$. In the following, these instances are denoted by "TTRP_n". Note that the number of locations is $1 + 3n$ and the number of vertices equals $2 + 8n$, as described in Section 4.3.2. For each value of n, there are 30 different instances. The vehicle fleet is heterogeneous, with two different truck types that can both be coupled with a suitable trailer, leading to four vehicle classes altogether. Costs are incurred for the covered distance, with additional costs for a coupled trailer. The customers have no time windows, but a maximal route duration is given. We do not apply the strict parking rule of Parragh and Cordeau (2017) for these instances.

The second instance set was created by Parragh and Cordeau (2017) from the Solomon instances (Solomon, 1987) by identifying truck customers as described by Lin *et al.* (2011): customers were sorted by increasing distance from their nearest neighbor, and from each instance, three new instances were derived by taking the first 25 %, 50 %, and 75 % of customers as truck customers and the remaining ones as trailer customers. These instances are henceforth referred to by "OriginalName_n_p", with n indicating the number of customers (25, 50, 100) and p standing for the rounded percentage of truck customers (25, 50, 75). The number of locations is $1+n$ and the number of vertices equals $2 + pn + 4(1 - p)n$. Besides, a trailer capacity was added, so that there are two vehicle classes, one representing a complete vehicle and one a truck without a trailer. As known, the Solomon instances have time windows, and costs and times are proportional to the

Euclidean distance between locations, independent of whether or not a trailer is attached. Note that we rounded up the times to one decimal place. In these instances, there are no dedicated transshipment locations. As mentioned, Parragh and Cordeau (2017) require that trailer customer locations be used as parking places only if their supply is collected during the stop. Thus, we also respect this strict parking rule for these instances.

4.6.2 Results

Preliminary tests showed that the influence of a reduced network, a maximal number of columns to add per pricing iteration, and the size of the ng-neighborhood (ng-size) can be significant. We found that the best results concerning the thinned-out network were obtained by keeping, for each customer, only the arcs leading to the 7 most promising neighbors (those with lowest reduced costs) and to the 5 nearest transshipment locations. Also the depot was connected only to these 12 best neighbors. The maximal number of columns to add per pricing iteration was chosen dependent on the instance size: it was best to add at most six times the number of vertices. The best ng-size for the TTRP-instances was 6, whereas the modified Solomon instances could be solved better with an ng-size of 8. We limited the overall number of subset-row cuts to 120 and generated a maximum of 10 cuts simultaneously. We required a minimum violation of 0.1 and stopped the separation after the third level of the branch-and-bound tree. We report all computation times in seconds. Moreover, we define the relative optimality gap between the best found upper bound ub and the lower bound lb at termination as $(ub - lb)/ub \cdot 100\,\%$.

In the following, we present aggregated results for the two instance sets. More detailed results, including the individual objective function values, can be found in the Appendix, Section 4.B. Note that, unfortunately, the extended dominance rules described in Section 4.3.2 did not consistently reduce solution times; for several instances, the stronger dominance was offset by the increased computing time for the dominance check. For this reason, all presented results were obtained without the extended dominance tests.

The aggregated results for the TTRP instances are reported in Table 4.3. Only ten of the TTRP instances with n between 6 and 10 could not be solved to optimality within one hour. Compared to the results of Drexl (2011), 36 new optimal solutions were found. The high number of instances solved in the extended root node by incorporating the subset-row cuts demonstrates the effectiveness of these inequalities. The highest remaining optimality gap overall was less than 5 %, and the largest instance that could be solved to optimality had 76 locations, i.e., 202 vertices.

The aggregated results for the Solomon-based instances with 25 customers are presented in Table 4.4. All in all, only 2 of these 168 Solomon-based instances could not be solved to optimality within one hour of CPU time. As expected, a higher percentage of truck customers simplifies the problem, because fewer parking possibilities exist. Also for the Solomon-based instances, the subset-row inequalities had a large influence on the results. In the Appendix, Section 4.B, in Tables 4.8 and 4.9, we also show our results for the 50- and 100-customer Solomon-based instances presented by Parragh and Cordeau (2017). Compared to the results of Parragh and Cordeau (2017), an optimal solution

Instance set	# instances solved to optimality			∅ CPU time	∅ gap
	Total	Root w/o cuts	Root with cuts		
TTRP_6	30/30	6	29	1.5	0.00
TTRP_7	30/30	6	28	36.7	0.00
TTRP_8	29/30	4	27	182.4	0.02
TTRP_9	27/30	3	25	392.5	0.05
TTRP_10	24/30	2	20	842.4	0.20
TTRP_25	1/30	0	1	3559.5	1.51
Total	141/180	21	130	835.8	0.29

Table 4.3: Aggregated results for TTRP instances

was found for 35 additional instances.

Running the Solomon-based instances for a two-day planning horizon yielded the results shown in Table 4.5. We assumed that every second customer was allowed to be visited every other day. A comparison of the effectiveness of the root stabilization described in Section 4.4 leads to a surprising result. Despite a much smaller root solution time when incorporating the stabilization methods, the total solution time was not reduced consistently, and even some significantly larger optimality gaps occurred. The reason is that many optimal solutions can be found already in the extended root node (with the SR cuts). In the case of root stabilization, the former integral solution is split evenly between both days, so that a potential integrality is mostly destroyed. Overall, nearly 70 % of the two-day instances could be solved to optimality within one hour of CPU time, although the number of tasks is twice as high as for the one-day planning horizon. Moreover, it is remarkable that more than 14 % of the costs can be saved by serving a part of the customers only every other day.

Results concerning the quantity-dependent transfer times were obtained with the Solomon-based instances, as the ones of Drexl (2011) have no time windows and are thus not influenced by non-constant transfer times. We assumed a transfer rate of one unit per second, i.e., $\rho = 1$. The computational results are summarized in Table 4.6. In the instance sets R1, R2, and RC2, all optimal solutions matched those for constant transfer times. This is simply because in both versions, all routes of optimal solutions are performed by single trucks. In the C1, C2, and RC1 instance groups, however, several optimal solutions differed from those for constant transfer times and had higher costs. Nevertheless, it is clear that the impact of quantity-dependent transfer times is limited if the time windows are not tight and if the amount collected during all subtours barely exceeds the truck capacity. Finally, it is noteworthy that, despite the more complicated resource extension functions, the number of solved instances and the solution times are nearly as good as those for constant transfer times.

Instance set	# instances solved to optimality			∅ CPU time	∅ gap
	Total	Root w/o cuts	Root with cuts		
C1_25_25	9/9	8	9	48.3	0
C1_25_50	9/9	3	9	43.2	0
C1_25_75	9/9	0	8	351.7	0
C2_25_25	7/8	0	8	846.9	0.10
C2_25_50	8/8	0	8	602.5	0
C2_25_75	8/8	0	8	335.6	0
R1_25_25	12/12	5	11	1.4	0
R1_25_50	12/12	5	12	0.5	0
R1_25_75	12/12	5	12	0.2	0
R2_25_25	10/11	1	10	531.3	0.00
R2_25_50	11/11	1	11	301.2	0
R2_25_75	11/11	1	11	52.4	0
RC1_25_25	8/8	7	8	3.1	0
RC1_25_50	8/8	7	8	1.4	0
RC1_25_75	8/8	7	8	0.5	0
RC2_25_25	8/8	8	8	339.3	0
RC2_25_50	8/8	8	8	227.9	0
RC2_25_75	8/8	8	8	37.0	0
Total	166/168	74	164	195.8	0.01

Table 4.4: Aggregated results for Solomon-based instances with 25 customers

4.7 Conclusions

In this paper, we have studied the truck-and-trailer routing problem with time windows and two new extensions: the consideration of a two-day planning horizon where it is allowed to visit some customers every other day, and the inclusion of a quantity-dependent time consumption for the transfer of load from a truck to its trailer. All variants were tackled by effective branch-and-price-and-cut algorithms, using subset-row inequalities to strengthen the lower bound of the linear relaxation. The two-day planning horizon leads to undesirable symmetries. To deal with these, we proposed a constraint-aggregation and stabilization procedure based on the concept of deep dual-optimal inequalities. Quantity-dependent load transfer times cause a tradeoff between the time needed for the transfer operation and the truck capacity gained. This requires additional resources and more complicated resource extension functions in the labeling algorithm for solving the pricing problems.

Computational experiments were performed with established TTRPTW benchmark instances. The results compare very favorably with those known from the literature; many instances could be solved to optimality for the first time. We attribute this success

	With stabilization				Without stabilization				
	# optimal		∅ CPU time		# optimal		∅ CPU time		
Instance set	Total	Root with cuts	Total	Root with cuts	Total	Root with cuts	Total	Root with cuts	∅ cost savings in %
C1_25	15/27	9	1695.6	137.5	15/27	14	1763.7	368.2	4.01
C2_25	10/24	7	2192.3	744.5	10/24	8	2282.7	1138.3	10.88
R1_25	29/36	2	1044.2	5.6	27/36	13	1175.6	19.3	17.46
R2_25	23/33	18	1537.9	912.9	21/33	15	1640.5	1047.0	22.13
RC1_25	20/24	18	609.2	8.9	21/24	21	480.3	31.0	14.56
RC2_25	18/24	18	916.1	776.1	18/24	18	917.0	849.6	14.76
Total	115/168	72	1329.4	421.1	112/168	89	1383.3	557.4	14.48

Table 4.5: Aggregated results for Solomon-based instances with 25 customers and two-day planning horizon

Instance set	# optimal	∅ CPU time	∅ gap	# changed solutions	∅ cost increase in %
C1_25	26/27	199.4	0.00	5	0.65
C2_25	23/24	749.3	0.07	1	0.04
R1_25	36/36	1.0	0	0	0
R2_25	32/33	297.6	0.01	0	0
RC1_25	24/24	6.8	0	22	6.35
RC2_25	24/24	249.1	0	0	0
Total	165/168	234.3	0.01	28	1.02

Table 4.6: Aggregated results for Solomon-based instances with 25 customers and quantity-dependent transfer times

mainly to the use of subset-row cuts and the new bidirectional labeling.

Further research can be done regarding algorithmic as well as problem aspects. As for algorithmics, for the two-day planning horizon, an improved transition from the symmetric to the asymmetric treatment of the problem might reduce the run times significantly. To this end, it would be necessary to devise a more sophisticated procedure for finding suitable subsets of the first-day columns to copy to the second day.

From a modeling point of view, the next big step would be to abandon the fixed assignment of trailers to trucks. Especially if the service at customers takes a significant amount of time, it may be useful to leave a trailer at a customer for loading, continue the route with the truck only, and recouple the trailer later with the same or a different compatible truck. This complicates the problem considerably, because a synchronization between all vehicles becomes necessary and the pricing problems for the different vehicles become interdependent.

Acknowledgments

We would like to thank the three anonymous reviewers for their very careful reading and constructive comments that helped to significantly clarify the description of the presented material. The research of the second and the third author was supported by the Deutsche Forschungsgemeinschaft (DFG) under grants DR 963/2-1 and IR 122/6-1 respectively. This support is gratefully acknowledged.

Bibliography

Baldacci, R., Mingozzi, A., and Roberti, R. (2011c). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

Batsyn, M. and Ponomarenko, A. (2014). Heuristic for a real-life truck and trailer routing problem. *Procedia Computer Science*, **31**, 778–792.

Belenguer, J., Benavent, E., Martínez, A., Prins, C., Prodhon, C., and Villegas, J. (2016). A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots. *Transportation Science*, **50**(2), 735–749.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, **60**(5), 1167–1182.

Bodin, L. and Levy, L. (2000). Scheduling of local delivery carrier routes for the united states postal service. In M. Dror, editor, *Arc Routing: Theory, Solutions, and Applications*, pages 419–442. Springer, Boston, MA.

Brønmo, G., Christiansen, M., and Nygreen, B. (2007). Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society*, **58**(9), 1167–1177.

Caramia, M. and Guerriero, F. (2010a). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, **61**(7), 1168–1180.

Caramia, M. and Guerriero, F. (2010b). A milk collection problem with incompatibility constraints. *Interfaces*, **40**(2), 130–143.

Chao, I. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, **29**(1), 33–51.

Cuda, R., Guastaroba, G., and Speranza, M. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, **55**, 185–199.

Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing—Problems, heuristics and computational experience. *Computers & Operations Research*, **40**(2), 536–546.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, **58**(1), 179–192.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016a). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, **64**(6), 1388–1405.

Drexl, M. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Journal of Quantitative Methods for Economics and Business Administration*, **12**, 5–38.

Gerdessen, J. (1996). Vehicle routing problem with trailers. *European Journal of Operational Research*, **93**(1), 135–147.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Hennig, F., Nygreen, B., and Lübbecke, M. (2012). Nested column generation applied to the crude oil tanker routing and scheduling problem with split pickup and split delivery. *Naval Research Logistics*, **59**(3–4), 298–310.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Lin, S., Yu, V., and Lu, C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, **38**(12), 15244–15252.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Mirmohammadsadeghi, S. and Ahmed, S. (2015). Memetic heuristic approach for solving truck and trailer routing problems with stochastic demands and time windows. *Networks and Spatial Economics*, **15**(4), 1093–1115.

Parragh, S. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, **83**, 28–44.

Pasha, U., Hoff, A., and Løkketangen, A. (2014). A hybrid approach for milk collection using trucks and trailers. *Annals of Management Science*.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, **33**(4), 894–909.

Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, **41**(4), 469–488.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**(2), 254–265.

Tan, K., Chew, Y., and Lee, L. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, **172**(3), 855–885.

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., and Meisel, F. (2016). Branch-and-price for the active-passive vehicle-routing problem. *Transportation Science*. Forthcoming.

Torres, I., Cruz, C., and Verdegay, J. L. (2015). Solving the truck and trailer routing problem with fuzzy constraints. *International Journal of Computational Intelligence Systems*, **8**(4), 713–724.

Tricoire, F. (2016). Private communication.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., and Velasco, N. (2010). GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, **23**(5), 780–794.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., and Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, **38**(9), 1319–1334.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, **230**(2), 231–244.

Villeneuve, D. and Desaulniers, G. (2005). The shortest path problem with forbidden paths. *European Journal of Operational Research*, **165**(1), 97–107.

Appendix

4.A Resource Extension Functions for Quantity-Dependent Load Transfer Times

In this section, we present the complete set of REFs for quantity-dependent load transfer times (see Section 4.5) grouped by resource.

$$E_j^{cost} = E_i^{cost} + \tilde{c}_{ij}^k(\delta_{E_i^{posH},\perp})$$

$$E_j^{time} = \begin{cases} \max\{e_j,\ E_i^{time} + t_{ij} + R_{ij}^{NLTB}(E_i)/\rho\}, & j \in N \text{ and } E_i^{posH} \neq \perp \\ \max\{e_j,\ E_i^{time} + t_{ij}\}, & \text{otherwise.} \end{cases}$$

$$E_j^{\overline{loadL}} = \begin{cases} E_i^{\overline{loadL}} + q_j - \max\left\{R_{ij}^{NLTB}(E_i),\ R_{ij}^{LTWT}(E_i)\right\}, & j \in N \text{ and } E_i^{posH} \neq \perp \\ E_i^{\overline{loadL}} - \min\{R_{ij}^{LTWT}(E_i), Q_H^k - E_i^{loadH} - q_j\}, & j \in N^H \text{ and } E_i^{posH} = \perp \\ E_i^{\overline{loadL}}, & j \in D \cup \mathcal{T} \cup C \cup d^- \end{cases}$$

$$E_j^{\underline{loadL}} = \begin{cases} \max\{E_i^{\underline{loadL}} + q_j, R_{ij}^{MinLL}(E_j)\}, & j \in N \text{ and } E_i^{posH} \neq \perp \\ \max\{E_i^{\underline{loadL}} + \max\{E_i^{loadH} + q_j - Q_H^k, 0\}, R_{ij}^{MinLL}(E_j)\}, & j \in N^H \text{ and } E_i^{posH} = \perp \\ R_{ij}^{MinLL}(E_j), & j \in D \cup \mathcal{T} \cup C \cup d^- \end{cases}$$

$$E_j^{loadH} = \begin{cases} E_i^{loadH} + \max\{R_{ij}^{NLTB}(E_i),\ R_{ij}^{LTWT}(E_i)\}, & j \in N \text{ and } E_i^{posH} \neq \perp \\ E_i^{loadH} + q_j + \min\{R_{ij}^{LTWT}(E_i), Q_H^k - E_i^{loadH} - q_j\}, & j \in N^H \text{ and } E_i^{posH} = \perp \\ E_i^{loadH}, & j \in D \cup \mathcal{T} \cup C \cup d^- \end{cases}$$

$$E_j^{posH} = \begin{cases} E_i^{posH}, & j \in N \cup \mathcal{T} \\ p, & j \in D \text{ where } j = d(p), p \in P \\ \perp, & j \in C \cup \{d^-\} \end{cases}$$

$$E_j^{cust_n} = \begin{cases} E_i^{cust_n} + 1, & j \in N \text{ and } j = n \\ \max\{E_i^{cust_n}, U_n(E_j)\}, & \text{otherwise.} \end{cases}$$

4.B Detailed Computational Results

Table 4.7 shows a comparison of the performance of the different labeling directions in the pricing problem. For the monodirectional forward, the monodirectional backward and the bidirectional labeling the number of instances solved to optimality, the average CPU time in seconds, the average relative gap between best upper bound and the lower bound at termination, and the number of generated labels per instance group in 10^6 are given. As expected, the bidirectional labeling could reduce the runtime by about 40 % compared to the monodirectional labeling.

	Forward				Backward				Bidirectional			
Instance set	Opt.	Time	Gap	#Lab.	Opt.	Time	Gap	#Lab.	Opt.	Time	Gap	#Lab.
C1_25	26/27	301.7	0.03	11.70	26/27	462.2	0.05	16.23	27	147.8	0	8.10
C2_25	21/24	937.5	0.20	24.67	20/24	1134.3	0.27	29.87	23	595.0	0.05	20.05
R1_25	36/36	1.0	0	0.46	36/36	1.6	0	0.54	36	0.7	0	0.36
R2_25	32/33	446.9	0.08	16.39	32/33	408.0	0.00	15.58	32	295.0	0,00	11.98
RC1_25	24/24	2.5	0	1.06	24/24	8.6	0	1.67	24	1.7	0	0.88
RC2_25	24/24	292.6	0	6.18	24/24	137.4	0	5.60	24	201.4	0	5.08
Total	163/168	312.6	0.05	9.76	162/168	337.7	0.05	11.09	166/168	195.9	0.01	7.45

Table 4.7: Comparison of labeling strategies for Solomon-based instances with 25 customers

The following tables show detailed computational results for the Solomon-based instances as well as for the TTRP instances with varying number of customers and additional transshipment locations. All columns have the same headers and inform about the tested instance, the lower bound, the upper bound, the resulting optimality gap, the solution time in seconds (with T.L. indicating the time limit of 1 hour), the number of added subset-row inequalities, the number of branch-and-bound nodes and the number of generated routes. Instances that could be solved to optimality for the first time are printed in bold.

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
C101_50_25	396.02	396.02	0	3.1	0	1	2,198
C101_50_50	426.08	426.08	0	3.6	20	1	2,347
C101_50_75	453.39	453.39	0	4.4	60	1	2,053
C201_50_25	391.97	391.97	0	145.8	20	1	20,891
C201_50_50	403.69	406.30	0.24	T.L.	110	1	20,315
C201_50_75	403.39	406.42	0.48	T.L.	110	2	27,505
R101_50_25	1046.70	1046.70	0	0.1	0	1	595
R101_50_50	1046.70	1046.70	0	0.1	0	1	525
R101_50_75	1046.70	1046.70	0	0	0	1	626
R201_50_25	794.34	794.34	0	16.3	10	1	4,882
R201_50_50	794.34	794.34	0	1.7	0	1	5,778
R201_50_75	794.34	794.34	0	0.6	0	1	4,718
RC101_50_25	945.68	959.38	1.43	T.L.	120	82	104,339
RC101_50_50	959.92	959.92	0	22.7	120	5	4,936
RC101_50_75	976.20	976.20	0	6.2	110	1	1,843
RC201_50_25	686.31	686.31	0	17.7	0	1	5,628
RC201_50_50	686.31	686.31	0	3.1	0	1	4,816
RC201_50_75	686.31	686.31	0	0.7	0	1	4,657
Average			0.12	612.6			

Table 4.8: Detailed results for Solomon-based instances with 50 customers

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
C101_100_25	899.75	899.75	0	80.7	20	1	7,646
C101_100_50	986.41	1003.86	1.74	T.L.	120	1	91,990
C101_100_75	1042.04	1043.28	0.12	T.L.	120	1	75,380
C201_100_25	662.46	695.51	1.34	T.L.	20	1	65,535
C201_100_50	692.67	719.04	3.67	T.L.	60	1	66,124
C201_100_75	704.40	737.08	0.90	T.L.	120	2	41,711
R101_100_25	1644.64	1644.64	0	28.9	23	1	11,048
R101_100_50	1644.64	1644.64	0	2.7	13	1	4,278
R101_100_75	1644.64	1644.64	0	1.3	14	1	3,055
R201_100_25	1146.69	1161.48	0.16	T.L.	50	1	53,157
R201_100_50	1147.65	1247.81	0.01	T.L.	120	1	61,870
R201_100_75	1147.80	1147.80	0	687.0	120	1	54,211
RC101_100_25	1653.93	1653.93	0	1337.9	120	82	48,098
RC101_100_50	1706.56	1734.14	1.59	T.L.	120	5	141,283
RC101_100_75	1770.71	1788.68	1.00	T.L.	120	1	88,300
RC201_100_25	1258.73	1290.68	0.93	T.L.	30	1	44,097
RC201_100_50	1264.60	1265.56	0.08	T.L.	120	1	91,870
RC201_100_75	1265.56	1265.56	0	581.3	120	1	42,941
Average			0.64	2351.1			

Table 4.9: Detailed results for Solomon-based instances with 100 customers

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
C101_25_25	205.21	205.21	0	0.1	0	1	714
C101_25_50	219.58	219.58	0	0.3	10	1	896
C101_25_75	235.18	235.18	0	0.4	50	1	815
C102_25_25	203.68	203.68	0	8.3	0	1	2,624
C102_25_50	218.34	218.34	0	3.2	10	1	1,755
C102_25_75	235.18	235.18	0	33.4	88	1	1,660
C103_25_25	203.68	203.68	0	28.8	0	1	2,477
C103_25_50	215.88	215.88	0	24.1	20	1	2,234
C103_25_75	235.18	235.18	0	120.8	77	1	2,506
C104_25_25	200.39	200.39	0	392.2	10	1	3,361
C104_25_50	212.89	212.89	0	356.5	20	1	4,026
C104_25_75	232.71	232.71	0	2,995.0	91	5	6,290
C105_25_25	204.92	204.92	0	0.2	0	1	977
C105_25_50	218.95	218.95	0	0.4	10	1	931
C105_25_75	235.18	235.18	0	0.9	70	1	795
C106_25_25	205.21	205.21	0	0.1	0	1	761
C106_25_50	219.58	219.58	0	0.4	10	1	1,042
C106_25_75	235.18	235.18	0	0.6	55	1	759
C107_25_25	204.92	204.92	0	0.5	0	1	684
C107_25_50	208.94	208.94	0	0.3	0	1	688
C107_25_75	227.08	227.08	0	0.6	40	1	738
C108_25_25	204.92	204.92	0	1.3	0	1	1,415
C108_25_50	208.94	208.94	0	0.7	0	1	925
C108_25_75	227.08	227.08	0	2.3	40	1	1,521
C109_25_25	204.31	204.31	0	3.6	0	1	1,624
C109_25_50	208.33	208.33	0	3.3	0	1	1,755
C109_25_75	227.08	227.08	0	11.2	50	1	2,133
C201_25_25	223.07	223.07	0	4.4	30	1	3,643
C201_25_50	223.07	223.07	0	1.5	30	1	2,423
C201_25_75	227.29	227.29	0	15.6	80	1	3,602
C202_25_25	220.52	220.52	0	84.1	30	1	6,076
C202_25_50	220.52	220.52	0	15.6	30	1	4,987
C202_25_75	220.52	220.52	0	4.2	30	1	4,344
C203_25_25	220.52	220.52	0	962.5	50	1	7,440
C203_25_50	220.52	220.52	0	109.7	40	1	7,096
C203_25_75	220.52	220.52	0	32.6	50	1	7,325
C204_25_25	217.38	217.38	0	1,347.2	10	1	11,579
C204_25_50	218.16	218.16	0	706.4	20	1	10,014
C204_25_75	220.32	220.32	0	645.6	60	1	9,130
C205_25_25	223.07	223.07	0	16.1	30	1	4,712
C205_25_50	223.07	223.07	0	5.5	30	1	5,329
C205_25_75	225.66	225.66	0	5.5	60	1	3,793
C206_25_25	223.07	223.07	0	26.2	30	1	5,437
C206_25_50	223.07	223.07	0	10.0	30	1	4,677
C206_25_75	225.66	225.66	0	10.0	60	1	4,895
C207_25_25	222.94	225.45	1.11	T.L.	50	1	9,860
C207_25_50	222.87	222.87	0	3,599.9	50	1	7,797
C207_25_75	225.45	225.45	0	1,903.6	70	1	5,321
C208_25_25	222.90	222.90	0	683.4	40	1	7,162
C208_25_50	222.90	222.90	0	369.0	40	1	6,808
C208_25_75	225.49	225.49	0	67.3	70	1	5,046

Table 4.10: Detailed results for Solomon-based instances C

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
R101_25_25	618.33	618.33	0	0.0	0	1	134
R101_25_50	618.33	618.33	0	0.0	0	1	134
R101_25_75	618.33	618.33	0	0.0	0	1	129
R102_25_25	548.11	548.11	0	0.1	7	1	506
R102_25_50	548.11	548.11	0	0.1	7	1	506
R102_25_75	548.11	548.11	0	0.0	9	1	513
R103_25_25	455.70	455.70	0	0.2	0	1	733
R103_25_50	455.70	455.70	0	0.1	0	1	642
R103_25_75	455.70	455.70	0	0.0	0	1	541
R104_25_25	417.96	417.96	0	0.7	0	1	1,022
R104_25_50	417.96	417.96	0	0.2	0	1	809
R104_25_75	417.96	417.96	0	0.1	0	1	825
R105_25_25	531.54	531.54	0	0.0	0	1	247
R105_25_50	531.54	531.54	0	0.0	0	1	247
R105_25_75	531.54	531.54	0	0.0	0	1	237
R106_25_25	466.48	466.48	0	0.3	10	1	682
R106_25_50	466.48	466.48	0	0.1	10	1	595
R106_25_75	466.48	466.48	0	0.0	10	1	507
R107_25_25	429.20	429.20	0	1.5	20	1	780
R107_25_50	429.20	429.20	0	0.5	20	1	772
R107_25_75	429.20	429.20	0	0.2	10	1	676
R108_25_25	404.28	404.28	0	3.8	20	1	994
R108_25_50	404.28	404.28	0	2.0	30	1	1,126
R108_25_75	404.28	404.28	0	0.4	20	1	1,208
R109_25_25	442.63	442.63	0	0.0	0	1	490
R109_25_50	442.63	442.63	0	0.0	0	1	490
R109_25_75	442.63	442.63	0	0.0	0	1	451
R110_25_25	445.18	445.18	0	2.7	24	7	4,020
R110_25_50	445.18	445.18	0	0.3	20	1	509
R110_25_75	445.18	445.18	0	0.1	20	1	509
R111_25_25	429.70	429.70	0	0.6	10	1	1,051
R111_25_50	429.70	429.70	0	0.3	10	1	852
R111_25_75	429.70	429.70	0	0.1	10	1	744
R112_25_25	402.85	402.85	0	7.2	40	1	956
R112_25_50	402.85	402.85	0	2.8	40	1	932
R112_25_75	402.85	402.85	0	1.2	40	1	719

Table 4.11: Detailed results for Solomon-based instances R1

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
R201_25_25	464.38	464.38	0	0.3	9	1	968
R201_25_50	464.38	464.38	0	0.1	9	1	817
R201_25_75	464.38	464.38	0	0.1	9	1	817
R202_25_25	411.49	411.49	0	1.1	0	1	4,557
R202_25_50	411.49	411.49	0	0.7	0	1	4,339
R202_25_75	411.49	411.49	0	0.2	0	1	3,202
R203_25_25	392.33	392.33	0	27.6	10	1	6,646
R203_25_50	392.33	392.33	0	7.4	10	1	4,579
R203_25_75	392.33	392.33	0	2.7	10	1	3,878
R204_25_25	355.89	355.89	0	1,133.3	40	1	7,529
R204_25_50	355.89	355.89	0	1,348.9	80	1	8,153
R204_25_75	355.89	355.89	0	69.0	70	1	7,338
R205_25_25	394.06	394.06	0	2.2	10	1	3,454
R205_25_50	394.06	394.06	0	0.8	10	1	2,920
R205_25_75	394.06	394.06	0	0.5	10	1	2,813
R206_25_25	375.48	375.48	0	32.0	10	1	6,468
R206_25_50	375.48	375.48	0	6.3	10	1	7,001
R206_25_75	375.48	375.48	0	1.6	10	1	5,141
R207_25_25	362.64	362.64	0	259.4	10	1	10,796
R207_25_50	362.64	362.64	0	92.7	10	1	7,781
R207_25_75	362.64	362.64	0	32.8	10	1	7,724
R208_25_25	329.33	329.33	0	658.3	10	1	11,670
R208_25_50	329.33	329.33	0	230.4	10	1	12,023
R208_25_75	329.33	329.33	0	39.4	10	1	9,973
R209_25_25	371.56	371.56	0	67.1	60	1	6,042
R209_25_50	371.56	371.56	0	14.4	50	1	5,990
R209_25_75	371.56	371.56	0	4.8	50	1	4,458
R210_25_25	405.48	405.48	0	16.9	10	1	6,808
R210_25_50	405.48	405.48	0	5.9	10	1	5,540
R210_25_75	405.48	405.48	0	2.4	10	1	4,730
R211_25_25	351.75	351.91	0.04	T.L.	30	1	8,561
R211_25_50	351.91	351.91	0	1,605.6	50	1	8,862
R211_25_75	351.91	351.91	0	422.9	60	1	7,067

Table 4.12: Detailed results for Solomon-based instances R2

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
RC101_25_25	473.54	473.54	0	0.7	46	1	621
RC101_25_50	478.73	478.73	0	0.4	40	1	686
RC101_25_75	487.76	487.76	0	0.3	50	1	632
RC102_25_25	363.86	363.86	0	0.8	0	1	1,078
RC102_25_50	369.68	369.68	0	0.5	0	1	1,300
RC102_25_75	391.38	391.38	0	0.1	0	1	847
RC103_25_25	346.51	346.51	0	1.6	0	1	1,474
RC103_25_50	354.13	354.13	0	0.7	0	1	1,500
RC103_25_75	373.28	373.28	0	0.2	0	1	1,013
RC104_25_25	320.61	320.61	0	6.2	0	1	1,860
RC104_25_50	342.02	342.02	0	2.4	0	3	1,461
RC104_25_75	361.07	361.07	0	0.6	0	1	800
RC105_25_25	412.56	412.56	0	0.2	0	1	475
RC105_25_50	419.72	419.72	0	0.2	0	1	798
RC105_25_75	434.35	434.35	0	0.1	0	1	608
RC106_25_25	355.32	355.32	0	0.3	0	1	618
RC106_25_50	361.25	361.25	0	0.2	0	1	608
RC106_25_75	388.71	388.71	0	0.1	0	1	691
RC107_25_25	318.46	318.46	0	2.2	0	1	1,335
RC107_25_50	333.23	333.23	0	1.2	0	1	1,395
RC107_25_75	345.55	345.55	0	0.5	0	1	1,091
RC108_25_25	314.64	314.64	0	13.0	0	1	2,898
RC108_25_50	331.25	331.25	0	5.9	0	1	2,101
RC108_25_75	345.55	345.55	0	2.1	0	1	1,494
RC201_25_25	361.24	361.24	0	0.1	0	1	745
RC201_25_50	361.24	361.24	0	0.1	0	1	745
RC201_25_75	361.24	361.24	0	0.0	0	1	745
RC202_25_25	338.82	338.82	0	0.9	0	1	1,273
RC202_25_50	338.82	338.82	0	0.4	0	1	1,273
RC202_25_75	338.82	338.82	0	0.1	0	1	1,273
RC203_25_25	327.69	327.69	0	19.9	0	1	2,230
RC203_25_50	327.69	327.69	0	2.7	0	1	2,170
RC203_25_75	327.69	327.69	0	1.0	0	1	2,235
RC204_25_25	300.24	300.24	0	66.7	0	1	7,560
RC204_25_50	300.24	300.24	0	38.0	0	1	10,317
RC204_25_75	300.24	300.24	0	15.0	0	1	9,491
RC205_25_25	338.93	338.93	0	2.4	0	1	2,371
RC205_25_50	338.93	338.93	0	1.2	0	1	1,852
RC205_25_75	338.93	338.93	0	0.2	0	1	1,496
RC206_25_25	325.10	325.10	0	1.1	0	1	2,049
RC206_25_50	325.10	325.10	0	0.4	0	1	2,709
RC206_25_75	325.10	325.10	0	0.2	0	1	2,285
RC207_25_25	298.95	298.95	0	7.7	0	1	7,442
RC207_25_50	298.95	298.95	0	2.2	0	1	2,640
RC207_25_75	298.95	298.95	0	0.6	0	1	1,858
RC208_25_25	269.57	269.57	0	2,615.1	0	1	10,601
RC208_25_50	269.57	269.57	0	1,778.4	0	1	11,410
RC208_25_75	269.57	269.57	0	278.6	0	1	9,518

Table 4.13: Detailed results for Solomon-based instances RC

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
TTRP_6_0	163,874	163,874	0	1.1	16	1	1,004
TTRP_6_1	143,909	143,909	0	1.0	10	1	1,152
TTRP_6_2	145,645	145,645	0	0.4	0	1	999
TTRP_6_3	197,850	197,850	0	0.3	9	1	619
TTRP_6_4	178,383	178,383	0	0.2	10	1	568
TTRP_6_5	142,888	142,888	0	1.8	15	1	1,190
TTRP_6_6	145,040	145,040	0	1.5	10	1	1,235
TTRP_6_7	148,384	148,384	0	1.2	10	1	1,094
TTRP_6_8	173,353	173,353	0	1.4	20	1	644
TTRP_6_9	147,385	147,385	0	1.2	20	1	1,056
TTRP_6_10	157,903	157,903	0	0.5	3	1	857
TTRP_6_11	152,266	152,266	0	1.5	25	1	975
TTRP_6_12	163,427	163,427	0	3.4	30	1	1,016
TTRP_6_13	143,010	143,010	0	0.6	10	1	1,537
TTRP_6_14	164,496	164,496	0	1.7	10	1	949
TTRP_6_15	155,263	155,263	0	1.1	10	1	938
TTRP_6_16	165,577	165,577	0	0.7	0	1	837
TTRP_6_17	173,967	173,967	0	0.2	0	1	840
TTRP_6_18	149,120	149,120	0	2.2	19	1	870
TTRP_6_19	136,069	136,069	0	1.5	17	1	867
TTRP_6_20	113,710	113,710	0	1.2	0	1	1,655
TTRP_6_21	154,589	154,589	0	0.5	10	1	1,012
TTRP_6_22	157,212	157,212	0	1.0	10	1	867
TTRP_6_23	152,416	152,416	0	0.5	0	1	1,077
TTRP_6_24	157,880	157,880	0	2.6	10	1	1,212
TTRP_6_25	150,638	150,638	0	0.8	0	1	903
TTRP_6_26	173,962	173,962	0	0.7	10	1	770
TTRP_6_27	171,155	171,155	0	0.7	9	1	973
TTRP_6_28	178,840	178,840	0	2.7	20	1	1,000
TTRP_6_29	165,292	165,292	0	9.6	28	3	1,607
TTRP_7_0	177,536	177,536	0	2.3	20	1	1,230
TTRP_7_1	167,016	167,016	0	5.3	20	1	1,241
TTRP_7_2	189,518	189,518	0	0.4	10	1	759
TTRP_7_3	198,714	198,714	0	0.6	20	1	1,209
TTRP_7_4	187,852	187,852	0	1.4	10	1	616
TTRP_7_5	178,057	178,057	0	1.1	19	1	1,285
TTRP_7_6	171,881	171,881	0	70.7	56	3	2,332
TTRP_7_7	197,741	197,741	0	0.8	10	1	1,333
TTRP_7_8	192,178	192,178	0	1.8	10	1	982
TTRP_7_9	189,818	189,818	0	1.1	8	1	1,252
TTRP_7_10	188,080	188,080	0	0.7	0	1	1,222
TTRP_7_11	183,926	183,926	0	0.9	10	1	931
TTRP_7_12	163,727	163,727	0	2.2	10	1	1,284
TTRP_7_13	181,886	181,886	0	0.6	0	1	859
TTRP_7_14	187,342	187,342	0	1.4	10	1	1,288
TTRP_7_15	192,112	192,112	0	0.3	0	1	930
TTRP_7_16	193,950	193,950	0	3.2	21	1	1,123
TTRP_7_17	172,506	172,506	0	0.7	0	1	811
TTRP_7_18	191,093	191,093	0	991.1	36	3	8,059
TTRP_7_19	190,032	190,032	0	1.4	20	1	1,106
TTRP_7_20	179,299	179,299	0	2.4	20	1	1,893
TTRP_7_21	186,217	186,217	0	0.7	10	1	1,195
TTRP_7_22	202,932	202,932	0	1.0	18	1	1,124
TTRP_7_23	186,509	186,509	0	0.7	10	1	995
TTRP_7_24	191,921	191,921	0	0.3	0	1	718
TTRP_7_25	191,909	191,909	0	3.7	18	1	1,180
TTRP_7_26	200,460	200,460	0	0.5	10	1	1,071
TTRP_7_27	188,632	188,632	0	0.2	0	1	821
TTRP_7_28	173,434	173,434	0	1.7	10	1	1,721
TTRP_7_29	159,213	159,213	0	1.6	10	1	1,440

Table 4.14: Detailed results for TTRP instances size 6 and 7

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
TTRP_8_0	204,862	204,862	0	3.5	10	1	2,494
TTRP_8_1	195,854	195,854	0	23.0	51	1	1,700
TTRP_8_2	191,907	191,907	0	8.8	30	1	2,448
TTRP_8_3	203,350	203,350	0	2.3	10	1	1,555
TTRP_8_4	194,293	194,293	0	2.4	24	1	1,583
TTRP_8_5	201,960	201,960	0	223.7	50	15	6,239
TTRP_8_6	205,829	205,829	0	7.1	20	1	1,399
TTRP_8_7	204,158	204,158	0	4.2	24	1	2,055
TTRP_8_8	193,122	193,122	0	2.8	10	1	2,495
TTRP_8_9	194,797	194,797	0	1.8	0	1	1,584
TTRP_8_10	163,595	163,595	0	25.3	30	1	2,909
TTRP_8_11	201,075	201,075	0	7.5	30	1	2,451
TTRP_8_12	185,829	185,829	0	36.3	48	1	2,475
TTRP_8_13	231,992	231,992	0	1,377.0	24	2,085	2,014,384
TTRP_8_14	227,510	227,510	0	3.6	29	1	1,337
TTRP_8_15	188,023	189,025	0.53	T.L.	32	978	2,039,139
TTRP_8_16	168,893	168,893	0	32.5	20	1	2,760
TTRP_8_17	220,641	220,641	0	2.7	22	1	1,185
TTRP_8_18	186,792	186,792	0	1.4	0	1	1,775
TTRP_8_19	203,236	203,236	0	1.6	0	1	2,588
TTRP_8_20	179,462	179,462	0	3.3	20	1	1,225
TTRP_8_21	208,664	208,664	0	28.7	38	1	1,631
TTRP_8_22	228,495	228,495	0	2.3	35	1	2,245
TTRP_8_23	210,647	210,647	0	12.1	18	1	2,873
TTRP_8_24	181,086	181,086	0	2.2	9	1	1,639
TTRP_8_25	185,788	185,788	0	1.5	10	1	1,660
TTRP_8_26	166,093	166,093	0	27.2	20	1	2,378
TTRP_8_27	195,398	195,398	0	1.9	0	1	988
TTRP_8_28	204,747	204,747	0	23.6	30	1	2,155
TTRP_8_29	191,064	191,064	0	1.3	10	1	1,647
TTRP_9_0	220,184	220,184	0	11.7	14	1	3,731
TTRP_9_1	192,652	192,652	0	41.7	30	1	2,162
TTRP_9_2	220,404	220,404	0	1.6	10	1	2,350
TTRP_9_3	195,389	197,215	0.93	T.L.	68	12	5,958
TTRP_9_4	207,619	207,619	0	6.6	10	1	3,019
TTRP_9_5	207,447	208,450	0.48	T.L.	87	22	13,701
TTRP_9_6	237,694	237,694	0	5.2	30	1	1,734
TTRP_9_7	229,948	229,948	0	57.3	52	1	1,654
TTRP_9_8	212,826	212,826	0	19.8	20	1	2,151
TTRP_9_9	234,673	234,673	0	1.6	10	1	1,873
TTRP_9_10	189,741	189,741	0	9.8	28	1	2,088
TTRP_9_11	190,269	190,269	0	2.9	0	1	2,240
TTRP_9_12	198,115	198,115	0	35.0	50	1	3,200
TTRP_9_13	199,631	199,631	0	3.7	0	1	2,753
TTRP_9_14	208,587	208,587	0	16.9	20	1	2,805
TTRP_9_15	199,285	199,285	0	8.6	20	1	3,539
TTRP_9_16	212,885	212,885	0	3.1	10	1	2,398
TTRP_9_17	208,499	208,499	0	21.3	30	1	3,348
TTRP_9_18	237,788	237,788	0	0.9	9	1	1,368
TTRP_9_19	214,741	214,741	0	118.0	30	1	2,783
TTRP_9_20	217,954	217,954	0	124.3	45	1	3,593
TTRP_9_21	220,973	220,973	0	10.0	20	1	2,641
TTRP_9_22	199,360	199,360	0	239.5	34	7	14,725
TTRP_9_23	265,177	265,177	0	1.0	0	1	2,123
TTRP_9_24	198,701	198,825	0.06	T.L.	70	68	128,880
TTRP_9_25	189,096	189,096	0	11.1	20	1	3,041
TTRP_9_26	239,434	239,434	0	3.5	10	1	2,300
TTRP_9_27	236,151	236,151	0	4.4	20	1	1,577
TTRP_9_28	179,036	179,036	0	51.2	20	1	4,271
TTRP_9_29	214,870	214,870	0	153.7	42	3	3,048

Table 4.15: Detailed results for TTRP instances size 8 and 9

Instance	LB	UB	Gap	Time	#cuts	#nodes	#columns
TTRP_10_0	250,815	250,815	0	254.5	63	3	2,893
TTRP_10_1	262,139	262,139	0	36.0	30	1	3,197
TTRP_10_2	237,509	237,509	0	37.8	36	1	2,840
TTRP_10_3	244,731	244,731	0	15.6	20	1	2,907
TTRP_10_4	266,077	266,077	0	9.7	30	1	2,480
TTRP_10_5	243,870	243,870	0	4.5	10	1	2,209
TTRP_10_6	247,492	249,605	0.85	T.L.	74	81	151,803
TTRP_10_7	231,129	231,568	0.19	T.L.	48	46	114,560
TTRP_10_8	217,021	217,021	0	11.1	20	1	3,299
TTRP_10_9	248,862	248,862	0	80.4	53	3	2,662
TTRP_10_10	262,310	262,310	0	135.8	42	3	4,644
TTRP_10_11	221,083	221,645	0.25	T.L.	66	136	316,716
TTRP_10_12	246,323	246,323	0	10.1	23	1	2,595
TTRP_10_13	261,842	261,842	0	5.4	10	1	3,916
TTRP_10_14	245,973	245,973	0	5.4	18	1	2,037
TTRP_10_15	213,753	213,753	0	51.2	0	1	4,565
TTRP_10_16	207,860	207,860	0	151.8	57	1	4,395
TTRP_10_17	209,614	210,253	0.3	T.L.	64	8	13,114
TTRP_10_18	216,106	219,933	1.74	T.L.	81	12	18,806
TTRP_10_19	207,686	207,686	0	17.1	10	1	3,727
TTRP_10_20	228,717	228,717	0	17.3	30	1	2,568
TTRP_10_21	205,494	205,494	0	57.4	10	1	5,312
TTRP_10_22	224,851	230,687	2.53	T.L.	57	1	9,499
TTRP_10_23	241,509	241,509	0	4.6	0	1	2,580
TTRP_10_24	221,766	221,766	0	202.4	40	1	6,756
TTRP_10_25	233,261	233,261	0	3.0	10	1	2,195
TTRP_10_26	238,139	238,139	0	27.7	30	1	3,456
TTRP_10_27	249,677	249,677	0	19.7	30	1	2,297
TTRP_10_28	209,008	209,008	0	2,006.1	51	3	7,833
TTRP_10_29	238,367	238,367	0	66.6	34	1	4,509
TTRP_25_0	492,306	498,259	1.19	T.L.	40	1	16,571
TTRP_25_1	484,775	487,918	0.64	T.L.	50	1	12,321
TTRP_25_2	510,294	520,253	1.91	T.L.	40	1	14,279
TTRP_25_3	523,059	524,886	0.35	T.L.	90	1	9,785
TTRP_25_4	548,321	550,491	0.39	T.L.	70	1	11,622
TTRP_25_5	533,485	559,913	4.72	T.L.	70	1	13,430
TTRP_25_6	544,724	548,438	0.68	T.L.	70	1	14,840
TTRP_25_7	534,811	534,811	0	2,291.9	60	1	11,583
TTRP_25_8	511,184	511,508	0.06	T.L.	60	1	11,061
TTRP_25_9	556,671	560,022	0.6	T.L.	50	1	13,569
TTRP_25_10	532,491	546,649	2.59	T.L.	40	1	14,853
TTRP_25_11	489,122	503,341	2.82	T.L.	40	1	13,766
TTRP_25_12	543,077	548,557	1	T.L.	113	1	9,673
TTRP_25_13	526,345	544,599	3.35	T.L.	30	1	11,882
TTRP_25_14	518,128	528,855	2.03	T.L.	50	1	14,155
TTRP_25_15	546,404	550,604	0.76	T.L.	70	1	12,882
TTRP_25_16	518,744	519,920	0.23	T.L.	50	1	12,336
TTRP_25_17	478,372	496,769	3.7	T.L.	20	1	14,174
TTRP_25_18	533,444	533,471	0.01	T.L.	60	1	12,144
TTRP_25_19	506,870	523,373	3.15	T.L.	50	1	13,181
TTRP_25_20	506,401	507,249	0.17	T.L.	80	1	11,259
TTRP_25_21	482,849	490,077	1.47	T.L.	40	1	14,300
TTRP_25_22	493,621	495,046	0.29	T.L.	70	1	14,914
TTRP_25_23	575,009	578,268	0.56	T.L.	80	6	22,709
TTRP_25_24	518,835	543,377	4.52	T.L.	40	1	13,769
TTRP_25_25	538,252	542,546	0.79	T.L.	80	1	15,357
TTRP_25_26	516,290	537,281	3.91	T.L.	40	1	14,175
TTRP_25_27	528,238	536,408	1.52	T.L.	60	1	13,041
TTRP_25_28	511,719	518,385	1.29	T.L.	50	1	12,054
TTRP_25_29	538,870	541,306	0.45	T.L.	100	1	12,048

Table 4.16: Detailed results for TTRP instances size 10 and 25

Chapter 5

Branch-and-Price-and-Cut for the Periodic Vehicle Routing Problem with Flexible Schedule Structures

Ann-Kathrin Rothenbächer

Abstract

This paper addresses the periodic vehicle routing problem with time windows (PVRPTW). Therein, customers require one or several visits during a planning horizon of several periods. The possible visiting patterns (schedules) per customer are limited. In the classical PVRPTW, it is common to assume that each customer requires a specific visit frequency and offers all corresponding schedules with regular intervals between the visits. In this paper, we permit all kinds of schedule structures and the choice of the service frequency. We present an exact branch-and-price-and-cut algorithm for the classical PVRPTW and its variant with flexible schedules. The pricing problems are elementary shortest path problems with resource constraints. They can be based on one of two new types of networks and solved with a labeling algorithm, which uses several known acceleration techniques such as the *ng*-path relaxation and dynamic half-way points within bidirectional labeling. For instances whose schedule sets fulfill a certain symmetry condition, we present specialized improvements of the algorithm such as constraint aggregation and symmetry breaking. Computational tests on benchmark instances for the PVRPTW show the effectiveness of our algorithm. Furthermore, we analyze the impact of different schedule structures on run times and objective function values.

5.1 Introduction

The periodic vehicle routing problem (PVRP) extends the well-known vehicle routing problem to a planning horizon of multiple periods, which repeats continuously over time. For the sake of simplicity, we refer to a *day* instead of a period in the following. Customers need to be served one or several times during a planning horizon, following one of their offered visiting patterns. By respecting time windows of the customers during each day, the PVRP becomes the periodic vehicle routing problem with time windows (PVRPTW). The problem occurs in many real-world applications, for example, in product delivery, in waste collection and in some health-care services.

The PVRP can be described as follows. Starting from a given depot, a set of capacitated vehicles can perform round trips with a limited duration on every day. These round trips, called *routes*, serve a subset of the customers and may differ from day to day. Every customer has a daily demand and offers a set of convenient visiting patterns that are called *schedules*. A schedule contains a subset of the considered days whose combination is allowed for visiting the customer. Through the visiting days specified by a schedule, the delivery amount for every visit is also determined. For example, regarding a planning horizon of six days from Monday to Saturday, a customer may offer to be visited either on Monday and Thursday, or on Tuesday and Friday, or on Wednesday and Saturday. In every case, the demand for three days has to be delivered at each visit. The goal is to minimize the overall routing costs.

In general, the PVRP combines an assignment and a vehicle routing decision. Exactly one schedule has to be selected for every customer. If this decision was fixed, a vehicle routing problem would need to be solved for every day to determine the set of customers served on the same route and the order in which they are served.

In the PVRP literature, it is common to assume a special periodic structure of the schedules. A schedule is called *regular* if the intervals between two visits are always equal (assuming a cyclic repetition of the planning horizon). Two schedules are *overlapping* if both allow a visit on the same day. Typically, the schedules of a customer are assumed to be regular and non-overlapping. Furthermore, the number of required visits to a certain customer during the planning horizon, called the *visit frequency*, is usually specified. In addition, former publications usually presumed that every customer can be visited on every day. We call a set of schedules *standard* if all schedules are regular, pairwise non-overlapping, require the same visit frequency and if every day is contained in at least one schedule. The customers' schedule sets in existing PVRPTW benchmark instances are always standard. In this paper, we will relax these assumptions and consider additionally irregular and overlapping schedules as well as different visit frequencies per customer.

A small example of a PVRP with a planning horizon of four days $(0, 1, 2, 3)$ and its optimal solution for one available vehicle are depicted in Figure 5.1. For every customer (A, B, C, D, E), the offered schedules are indicated close to each node and the chosen schedule is underlined. Some customers offer standard schedule sets (i.e., customers A and E). In contrast, the schedule set of customer B fulfills none of the properties for standard schedule sets. The schedule $\{0, 2, 3\}$ is irregular because it induces visit

intervals of 1 and 2 days. All schedules of B are overlapping because each allows a visit on day 0. The visit frequency varies between 2 and 3 visits during the planning horizon. Moreover, none of his schedules allows a visit on day 1. The optimal solution for the visits at customer D shows that it may be beneficial to not choose the schedule with the lowest visit frequency.

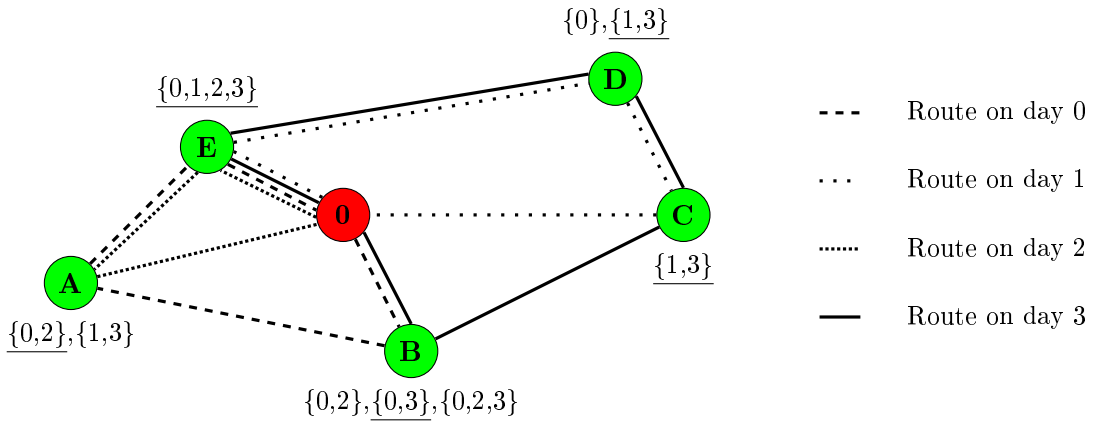


Figure 5.1: Example of a PVRP

The main contribution of the paper at hand is the presentation of the first exact algorithm for the classical PVRPTW and for variants with all possible schedule structures. We propose a branch-and-price-and-cut algorithm based on a new extended set-partitioning formulation. The pricing problem is an elementary shortest path problem with resource constraints (ESPPRC). It is solved with a bidirectional labeling algorithm, which can operate on two new types of auxiliary networks. Furthermore, we define a symmetry concept depending on certain properties of the customers' schedule sets. For instances that have this symmetry property, we present some techniques to reduce the number of pricing problems and the size of the branch-and-bound tree.

The remainder of this paper is structured as follows. Section 5.2 gives an overview of the related literature. In Section 5.3, we formally introduce the problem and present the master problem. Section 5.4 describes our exact branch-and-price-and-cut algorithm. It includes the definition of two new types of pricing networks, which are especially designed for non-standard schedule sets, and the labeling algorithm for the solution of the arising ESPPRC. Moreover, it presents our branching strategy and the incorporation of subset-row inequalities (Jepsen *et al.*, 2008). Section 5.5 explains ideas for tackling instances that fulfill a symmetry property. In Section 5.6, we present the results of our computational study. Finally, we finish with a conclusion and a short outlook in Section 5.7.

5.2 Literature Review

Research on the PVRP started with Beltrami and Bodin (1974) who tackled a problem of waste collection over several days through heuristics, treating the assignment and

the routing successively. Russell and Igo (1979) and Christofides and Beasley (1984) formalized the problem, coined the name PVRP, and presented further heuristics. Since then a lot of applications and sophisticated heuristics have been described in the literature on the classical PVRP without time windows (Hemmelmayr *et al.*, 2009; Cacchiani *et al.*, 2014), and with time windows (Cordeau *et al.*, 2004; Pirkwieser and Raidl, 2009a; Vidal *et al.*, 2013). Surveys on the PVRP can be found in (Francis *et al.*, 2008), (Campbell and Wilson, 2014), and (Irnich *et al.*, 2014b). Pirkwieser and Raidl (2009b) were the first to compute lower bounds for the PVRPTW by solving the linear relaxation of a set-covering formulation with column generation. The ESPPRC subproblems were solved with a labeling algorithm. The only exact approach for the pure PVRP so far was presented in (Baldacci *et al.*, 2011b). They used three relaxations based on a set-partitioning formulation to obtain lower bounds, generated all remaining routes with reduced costs smaller than the optimality gap, and finally solved the problem with an IP solver.

The literature on non-standard schedule structures is very sparse. Gaudioso and Paletta (1992) proposed a three-phase heuristic for a PVRP, in which schedules were allowed to be irregular and overlapping, but every customer still had a fixed visit frequency. More recently, Francis *et al.* (2006) studied selectable visit frequencies and irregular schedules in the so-called PVRP with Service Choice. With their exact approach based on Lagrangean relaxation and branch-and-bound, they showed possible savings through an increased visit frequency. However, they limited their study to non-overlapping schedules and used an estimation of customer demand per visit. The same authors published a Tabu Search heuristic, which is able to tackle the full range of possible schedules (Francis *et al.*, 2007). Moreover, they analyzed the impact of different dimensions of operational flexibility on different aspects of service consistency.

A summarizing comparison of the paper at hand and the most related papers, concerning the solution approach and the considered problem aspects, can be found in Table 5.1.

Some other problems are closely related to the PVRP. The *Multi-Period VRP* (MPVRP) deals with a planning horizon of several days during which every customer offers a time span of several consecutive days in which he has to be served once. Bostel *et al.* (2008) solved an MPVRP with multiple depots, taking into account labor rules with a metaheuristic and an exact column-generation approach. A dynamic variant of the MPVRP, in which customer demands are revealed over time, was tackled with a three-phase heuristic by Wen *et al.* (2010) and with an exact approach by Baldacci *et al.* (2011b) (who called it Tactical Planning VRP). A branch-and-price algorithm for the MPVRP with time windows was presented by Athanasopoulos and Minis (2013). They also described some techniques to reduce the number of pricing-problem solutions using the similarities of the pricing problems of the different days. Recently, Archetti *et al.* (2015) introduced a branch-and-cut algorithm for a MPVRP variant that considers inventory costs.

The *inventory routing problem* (IRP) also deals with a planning horizon of several days, but in this case the customers need to be served such that no stock-out occurs.

Table 5.1: Comparison of our paper with the related literature

	Our paper	(Pirkwieser and Raidl, 2009b)	(Baldacci *et al.*, 2011b)	Gaudioso and Paletta (1992)	Francis *et al.* (2006)	(Francis *et al.*, 2007)
Computes lower bounds	✓	✓	✓		✓	
Computes upper bounds	✓		✓	✓	✓	✓
Considers time windows	✓	✓				
Allows irregular schedules	✓			✓	✓	✓
Allows overlapping schedules	✓			✓		✓
Allows different visit frequencies	✓				✓	✓

Delivery quantities, visiting days, and visit frequencies are not predetermined but part of the decision problem. Recent surveys can be found in (Bertazzi *et al.*, 2008) and (Coelho *et al.*, 2014).

Closely related to the IRP is the *flexible PVRP*, which was recently introduced and solved by Archetti *et al.* (2017) with a load-based MIP formulation and some valid inequalities. Again, given a planning horizon of several days and a total demand for every customer for this time span, the visiting days and the delivery quantities can be chosen freely. Only the amount to deliver per customer and day is limited whereas the visiting day combinations are not limited to schedules offered by the customers.

5.3 Problem Formulation

The PVRPTW can be stated as follows. Let $\Pi = \{0, \ldots, \pi - 1\}$ be a π-day planning horizon, which repeats continuously meaning that the last day $(\pi - 1)$ is again followed by the first day 0. A set of customers N and a single depot d represent all relevant locations $O = N \cup \{d\}$. On each day, a set of m homogeneous vehicles with capacity Q and a maximal route duration D is available to perform routes starting from and ending at the depot d during its time window $[a_d, b_d]$ and visiting a subset of the customers N in between. A route is defined by the sequence of visited customers and the number of days the deliveries cover. Let R be the set of all feasible routes and $R^p \subset R$ be the subset feasible for day $p \in \Pi$. The costs c_r for a route r depend exclusively on the traveled distance. The objective is to minimize the sum of the costs of all routes that are executed.

Every customer $n \in N$ has a specific demand per day q_n, a time window $[a_n, b_n]$ and a set of offered schedules $S_n \subseteq \mathcal{P}(\Pi)$, with $\mathcal{P}$ indicating the power set. Both the customers' demands and time windows could be different for every day without affecting the appli-

cability of our solution approach (except symmetry handling, see Section 5.5). However, the classical PVRPTW assumes identical demands and time windows per customer over the planning horizon such that we omit a day index for the ease of notation. The delivery of the demand of customer n for one day p is called the *task* n^p. The task n^p does not need to be fulfilled on day p. Instead, we assume that at every visit to a customer, his tasks for all days up to the next visit have to be fulfilled (taking into account the cyclic repetition of the planning horizon). Hence, in a planning horizon of four days $(0, 1, 2, 3)$, visits to a customer A on the days 1 and 2 require the fulfillment of the day-1 task (A^1) at the first visit and the fulfillment of the tasks of the days 2, 3 and 0 (A^2, A^3, A^0) at the second visit. Assuming a daily demand of one unit for this customer, one unit has to be delivered at the first visit and three units have to be delivered at the second visit.

As mentioned in the introduction, the offered schedule sets of a customer S_n are allowed to contain any number and structure of schedules. For a given schedule, we call every induced pair of visiting day and number of fulfilled tasks a *schedule part*. For example, in a planning horizon from Monday to Saturday, the schedule {Tuesday, Friday} induces the schedule parts (Tuesday : 3) and (Friday : 3), because it represents the delivery of the demand for three days on Tuesday and Friday, respectively. The schedule part of a customer n that represents the delivery of the demand of l days at a visit on day p is denoted by $n^{p:l}$. The number of covered demand days l of a schedule part $n^{p:l}$ is also called the *length* of the schedule part and is an element of the set $L = \{1, ..., \pi\}$. Let the set $S_n^{p:l}$ identify the subset of schedules S_n offered by customer n that include the schedule-part $n^{p:l}$.

Note that for regular schedules and a fixed visit frequency per customer, the amount to deliver per visit (and thus the length of each schedule part) is fixed and independent of the chosen schedule. Obviously, this is not the case for irregular schedules as different numbers of tasks can be fulfilled by one visit. Furthermore, by allowing overlapping schedules, the demand to deliver is not determined even if a visiting day is chosen.

Route-based Formulation Let the binary variable λ_r^p indicate whether the route $r \in R^p$ is performed on day $p \in \Pi$. Moreover, let the variable z_n^s indicate whether the schedule $s \in S_n$ is chosen for customer $n \in N$. Then, a route-based extended set-partitioning model for the PVRPTW can be formulated as follows:

$$\min \quad \sum_{p \in \Pi} \sum_{r \in R^p} c_r \lambda_r^p \tag{5.1a}$$

$$\text{s.t.} \quad \sum_{s \in S_n} z_n^s = 1 \qquad \forall\, n \in N \tag{5.1b}$$

$$\sum_{r \in R^p} a_{rn}^{p:l} \lambda_r^p = \sum_{s \in S_n} b_s^{p:l} z_n^s \qquad (\rho_n^{p:l}) \qquad \forall\, n \in N, p \in \Pi, l \in L : S_n^{p:l} \neq \varnothing \tag{5.1c}$$

$$\sum_{r \in R^p} \lambda_r^p \leq m \qquad (\mu_p) \qquad \forall\, p \in \Pi \tag{5.1d}$$

$$z_n^s \in \{0, 1\} \qquad \forall\, n \in N, s \in S_n \tag{5.1e}$$

$$\lambda_r^p \in \{0,1\} \qquad \forall\, p \in \Pi, r \in R^p \qquad (5.1f)$$

with the parameter a_{rn}^{pl} stating how often the route r includes the schedule part $n^{p:l}$ and the parameter b_s^{pl} stating whether schedule s induces the schedule part $(p{:}l)$.

The objective function (5.1a) minimizes the total routing costs. The first constraints (5.1b) ensure that for each customer exactly one of its offered schedules is chosen. Constraints (5.1c) link the route and schedule variables. A schedule variable may only be selected if all its induced schedule parts are included in a realized route. Note that the equality signs in the former constraints can be replaced by a $\geq$-sign whenever removing a customer from a route does not increase the routing costs.The constraints (5.1d) limit the number of available vehicles on every day. Finally, (5.1e) and (5.1f) restrict the variables to binary values.

Note that summing up constraints (5.1c) over all possible schedule part lengths $l \in L$ leads to the equalities

$$\sum_{r \in R^p} a_{rn} \lambda_r^p = \sum_{s \in S_n} b_s^p z_n^s \qquad \forall\, n \in N, p \in \Pi \qquad (5.2)$$

where the parameter a_{rn} states how often the route r visits the customer n and the parameter b_s^p states whether the schedule s requires a visit on day p. This aggregated variant is common in the literature (see Pirkwieser and Raidl, 2009b; Baldacci *et al.*, 2011b). However, it is not suitable for the case of overlapping and irregular schedules, since visiting a customer on a certain day does not specify how much demand is delivered. Thus, every visit would be accepted as part of every schedule that includes this day without consideration of the fulfilled tasks. For example, consider $\pi = 4$ and a customer who offers the two overlapping schedules {0} and {0,2}, and a route performed on day 0 which delivers the customer's demand for two days. The constraints (5.2) allow to set the schedule variable referring to the schedule {0} to 1, but this would lead to unfulfilled demand on the last two days. Nevertheless, for the case of standard schedule sets, both constraints (5.1c) and (5.2) are identical, because for every customer and visiting day there exists exactly one schedule part.

Note further that the other intuitive idea to use covering-of-tasks constraints instead of (5.1c), i.e.

$$\sum_{p \in \Pi} \sum_{r \in R^p} a_{rn^{p'}} \lambda_r^p \geq 1 \qquad \forall\, n \in N, p' \in \Pi$$

where the parameter $a_{rn^{p'}}$ states how often the route r fulfills the task $n^{p'}$, is also not valid in general because it would enable unallowed combinations of schedule parts. For example, consider $\pi = 3$ and a customer who offers the schedules {0,1}, {1,2}, and {0,2}. Then the combination of three routes fulfilling each one of the allowed schedule parts 0:1, 1:1 and 2:1 would lead to the schedule {0,1,2}, which is not offered by the customer.

5.4 Branch-and-Price-and-Cut Algorithm

The route-based formulation (5.1) is called the master problem and is solved with a branch-and-price-and-cut algorithm. The column-generation procedure manages the solution of the LP relaxation of this master problem. Valid inequalities can be separated and added to the formulation to tighten the LP bound. Branching decisions force the variables to have integer values. In the following, we describe in Section 5.4.1 the solution of the pricing problem to generate columns, in Section 5.4.2 the incorporation of the subset-row inequalities introduced by Jepsen *et al.* (2008), and in Section 5.4.3 the branching strategy for the PVRPTW.

5.4.1 Column Generation

Because the number of feasible routes is huge, promising routes should be added dynamically. Thus, we use column generation for the route variables (see Lübbecke and Desrosiers, 2005). The restricted master problem (RMP) is defined as model (5.1) modified by including only a subset of all possible routes $R' \subseteq R$ and by relaxing the integrality conditions for all variables. Let the dual variables $\rho_n^{p:l}$ and μ_p be associated with constraints (5.1c) and (5.1d), respectively. Then, the reduced costs $\tilde{c}_r$ of a route $r \in R^p$ performed on day p are given by

$$\tilde{c}_r = c_r - \sum_{n \in N} \sum_{l \in L} a_{rn}^{p:l} \cdot \rho_n^{p:l} - \mu_p.$$

In general, pricing problems have to find columns with negative reduced costs or need to detect that no more such columns exist. Here, an ESPPRC pricing problem needs to be solved for every day (see Irnich and Desaulniers, 2005). In the following, we introduce two different ways to model the underlying pricing networks that respect the challenges resulting from irregular and overlapping schedules, and the correct treatment of the demand. Afterward, we present the dynamic-programming labeling algorithm to solve the ESPPRC.

Pricing Networks

For standard schedule sets, a route is uniquely determined by the sequence of visited customers. The reason is that for every customer the number of fulfilled tasks by each visit is identical and thus the demand to deliver is known. For this case, a pricing network consisting of one vertex for every customer and the depot would suffice. However, for irregular schedules, a route has to specify how many tasks are fulfilled by each visit, i.e., the length of the associated schedule part. Therefore, the vertices in the pricing network need to distinguish for every customer which of his tasks are fulfilled when the vertex is visited. Moreover, for non-standard schedule sets, the pricing networks differ per day. We propose two structurally different layouts for the pricing networks, namely the *schedule part network* and the *task network*.

Let $t_{o_1o_2}$ and $c_{o_1o_2}$ express the time and cost for traveling from location $o_1 \in O$ to location $o_2 \in O$, respectively (with $t_{o_1o_2}$ including the service time at location o_1).

Schedule Part Network We define the schedule part network for each day $p \in \Pi$ as the graph $G^p_{SP} = (V^p_{SP}, A^p_{SP})$ with vertices V^p_{SP} and arcs A^p_{SP}. The set of vertices V^p_{SP} consists of one vertex for every allowed schedule part of a customer with a visit on day p and two vertices d^+ and d^- modeling two copies of the depot. A schedule part $n^{p:l}$ is allowed, if at least one schedule $s \in S_n$ exists, that induces the schedule part $(p : l)$. Let $o(j) \in O$ be the location and $l(j)$ be the length of the schedule part represented by a vertex $j \in V^p_{SP} \setminus \{d^+, d^-\}$, respectively. The set of arcs A^p_{SP} contains all arcs $(i, j) \in (V^p_{SP} \setminus \{d^-\}) \times (V^p_{SP} \setminus \{d^+\})$ between two vertices of different locations, i.e., with $o(i) \neq o(j)$. Obviously, some arcs can be eliminated due to the customers' time windows.

An example of the schedule part networks of an instance with $\pi = 4$ and two customers A and B is depicted in Figure 5.2. For the sake of conciseness, we assume that the time windows require customer A to be visited before customer B. The figure shows that the networks of the different days are completely separated. It becomes clear that the number of vertices and the number of arcs strongly depends on the number and structure of offered schedules. For a customer with a standard schedule set as customer A, exactly one vertex in the network of each day is necessary. On the other hand, a customer who offers all possible schedules needs π vertices per daily network. Because all vertices of one customer need an outgoing arc to all vertices of all (time-feasible) successive customers, the number of arcs per day can reach $\mathcal{O}(|N|^2\pi^2)$.

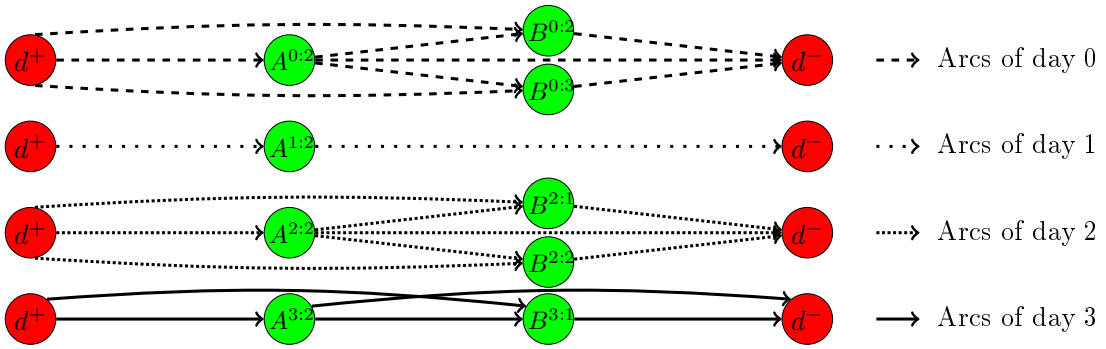


Figure 5.2: Example of schedule part networks with $S_A = \{\{0, 2\}, \{1, 3\}\}$ and $S_B = \{\{0, 2\}, \{0, 3\}, \{0, 2, 3\}\}$

The cost and time for traveling along arc (i, j) are identical to the costs and time for traveling from location $o(i)$ to location $o(j)$. The dual price for the fleet constraint (5.1d) is subtracted from the costs of all arcs leaving the start depot whereas the dual prices for constraints (5.1c) are considered on the arcs leaving the associated schedule parts. The load collected at a vertex is the product of the customer's demand per day and the length of the associated schedule part. Thus, the arc weights for the reduced costs $\tilde{c}^p_{ij}$,

the load d_{ij}, and the time t_{ij} in the graph G_{SP}^p are:

$$\begin{aligned}\tilde{c}_{ij}^p &:= \begin{cases} c_{o(i)o(j)} - \mu_p, & \text{if } i = d^+ \\ c_{o(i)o(j)} - \rho_{o(i)}^{pl(i)}, & \text{otherwise} \end{cases} \\ d_{ij} &:= q_{o(j)} \cdot l(j) \\ t_{ij} &:= t_{o(i)o(j)}.\end{aligned}$$

Task Network Another variant to model the pricing networks is based on the tasks that need to be fulfilled. We define the task network for day p as the graph $G_T^p = (V_T^p, A_T^p)$ with the vertex set V_T^p, which is a subset of the day-independent vertex set V_T, and the arc set A_T^p. The set of vertices V_T contains two depot vertices and one vertex for every customer $n \in N$ and day $p \in \Pi$, which refers to the fulfillment of the associated task n^p. The day-dependent vertex set V_T^p contains the two depot vertices d^+ and d^- and all vertices referring to tasks that can be fulfilled during a route on day p, i.e., it contains a vertex for the task $n^{p'}$ whenever the customer offers a schedule that allows a visit on day p that includes the demand fulfillment of the day p'. Let again the location of a vertex $i \in V_T$ be given by $o(i)$ and let the associated day be determined by $p(i)$. Moreover, let $\Delta(p, p')$ indicate the number of days spanned by p and p' including these two, i.e. we set

$$\Delta(p, p') := \big[(p' - p) \bmod \pi\big] + 1.$$

An arc (i, j) between vertices of different locations exists in A_T^p, whenever a schedule part of the customer $o(i)$ starts on day p and fulfills all demands up to period $p(i)$, and a schedule part of the customer $o(j)$ starts on day p. More formally, an arc (i, j) with $o(i) \neq o(j)$ exists in A_T^p, if there exists a schedule part $o(i)^{p:l}$ with $(p + l - 1) \equiv p(i)$ and $p(j) = p$. An arc (i, j) between vertices of the same customer exists in A_T^p, whenever day $p(i)$ is followed by day $p(j)$ and both days are covered by a schedule part of customer $o(i)$ starting on day p. Formally, an arc (i, j) with $o(i) = o(j)$ exists in A_T^p, if $p(i) + 1 \equiv p(j)$ and there exists a schedule part $o(i)^{p:l}$ with $\Delta(p, p(i)) \leq l$ and $\Delta(p, p(j)) \leq l$.

The four task networks for the example from above are illustrated in Figure 5.3. In contrast to the schedule part network, the task networks of the different days share the vertex set. However, every arc exists exclusively in the network of a certain day. Here, even for a customer with standard schedule set, the number of vertices in a daily network depends on the customer's visit frequency and can range from 1 to π. The main advantage of the task network is that a customer can only be entered through one single vertex in every network. Thus, when all customers offer all possible schedules, the number of arcs is in $\mathcal{O}(|N|^2\pi)$ and thus smaller than in the schedule part network. Altogether, there is no clear preference between the two pricing network types as the network sizes depend very much on the offered schedule sets. Their performance for instances with varying schedule sets is evaluated in Section 5.6.2.

The cost and time for traveling between two vertices belonging to different locations equal again the corresponding values for traveling between these locations. Accordingly, no cost and time consumption occurs on arcs between two vertices of the same location.

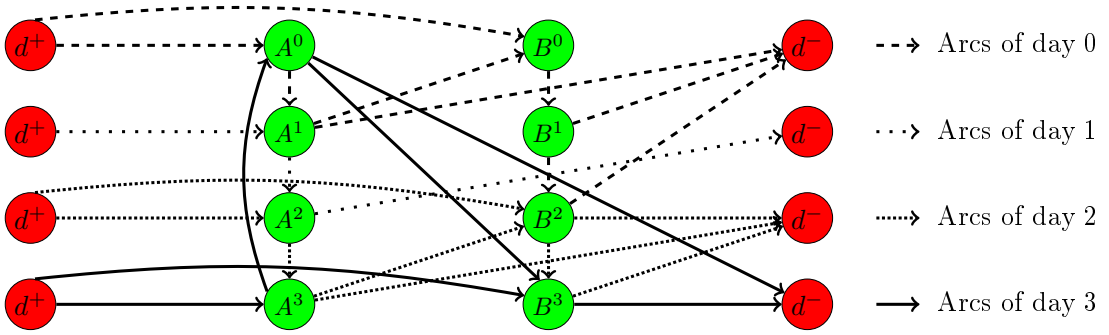


Figure 5.3: Example of task networks with $S_A = \{\{0,2\},\{1,3\}\}$ and $S_B = \{\{0,2\},\{0,3\},\{0,2,3\}\}$

All dual prices are again subtracted from the arcs leaving a location. Since every vertex represents the fulfillment of a single task, the demand per day is assigned to every vertex. Note that in this pricing network, the covered schedule part is determined only when a customer location is left because for overlapping schedules several schedule parts are possible when entering a customer location. Thus, the arc weights for the reduced costs $\tilde{c}^p_{ij}$, the load d_{ij}, and the time t_{ij} in the graph G^p_T are:

$$\tilde{c}^p_{ij} := \begin{cases} c_{o(i)o(j)} - \mu_p, & \text{if } i = d^+ \\ c_{o(i)o(j)} - \rho^{p\,\Delta(p,p(i))}_{o(i)}, & \text{if } i \neq d^+ \text{ and } o(i) \neq o(j) \\ 0, & \text{otherwise} \end{cases}$$

$$d_{ij} := q_{o(j)}$$

$$t_{ij} := \begin{cases} t_{o(i)o(j)}, & \text{if } o(i) \neq o(j) \\ 0, & \text{otherwise.} \end{cases}$$

Labeling Algorithm

The dynamic-programming labeling algorithm for our PVRPTW pricing problems of the different days can operate on both pricing network types presented above to find feasible routes with negative reduced costs. Thus, let the graph $G^p = (V^p, A^p)$ for day p be either the schedule part network $G^p_{SP} = (V^p_{SP}, A^p_{SP})$ or the task network $G^p_T = (V^p_T, A^p_T)$. The feasibility of a route is given if both the capacity and the route duration limit of the vehicle are respected, if all customers are visited within their time window, and if the visiting day and the delivery amount fit to at least one offered schedule. While the schedule feasibility is ensured by the structure of the pricing networks, the other restrictions are guaranteed by the following labeling algorithm. To obtain a bidirectional labeling algorithm, which is the most successful version for ESPPRCs (Righini and Salani, 2006), we explain the forward labeling, the backward labeling, and finally the concatenation procedure subsequently.

A partial forward path r from the start depot d^+ to a vertex $i \in V$ is represented by a label $E_i = (E_i^{cost}, E_i^{load}, E_i^{time}, E_i^{dur}, E_i^{help}, (E_i^{cust_n})_{n \in N})$, where the label components are:

E_i^{cost}: reduced cost of path r;
E_i^{load}: delivered demand on the path r;
E_i^{time}: earliest service start time at vertex i on the path r;
E_i^{dur}: minimum route duration of the path r;
E_i^{help}: negative of the latest possible departure time at the depot for the path r;
$E_i^{cust_n}$: number of times that customer $n \in N$ is visited along path r.

The resources E_i^{dur} and E_i^{help} are taken from (Tilk and Irnich, 2017) and are necessary for handling the route duration. In the initial label at vertex d^+, all components are set to 0 except $E_i^{time} = a_d$ and $E_i^{help} = -b_d$. The forward extension of a label E_i along an arc $(i,j) \in A^p$ is performed using the following resource extension functions (REFs):

$$
\begin{aligned}
E_j^{cost} &= E_i^{cost} + \tilde{c}_{ij}^p \\
E_j^{load} &= E_i^{load} + d_{ij} \\
E_j^{time} &= \max\{a_{o(j)}, E_i^{time} + t_{ij}\} \\
E_j^{dur} &= \max\{E_i^{dur} + t_{ij}, E_i^{help} + a_{o(j)}\} \\
E_j^{help} &= \max\{E_i^{dur} + t_{ij} - b_{o(j)}, E_i^{help}\} \\
E_j^{cust_n} &= \begin{cases} E_i^{cust_n} + 1, & o(j) \in N,\ o(j) = n, \text{ and } o(i) \neq o(j) \\ E_i^{cust_n}, & \text{otherwise.} \end{cases} \qquad \text{for all } n \in N
\end{aligned}
$$

A forward label E_j is feasible if and only if $E_j^{load} \leq Q$, $E_j^{time} \leq b_{o(j)}$, $E_j^{dur} \leq D$, and $E_j^{cust_n} \leq 1$ for all customers $n \in N$. Because of the non-decreasing REFs, a forward label E_j dominates another forward label E'_j belonging to the same vertex j, if $E_j \leq E'_j$ component-wise. Note that, when using the task network, it suffices to check for dominance between labels referring to the entrance vertex of each customer because the relations between the labels do not change while they are propagated at the same location.

A partial backward path r from a vertex $i \in V$ to the end depot d^- is represented by a label $\bar{E}_i = (\bar{E}_i^{cost}, \bar{E}_i^{load}, \bar{E}_i^{time}, \bar{E}_i^{dur}, \bar{E}_i^{help}, (\bar{E}_i^{cust_n})_{n \in N})$, where the three time-related resources have a slightly different meaning than in the forward labeling:

$\bar{E}_i^{time}$: latest service start time at vertex i;
$\bar{E}_i^{dur}$: difference between the maximal route duration D and the minimum route duration of the path r;
$\bar{E}_i^{help}$: difference between maximal route duration D and the earliest possible arrival time at the depot with route duration $\bar{E}_i^{dur}$;

The initial backward label at the end depot d^- is $\bar{E}_{d^-} = (0, 0, b_d, D, D - a_d, 0)$. The backward extension of a label $\bar{E}_j$ along an arc $(i,j) \in A^p$ is performed using the following

REFs:

$$
\begin{aligned}
\bar{E}_i^{cost} &= \bar{E}_j^{cost} + \tilde{c}_{ij}^{p} \\
\bar{E}_i^{load} &= \bar{E}_j^{load} + d_{ij} \\
\bar{E}_i^{time} &= \min\{b_{o(i)}, \bar{E}_j^{time} - t_{ij}\} \\
\bar{E}_i^{dur} &= \min\{\bar{E}_j^{dur} - t_{ij}, \bar{E}_j^{help} - t_{ij} + b_{o(j)}\} \\
\bar{E}_i^{help} &= \min\{\bar{E}_j^{dur} - a_{o(j)}, \bar{E}_j^{help}\} \\
\bar{E}_i^{cust_n} &= \begin{cases} \bar{E}_j^{cust_n} + 1, & o(i) \in N,\ o(i) = n, \text{ and } o(i) \neq o(j) \\ \bar{E}_j^{cust_n}, & \text{otherwise.} \end{cases} \qquad \text{for all } n \in N
\end{aligned}
$$

A backward label $\bar{E}_i$ is feasible if and only if $\bar{E}_i^{load} \leq Q$, $\bar{E}_i^{time} \geq a_{o(i)}$, $\bar{E}_i^{dur} \leq D$, and $\bar{E}_i^{cust_n} \leq 1$ for all customers $n \in N$. A backward label $\bar{E}_i$ dominates another backward label $\bar{E}_i'$ belonging to the same vertex i, if the time-related resources of the former label are greater equal and all other resources are less equal than the corresponding resources of the second label, i.e., if $\bar{E}_i^{cost} \leq \bar{E}_i'^{cost}$, $\bar{E}_i^{load} \leq \bar{E}_i'^{load}$, $\bar{E}_i^{cust_n} \leq \bar{E}_i'^{cust_n}$ $\forall n \in N$, $\bar{E}_i^{time} \geq \bar{E}_i'^{time}$, $\bar{E}_i^{dur} \geq \bar{E}_i'^{dur}$ and $\bar{E}_i^{help} \geq \bar{E}_i'^{help}$.

The concatenation of a forward partial path represented by the forward label E_j and a backward partial path represented by the backward label $\bar{E}_j$ at the vertex j is feasible, if and only if

$$
\begin{aligned}
E_j^{load} + \bar{E}_j^{load} - q_{o(j)} &\leq Q \\
E_j^{time} &\leq \bar{E}_j^{time} \\
E_j^{dur} &\leq \bar{E}_j^{dur} \\
E_j^{help} &\leq \bar{E}_j^{help} \\
E_j^{cust_n} + \bar{E}_j^{cust_n} &\leq 1 \quad \forall\ n \in N \setminus \{o(j)\}.
\end{aligned}
$$

Further Acceleration Techniques

To accelerate the solution of the pricing problem, we include several known techniques. First, we incorporate the *ng*-path relaxation to achieve more dominance while we need to accept some non-elementary routes (see Baldacci *et al.*, 2011d). The size of the *ng*-neighborhoods is dynamically increased after the solution of each branch-and-bound-node, as we always search for all cycles in the chosen routes and add the double visited vertex to the *ng*-neighborhoods of the vertices in between (Baldacci *et al.*, 2011a; Bode and Irnich, 2015). Second, we accelerate the bidirectional labeling by a dynamic choice of the half-way point as described in Chapter 2. In this approach, the half-way point is not predetermined to the middle of a resource domain but is dynamically adjusted due to a heuristic criterion that aims to balance the forward and backward workload. Third, we use restricted networks as a heuristic pricing as long as this generates columns with negative reduced costs. Fourth, we use the special pricing problem order suggested

by Pirkwieser and Raidl (2009b) to always complete the pricing problem of one day (with a certain network size) until it finds no more columns before changing to the next day. This is a variant of partial pricing (Gamache *et al.*, 1999).

5.4.2 Subset-Row Cuts

Valid inequalities are added to a linear program to cut off fractional solutions and thus strengthen the LP relaxation. For the PVRPTW, we add the non-robust subset-row (SR) inequalities introduced by Jepsen *et al.* (2008). These cuts are usually defined on a subset of customers and limit the number of routes serving a given number of these customers. However, a precondition is that every customer has to be served at most once, which is obviously not the case for the PVRPTW. We therefore need to adjust the SR cuts to our context and look at subsets of tasks or schedule parts, because both a single task and a single schedule part can be served at most once in the PVRP. The SR cut defined on a subset S of either tasks or schedule parts with $0 < k \leq |S|$ can be stated as

$$\sum_{p \in \Pi} \sum_{r \in R^p} \left\lfloor \frac{\sum_{i \in S} a_{ri}^p}{k} \right\rfloor \lambda_r^p \leq \left\lfloor \frac{|S|}{k} \right\rfloor, \tag{5.3}$$

where the parameter a_{ri}^p indicates how often route r performed on day p contains the task/the schedule part i.

Because of pretests, we consider SR cuts defined for subset of tasks in the following. As common in the literature, we restrict ourselves to SR cuts with subsets of cardinality $S = 3$ and $k = 2$, because they can be quickly separated through enumeration. However, the number of task subsets is $\mathcal{O}(\pi^3)$ times larger than the number of customer subsets. Therefore, we only separate subsets of tasks belonging to the same day, even though there can be violated SR cuts with subsets consisting of tasks of different days.

The necessary adaptations to integrate SR cuts in the labeling algorithm can be found in (Jepsen *et al.*, 2008). The only PVRP-specific point is that all task subsets need to be taken into account in all pricing networks and not just those with tasks belonging to the appropriate day.

5.4.3 Branching Strategy

To achieve integrality of the decision variables, a branching strategy needs to be implemented. Here, a hierarchical four-level based strategy is used, which tries to fix decisions in decreasing order of influence on the solution space.

On the first level, we branch on the total number of routes whenever it is fractional. On the second level, we branch on the number of routes performed on one day. When an integer number of vehicles is used on all days, we branch on a visiting day of a customer on the third level by forbidding the respective schedule columns. After no more fractional values on visiting days of individual customers can be detected, exactly one schedule is chosen for every customer. Thus, for every day remains the solution of a VRPTW with known customers and demands. Therefore, it suffices to finally branch on the decision

to visit two customers successively on a certain day. This is realized by removing the respective arcs from the pricing networks and by excluding all already generated route columns that contradict this decision. After these four branching levels there is either an integer solution or the problem is infeasible. A best-first search is executed to determine the next branch-and-bound node to process.

5.5 Symmetry

In this section, we assume that both demands and time windows of the customers do not vary from day to day, which is a common assumption in the PVRP literature. When the schedule sets of all customers are standard, the problem is symmetric in the way that the routes in every feasible solution can be shifted by any number of days without changing the objective function value or destroying the feasibility. In general, even non-standard instances may possess a (weaker) form of symmetry. For example, in a planning horizon of six days, the schedule set $S = \{\{0, 1\}, \{3, 4\}\}$ could be shifted by three days and would remain unchanged. Thus, several days may be equivalent, such that the solution of a pricing problem for every single day creates unnecessary effort. Furthermore, symmetry causes multiple solutions with the same structure and objective function value at different nodes in the branch-and-bound tree. To avoid these drawbacks, this section gives a general definition of symmetry in the PVRP context and introduces some techniques to deal with symmetric instances.

We define the shift function $f_k : \mathcal{P}(\Pi) \to \mathcal{P}(\Pi)$ with $k \in \Pi$ for a schedule $s \in S$ as:

$$f_k(s) = \{(p + k) \bmod \pi : p \in s\}.$$

A set of schedules S is called *k-shift-symmetric*, if and only if the mapping of all schedules $s \in S$ via $f_k(s)$ yields the original schedule set S again, i.e., if $S = \{f_k(s) : s \in S\}$. Accordingly, a problem instance is called k-shift-symmetric, if k is the minimal value for that all the schedule sets S_n of all customers $n \in N$ are shift-symmetric. If a problem instance is 1-shift-symmetric, we call it *completely symmetric*.

For the remainder of this section, we consider a k-shift-symmetric instance. For a given schedule $s \in S_n$, we define the set of equivalent schedules as $M_s = \{\bar{s} \in S_n : \exists m \in \mathbb{N} : f_{mk}(s) = \bar{s}\}$. The number m_s of elements in this set M_s also represents the minimal number of k-shifts such that the schedule s is mapped to itself, i.e., $m_s = \text{argmin}\{m \in \mathbb{N} : f_{mk}(s) = s\}$.

Two days $p, \bar{p} \in \Pi$ are equivalent in a k-shift-symmetric instance, written as $p \sim \bar{p}$, if there exists a multiplier $m \in \mathbb{N}_0$, such that $|\bar{p} - p| = mk$. This means, that every route performed on day p could also be performed on day $\bar{p}$ while still fitting to offered schedule parts. The equivalence class $[p]$ of a day $p \in \Pi$ is the set of all days that are equivalent to p, i.e., $[p] = \{\bar{p} \in \Pi : p \sim \bar{p}\} = \{(p + mk) \bmod \pi : m \in \mathbb{N}_0\}$. Let $U \subseteq \Pi$ be a set of class representatives.

In Section 5.5.1, we explain a valid aggregation of constraints to reduce the number of considered days and in Section 5.5.2, we describe two further techniques to exploit the

symmetry.

5.5.1 Aggregation

Even in symmetric instances, the pricing problems of two equivalent days may generate different columns and the same route may have different reduced costs when assigned to these days. The reason is primal degeneracy leading to several optimal dual solutions. However, the equivalence of days (in the sense $p \sim \bar{p}$) can be used so that only one pricing problem per equivalence class needs to be considered. The idea is to aggregate all day-specific constraints in the linear relaxation of the PVRP model (5.1) over all days p in the same equivalence class $[p]$. The same aggregation approach was successfully applied in Chapter 4 for the truck-and-trailer routing problem with a two-day planning horizon. In the following, we first present the general aggregation for a k-shift-symmetric instance and then a further simplification for completely symmetric instances with regular schedules.

General Aggregation

Aggregating (summing up) all day-equivalent constraints in the linear relaxation of the PVRP model (5.1) leads to the following *aggregated model*:

$$\min \quad \sum_{p \in U} \sum_{r \in R^p} c_r \lambda_r^p \tag{5.4a}$$

$$\text{s.t.} \quad \sum_{s \in S_n} z_n^s = 1 \qquad \forall\, n \in N \tag{5.4b}$$

$$\sum_{r \in R^p} a_{rn}^{p:l} \lambda_r^p = \sum_{s \in S_n} \sum_{\bar{p} \in [p]} b_s^{\bar{p}:l} z_n^s \qquad \forall\, n \in N, p \in U, l \in L : S_n^{pl} \neq \varnothing \tag{5.4c}$$

$$\sum_{r \in R^p} \lambda_r^p \leq |[p]| m \qquad \forall\, p \in U \tag{5.4d}$$

$$0 \leq z_n^s \leq 1 \qquad \forall\, n \in N, s \in S_n \tag{5.4e}$$

$$0 \leq \lambda_r^p \leq |[p]| \qquad \forall\, p \in U, r \in R^p \tag{5.4f}$$

It suffices to include the route variables for the class representatives, because two identical routes $r \in R^p$ and $\bar{r} \in R^{\bar{p}}$ performed on equivalent days $p \sim \bar{p}$ have the same coefficients, i.e., $a_{rn}^{p:l} = a_{\bar{r}n}^{\bar{p}:l}$ for all $n \in N, l \in L$, and identical costs $c_r = c_{\bar{r}}$. The advantages of the aggregated model are a smaller problem size regarding the number of variables and constraints and a reduced number of pricing problems that have to be solved.

Proposition 5.1. The linear relaxations of the PVRP model (5.1) and the aggregated model (5.4) are equivalent for k-shift symmetric instances in the sense that

(i) they provide identical lower bounds and

(ii) there exists a constructive procedure to transform a solution for one of them to a solution for the other one.

Proof. The original model and the aggregated model are equivalent for k-shift-symmetric instances, because from a solution for one of them, a solution for the other with the same

objective function value can be generated. The direction from the original model to the aggregated model is clear, because aggregation of constraints is always a relaxation. The other direction can be shown by stating a feasible transformation. For any solution $\bar{\lambda}_r^p$ with $p \in U, r \in R^p$ and $\bar{z}_n^s$ with $n \in N, s \in S_n$ of (5.4), a feasible solution of the linear relaxation of (5.1) with the same objective function value is given by setting

$$\hat{\lambda}_r^{\bar{p}} = \frac{1}{|[p]|}\bar{\lambda}_r^p \quad \forall p \in U, r \in R^p, \bar{p} \in [p] \qquad \text{and} \qquad \hat{z}_n^s = \frac{1}{m_s}\sum_{\bar{s} \in M_s} \bar{z}_n^s \quad \forall n \in N, s \in S_n.$$

The proof of the feasibility and the unchanged objective function value can be performed by recalculation. □

Example Consider an instance with a planning horizon of four days and two customers A and B with schedule sets $S_A = \{s_A^1 = \{0,1\}, s_A^2 = \{1,2\}, s_A^3 = \{2,3\}, s_A^4 = \{3,0\}\}$ and $S_B = \{s_B^1 = \{0,2\}\}$. Although the schedules of customer A are irregular, S_A is 1-shift-symmetric, while S_B is only 2-shift-symmetric even though customer B offers a regular schedule. Consequently, the whole instance is only 2-shift-symmetric. Thus, the two days 0 and 1 constitute a set of representatives and it suffices to consider these two days in the algorithm until the symmetry is broken. An optimal solution of the aggregated model (5.4) is given by the following variable assignments:

$$\begin{aligned} &\bar{\lambda}_{r_1}^0 = 1 \text{ with } r_1 = (d^+, A^{l=1}, B^{l=2}, d^-), && \bar{z}_A^1 = 1, \quad \bar{z}_A^2 = \bar{z}_A^3 = \bar{z}_A^4 = 0, \\ &\bar{\lambda}_{r_2}^0 = 1 \text{ with } r_2 = (d^+, B^{l=2}, d^-), && \bar{z}_B^1 = 1, \\ &\bar{\lambda}_{r_3}^1 = 1 \text{ with } r_3 = (d^+, A^{l=3}, d^-). \end{aligned}$$

Note that setting $\bar{z}_A^1 = 1$ is not the only possibility concerning the schedule variables of customer A in the aggregated model. All schedule variables with schedules in the same set M_s are treated equally, such that, because of $M_{s_A^1} = \{s_A^1, s_A^3\}$, every variable assignment fulfilling $\bar{z}_A^1 + \bar{z}_A^3 = 1, \bar{z}_A^2 = \bar{z}_A^4 = 0$ is feasible for the given routes. An equivalent solution for the original, disaggregated model (5.1) is given by considering all four days and redistributing the variable values as described above:

$$\begin{aligned} &\hat{\lambda}_{r_1}^0 = \hat{\lambda}_{r_1}^2 = 0.5, && \hat{z}_A^1 = \hat{z}_A^3 = 0.5, \quad \hat{z}_A^2 = \hat{z}_A^4 = 0, \\ &\hat{\lambda}_{r_2}^0 = \hat{\lambda}_{r_2}^2 = 0.5, && \hat{z}_B^1 = 1, \\ &\hat{\lambda}_{r_3}^1 = \hat{\lambda}_{r_3}^3 = 0.5. \end{aligned}$$

The SR cuts can also be used in an aggregated version. Recall that we restrict ourselves to SR cuts based on subsets of tasks referring to the same day. The addition of a single SR cut for a subset S^p of tasks belonging to one certain day p would destroy the symmetry of the problem. Thus, the corresponding SR cuts with subsets $S^{\bar{p}}$ of tasks referring to the same customers, but to the other days $\bar{p} \in [p] \setminus p$ in the equivalence class, have to be generated too. Then, the associated cuts of the same equivalence class have to be

aggregated as above to

$$\sum_{p\in U}\sum_{r\in R^p}\sum_{\bar{p}\in[p]}\left\lfloor\frac{\sum_{i\in S^{\bar{p}}}a^p_{ri}}{k}\right\rfloor\lambda^p_r \le |[p]|\left\lfloor\frac{|S^p|}{k}\right\rfloor.$$

As soon as the symmetry of the problem is broken, for example due to day-specific branching decisions, the constraints have to be disaggregated. Thus, formulation (5.1) has to be used again. To obtain the same objective function value, route variables referring to the days that were ignored, have to be added. More precisely, the route r of a variable λ^p_r with positive value in the solution of the aggregated model, has to be shifted to all other days in the equivalence class of the day p. The resulting route variables $\lambda^{\bar{p}}_r$ with $\bar{p} \in [p] \setminus p$ have to be added to the formulation.

Aggregation for Completely Symmetric Instances with Regular Schedules

In the special case of completely symmetric instances with only regular schedules, the linking constraints (5.4c) can be further aggregated. Because of the complete symmetry, there is only one equivalence class, i.e., $U = \{0\}$. By multiplying both sides of constraints (5.4c) with the schedule part length and aggregating over the lengths, the following constraints result:

$$\sum_{r\in R^0}\sum_{l\in L} l\, a^{0:l}_{rn}\lambda^0_r = \sum_{s\in S_n}\underbrace{\sum_{l\in L} l \sum_{\bar{p}\in[0]} b^{\bar{p}:l}_s}_{\text{regularity of schedules}\;=\;\pi} \quad z^s_n \overset{(5.1b)}{=} \pi \qquad \forall\, n\in N \tag{5.5}$$

because, for a fixed schedule, the unique schedule part length times the number of schedule parts in this schedule gives the total number of days. In consequence, route variables and schedule variables become independent. Since the schedule variables have no influence on the objective function, they can be ignored in the aggregated model for completely symmetric instances with only regular schedules. Thus, constraints (5.4b) and (5.4e) can be discarded in this case. All in all, the resulting *aggregated model for completely symmetric instances with only regular schedules* is given by:

$$\begin{aligned} \min \quad & \sum_{r\in R^0} c_r\lambda^0_r \\ \text{s.t.} \quad & (5.5), (5.4\text{d}) \text{ and } (5.4\text{f}) \end{aligned} \tag{5.6}$$

Proposition 5.2. The linear relaxations of the PVRP model (5.1) and the aggregated model (5.6) are equivalent for completely symmetric instances with only regular schedules in the sense that

(i) they provide identical lower bounds and

(ii) there exists a constructive procedure to transform a solution for one of them to a solution for the other one.

Proof. Again, the feasibility of this expanded aggregation can be proven by showing the

necessary transformation to obtain a solution of the original model (5.1). For any solution $\bar{\lambda}_r^0$ with $r \in R^0$ of (5.6) of a completely symmetric instance with only regular schedules, a feasible solution of the linear relaxation of (5.1) with the same objective function value is given by setting

$$\hat{\lambda}_r^{\bar{p}} = \frac{1}{\pi}\bar{\lambda}_r^0 \quad \forall r \in R^0, \bar{p} \in [0] \qquad \text{and} \qquad \hat{z}_n^s = \frac{1}{\pi}\sum_{r \in R^0} a_{rn}^{0:l_s}\bar{\lambda}_r^0 \quad \forall n \in N, s \in S_n,$$

where l_s is the unique schedule part length of all schedule parts of a regular schedule. Again, the validity can be seen by recalculation. □

Example Consider an instance with a planning horizon of four days and one customer A with the schedule set $S_A = \{s_A^1 = \{0,1,2,3\}, s_A^2 = \{0,2\}, s_A^3 = \{1,3\}\}$. Despite different visit frequencies of the offered schedules, all schedules are regular and the instance is completely symmetric. An optimal solution of the aggregated model for completely symmetric instances with only regular schedules (5.6) is given by the following variable assignments:

$$\bar{\lambda}_{r_1}^0 = 2 \text{ with } r_1 = (d^+, A^{l=1}, d^-),$$
$$\bar{\lambda}_{r_2}^0 = 1 \text{ with } r_2 = (d^+, A^{l=2}, d^-).$$

Thereby, route r_1 fulfills one task of customer A and is performed two times and route r_2 fulfills the remaining two tasks. An equivalent solution for the original model (5.1) is given by considering all four days, redistributing the route variable values equally to all days, and including the schedule columns as described above:

$$\hat{\lambda}_{r_1}^0 = \hat{\lambda}_{r_1}^1 = \hat{\lambda}_{r_1}^2 = \hat{\lambda}_{r_1}^3 = 0.5, \qquad \hat{z}_A^1 = 0.5, \quad \hat{z}_A^2 = \hat{z}_A^3 = 0.25,$$
$$\hat{\lambda}_{r_2}^0 = \hat{\lambda}_{r_2}^1 = \hat{\lambda}_{r_2}^2 = \hat{\lambda}_{r_2}^3 = 0.25.$$

When the symmetry is broken, all schedule variables have to be included in the model and certain route variables have to be added as described for the general aggregation above.

5.5.2 Further Techniques Handling Symmetry

For completely symmetric instances, the model in its original formulation (5.1) offers more degrees of freedom than necessary. As mentioned, every solution can be shifted by any number of days to which the instance is symmetric. Thus, an optimal solution remains reachable if for one single customer the decision to visit him on one certain day is fixed. Therefore, we choose a customer with the highest number of possible schedules and among them one with the highest demand, and fix him to be visited on a certain day that is allowed by at least one of his offered schedules. This partially breaks the symmetry of the problem and can significantly reduce the size of the branch-and-bound

tree.

Furthermore, for k-shift-symmetric instances, as long as no day-specific branching is applied (i.e., including the first branching level), all generated routes that do not contain the fixed customer are valid for all days in the same equivalence class. Hence, we shift every route generated in the pricing problem of a specific day to all other days in the same equivalence class and compute the reduced costs of the corresponding column. If its reduced costs are also negative, we directly add it to the RMP. This can reduce the number of necessary pricing iterations.

5.6 Computational Study

The branch-and-price-and-cut algorithm was implemented in C++ and uses CPLEX 12.6.0. All computations were performed in single thread mode on a PC with an Intel Core i7-4790K CPU clocked at 3.60 GHz, with 8 GB RAM, running Windows 7 Enterprise. Apart from the single-thread mode, default settings were used for the CPLEX solver. Preliminary tests revealed that the best results can be obtained with a size of 12 for the ng-neighborhoods which may be increased dynamically up to 18, and reduced pricing networks with 5, 10, 20, and 40 neighbors per vertex. The overall number of SR cuts was limited to 100, whereby at most four SR cuts per day were allowed to be added per pricing iteration.

We start with a description of the considered PVRPTW instances in Section 5.6.1. In Section 5.6.2, the success of the proposed PVRP-specific settings is evaluated and the two introduced pricing network types are compared. Section 5.6.3 shows the effects of different schedule structures and presents the results for PVRPTW benchmark instances.

5.6.1 Instances

The basis for the computational results are the 20 PVRPTW benchmark instances published by Cordeau *et al.* (2001), called *PR* in the following. Herein, the number of customers ranges from 48 to 288 customers and planning horizons of four and six days are considered. The time windows in the first 10 instances are narrow and larger in the remaining 10 instances. All customers offer standard schedule sets, such that the instances are completely symmetric. For the instances with four days, there exist three groups of customers requiring the different possible visit frequencies 1, 2, and 4. For the instances with six days, there exist four groups of customers requiring the different possible visit frequencies 1, 2, 3, and 6. Up to now, these instances were not tackled with exact approaches. Concluding from the successes on VRPTW instances with 100 customers and more, most of the instances in *PR* are too large to be solved to optimality in reasonable times (see Desaulniers *et al.*, 2014).

Thus, we created 20 smaller instances by selecting a subset of 36 customers for the instances with a planning horizon of four days and a subset of 24 customers for the instances with a planning horizon of six days. We took the same number of customers of each of the aforementioned groups with different visit frequencies to keep the ratios

between these groups unchanged. Moreover, we limited the number of vehicles in these smaller instances to three. We refer to these smaller instances as PR_s.

To test the effects of different schedule structures, we created two more variants of the instance set PR_s by modifying the offered schedule sets. In the instance set PR_s^+, we added for each customer all regular schedules with a higher visit frequency than originally required. The intention is the assumption that customers need a certain minimal visit frequency but are willing to be visited more often. Indeed, even a bonus for more frequent visits seems realistic and was presumed for example by Francis *et al.* (2006).

Furthermore, in the instance set PR_s^{++}, we extended PR_s by adding for each customer all irregular schedules with the same number of visits as originally required. We assume that customers do not require exactly the same intervals between each two visits, but need only a certain number of visits during the planning horizon. Obviously, both modified instance sets are harder because of the increased number of allowed schedules. On the other hand, the solution values can only be decreased, since all solutions that are feasible for the original instances in PR_s stay feasible for the appropriate modified instances. Furthermore, to the best of our knowledge, these two sets could not be tackled by exact approaches from the literature because of the overlapping (and in the second case irregular) schedules.

The most important properties of the regarded instance sets are summarized in Table 5.2.

Table 5.2: Properties of the considered PVRPTW instance sets

Instance set	#Days	#Customers	Avg. $\lvert S_n \rvert$	Irregular schedules	Overlapping schedules	Varying visit frequency
PR	4	48–288	2.75			
	6	72–288	3.00			
PR_s	4	36	2.75			
	6	24	3.00			
PR_s^+	4	36	4.50		x	x
	6	24	5.50		x	x
PR_s^{++}	4	36	3.75	x	x	
	6	24	10.50	x	x	

5.6.2 PVRP Specific Settings Evaluation

In this paper, several options to tackle the symmetry and to model the pricing networks were discussed that are specific to PVRPs. The effectiveness of these settings is measured through extensive tests that are presented in the following. A run time limit of one hour was used for all tests in this section.

Table 5.3 shows the results of all combinations of the following settings: (i) aggregation of constraints (Section 5.5.1), (ii) symmetry breaking through fixation of one

customer (Section 5.5.2), (iii) addition of columns shifted to other days (Section 5.5.2), and (iv) the pricing problem order that continues with one day until no more columns for this day are found (Section 5.4.1). Each entry shows the average run time and the number of solved instances from 80 test runs, consisting of the 40 instances of the sets PR_s and PR_s^+, tested with both the task network and the schedule part network. Note that we did not include the instance set PR_s^{++} in this test, because we wanted to apply a consistent form of aggregation for the comparison (and the aggregated model (5.6) is not applicable for PR_s^{++} because of the irregular schedules).

Table 5.3: Comparison of run times for different solving strategies

		Symmetry breaking					
		true	true	false	false		
Aggregation	true	976.4 (68)	1027.7 (65)	1370.1 (60)	1417.6 (59)	true	Add shifted columns
	true	982.7 (67)	1045.7 (66)	1371.4 (60)	1446.4 (58)	false	
	false	1035.7 (67)	1116.6 (66)	1652.5 (55)	1668.1 (55)	true	
	false	1051.1 (67)	1058.9 (66)	1704.5 (51)	1605.1 (53)	false	
		true	false	true	false		
		Special pricing problem order					

The values show the average run times over forty test instances solved with both the task network and the schedule part network with the PVRP specific settings indicated at the borders. Instances that were not solved to optimality in the time limit of 1 hour were counted with the time limit. The number in brackets gives the number of instances solved to optimality.

The best results regarding both the run times and the number of solved instances were obtained with the setting using all the options listed above. Switching off all these options leads to an increase of the average run time by more than 60 %. Obviously, the symmetry breaking has the biggest influence, but also the use of the aggregated model leads to an improvement independent from the other settings. The addition of shifted columns is especially beneficial when the special pricing problem order is used. All in all, we concluded to use all the options (i) to (iv) in all following tests.

The next test evaluates the performance of the two different pricing networks introduced in Section 5.4.1. All small instance sets were run with the schedule part network and the task network. The results of this comparison can be found in Table 5.4. Average run times were calculated only over the instances solved with both network types.

Overall, the algorithm performed better using the task network in the pricing problem. When one extreme outlier of the tests with the schedule part network is ignored (which is due to an unfavorable branching process), the run times were similar for the small instance set with the standard schedule sets. However, with an increasing number of offered schedules (in instance sets PR_s^+ and PR_s^{++}), the difference between the applications of the two pricing networks becomes larger. In the instance set with the highest average number of offered schedules per customer, PR_s^{++}, the algorithm with the task network is more than twice as fast as the algorithm with the schedule part network. Therefore,

Table 5.4: Task network versus schedule part network on different instance sets

		SP network		Task network	
Instance set	TW	#Opt	Avg. time[s]	#Opt	Avg. time[s]
PR_s	small	10 / 10	25.6	10/ 10	26.0
	large	9 / 10	1059.5*	9/ 10	732.4
PR_s^+	small	10 / 10	377.7	10/ 10	292.6
	large	4 / 10	2057.2	6/ 10	1505.1
PR_s^{++}	small	6 / 10	821.3	6/ 10	382.7
	large	4 / 10	302.7	4/ 10	121.9
Total		43 / 60	774.0	45 / 60	510.1

*Ignoring one outlier instance: 776.9s
(this instance needed nearly 8 times as many B&B nodes with the SP network as with the task network)

we decided to use the task network for all further computations.

5.6.3 Results

This section presents the computational results for the small instance sets and the original instances. Table 5.5 shows the aggregated results for the small instances. The columns state the name of the instance set, the number of instances solved to optimality during one hour, the average run time (unsolved instances are counted with the time limit), and the average optimality gap over all instances. For the instance sets $PR_s^+$and PR_s^{++}, the table gives additionally the information about the improvement compared to the basis instance set PR_s. We present the number of instances with improved objective value and for these instances the average percental improvement. In cases where instances were not solved to optimality, we compared the lower bound of an instance in PR_s with the best found upper bound of the corresponding instance in $PR_s^+$and PR_s^{++}, respectively.

Table 5.5: Comparison of the results for different schedule structures

instance set	#Opt	Time[s]	Gap[%]	#Improved solutions	Avg. improvement[%]
PR_s	19 / 20	522.6	0.03		
PR_s^+	16 / 20	1317.8	0.52	4	1.2
PR_s^{++}	10 / 20	1939.2	2.23	18	1.4

As expected, increasing flexibility of the customers' schedule sets leads to higher run

times and less instances solved to optimality. On the other hand, several instances produced better solution values when more flexible schedule sets were allowed. Even in the instance set PR_s^+, where only schedules with higher visit frequencies were added, four improved solutions could be detected. In the instance set PR_s^{++}, where the visit frequency stayed unchanged, for nearly all instances a better solutions could be found. This is especially surprising because our algorithm does not focus on the search for good upper bounds and half of the instances were not solved to optimality. This could also be a reason for the rather small percental improvements. Detailed results for the individual instances can be found in Table 5.7 in the Appendix.

In Table 5.6, we present new lower bounds and two optimal solutions for the benchmark set *PR*. Beside the instance characteristics, name, number of customers, number of days, number of vehicles, and the size of the time windows, we list the root lower bounds computed by Pirkwieser and Raidl (2009b) (if available), the lower bounds computed with our algorithm after two hours of runtime, the best known upper bounds that were published by Vidal *et al.* (2013), and the optimality gap between these upper and our lower bounds.

Table 5.6: Results for PVRPTW benchmark instances *PR*

Instance	#Cust.	#Days	#Veh.	TW	LB[PR09]	Our LB	UB[V13]	Gap[%]
pr01	48	4	3	small	2882.01	2909.02	2909.02	0
pr02	96	4	6	small	4993.48	5010.85	5026.57	0.3
pr03	144	4	9	small	6841.44	6853.10	7023.90	2.5
pr04	192	4	12	small		7536.01	7755.77	2.9
pr05	240	4	15	small		7991.78	8311.17	4.0
pr06	288	4	18	small			10473.24	
pr07	72	6	5	small	6641.39	6724.24	6783.23	0.9
pr08	144	6	10	small	9153.79	9159.92	9574.80	4.5
pr09	216	6	15	small		12439.90	13201.06	6.1
pr10	288	6	20	small		15893.90	16920.96	6.5
pr11	48	4	3	large	2258.85	2277.44	2277.44	0
pr12	96	4	6	large		4034.77	4121.50	2.1
pr13	144	4	9	large		5335.27	5489.33	2.9
pr14	192	4	12	large			6347.77	
pr15	240	4	15	large			6777.54	
pr16	288	4	18	large			8582.72	
pr17	72	6	4	large		5323.57	5481.61	3.0
pr18	144	6	8	large			7599.01	
pr19	216	6	12	large			10532.51	
pr20	288	6	16	large			13406.89	

PR09 = Pirkwieser and Raidl (2009b), V13 = Vidal *et al.* (2013)

For seven out of these large instances, the first column-generation-based lower bounds could be computed. The optimality of two known solutions was proven for the first time. The small gaps together with the two optimality proofs indicate that both the heuristic of Vidal *et al.* (2013) and our lower bounds are satisfactory. However, it becomes apparent that larger time windows complicate the solution a lot because the number of instances whose root node could be solved decreased to less than 50% by enlarging the time windows.

5.7 Conclusions

In this paper, the classical PVRPTW and its extension to fully flexible schedule sets was studied. We presented a slightly modified master problem and a new exact branch-and-price-and-cut algorithm. For the pricing problem, two new underlying networks were described on which a bidirectional labeling operates to find new columns. Moreover, we presented efficient methods to deal with symmetric instances using constraint aggregation and a symmetry breaking variable fixing.

An extensive computational study was performed with PVRPTW benchmark instances from the literature and smaller instances with varying schedule structures. The effectiveness of the proposed PVRP-specific methods could be confirmed and benefits of the task-based network for increasing schedule sets per customer were shown. While raising flexibility through more offered schedules per customer increases the difficulty and thus the solution times, we showed improvements in terms of costs in many instances. For the large benchmark instances, several lower bounds that are close to known upper bounds were found and the optimality of two solutions was proven.

Future research could address the branching and, for symmetric instances, the transition from the aggregated model to the original model. During preliminary tests, we observed that slightly different tie breakers in the branching decisions can have a huge influence on the run times. Therefore, a detailed study of branching decisions and the usage of strong branching could lead to significant improvements. Concerning the transition from the aggregated to the original model, the current approach often produces a very fractional solution of the original model, even when the solution of the aggregated model was integer. Finally, a good heuristic could help to reduce the effort for finding an integer solution after the transition.

Acknowledgement

The PhD supervisor and his research group at JGU Mainz work on column-generation stabilization methods which have strongly influenced the symmetry considerations in this paper. This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/6-1. The support is gratefully acknowledged.

Bibliography

Archetti, C., Jabali, O., and Speranza, M. G. (2015). Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, **61**, 122–134.

Archetti, C., Fernández, E., and Huerta-Muñoz, D. L. (2017). The flexible periodic vehicle routing problem. *Computers & Operations Research*, **85**, 58–70.

Athanasopoulos, T. and Minis, I. (2013). Efficient techniques for the multi-period vehicle routing problem with time windows within a branch and price framework. *Annals of Operations Research*, **206**(1), 1–22.

Baldacci, R., Bartolini, E., Mingozzi, A., and Valletta, A. (2011a). An exact algorithm for the period routing problem. *Operations Research*, **59**(1), 228–241.

Baldacci, R, Mingozzi, A., and Roberti, R. (2011). Dynamic ng-path relaxation. *https://www.uv.es/route2011/Roberti.pdf*.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011c). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

Beltrami, E. J. and Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, **4**(1), 65–94.

Bertazzi, L., Savelsbergh, M., and Speranza, M. G. (2008). Inventory routing. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 49–72. Springer US, Boston, MA.

Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, **49**(2), 369–383.

Bostel, N., Dejax, P., Guez, P., and Tricoire, F. (2008). Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 503–525. Springer US, Boston, MA.

Cacchiani, V., Hemmelmayr, V., and Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, **163**, 53–64. Matheuristics 2010.

Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, **63**(1), 2–15.

Christofides, N. and Beasley, J. E. (1984). The period routing problem. *Networks*, **14**(2), 237–256.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, **48**(1), 1–19.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *The Journal of the Operational Research Society*, **52**(8), 928–936.

Cordeau, J. F., Laporte, G., and Mercier, A. (2004). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *The Journal of the Operational Research Society*, **55**(5), 542–546.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). The vehicle routing problem with time windows. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Francis, P., Smilowitz, K., and Tzur, M. (2006). The period vehicle routing problem with service choice. *Transportation Science*, **40**(4), 439–454.

Francis, P., Smilowitz, K., and Tzur, M. (2007). Flexibility and complexity in periodic distribution problems. *Naval Research Logistics (NRL)*, **54**(2), 136–150.

Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer US, Boston, MA.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large scale aircrew rostering problems. *Operations Research*, **47**, 247–262.

Gaudioso, M. and Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, **26**(2), 86–92.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, **195**(3), 791–802.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York.

Irnich, S., Schneider, M., and Vigo, D. (2014b). Four variants of the vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing*, chapter 9, pages 241–271. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Pirkwieser, S. and Raidl, G. R. (2009a). Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ILP techniques. In S. Voss and M. Caserta, editors, *Proceedings of the 8th Metaheuristic International Conference*.

Pirkwieser, S. and Raidl, G. R. (2009b). A column generation approach for the periodic vehicle routing problem with time windows. In M. Scutellà et al, editor, *Proceedings of the International Network Optimization Conferencee*.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, **9**(1), 1–17.

Tilk, C. and Irnich, S. (2017). Dynamic programming for the minimum tour duration problem. *Transportation Science*, **51**(2), 549–565.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, **40**(1), 475–489.

Wen, M., Cordeau, J.-F., Laporte, G., and Larsen, J. (2010). The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, **37**(9), 1615–1623.

Appendix

Table 5.7 lists the lower and upper bounds as well as the solution times for the small instances PR_s, PR_s^+, and PR_s^{++} for a run time limit of one hour. Upper bounds for instances in PR_s^+ and PR_s^{++} that were smaller than the corresponding solution in PR_s are indicated in bold.

Table 5.7: Detailed results for different schedule structures

	PR_s			PR_s^+			PR_s^{++}		
Instance	LB	UB	Time[s]	LB	UB	Time[s]	LB	UB	Time[s]
$pr01_s$	2510.27	2510.27	3.9	2510.27	2510.27	29.0	2506.60	**2509.06**	TL
$pr02_s$	3324.68	3324.68	43.8	3324.68	3324.68	573.5	3306.73	**3306.73**	241.3
$pr03_s$	2802.02	2802.02	6.3	2802.02	2802.02	35.5	2797.23	**2797.23**	313.6
$pr04_s$	3012.19	3012.19	25.0	3012.19	3012.19	270,0	2996.28	**2996.28**	793.3
$pr05_s$	2387.90	2387.90	35.6	2387.90	2387.90	559.7	2353.54	**2353.54**	9.6
$pr06_s$	2891.15	2891.15	3.9	2891.15	2891.15	23.8	2881.00	**2881.00**	936.6
$pr07_s$	4268.41	4268.41	0.6	4251.06	**4251.06**	1.8	4239.22	**4239.22**	1.8
$pr08_s$	3065.62	3065.62	32.3	3056.07	**3056.07**	275.6	3001.15	**3007.07**	TL
$pr09_s$	3227.32	3227.32	85.7	3227.32	3227.32	956.4	3132.44	**3134.56**	TL
$pr10_s$	3775.87	3775.87	22.9	3775.87	3775.87	200.6	3665.55	**3672.33**	TL
$pr11_s$	2042.37	2042.37	38.9	2042.37	2042.37	642.9	2034.53	**2034.53**	269.5
$pr12_s$	2716.89	2733.35	TL	2705.63	2766.34	TL	2703.27	3492.08	TL
$pr13_s$	2419.54	2419.54	100.0	2419.54	2419.54	1730.6	2400.41	**2400.41**	54.5
$pr14_s$	2404.58	2404.58	110.2	2404.58	2404.58	2450.8	2394.43	**2396.22**	TL
$pr15_s$	1812.11	1812.11	18.5	1812.11	1812.11	565.3	1803.34	**1808.62**	TL
$pr16_s$	2474.41	2474.41	3534.0	2466.35	2483.04	TL	2454.40	**2458.48**	TL
$pr17_s$	3774.49	3774.49	37.2	3766.64	**3766.64**	698.9	3724.02	**3724.02**	79.5
$pr18_s$	2707.06	2707.06	562.7	2646.14	2713.36	TL	2634.74	3011.30	TL
$pr19_s$	2955.68	2955.68	840.7	2842.27	**2842.27**	2942.2	2782.91	**2784.24**	TL
$pr20_s$	3198.93	3198.93	1349.8	3081.18	3233.57	TL	3069.71	**3069.71**	84.0

Chapter 6

Branch-and-Price-and-Cut for a Service Network Design and Hub Location Problem

Ann-Kathrin Rothenbächer, Michael Drexl, Stefan Irnich

Abstract

In the context of combined road-rail freight transport, we study the integrated tactical planning of hub locations and the design of a frequency service network. We consider a number of real-world constraints such as multiple transshipments of requests at hubs, transport time limits for requests, request splitting, and outsourcing possibilities. We present a path-based model and solve it with a branch-and-price-and-cut algorithm. Computational experiments show that large realistic instances from a major German rail freight company can be solved close to optimality within one hour on a standard desktop computer, allowing our algorithm to be used for practical planning purposes.

6.1 Introduction

In this paper, we consider a real-world integrated tactical service network design and hub location problem (SNDHLP) for combined transport by road and rail. Combined transport is a special kind of intermodal transport. The latter is defined by the Economic Commission for Europe (2001) as the movement of freight in one and the same loading unit, such as a container or swap body, which uses successively two or more modes of transport without handling the freight itself in changing modes. Combined road-rail transport is characterized by the additional condition that the main part of the transport is by rail, and only the initial and the final leg are performed by road, see Figure 6.1. Intermodal and combined transport are economically attractive because of the consolidation effects they offer and the resulting potential for cost savings and efficiency gains. Combined transport traffic has more than doubled in the European Union since 1990. In 2011, about 6.4 million containers were transported with a freight performance of 44.7 billion tonne-kilometres. Combined transport has thus become a well-established sector of freight transport, with a market encompassing dozens of companies generating a joint turnover of several billion Euros per year (European Commission, 2014). A further aspect of combined transport is its presumed environment-friendliness. Although this view is not undisputed (Dekker *et al.*, 2012), combined transport is commonly considered more ecologically beneficial than pure road transport (Craig *et al.*, 2013), and the European Union and national governments have set up programmes for fostering combined transport (Council of the European Communities, 1992). In short, planning problems in combined transport are of considerable significance for a large number of enterprises as well as the economy and society as a whole.

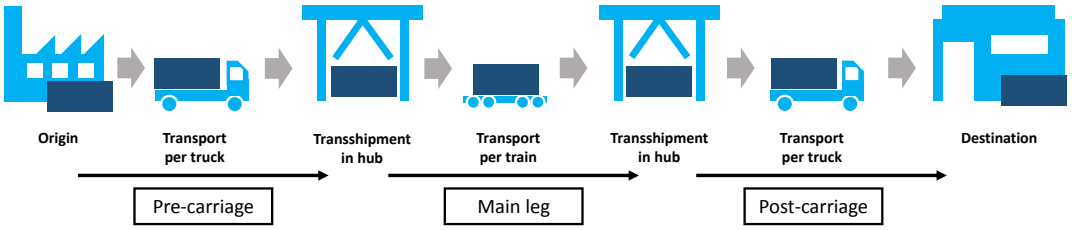


Figure 6.1: Schematic structure of combined transport

The SNDHLP we consider is as follows. We are given a set of transport requests (or shipments). Each request consists of one or more containers that must be transported from an origin location to a destination location. Each location may be the origin and/or the destination of more than one request, but all containers with the same origin and the same destination constitute one request. There are two possibilities for transporting a container: combined road-rail transport and direct transport from origin to destination by road. In the former case, the number of *hops*, i.e., the number of times a container may be transshipped at hubs, is limited to at most four (this is an operational requirement of our industry partner). This number includes the transshipments from road to rail at the

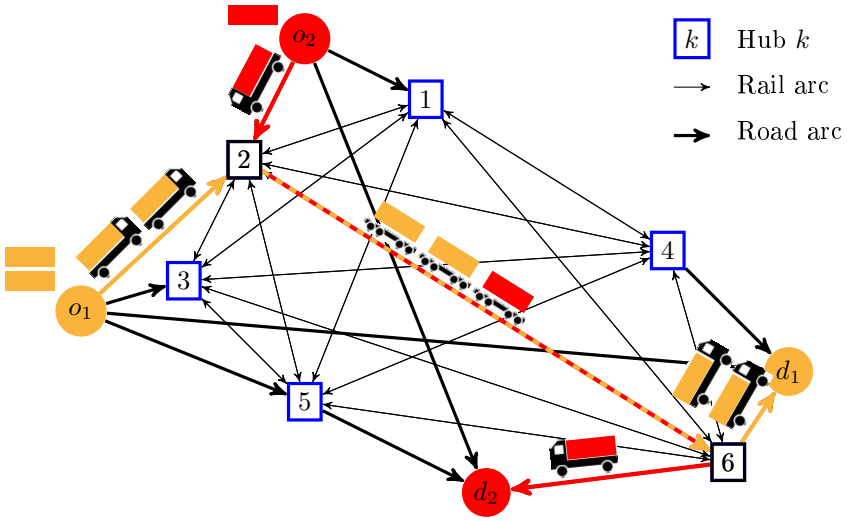
start hub and from rail to road at the end hub. Combined transport is relevant only for requests beyond a certain origin-destination distance, so that, for any request, the start and the end hub used are always different, and the number of hops is always at least two. There is a maximal time en route, including travelling and transshipment, for each request. This time is assumed non-restrictive in the case of road transport. The decision whether to transport a request by combined transport or entirely by road is part of the problem.

There are two types of hubs, *fixed* and *potential* ones. Fixed hubs are open and may be used in any solution. In addition, a number of potential hubs is available and may be opened and used. Because of the considered tactical planning horizon of between six months and one year, no fixed costs are incurred for using or not using a fixed hub or for opening and using a potential hub. From the practical point of view, the reason is that these costs are not known in advance. This is because using a fixed or opening a potential hub actually means that the rail company books a certain part of a hub's overall processing capacity, so that the costs are negotiable and depend on the booked capacity. Therefore, instead of opening costs, the number of hubs used in the solution is limited. The decision which hubs to use is part of the problem whenever the number of allowed hubs lies between the number of fixed hubs and the total number of available hubs. In the special case when only the fixed or all hubs may be used, the location aspect of the problem disappears, and only the service network design has to be determined. Each request can enter the rail network, i.e., be transshipped from road vehicle to rail wagon, at a given request-specific set of start hubs. Similarly, a request can leave the rail network, i.e., be transshipped from rail wagon to road vehicle, at a given request-specific set of end hubs. For any request, these two sets are disjoint, and they may each contain one element only. The selection of the start and the end hub for each request that uses combined transport is part of the problem.

We consider two problem variants, with *unsplittable* and *splittable* requests. 'Unsplittable' means that all containers belonging to the same request must take the same itinerary. 'Splittable' means that the itineraries of two containers from one request may be different (including different start and end hubs), and that some containers from one request may be transported by road and some by combined transport. Unsplit requests are required by customers who want to receive all containers of a request at the same time. On the other hand, the option to split requests can lead to cost savings for the transport company.

Owing to the tactical nature of the problem, the capacities of the rail links are assumed nonrestrictive. This means that at any point in time, an unlimited number of trains may use a rail link without affecting the travel time of a train. The number of trains to operate between an ordered pair of hubs is part of the problem. Similarly, the transshipment capacities at the hubs are assumed nonrestrictive, too. This means that an unlimited number of requests can be transshipped at a hub without any requests having to wait for transshipment. Moreover, due to the tactical nature of the problem and the averaged request data, our industry partner assumes homogeneous equipment, i.e., there is only one type of container, road vehicle, and rail wagon. In particular, we assume that each

Figure 6.2: Example problem instance with solution



available road vehicle and rail wagon can carry only one container at a time. The number of wagons, i.e., the capacity of a train, is limited and identical for all rail connections. The cost of a train trip between two hubs is independent of the actual capacity utilization of the train, i.e., of the number of rail wagons actually pulled. Costs for direct road transport are known for each request. Hence, from the point of view of a combined transport operator, direct road transport can be seen as outsourcing. The time for transshipping one container is hub-specific, but at each hub, it is the same for all requests. Hub-specific transshipment costs could be incorporated analogously but were not provided by our industry partner.

Overall, the problem is to decide on the itineraries of the requests, the usage of hubs, and the number of trains operating between hubs, with the objective of minimizing the overall costs for fulfilling all requests, either by combined transport or by pure road transport. These overall costs are comprised of the costs for operating trains between hubs and of the road transport costs. In the well-known classification of network design problems by Crainic (2000), the SNDHLP we consider is a *frequency service network design* problem, because it focuses on determining which traffic itineraries to operate and how often over the planning horizon to offer a train service. Scheduling aspects referring to when services depart, which are the subject of *dynamic service network design*, are not considered. Additionally, our problem possesses a hub location component, because a restricted number of hubs to use must be selected from a larger set.

Figure 6.2 shows a small example network consisting of six hubs and two requests with different numbers of containers (orange: two containers from o_1 to d_1; red: one container from o_2 to d_2). A solution is depicted in which both requests are routed via combined road-rail transport via hubs 2 and 6, making use of consolidation.

The contributions of the present paper consist in the development of (i) a path-based mixed-integer programming model for the SNDHLP, and (ii) a branch-and-price-and-cut

algorithm capable of solving instances with up to 100 requests and 10 potential hubs to optimality. For real-world instances with up to 5300 requests and 43 hubs, feasible solutions with a gap of less than 2 % are obtained within 60 minutes. This allows to use our implementation for practical planning purposes. Moreover, our solution approach consistently outperforms a state-of-the-art commercial solver applied out-of-the-box to an arc flow model.

The remainder of this paper is organized as follows. The next section provides a review of literature on hub location and service network design problems related to the SNDHLP. Section 6.3 presents our path-based formulation of the SNDHLP. In Section 6.4, the branch-and-price-and-cut algorithm is explained. Computational results are presented in Section 6.5, and we give a conclusion in Section 6.6.

6.2 Literature review

Assad (1980) was the first to present a survey of planning problems in rail transport. Further reviews were given by Crainic and Laporte (1997), Crainic (2003), and, especially for intermodal transport, by Bontekoning *et al.* (2004), Crainic *et al.* (2006), Caris *et al.* (2013), and SteadieSeifi *et al.* (2014). The problem we consider addresses both service network design and hub location aspects. Often, these two topics are treated independently. Wieberneit (2008) and Yaghini and Akhavan (2012) gave reviews of service network design problems occurring in freight transport. A good overview of hub location problems can be found in Alumur and Kara (2008). The simultaneous consideration of hub location and request routing decisions, as in our SNDHLP, increases the complexity of the models, but makes much more efficient solutions possible. There are several problem variants studied in the literature. In the following, we briefly describe important variants and then refer chronologically to papers dealing with problems that share properties with the one we study. We did not find a paper dealing with the same combination of problem features as our SNDHLP. A summarizing comparison of papers with respect to the considered problem aspects and the applied solution procedures is provided in Table 6.1.

As mentioned, requests can either be enforced to be transported as a whole (unsplittable requests) or the loading units of a request can use different routes (splittable requests). If several requests starting or ending at the same location have to be assigned to the same hub, this is called *single allocation*, whereas the opposite is called *multiple allocation*. Most papers model economies of scale caused by consolidation with the help of a discount factor smaller than 1, which is multiplied with the transport costs incurred for the use of links between two hubs. In this case, there is no reason for transshipments when a fully connected network is considered. A more accurate treatment determines the number of required means of transport to provide the needed capacity for each link individually. This causes discretely increasing costs and thereby a significantly higher complexity (Crainic, 2000). With this discrete capacity determination of the links, transshipping at hubs between means or modes of transport can improve the utilization and should be taken into account, though this also increases the difficulty. Furthermore, as

in our problem, some papers view the option of a direct transport from origin to destination at potentially higher costs, and there are also papers considering bounds for the transport times.

The first to consider hub location and routing of goods simultaneously was O'Kelly (1986). The positions of the hubs were not chosen out of a discrete set but located freely in the plane. Only the cases of one and two hubs were considered. Campbell (1994) gave several formulations for different variants of a hub location problem, regarding explicitly the decision of the usage of links. Brockmüller *et al.* (1996) studied a network design problem arising in the telecommunication sector, addressing the decision over integral numbers of capacities to be installed on the arcs. The problem was solved heuristically with the help of valid inequalities. Barahona (1996) considered a similar problem with the same application regarding both splittable and unsplittable flows. Yoon and Current (2008) were the first to study an integrated hub location and network design problem similar to our setting. However, they decided only on the usage of arcs and not on arc capacities. The first author looking at transport time limits in the context of hub location and network design was Campbell (2009). Ishfaq and Sox (2011) also addressed time limits. They used the context of combined road-rail transport and supported the possibility to choose between both modes of transport on each arc.

Bauer *et al.* (2010), in a follow-up to earlier work by Andersen *et al.* (2009), studied a problem similar to our SNDHLP. They considered two different objective functions: (i) the minimization of greenhouse gas emissions, disregarding any costs, and (ii) the minimization of the total costs of providing the services plus the sum of the time-en-route of all requests weighted by corresponding values of time. They developed an arc flow model based on a time-space network. Using a commercial solver and two types of valid inequalities, they were able to solve small real-world instances with up to 15 requests and 5 hubs to optimality. Bauer *et al.* (2010) compared the results obtained with the two objective functions and concluded that the cost minimization version leads to more services being offered, so that the capacity utilization of the services is higher when emissions are minimized. Another problem closely related to our setting was regarded by Üster and Agrahari (2011). While looking at the same decisions, they restricted themselves to the case of splittable requests and assumed that only one long-haul link is used by each request. They solved the problem with Benders decomposition.

In his doctoral thesis, Homfeld (2012) addressed rail freight services with explicit decisions on the number of trains running on each link between hubs. He looked at a special hierarchical order of consolidation centres and considers the requirement that two requests with the same destination have to use the same remaining route, once they meet at a hub. By their own account, the most general form of a hub location and network design problem addressed in the literature was regarded by Alumur *et al.* (2012). They allowed multiple modes of transport on each link. Contreras and Fernández (2012) studied the simultaneous optimization of hub locations and network design under the additional requirement that the chosen arcs result in a tree, connecting all hubs with a unique path between them. Zhang *et al.* (2013) studied the optimization of intermodal networks considering road, rail, and inland waterway, taking into account costs for CO_2

Table 6.1: Characteristics of SNDHLP and of similar models from the literature

	Our paper	Brockmüller *et al.* (1996)	Barahona (1996)	Yoon and Current (2008)	Campbell (2009)	Bauer *et al.* (2010)	Ishfaq and Sox (2011)	Üster and Agrahari (2011)	Alumur *et al.* (2012)	Homfeld (2012)	Contreras and Fernández (2012)	Zhang *et al.* (2013)
Network design	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
Hub location	p-median			Fixed opening costs	p-median	p-median	p-median, fixed costs	Fixed opening costs	p-median, fixed costs		p-median	
Application	Intermodal transport	Telecommunication network	Telecommunication network	Not specified	Time definite motor carriers	Intermodal transport	Intermodal transport road-rail	Freight transport	Multimodal transport	Rail freight transport	Not specified	Intermodal transport road-rail-inland waterway
Direct transport possible	✓	✓		✓		✓	✓					✓
Splittable requests	✓		✓	✓	✓	✓		✓				
Unsplittable requests	✓	✓	✓			✓	✓		✓	Extended, see text	✓	✓
Multiple allocation	✓	✓	✓	✓	✓	✓	✓	✓				✓
Discretely increasing link costs	✓	✓	✓					✓		✓		
Maximal number of hops	4						2	2		9		
Transport time limit	✓				✓		✓		✓	✓		
Solution approach	Branch-and-price-and-cut	Mat-heuristic	Heuristic	Dual ascent heuristic	MIP solver	MIP solver with static cuts	Tabu search	Benders decomposition	Mat-heuristic	Mat-heuristic	MIP solver	Heuristic bilevel programming

emissions. As in our problem, they considered multiple commodities and request-specific sets of start and end hubs. However, they did not consider splittable requests, time-en-route limits, or transport economies of scale through consolidation. They solved their problem with heuristic bilevel programming. The upper level, which is solved by a genetic algorithm, searches for hubs to be opened and adequate CO_2 emission prices. The lower level, which is solved by a shortest-path algorithm for each request, computes minimal transport and CO_2 costs given a certain network structure and CO_2 prices.

The recent paper by Crainic *et al.* (2015) presents a more integrated model for combined location and service network design. The model links strategic and tactical decisions, considers different types of transport resources, i.e., a heterogeneous fleet, and takes into account vehicle scheduling and repositioning. Their solution approach consists of representing the problem as a time-space network to which a matheuristic combining column generation and slope scaling is applied.

6.3 Mathematical model

Let $G = (V, A)$ be a digraph with node set V and arc set A. The node set V contains one origin and one destination node for each request, and one node for each hub. The set of all hub nodes is denoted by H, and the set of fixed hubs by H_f, with H_f a subset of H. The number of hubs that may be used is given by n_H. The set of arcs contains the rail arcs between any two hubs, A_t, with costs per kilometre c_t and the road arcs, A_s, with costs per kilometre c_s. The road arcs are composed of the direct road arcs from the origins to the destinations of the requests and the arcs connecting the origins and destinations to the hub network. Each request $r \in R$ comprises m^r containers. The length of an arc a is given by l_a. The train capacity is k_t.

We tackle the problem with a path-based model, due to past experiences (Homfeld, 2012) as well as tests and comparisons with arc-based models (cf. Section 6.5). A feasible combined transport path for a request is an elementary path from the origin of the request to the destination, passing through an allowed start and an allowed end hub, performing at most the allowed number of hops, and maintaining the travel time constraint. A road transport path for a request is simply the single-arc path from the origin of the request to the destination and is feasible by definition. The set of feasible paths for a request r is denoted by P^r. P_i and P_a indicate the sets of paths containing hub i and arc a respectively, with $P_i^r = P_i \cap P^r$ and $P_a^r = P_a \cap P^r$.

The model contains three types of decision variables: binary variables h_i indicate whether hub $i \in H$ is used. General integer variables t_a count the number of trains that use arc $a \in A_t$. Continuous variables $y_p^r \in [0, 1]$ indicate the fraction of the containers of request $r \in R$ which are transported via path $p \in P^r$.

Given these definitions, the SNDHLP can be formulated as follows:

$$\min \quad \sum_{a \in A_t} c_t l_a t_a + \sum_{r \in R} \sum_{p \in P^r} \sum_{a \in p \cap A_s} c_s l_a m^r y_p^r \tag{6.1a}$$

$$\text{s.t.} \quad h_i = 1 \qquad \forall\, i \in H_f \tag{6.1b}$$

$$\sum_{i \in H} h_i \leq n_H \tag{6.1c}$$

$$\sum_{p \in P^r} y_p^r = 1 \qquad \forall\, r \in R \qquad (\gamma_r) \tag{6.1d}$$

$$\sum_{r \in R} \sum_{p \in P_a^r} m^r y_p^r \leq k_t t_a \qquad \forall\, a \in A_t \qquad (\delta_a) \tag{6.1e}$$

$$\sum_{p \in P_i^r} y_p^r \leq h_i \qquad \forall\, i \in H, r \in R \qquad (\epsilon_i^r) \tag{6.1f}$$

$$h_i \in \{0,1\} \qquad \forall\, i \in H \tag{6.1g}$$

$$t_a \in \mathbb{N}_0 \qquad \forall\, a \in A_t \tag{6.1h}$$

$$y_p^r \in \{0,1\} \qquad \forall\, r \in R, p \in P^r \tag{6.1i}$$

The objective function, (6.1a), seeks to minimize the total costs incurred by the covered rail and road kilometres. Conditions (6.1b) and (6.1c) ensure that all fixed hubs are opened and that at most the allowed number of hubs is used. Constraint (6.1d) makes sure that every request is transported. Constraint (6.1e) calculates the used capacity and determines the number of needed trains per arc. By constraint (6.1f), a hub is forced to be opened if a path passes through it. The last three constraints determine the domains of the variables.

The above model targets the case of unsplittable requests, but the following simple domain modification approximates the case of splittable requests:

$$y_p^r \in [0,1] \qquad \forall\, r \in R, p \in P^r$$

Note that this condition does not model the option to select different itineraries for the containers of one and the same request in an exact manner, because it allows all fractional solutions, not only multiples of $\frac{1}{m^r}$. Hence, solutions with distinct routes for *parts* of containers can occur, which is impossible in reality. Nevertheless, this inaccuracy is acceptable, because train costs are independent of train load, and train capacity is high in relation to the typical number of containers of single requests (40 to 3). Therefore, most requests are not split anyway, and any remaining inaccuracies result from the road transport parts only.

The problem can also be formulated with arc variables. Then, the feasibility of the routes concerning transport times and limits for the number of hops has to be expressed explicitly in the model. For the unsplittable case, this is straightforward, whereas for the splittable case, additional time and hop counter variables are needed. Linear formulations for these models can be found in the Appendix.

6.4 Branch-and-Price-and-Cut

The number of feasible paths can be huge, so an enumeration of all possible routes is not practicable. Instead, we use delayed column generation to dynamically generate

promising paths. To achieve integrality, this is embedded in branch-and-bound. Besides, we use valid inequalities to strengthen the linear relaxation.

To solve our path-based SNDHLP model (6.1), we developed a branch-and-price-and-cut algorithm, see Barnhart *et al.* (1996), Desaulniers *et al.* (2005), and Lübbecke and Desrosiers (2005). In the following subsection, we illustrate our pricing algorithm to find new paths with negative reduced costs. Afterwards, we present our branching scheme and some valid inequalities, for which we describe their influence on the pricing problem. At the end of this section, we mention some techniques that enhance the performance of the branch-and-price-and-cut algorithm.

6.4.1 Generation of new columns

For the column generation, we consider the linear relaxation of the master problem. To have a feasible problem initially, we start with a small subset of paths, containing, for each request, one path that represents the complete transport by road. Recall that the pricing problem searches for variables with negative reduced costs to add them to the current set of columns of the master problem. Let γ_r be the dual variable associated to constraint (6.1d) for request $r \in R$, δ_a the dual variable corresponding to the capacity restriction (6.1e) of arc $a \in A$, and ϵ_i^r the dual belonging to constraint (6.1f) for hub $i \in H$ and request $r \in R$. Then, the reduced costs for a path p for request r are

$$\tilde{c}_p = \sum_{a \in p \cap A_s} c_s m^r l_a - \gamma_r - \sum_{a \in p \cap A_t} m^r \delta_a - \sum_{i \in p \cap H} \epsilon_i^r. \tag{6.2}$$

The subproblem for a certain request $r \in R$ is a shortest path problem with resource constraints (SPPRC). SPPRCs are well studied, because they appear in many applications, see Irnich and Desaulniers (2005). They are usually solved by dynamic-programming based labeling algorithms. One of the first algorithms in this category is the one proposed by Aneja *et al.* (1983). More recent papers compare mono- and bi-directional extensions (Righini and Salani, 2006) and deal with special properties of such algorithms in the context of branch-and-price-and-cut algorithms (Zhu and Wilhelm, 2013).

Let $r \in R$ be a request for which we solve the pricing problem. We define a label $l = (v, \bar{l}, z)$ that marks a path from the origin to a node $v \in V$ with $\bar{l}$ as the previous label on the path and $z \in \mathbb{R}_+^m$ as the vector of resource consumptions of the path. For our purposes, the resource vector $z = (z_1, z_2, z_3) = (\tilde{c}, d, h)$ contains the sum of the modified arc costs $\tilde{c}$, the transport time d and the number of hops h of the partial path up to the associated node. As most dual variables are zero, particularly at the beginning of the column generation process, and as shorter paths tend to be cheaper and are more likely to be part of an optimal solution, it seems reasonable to search for paths with the most negative reduced costs and, among these, the one with the shortest transport time and the lowest number of hops. Therefore, we consider a lexicographic order of objectives with bounds for the downstream targets. All labels belonging to the same node can be compared according to the order of the objectives ($\tilde{c} - d - h$):

Definition 6.4.1 (Order of labels)**.** A label $l = (v, \bar{l}, z)$ is better than a label $l' = (v, \bar{l}', z')$, written $l < l'$, if

$$\exists\, i \in \{1,2,3\} : (z_j = z'_j\ \forall\ j < i) \wedge (z_i < z'_i).$$

Besides, we say a label dominates an other label belonging to the same node, if the former is at least as good as the other label in all resource categories and strictly better in at least one resource consumption:

Definition 6.4.2 (Dominance between labels)**.** A label $l = (v, \bar{l}, z)$ dominates a label $l' = (v, \bar{l}', z')$, written $l \prec l'$, if

$$z \leq z' \wedge \exists\, i \in \{1,2,3\} : z_i < z'_i.$$

In general, an SPPRC can be solved with a monodirectional or a bidirectional algorithm. Because of the special structure respecting the limited hop-number, we implemented both a monodirectional forward and a bidirectional pricing routine in order to compare them. In the former case, we extend partial paths only from the origin, always continuing with the best unprocessed label as in the Dijkstra algorithm. In the latter case, we start from both the origin and the destination, make a predefined number of hops from both sides, determine all non-dominated paths, and choose the best alternative from a concluding merge step.

In the following, we describe the forward extension of the paths. The dual prices corresponding to the hub usage, the transshipment times on the hubs, and the increase of the hop counter are assigned to the ingoing arcs. Starting from a label l, we look at all outgoing arcs to other hubs, add the weight and resource consumptions for this path extension, and create a new label. Let the hub indicator ζ_i be 1 if $i \in H$ and 0 otherwise, and let the arc indicator ξ_a be 1 if $a \in A_t$ and 0 if $a \in A_s$. For the sake of readability, we write ξ_{ij} for $\xi_{(i,j)}$ (similar for other variables). With d_a as the transport time over arc a and d_i as the transshipment time on hub i, the new label is determined as follows:

$$l_{\text{new}} = (j,\ l, \tilde{c}(l_{\text{new}}),\ d(l_{\text{new}}),\ h(l_{\text{new}})) \tag{6.3}$$

with

$$\tilde{c}(l_{\text{new}}) = \tilde{c}(l) + (1 - \xi_{ij}) c_s m^r l_{ij} - \xi_{ij} m^r \delta_{ij} - \zeta_j \epsilon^r_j \tag{6.4a}$$
$$d(l_{\text{new}}) = d(l) + d_{ij} + \zeta_j d_j \tag{6.4b}$$
$$h(l_{\text{new}}) = h(l) + \zeta_j. \tag{6.4c}$$

Then we test whether this is feasible with regard to the resource limits. In addition, we check whether a feasible extension of the new label to the destination exists. Afterwards, we check whether the new label is dominated by one of the labels already assigned to the node. If yes, it is discarded, otherwise it is added to the label list of the node. Finally, all other labels are tested for dominance by the new one and are deleted in this case. We store the label list of a node as an ordered list, so that insertion and dominance are

performed simultaneously and take time linear in the number of labels at the node, see Algorithm 2. The backward extension of paths works analogously.

Algorithm 2: Potential insertion of a new label

Input: An ordered label list *list* belonging to a node, a new label l_{new}

```
1  l = list.first();
2  while l < l_new do
3  |   if l ≺ l_new then
4  |   |   Delete l_new;
5  |   |   return;
6  |   l = list.next();
7  Insert l_new before l;
8  while l ≠ list.end() do
9  |   if l_new ≺ l then
10 |   |   Delete l;
11 |   l = list.next();
```

The monodirectional pricer is applied on the network defined in Section 6.3. For the bidirectional pricer, we use a layered network constructed as follows: if $u_{\max}$ is the maximal allowed number of hops, we copy the hubs $u_{\max}+1$ times to create $u_{\max}+1$ hub node layers, numbered from 1 to $u_{\max}+1$. The first (last) layer contains only copies of hubs in the request-specific set of start (end) hubs. The bidirectional pricer propagates labels forward from the origin vertex only up to layer $\lfloor u_{\max}/2 \rfloor + 1$, and backward from the destination vertex only up to layer $\lfloor u_{\max}/2 \rfloor + 2$. Hence, forward (backward) road arcs are inserted from the origin (destination) vertex to all copies in the first (last) layer, and forward (backward) rail arcs are inserted between any two copies of different hubs in neighbouring layers from layer 1 up to layer $\lfloor u_{\max}/2 \rfloor + 1$ (from layer $u_{\max}$ back to layer $\lfloor u_{\max}/2 \rfloor + 2$). This yields a network as depicted in Figure 6.3, on which the shortest path search is done in both directions.

In this approach, we store label lists for each hub on each layer, so that the dominance criterion only needs to consider the modified arc costs and the transport time of the labels, but not the hop counter resource. To take into account all possible numbers of hops $(2, \ldots, u_{\max})$, we merge the label lists of all hubs at the last backward layer $(\lfloor u_{\max}/2 \rfloor + 2)$ with all forward layers except the first (1) and the label lists of the first forward layer with all backward layers except the first $(u_{\max}+1)$. The exceptions are motivated by the requirement that start and end hub of a request path are always different, as mentioned in the Introduction.

6.4.2 Branching

For a feasible solution, the hub variables (h) must be binary, the number of trains running between any two hubs (t) must be integer, and, in the case of unsplittable requests,

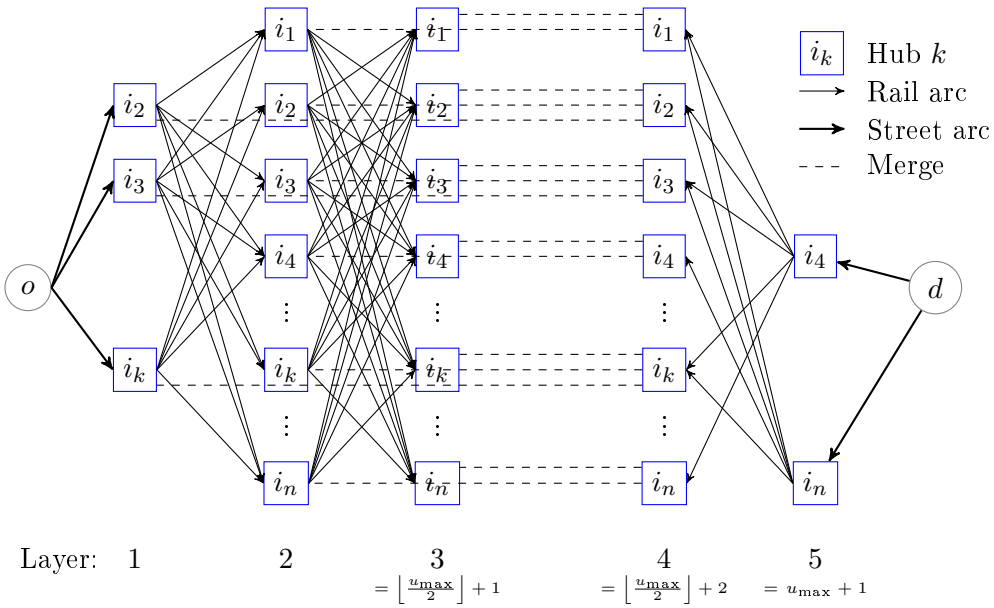


Figure 6.3: Layered network for the bidirectional pricing algorithm, $u_{\max} = 4$

also the path variable values (y) must be binary. To achieve this, we use the following branching scheme.

Branching on hub variables

Using or not using a hub has the biggest influence on the solution, so these decisions are taken first. To identify the most important hub, we calculate the number of containers transshipped at each hub in the current LP solution. For the hub i with the largest such number and fractional h_i, we create two branches, one fixing the value of the corresponding variable h_i to 0 and the other one to 1 by changing the upper and lower bounds, respectively. Whereas the 1-branch has no impact on the pricing algorithm, the 0-branch requires arcs ending at the corresponding hub to be excluded from the network.

Branching on train variables

For the number of trains, we first use a global approach, regarding the total number of trains running in the whole network, n_t. Whenever n_t is fractional, we create two branches and add a constraint to the master problem in both branches, setting the upper bound on the number of trains to $\lfloor n_t \rfloor$ in one branch, and setting the lower bound to $\lceil n_t \rceil$ in the other. This requires no change in the pricing problems. To guarantee integrality of the train variables on each arc, we have to branch further by setting similar bounds on individual train variables. If the upper bound of a train variable is set to zero, the corresponding arc has to be removed from the network of the pricing problem. In all other cases, the branching decision does not affect the pricing problem. For splittable

requests, the described branching rules suffice to obtain a feasible solution.

Branching on path variables

In the case of unsplittable requests, variables generated in the pricing routine cannot simply be forbidden by setting their upper bound to zero, because the excluded path will be regenerated by the pricing problem solver. We devised two approaches to treat integrality requirements on path variables, both concerning single arcs of a path.

The first possibility directly regards single arcs. To forbid an arc, all paths containing this arc are excluded from the network, and the pricing problem ignores this arc. The branch that enforces the arc a for request r, however, needs an additional constraint in the master problem:

$$\sum_{p \in P_a^r} y_p^r \geq 1.$$

The dual variable corresponding to this constraint takes nonnegative values, which can cause negative arc weights in the subproblem. To avoid cycles in the resulting paths, the algorithm searching for the best new variable entering the master problem would have to be adapted.

We therefore chose a second possibility, one that does not change the nature of the pricing problem. If u indicates the number of hubs in a path, this path will contain $u+1$ *stages*. The first stage consists of the arc from the origin to the first hub, the $(u+1)$th stage is the arc from the uth hub to the destination. The second possibility to branch on path variables now uses the idea to forbid and enforce arcs for use on certain stages of a path for a request. In the branch in which an arc is forbidden, the upper bound of all paths using this arc on the chosen stage is set to zero, and the arc is excluded from the pricing algorithm at this stage. In the other branch, in which the arc is enforced to be used by this request on the requested stage, all path variables not fulfilling this requirement are set to zero, and the pricing problem excludes all other arcs on this stage.

6.4.3 Cuts

To strengthen the linear relaxations of the master problem, we use several types of valid inequalities that are based on the main source of fractionality, the capacity restriction (6.1e). With $b_r = \frac{m^r}{k_t}$, this constraint for an arc a can be rewritten as

$$\sum_{r \in R} b_r \sum_{p \in P_a^r} y_p^r \leq t_a.$$

Of course, because of the positive costs for the trains, this inequality will always be fulfilled with equality in the LP relaxation, resulting in mostly fractional train variable values. Moreover, we see that not individual paths, but the sum of all path variables, i.e., the total flow, for the same request containing an arc has to be considered. In the following, we present three cuts with different application areas. Additional cuts for fixed-charge network design problems are described in Chouman *et al.* (2017).

Simple cuts

The simplest valid inequality, adapted from the *strong cuts* described in (Chouman *et al.*, 2003), is only useful for arcs with light traffic, i.e., for those arcs with train variable values of less than one. In those cases, we can add the following cut:

$$t_a \geq \sum_{p \in P_a^r} y_p^r \quad \forall\, r \in R_a. \tag{6.5}$$

The separation of these cuts is done by inspection: we look at all train variables with values in $(0, 1)$, compare them with the flows of the requests over this arc, and add all violated ones. In the pricing routine, these cuts need to be incorporated through additional arc costs. Of course, the dual price has to be added only for the request and the arc that are addressed by the simple cut.

Residual capacity cuts

For splittable requests, we use the so called *residual capacity cuts*, developed by Magnanti *et al.* (1993). For a subset $S \subseteq R$, define the sum $B_S := \sum_{r \in S} b_r$. Then, the cuts can be written as

$$(B_S - \lfloor B_S \rfloor) t_a + B_S - (B_S - \lfloor B_S \rfloor) \lceil B_S \rceil \geq \sum_{r \in S} b_r \sum_{p \in P_a^r} y_p^r. \tag{6.6}$$

To get the most violated inequality for an arc a, Atamtürk and Rajan (2002) showed that one has to choose the set $S = \left\{ r \in R : \sum_{p \in P_a^r} y_p^r > t_a - \lfloor t_a \rfloor \right\}$. Then, it suffices to check whether the current solution satisfies

$$\lfloor t_a \rfloor < B_S < \lceil t_a \rceil \quad \text{and}$$

$$\sum_{r \in R} b_r (1 - \sum_{p \in P_a^r} y_p^r - \lceil t_a \rceil + t_a) < (\lceil t_a \rceil - t_a) \lfloor t_a \rfloor .$$

If so, the corresponding residual capacity cut can be added to the master problem. Otherwise, no inequality of this type is violated by the current LP solution for the considered arc. Let μ_{Sa} be the dual price for a residual capacity cut represented by (S, a). Then, in the pricing algorithm for a request r, the weight $b_r \mu_{Sa}$ has to be added to the arc costs of an arc a, if $r \in S$.

c-Strong cuts

In the case of unsplittable requests, a stronger valid inequality can be used. Brockmüller *et al.* (1996) presented the so called *c-strong-inequality* in the application area of telecommunication networks, and Homfeld (2012) applied them to a rail freight transport problem.

For a $c \in \mathbb{N}_0$, a set $S \subset R$ is called c-strong if $c = \sum_{r \in R} \lceil b_r \rceil - \left\lceil \sum_{r \in R} b_r \right\rceil$. For such a c-strong set S, a valid inequality for our problem for unsplittable requests can be written

as

$$t_a + c \geq \sum_{r \in S} \lceil b_r \rceil \sum_{p \in P_a^r} y_p^r + \sum_{r \in R_a \setminus S} (\lceil b_r \rceil - 1) \sum_{p \in P_a^r} y_p^r. \tag{6.7}$$

To separate these inequalities, the following knapsack problem has to be solved:

$$\begin{aligned} \max \quad & \sum_{r \in R_a} \pi_r \sum_{p \in P_a^r} y_p^r \\ \text{s.t.} \quad & \sum_{r \in R_a} (\lceil b_r \rceil - b_r) \pi_r < c + 1 \\ & \pi_r \in \{0, 1\} \qquad \forall\, r \in R_a \end{aligned}$$

Then, the set $S = \{r \in R_a \mid \pi_r = 1\}$ yields the most violated c-strong cut for the arc a. The consideration of these cuts in the subproblem solver is similar to above. Let λ_{Sa} be the dual variable belonging to a c-strong cut with request-subset S and arc a. Then all requests in S must add $\lceil b_r \rceil \lambda_{Sa}$ to the modified costs of arc a, whereas all other requests have to add the rounded-down variant $\lfloor b_r \rfloor \lambda_{Sa}$.

6.4.4 Algorithmic improvements

In this section, we mention some techniques we use to improve our branch-and-price-and-cut algorithm. These are partial pricing, dynamic constraint generation, early branching, and usage of a MIP solver to obtain upper bounds.

Partial pricing

To accelerate the branch-and-price-and-cut algorithm, we perform partial pricing. Instead of searching for a negative reduced cost path for all requests in each iteration, we sometimes restrict ourselves to the most promising ones: requests for which an entering path variable was found in the last pricing iteration. The reason is that these requests are more likely to yield another negative reduced cost path in the following pricing step. Nevertheless, single path changes can cause completely different dual variables and, hence, there may also be other requests with improving columns in later iterations. Thus, with probability

$$1 - \frac{\text{Number of requests with new path in last iteration}}{\text{Total number of requests}},$$

we search for new columns only for the most promising requests. Naturally, when the partial pricing does not generate new columns, a full search has to be done to guarantee optimality.

Dynamic constraint generation

The number of constraints ensuring that hubs on used paths are opened can be very high (number of hubs · number of requests) (6.1f). Because this can make the model unnecessarily large, we generate them on demand. When no more improving paths are

found, we enumerate all hub-request pairs and add the corresponding constraint (6.1f) if it is violated. Since one request suffices to enforce a hub to be opened, we limit the number of such constraints added in one iteration.

Early branching

Although in general the cuts significantly increase the lower bound, they enlarge the model and herewith increase the solution time for the LP relaxation. Sometimes, also the branching decision has a considerable impact on the lower bound. Therefore, we restrict the separation of cuts in two ways. Firstly, we set a fixed bound on the minimal violation required for a cut to be added to the model. Secondly, we limit the number of cut-generation iterations per branch-and-bound node.

Usage of a MIP solver

Our branch-and-bound algorithm works with a best-first branching strategy, so that always the node with the smallest lower bound is explored next. Thus, there is no need for a good upper bound to reject nodes early, as those nodes with lower bounds above the optimal solution are simply fathomed as soon as the latter one is found. Nevertheless, the branch-and-price-and-cut algorithm may stop because of a timeout without having found an optimal or even a feasible solution. To improve the chances of obtaining a good feasible solution in such a case, we apply a MIP solver on the restricted problem (6.1) with the current set of columns at different stages of the algorithm: after solving the root node, we apply the MIP solver with a running time limited to 60 seconds. In the case of a timeout, we start the MIP solver again. Because the model is much larger at this point (compared to the root node) and the best found solution directly decreases the optimality gap, we allow a running time of five minutes.

6.5 Computational study

In this section, we present computational experiments to analyze the effectiveness of the proposed branch-and-price-and-cut procedure. The algorithm was implemented in C++, compiled with Microsoft Visual Studio 2010, and used the CPLEX 12.6 library for solving linear programs and as a MIP solver as described in the previous section. The experiments were performed on a standard PC with an Intel Core i7-4790 CPU at 3.60 GHz, 8 GB of RAM, running Windows 7 Enterprise.

6.5.1 Instances

Europe's largest railway company, DB Mobility Logistics AG, kindly provided us with two sets of requests based on real-life data with different container-per-request ratios. According to company information, the requests were obtained by computing daily averages over a representative period of time. These instances respectively contain 3,300 and 5,300 shipments inside Germany, suitable for combined transport, and 43 hub locations,

18 of which are considered fixed. By randomly choosing subsets of requests and selecting the hubs with the largest capacity, we generated smaller instances. The hubs selected in this way were still evenly spread throughout Germany. Furthermore, we varied the maximal allowed number of hubs to be used. In total, we generated 144 smaller instances, ranging from 100 to 500 requests and from 10 to 21 hubs. For these small instances, we assumed no fixed hubs. The request-specific sets of allowed start and end hubs were constructed by considering all hubs that were within 1/6 of the Euclidean origin-destination distance of a request. This resulted in an average of 2.5 potential start and end hubs per request for the largest hub set. For 21 and 10 hubs, the averages were 1.9 and 1.1, respectively. The maximal number of hops was set to $u_{\max} = 4$. The time-en-route limits were calculated by multiplying the road transport time with a factor of between 1.5 and 2.5, increasing with growing origin-destination distances. To be precise, the factor is 1.5 for requests with origin-destination distances of up to 300 km, 2.0 for those with distances of up to 600 km, and 2.5 for all others. The ratio of the costs per kilometre on street to the costs per kilometre on rail was assumed to be 3/7. All test instances were tackled by our algorithm regarding the requests as splittable as well as unsplittable.

6.5.2 Algorithmic settings

We tried several algorithmic variants to solve the problem. Table 6.2 shows a comparison of different settings: (i) use of bidirectional or monodirectional pricer, (ii) branching on the sum of all trains, (iii) using the cuts from Section 6.4.3, and (iv) dynamically adding constraints (6.1f) to the model. The results are based on six instances with 100 requests and 21 potential hubs from which 10 have to be chosen, and on six instances with 500 requests and 21 potential hubs from which 15 have to be chosen, with a runtime of 15 minutes. The deviations listed in the table demonstrate clearly that the cutting planes presented in Section 6.4.3 have a significant impact on the lower bound. The benefits of the dynamically generated constraints are smaller, but appear in all instances. Furthermore, the results indicate that branching on the sum of all used trains does not improve the lower bound. The choice of the best pricing algorithm depends on the instance; the bidirectional pricer produces slightly better lower bounds on average. Therefore, we decided to choose as 'best setting' the one with the bidirectional pricing algorithm, without branching on the sum of all trains, with the use of all described cuts, and with dynamic generation of constraints (6.1f).

6.5.3 Results

To assess the capabilities of our branch-and-price-and-cut algorithm, we compared its performance, using the best setting mentioned above, to that of the standard solver Cplex, applied out-of-the-box to arc-based formulations for the SNDHLP (specified in the Appendix). The findings were unequivocal: for all test instances, our algorithm yielded much better lower bounds, up to 86 % above those obtained by Cplex, and 23.5 % higher on average. Both our algorithm and Cplex had no problems in finding feasible solutions, but the upper bounds from Cplex were, on average, 27 % worse for splittable and 14.2 %

Table 6.2: Comparison of lower bounds for different solving strategies

		Use bidirectional pricer					
		true	true	false	false		
Branch on sum of trains	true	1.57 (1.71)	49.34 (49.71)	1.51 (1.72)	49.54 (49.66)	true	Generate dynamic constraints
	true	1.69 (1.91)	50.22 (51.13)	1.75 (2.10)	50.51 (50.77)	false	
	false	0.13 (0.11)	42.20 (42.70)	0.22 (0.10)	42.33 (42.48)	true	
	false	0.15 (0.38)	42.94 (43.29)	0.38 (0.29)	43.22 (43.95)	false	
		true	false	true	false		
		Use cuts					

The values show the sum (over the twelve test instances used) of the relative deviations of the lower bound of this setting from the best lower bound found among all settings for splittable requests (and, in parentheses, for unsplittable requests).
For example, the first value in the upper right corner indicates that the lower bounds for splittable requests are on average 49.54 % above the best found lower bounds when branching on the sum of trains and generating constraints (6.1f) on demand is activated while the monodirectional pricer is used and no cuts are included.

worse for unsplittable requests. This justifies the modelling and implementation efforts spent on the path-based formulation and the branch-and-price-and-cut algorithm.

We then applied our algorithm (with the best setting) to all instances described in Section 6.5.1. Tables 6.3 and 6.4 present the results. The time limit for the large instances was set to one hour, whereas for the smaller instances, 30 minutes were allowed. Table 6.3 indicates, for each large instance, the solution time (where 'T.L.' indicates that the time limit was reached), the relative gap between the final lower bound and the best found feasible solution, the percentage of the overall computation time needed for cut separation, pricing, and reoptimization, the number of added cuts, the number of pricing runs, the number of generated paths, the number of solved branch-and-bound nodes, and the number of requests that were transported completely by road in the best integer solution. In Table 6.4, the same information is displayed for each group of smaller instances. In addition, the number of instances with a gap of less than 3 % is indicated.

The most important insights are the following:

- Our algorithm was able to solve to optimality instances with up to 100 requests and 21 hubs. For almost all instances, an optimality gap in the single-digit area was reached within one hour.
- Of the 36 instances with 100 requests, 18 split (19 unsplit) could be solved to optimality. For 500 requests, only 6 (10) of the 36 instances finished with a gap greater than 3 %.

Table 6.3: Results for large instances

Instance	$\|H\|$	n_H	Time in s	%Gap	%Separation time	%Pricing time	%Reopt. time	#Cuts	#Pricings	#Paths	#Nodes	#Only Road
5392-Split	43	18	T.L.	0.28	0.6	72.7	2.4	2219	426	95427	93	1705
3351-Split	43	18	T.L.	0.26	1.6	65.8	4.2	3577	757	34746	369	1552
5392-Split	43	20	T.L.	2.58	0.7	71.4	1.6	2708	379	80812	101	689
3351-Split	43	20	T.L.	5.26	1.3	68.2	4.6	4501	671	35020	315	1130
5392-Split	43	25	T.L.	6.31	0.4	78.0	3.2	4710	395	81214	63	279
3351-Split	43	25	T.L.	12.82	0.7	66.7	17.0	7131	591	41494	163	692
5392-Unsplit	43	18	T.L.	0.33	0.8	63.7	2.9	1696	664	99787	113	1699
3351-Unsplit	43	18	T.L.	0.58	1.9	56.8	5.2	2846	973	34129	447	1549
5392-Unsplit	43	20	T.L.	0.85	0.8	70.7	2.1	2149	780	89925	112	677
3351-Unsplit	43	20	T.L.	0.92	1.6	63.8	4.6	3806	1068	37656	395	1137
5392-Unsplit	43	25	T.L.	1.80	0.4	80.8	2.9	3894	901	79626	69	272
3351-Unsplit	43	25	T.L.	1.92	0.9	63.9	15.6	5572	1106	42158	219	616

Table 6.4: Average results for small instances, 6 instances per group

Instance group	$\|H\|$	n_H	Time in s	%Gap	#Gap $< 3\%$	%Separation time	%Pricing time	%Reopt. time	#Cuts	#Pricings	#Paths	#Nodes	#Only Road
100-Split	10	5	373	0.00	6/6	0.1	44.5	42.0	933	2405	2856	1754	44
100-Split	10	10	851	0.86	6/6	0.1	33.9	49.4	2357	2602	2788	1961	2
100-Split	21	5	636	0.14	6/6	0.1	28.2	65.7	1658	2033	12484	288	57
100-Split	21	10	1263	1.34	5/6	0.2	21.2	71.5	3575	2507	12022	458	27
100-Split	21	15	1289	3.31	3/6	0.2	16.6	75.8	4920	2135	8761	570	12
100-Split	21	21	T.L.	6.00	2/6	0.2	13.4	77.3	5946	2671	8525	922	7
500-Split	10	5	T.L.	0.31	6/6	0.2	41.3	38.0	1770	4945	11632	3900	223
500-Split	10	10	T.L.	1.56	6/6	0.2	65.5	16.1	3964	3723	5953	3387	5
500-Split	21	5	T.L.	0.45	6/6	0.1	8.8	86.6	1835	1392	40195	157	293
500-Split	21	10	T.L.	2.12	4/6	0.1	15.7	77.5	2714	1444	28879	281	128
500-Split	21	15	T.L.	2.38	6/6	0.1	23.2	68.2	4635	1166	18903	304	43
500-Split	21	21	T.L.	14.47	2/6	0.1	28.9	63.5	7956	773	11700	218	7
100-Unsplit	10	5	665	0.00	6/6	0.1	31.1	40.5	769	4142	2893	3459	44
100-Unsplit	10	10	1041	1.12	5/6	0.1	26.4	42.4	1925	4184	2809	3504	2
100-Unsplit	21	5	634	0.20	6/6	0.1	26.0	68.0	1416	2058	12616	295	57
100-Unsplit	21	10	1256	2.51	3/6	0.1	17.8	73.4	2983	2681	12504	497	27
100-Unsplit	21	15	1270	4.29	2/6	0.1	13.8	80.2	4025	2300	8998	645	13
100-Unsplit	21	21	T.L.	10.14	2/6	0.2	10.5	81.9	4734	3038	8892	1146	5
500-Unsplit	10	5	T.L.	0.29	6/6	0.3	34.1	40.3	1494	5323	11668	4265	226
500-Unsplit	10	10	T.L.	1.84	6/6	0.3	59.4	19.5	3178	4913	6347	4542	4
500-Unsplit	21	5	T.L.	0.54	6/6	0.1	6.9	88.2	1504	1469	41146	165	292
500-Unsplit	21	10	T.L.	1.67	6/6	0.1	12.5	80.5	2241	1607	30054	325	127
500-Unsplit	21	15	T.L.	4.31	2/6	0.1	18.6	72.4	3742	1384	19730	367	44
500-Unsplit	21	21	T.L.	10.80	0/6	0.1	26.4	65.8	6617	994	12368	321	6

Of the 12 large instances, 2 (4) could be solved with a gap of less than 1 %, and only three had a gap of more than 3 %.

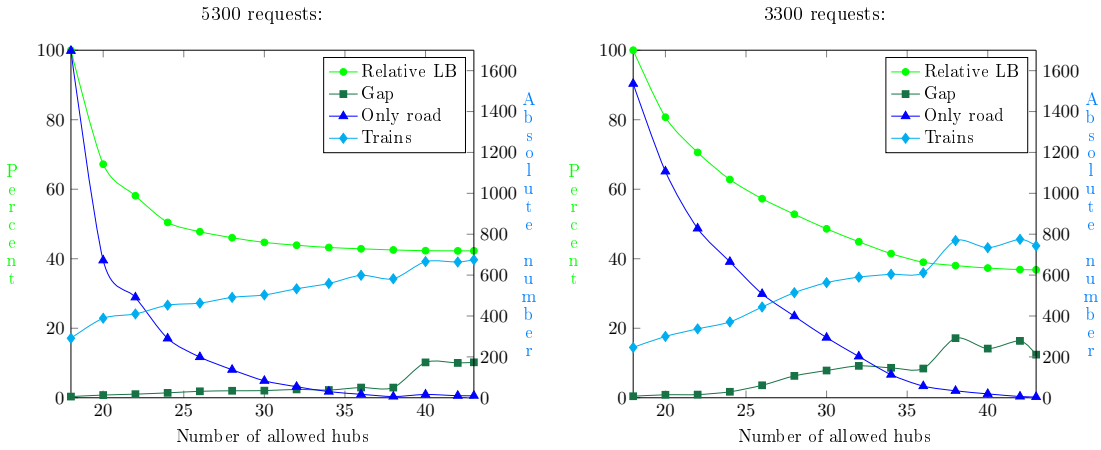
- Naturally, with an increasing number of requests, the problem becomes harder, and fewer instances were solved to optimality. The same happened with growing numbers of hubs. This is because more hubs allow disproportionately more connections and thus require a higher number of train variables.
- A small number of potential hubs permitted the solution of much more branch-and-bound nodes, as both the pricing algorithm and the reoptimization consumed less time. On the other hand, the number of paths found was smaller, and much more requests were transported entirely by road in solutions to such instances.
- The difference between settings with splittable and unsplittable requests was insignificant. Both the solution times and the solution values were very similar. This is probably caused by the fact that there are only few requests with a large number of containers relative to the train capacity (there are only 38 requests overall with a number of containers exceeding 25 % of the train capacity). Actually, many instances that could be solved to optimality even had the same solution. For the large instances, however, the solution structure was different. On average, about 150 fractional path variables occurred in the best found feasible solutions. Among those, approximately 100 paths represented the transport of a fractional number of containers, which is not practicable, i.e., the path variable is no multiple of $\frac{1}{m^r}$.
- Finally, we remark that branching on hub variables is decisive, even for unsplittable requests for which the hub variable integrality is a consequence of the feasibility of the path variables. Without this, results were much worse: fewer instances could be solved to optimality, and the average gap increased by more than 40 %.

6.5.4 Sensitivity analysis

We also tested some variations of the instance parameters for the two large instances:

- Regarding the maximal tour duration, we tried both instances with twice as well as half the original time values. The results showed that the standard values are not very restrictive, because the enlarged durations did not lead to significantly different solutions. On the other hand, the shortened maximal tour durations caused many additional requests to be transported by road (about 50 % more), and also the costs increased.
- Focussing on the cost relations between truck and train, it is remarkable that one kilometre on the rail network costs less than the twentieth part of a road kilometre for one container, assuming the train is used to its capacity. Therefore, modifying the cost for road transport does not change the solution structure significantly. Even when solving the instances with a road cost parameter of only 10 % of the original value, the number of requests being transported completely by trucks increases by a mere 10 % on average. The problem becomes more difficult, though, as the decision between the modes of transport is less clear, and we observe a larger gap.
- The two diagrams in Figure 6.4 visualize how different solution characteristics depend on the number of allowed hubs. This number is displayed along the x-axes. The

Figure 6.4: Solution characteristics for the two large instances with different numbers of hubs



algorithm was run on both large instances with the best setting with unsplittable requests and 18, 20, 22, ..., 40, 42, and 43 allowed hubs. The bright green lines (–●–) labelled 'Relative LB' indicate the development of the lower bound at timeout, where 100 % (left y-axes) correspond to the lower bound obtained when only the 18 fixed hubs may be used. As can be seen, the lower bound decreases steadily with increasing number of allowed hubs. When all 43 hubs may be used, the lower bound is only about 40 % of the one for 18 hubs. Hence, the number of allowed hubs has a significant impact on the total transportation costs. The dark green lines (–■–) labelled 'Gap' depict the development of the remaining percentage gap at timeout (left y-axes). The gap increases steadily with increasing number of hubs, indicating that the instances become more difficult to solve. This is not surprising, given that the solution space regarding the train decisions grows exponentially with the number of allowed hubs. From these lines, it can also be seen that the hub location aspect does not strongly affect the difficulty of the problem: when no hub decision remains (because all 43 hubs may be used), the gap is at least as high as in scenarios where a proper subset of the hubs must be selected. The blue lines (–▲–) labelled 'Only road' indicate the number of requests transported only by road (right y-axes). Clearly, this number decreases strongly with increasing number of hubs, because more hubs make the use of trains cheaper and faster. Correspondingly, the cyan lines (–◆–) labelled 'Trains' show the number of trains running (right y-axes). As was to be expected, this number increases with increasing number of hubs, albeit only moderately. Finally (this can not be seen in the figure), we observed that the sets of selected hubs are quite stable, as in most cases, allowing one more hub causes a superset of the former hub set to be chosen.

6.6 Conclusions

We have studied a specific variant of an integrated tactical service network design and hub location problem for combined road-rail transport, considering several practically relevant constraints. We have developed a branch-and-price-and-cut algorithm using different pricing algorithms, tailored branching strategies, and problem-specific cutting planes. Computational experiments with instances using real-world data show that our path-based algorithm clearly and consistently outperforms a commercial solver using an arc-based model. Our algorithm is capable of finding close-to-optimal solutions in less than one hour of computation time, even for instances of realistic size, and can thus be applied for practical planning.

Further research could be done with respect to model extensions and with respect to algorithmic enhancements. As for model extensions, balancing constraints for locomotives and wagons could be added, as well as capacities at hubs and fixed costs for hub usage and train services. Focusing on a model and solution approach for solving a concrete real-world problem, we have abstained from doing so. However, in different application contexts, these aspects may be relevant, and they can rather easily be integrated into our model.

With regard to algorithmics, our procedure could be improved by developing a primal heuristic that runs faster than the current approach of solving an extended set-covering problem with existing columns using a MIP solver. Furthermore, a method for which successful applications to network design and hub location problems are reported in the literature is Benders decomposition. Therefore, applying this technique to the problem described in this paper constitutes an interesting research avenue.

Acknowledgments

We thank Hanno Schülldorf from Deutsche Bahn Mobility Logistics AG for providing us the realistic network and request data.

During earlier work on the problem described in this paper, M. Drexl was partially funded by the German Federal Ministry of Education and Research (BMBF) within the project 'E-Motion' under grant no. 05M13ANA. He thanks the BMBF for the opportunity to participate in the project E-Motion.

Bibliography

Alumur, S. and Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, **190**(1), 1–21.

Alumur, S. A., Kara, B. Y., and Karasan, O. E. (2012). Multimodal hub location and hub network design. *Omega*, **40**(6), 927–939.

Andersen, J., Crainic, T. G., and Christiansen, M. (2009). Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, **193**(2), 377–389.

Aneja, Y. P., Aggarwal, V., and Nair, K. P. K. (1983). Shortest chain subject to side constraints. *Networks*, **13**(2), 295–302.

Assad, A. A. (1980). Models for rail transportation. *Transportation Research Part A*, **14A**, 205–220.

Atamtürk, A. and Rajan, D. (2002). On splittable and unsplittable flow capacitated network design arc-set polyhedra. *Mathematical Programming*, **92**(2), 315–333.

Barahona, F. (1996). Network design using cut inequalities. *SIAM Journal on Optimization*, **6**(3), 823–837.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1996). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**, 316–329.

Bauer, J., Bektaş, T., and Crainic, T. G. (2010). Minimizing greenhouse gas emissions in intermodal freight transport: An application to rail service design. *Journal of the Operational Research Society*, **61**(3), 530–542.

Bontekoning, Y., Macharis, C., and Trip, J. (2004). Is a new applied transportation research field emerging?–A review of intermodal rail-truck freight transport literature. *Transportation Research Part A*, **38**, 1–34.

Brockmüller, B., Günlük, O., and Wolsey, L. (1996). Designing private line networks – polyhedral analysis and computation. Technical report, Université Catholique de Louvain, Belgium.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, **72**, 387–405.

Campbell, J. F. (2009). Hub location for time definite transportation. *Computers & Operations Research*, **36**(12), 3107–3116.

Caris, A., Macharis, C., and Janssens, G. K. (2013). Decision support in intermodal transport: A new research agenda. *Computers in Industry*, **64**(2), 105–112.

Chouman, M., Crainic, T. G., and Gendron, B. (2003). A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design. Technical Report CIRRELT-2003-16, Centre de la recherche sur les transports.

Chouman, M., Crainic, T. G., and Gendron, B. (2011). Commodity representations and cutset-based inequalities for multicommodity capacitated fixed-charge network design. Technical Report CIRRELT-2011-56, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport.

Contreras, I. and Fernández, E. (2012). General network design: A unified view of combined location and network design problems. *European Journal of Operational Research*, **219**(3), 680–697.

Council of the European Communities (1992). *Council Directive 92/106/EEC of 7 December 1992 on the establishment of common rules for certain types of combined transport of goods between Member States.* Available at http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:01992L0106-20130701&from=EN. Accessed 4th February 2016.

Craig, A., Blanco, E., and Sheffi, Y. (2013). Estimating the CO_2 intensity of intermodal freight transportation. *Transportation Research Part D*, **22**, 49–53.

Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research*, **122**(2), 272–288.

Crainic, T. G. (2003). Long-haul freight transportation. In R. W. Hall, editor, *Handbook of Transportation Science*, pages 451–516. Springer, Boston, MA.

Crainic, T. G. and Laporte, G. (1997). Planning models for freight transportation. *European Journal of Operational Research*, **97**, 409–438.

Crainic, T. G., Kim, K. H., *et al.* (2006). Intermodal transportation. *Transportation*, **14**, 467–537.

Crainic, T. G., Hewitt, M., Toulouse, M., and Vu, D. M. (2015). Location and service network design. Technical Report CIRRELT-2015-45, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport.

Dekker, R., Bloemhof, J., and Mallidis, I. (2012). Operations research for green logistics – an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*, **219**(3), 671–679.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York.

Economic Commission for Europe (2001). *Terminology on Combined Transport*. Available at http://www.unece.org/fileadmin/DAM/trans/wp24/documents/term.pdf. Accessed 4th February 2016.

European Commission (2014). *EU Transport in Figures – Statistical Pocketbook 2014*. Available at http://ec.europa.eu/transport/facts-fundings/statistics/doc/2014/pocketbook2014.pdf. Accessed 4th February 2016.

Homfeld, H. (2012). *Consolidating Car Routes in Rail Freight Service by Discrete Optimization*. Ph.D. thesis, Technische Universität Darmstadt.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York.

Ishfaq, R. and Sox, C. R. (2011). Hub location-allocation in intermodal logistic networks. *European Journal of Operational Research*, **210**(2), 213–230.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Magnanti, T., Mirchandani, P., and Vachani, R. (1993). The convex hull of two core capacitated network design problems. *Mathematical Programming*, **60**(1-3), 233–250.

O'Kelly, M. E. (1986). The location of interacting hub facilities. *Transportation Science*, **20**, 92–105.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

SteadieSeifi, M., Dellaert, N., Nuijten, W., Woensel, T. V., and Raoufi, R. (2014). Multimodal freight transportation planning: A literature review. *European Journal of Operational Research*, **233**(1), 1–15.

Üster, H. and Agrahari, H. (2011). A benders decomposition approach for a distribution network design problem with consolidation and capacity considerations. *Operations Research Letters*, **39**(2), 138–143.

Wieberneit, N. (2008). Service network design for freight transportation: A review. *OR Spectrum*, **30**(1), 77–112.

Yaghini, M. and Akhavan, R. (2012). Multicommodity network design problem in rail freight transportation planning. *Procedia – Social and Behavioral Sciences*, **43**(0), 728–739.

Yoon, M.-G. and Current, J. (2008). The hub location and network design problem with fixed and variable arc costs: Formulation and dual-based solution heuristic. *The Journal of the Operational Research Society*, **59**(1), pp. 80–89.

Zhang, M., Wiegmans, B., and Tavasszy, L. (2013). Optimization of multimodal networks including environmental costs: A model and findings for transport policy. *Computers in Industry*, **64**(2), 136–145.

Zhu, X. and Wilhelm, W. E. (2013). Implementation of a three-stage approach for the dynamic resource-constrained shortest-path sub-problem in branch-and-price. *Computers & Operations Research*, **40**(1), 385–394.

Appendix: Arc-based models

Here, we describe the arc-based models for the unsplit and the split request variants of our SNDHLP. We need the following additional notation: let $d^r_{\max}$ be the maximal allowed transport time, and let A^r_{start} and A^r_{end} be the possible street arcs connecting the origin and destination of request r to the hub network. Then, the problem proposed in this paper can be formulated with an arc-based approach as follows:

6.A Arc-based model with unsplittable requests

The following variables are used:

- $h_i \in \{0,1\}$ indicates whether or not hub $i \in H$ is used
- $t_a \in \mathbb{N}_0$ counts the number of trains running on arc $a \in A_t$
- $x^r_a \in \{0,1\}$ decides whether or not request $r \in R$ uses arc $a \in A$

The model is then:

$$\min \quad \sum_{a\in A_t} c_t l_a t_a + \sum_{r\in R}\sum_{a\in A_s} c_s m^r l_a x^r_a \tag{6.8a}$$

$$\text{s.t.} \quad h_i = 1 \qquad \forall\, i \in H_f \tag{6.8b}$$

$$\sum_{i\in H} h_i = n_H \tag{6.8c}$$

$$\sum_{a\in A^r_{\text{start}}} x^r_a = 1 \qquad \forall\, r \in R \tag{6.8d}$$

$$\sum_{a\in A^r_{\text{end}}} x^r_a = 1 \qquad \forall\, r \in R \tag{6.8e}$$

$$\sum_{a\in A^r\cap\delta^-(i)} x^r_a - \sum_{a\in A^r\cap\delta^+(i)} x^r_a = 0 \qquad \forall\, r \in R, i \in H \tag{6.8f}$$

$$\sum_{r\in R} m^r x^r_a \le k_t t_a \qquad \forall\, a \in A_t \tag{6.8g}$$

$$\sum_{a=(i,j)\in A^r} (d_a + d_j) x^r_a \le d^r_{\max} \qquad \forall\, r \in R \tag{6.8h}$$

$$\sum_{a\in A_t} x^r_a \le u_{\max} - 1 \qquad \forall\, r \in R \tag{6.8i}$$

$$x^r_a \le h_i \qquad \forall\, r \in R, a = (i,j) \in A_t \tag{6.8j}$$

$$x^r_a \le h_j \qquad \forall\, r \in R, a = (i,j) \in A_t \tag{6.8k}$$

$$h_i \in \{0,1\} \qquad \forall\, i \in H \tag{6.8l}$$

$$t_a \in \mathbb{N}_0 \qquad \forall\, a \in A_t \tag{6.8m}$$

$$x^r_a \in \{0,1\} \qquad \forall\, r \in R, a \in A^r \tag{6.8n}$$

The minimization objective (6.8a) sums the costs for the covered kilometres. Constraints (6.8b) and (6.8c) respectively ensure that the fixed hubs are opened and that the correct number of hubs is available. The next three constraints ensure flow conservation. The train capacities are maintained by constraints (6.8g). Constraints (6.8h) and (6.8i) limit the total transport time and the number of transshipments for each request. Constraints (6.8j) and (6.8k) ensure that a hub is open if a request traverses it, and the last three constraints describe the domains of the variables.

6.B Arc-based model with splittable requests

The following variables are used:

- $h_i \in \{0,1\}$ indicates whether or not hub $i \in H$ is used
- $t_a \in \mathbb{N}_0$ counts the number of trains running on arc $a \in A_t$
- $x_a^r \in [0,1]$ describes the percentage of request $r \in R$ transported over arc $a \in A$
- $u_a^r \in \{0,1\}$ indicates whether or not a part of request $r \in R$ uses arc $a \in A$
- $m_i^r \in \mathbb{N}_0$ measures the time en route of a part of request $r \in R$ upon reaching hub $i \in H$
- $n_i^r \in \mathbb{N}_0$ counts the number of transshipments that a part of request $r \in R$ has performed up to hub $i \in H$

The model is then:

$$\min \quad \sum_{a \in A_t} c_t l_a t_a + \sum_{r \in R} \sum_{a \in A_s} c_s m^r l_a x_a^r \tag{6.9a}$$

$$\text{s.t.} \quad h_i = 1 \qquad \forall\, i \in H_f \tag{6.9b}$$

$$\sum_{i \in H} h_i = n_H \tag{6.9c}$$

$$\sum_{a \in A^r_{\text{start}}} x_a^r = 1 \qquad \forall\, r \in R \tag{6.9d}$$

$$\sum_{a \in A^r_{\text{end}}} x_a^r = 1 \qquad \forall\, r \in R \tag{6.9e}$$

$$\sum_{a \in A^r \cap \delta^-(i)} x_a^r - \sum_{a \in A^r \cap \delta^+(i)} x_a^r = 0 \qquad \forall\, r \in R, i \in H \tag{6.9f}$$

$$\sum_{r \in R} m^r x_a^r \le k_t t_a \qquad \forall\, a \in A_t \tag{6.9g}$$

$$x_a^r \le u_a^r \qquad \forall\, r \in R, a \in A_t \tag{6.9h}$$

$$(d_a + d_j) u_a^r \le m_j^r \qquad \forall\, r \in R, a = (i,j) \in A^r_{\text{start}} \tag{6.9i}$$

$$m_i^r + (d_a + d_j)u_a^r \leq m_j^r + d_{\max}^r(1 - u_a^r) \qquad \forall\, r \in R, a = (i,j) \in A_t \tag{6.9j}$$

$$m_i^r + d_a u_a^r \leq d_{\max}^r \qquad \forall\, r \in R, a = (i,j) \in A_{\text{end}}^r \tag{6.9k}$$

$$u_a^r \leq n_j^r \qquad \forall\, r \in R, a = (i,j) \in A_{\text{start}}^r \tag{6.9l}$$

$$n_i^r + u_a^r \leq n_j^r + u_{\max}(1 - u_a^r) \qquad \forall\, r \in R, a = (i,j) \in A_t \tag{6.9m}$$

$$n_i^r \leq u_{\max} \qquad \forall\, r \in R, a = (i,j) \in A_{\text{end}}^r \tag{6.9n}$$

$$u_a^r \leq h_i \qquad \forall\, r \in R, a = (i,j) \in A_t \tag{6.9o}$$

$$u_a^r \leq h_j \qquad \forall\, r \in R, a = (i,j) \in A_t \tag{6.9p}$$

$$h_i \in \{0,1\} \qquad \forall\, i \in H \tag{6.9q}$$

$$t_a \in \mathbb{N}_0 \qquad \forall\, a \in A_t \tag{6.9r}$$

$$x_a^r \in [0,1] \qquad \forall\, r \in R, a \in A^r \tag{6.9s}$$

$$u_a^r \in \{0,1\} \qquad \forall\, r \in R, a \in A^r \tag{6.9t}$$

$$m_i^r \in \mathbb{N}_0 \qquad \forall\, r \in R, i \in H \tag{6.9u}$$

$$n_i^r \in \mathbb{N}_0 \qquad \forall\, r \in R, i \in H \tag{6.9v}$$

Again, the objective is to minimize the total costs for the covered kilometres. The next constraints, (6.9b)–(6.9g), are also the same as above. Constraint (6.9h) determines the correct values for the binary flow variables on the basis of the fractional flow variables. The following three constraints ensure that the time en route of a request upon reaching a certain node is set correctly, and that the time at the destination does not exceed the allowed maximal time en route. This requirement is split into the first road leg (6.9i), the rail legs in between (6.9j), and the last road leg (6.9k). The same is done for the number of transshipments per request on each hub in the following three constraints, (6.9l)–(6.9n). Then, the relation between the hub and path variables is expressed as in the unsplittable case above by constraints (6.9o) and (6.9p). The last six constraints regulate the domains of the variables.

Chapter 7

Conclusions

The goal of this thesis was to contribute to finding optimal solutions of routing problems. Branch-and-price-and-cut algorithms became the method of choice, because of their former successes on these types of problems (Desrosiers and Lübbecke, 2011). Thereby, existing procedures should be improved and new problem variants should become solvable.

New improvements on existing procedures were introduced in Chapters 2 and 3. The ideas described there are applicable to all variants of the corresponding problems.

In Chapter 2, the dynamic half-way point selection in bidirectional labeling algorithms for the solution of ESPPRCs provides an acceleration of the standard solution procedure (Irnich and Desaulniers, 2005). It provides a better balance between the workload in forward and backward directions and can thus reduce the overall number of generated labels. Extensive tests on several vehicle routing problem variants could show significant reductions of the overall runtimes of full-fledged branch-and-price-and-cut algorithms when the dynamic half-way point was used in the subproblem. Moreover, observations indicate that the effectiveness of the dynamic half-way point increases with the difficulty of the ESPPRC instance. The applicability of this new half-way point selection benefits from a very small implementation effort on top of a bidirectional labeling.

Chapter 3 enables the usage of bidirectional labeling algorithms for ESPPRCs with a pickup-and-delivery structure. While several works have shown the advantages of bidirectional labeling compared to monodirectional labeling (Righini and Salani, 2006), no efficient way to apply it for pickup-and-delivery problems was known. The new idea to use different cost matrices in forward and backward labeling, leading to the usability of a strong dominance relation in both directions, allowed for an efficient bidirectional labeling. The computational study showed run time savings around 40% by using the bidirectional labeling with the dynamic half-way point instead of a monodirectional labeling. Beyond the basic PDPTW, the proposed method can be applied to more challenging problems with a pickup-and-delivery structure, such as the dial-a-ride problem or PDPs with loading constraints.

New and challenging problem variants were tackled in Chapters 4, 5, and 6. The extensions are motivated by real world requirements and implied adaptations to several parts of the branch-and-price-and-cut algorithm. Thereby, the proposed algorithms always applied the state-of-the-art techniques for accelerations.

In Chapter 4, an efficient branch-and-price-and-cut algorithm for the TTRPTW was presented, which could solve several benchmark instances to optimality for the first time.

Moreover, the extension of the planning horizon from one to two days was considered for this problem. The arising symmetry could successfully be tackled through a stabilization procedure. The second extension was the incorporation of a quantity-dependent time for the transfer of load from the truck to the trailer. Thereby, a tradeoff between the consumed time and the free space in the truck occurs, which implied more resources and a more complex propagation in the labeling algorithm for the subproblem. The results regarding the extensions demonstrated their influences on the overall routing costs and showed that the solvability does not change excessively despite the more complicated problems.

Chapter 5 proposed the first exact algorithm for the classical PVRPTW. In addition, the formerly strict requirements for the visiting patterns of the customers were relaxed, such that more real-world situations are treatable. To deal with this extension, new structures for the pricing networks were introduced as basis for a bidirectional labeling algorithm. Moreover, a PVRP-specific symmetry was defined and methods to deal with this symmetry were presented. In particular, constraint aggregation reduces significantly the computational effort without changing the linear relaxation. In the computational tests, some benchmark instances were solved for the first time and savings through more flexible visiting patterns were proven.

In Chapter 6, a real-world network design and hub location problem in context of combined road-rail transport was tackled with a branch-and-price-and-cut algorithm. The combination of problem aspects, such as the task to determine the arc frequencies and the limitation of the number of transshipment, was not handled before. The solution approach includes different pricing algorithms, tailored branching strategies, and problem-specific cuts. Real-world instances could be solved close to optimality within one hour. Furthermore, a sensitivity analysis evaluated the influences of different parameters.

Altogether, all three parts of a branch-and-price-and-cut algorithm have a strong influence on the run time. The solution of the pricing problem to generate new routes is usually by far the most time consuming part of the algorithm. Thus, the development of a good pricing network and the application of an efficient solution method like a labeling algorithm are very important. Using techniques for accelerating the solution of the pricing problem such as the ng-path relaxation and a bidirectional labeling is indispensable for obtaining good results. At the same time, strong cuts and appropriate branching rules can significantly reduce the number of branch-and-bound nodes. As the fastest pricing problem solver does not help when the branch-and-bound tree is huge, these two parts should not be neglected.

Bibliography

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Alumur, S. and Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, **190**(1), 1–21.

Alumur, S. A., Kara, B. Y., and Karasan, O. E. (2012). Multimodal hub location and hub network design. *Omega*, **40**(6), 927–939.

Andersen, J., Crainic, T. G., and Christiansen, M. (2009). Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, **193**(2), 377–389.

Aneja, Y. P., Aggarwal, V., and Nair, K. P. K. (1983). Shortest chain subject to side constraints. *Networks*, **13**(2), 295–302.

Archetti, C., Jabali, O., and Speranza, M. G. (2015). Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, **61**, 122–134.

Archetti, C., Fernández, E., and Huerta-Muñoz, D. L. (2017). The flexible periodic vehicle routing problem. *Computers & Operations Research*, **85**, 58–70.

Assad, A. A. (1980). Models for rail transportation. *Transportation Research Part A*, **14A**, 205–220.

Atamtürk, A. and Rajan, D. (2002). On splittable and unsplittable flow capacitated network design arc-set polyhedra. *Mathematical Programming*, **92**(2), 315–333.

Athanasopoulos, T. and Minis, I. (2013). Efficient techniques for the multi-period vehicle routing problem with time windows within a branch and price framework. *Annals of Operations Research*, **206**(1), 1–22.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011a). Dynamic ng-path relaxation. ROUTE 2011.

Baldacci, R., Bartolini, E., Mingozzi, A., and Valletta, A. (2011b). An exact algorithm for the period routing problem. *Operations Research*, **59**(1), 228–241.

Baldacci, R., Bartolini, E., and Mingozzi, A. (2011c). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, **59**(2), 414–426.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011d). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012a). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, **24**(3), 356–371.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012b). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, **218**, 1–6.

Barahona, F. (1996). Network design using cut inequalities. *SIAM Journal on Optimization*, **6**(3), 823–837.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1996). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**, 316–329.

Batsyn, M. and Ponomarenko, A. (2014). Heuristic for a real-life truck and trailer routing problem. *Procedia Computer Science*, **31**, 778–792.

Battarra, M., Cordeau, J.-F., and Iori, M. (2014). Pickup-and-delivery problems for goods transportation. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 6, pages 161–191. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Bauer, J., Bektaş, T., and Crainic, T. G. (2010). Minimizing greenhouse gas emissions in intermodal freight transport: An application to rail service design. *Journal of the Operational Research Society*, **61**(3), 530–542.

Belenguer, J., Benavent, E., Martínez, A., Prins, C., Prodhon, C., and Villegas, J. (2016). A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots. *Transportation Science*, **50**(2), 735–749.

Beltrami, E. J. and Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, **4**(1), 65–94.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, **23**(4), 214–229.

Bertazzi, L., Savelsbergh, M., and Speranza, M. G. (2008). Inventory routing. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 49–72. Springer US, Boston, MA.

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, **60**(5), 1167–1182.

Bode, C. and Irnich, S. (2014). The shortest-path problem with resource constraints with (k,2)-loop elimination and its application to the capacitated arc-routing problem. *European Journal of Operational Research*, **238**(2), 415–426.

Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, **49**(2), 369–383.

Bodin, L. and Levy, L. (2000). Scheduling of local delivery carrier routes for the united states postal service. In M. Dror, editor, *Arc Routing: Theory, Solutions, and Applications*, pages 419–442. Springer, Boston, MA.

Boland, N., Dethridge, J., and Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, **34**(1), 58–68.

Bontekoning, Y., Macharis, C., and Trip, J. (2004). Is a new applied transportation research field emerging?–A review of intermodal rail-truck freight transport literature. *Transportation Research Part A*, **38**, 1–34.

Bostel, N., Dejax, P., Guez, P., and Tricoire, F. (2008). Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 503–525. Springer US, Boston, MA.

Brockmüller, B., Günlük, O., and Wolsey, L. (1996). Designing private line networks – polyhedral analysis and computation. Technical report, Université Catholique de Louvain, Belgium.

Brønmo, G., Christiansen, M., and Nygreen, B. (2007). Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society*, **58**(9), 1167–1177.

Cacchiani, V., Hemmelmayr, V., and Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, **163**, 53–64. Matheuristics 2010.

Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, **63**(1), 2–15.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, **72**, 387–405.

Campbell, J. F. (2009). Hub location for time definite transportation. *Computers & Operations Research*, **36**(12), 3107–3116.

Caramia, M. and Guerriero, F. (2010a). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, **61**(7), 1168–1180.

Caramia, M. and Guerriero, F. (2010b). A milk collection problem with incompatibility constraints. *Interfaces*, **40**(2), 130–143.

Caris, A., Macharis, C., and Janssens, G. K. (2013). Decision support in intermodal transport: A new research agenda. *Computers in Industry*, **64**(2), 105–112.

Chao, I. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, **29**(1), 33–51.

Cherkesly, M., Desaulniers, G., and Laporte, G. (2015). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, **49**(4), 752–766.

Cherkesly, M., Desaulniers, G., Irnich, S., and Laporte, G. (2016). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. *European Journal of Operational Research*, **250**(3), 782–793.

Chouman, M., Crainic, T. G., and Gendron, B. (2003). A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design. Technical Report CIRRELT-2003-16, Centre de la recherche sur les transports.

Chouman, M., Crainic, T. G., and Gendron, B. (2017). Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, **51**(2), 650–667.

Christofides, N. and Beasley, J. E. (1984). The period routing problem. *Networks*, **14**(2), 237–256.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, **4**(4), 305 – 337.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, **48**(1), 1–19.

Contardo, C., Desaulniers, G., and Lessard, F. (2015). Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks*, **65**(1), 88–99.

Contreras, I. and Fernández, E. (2012). General network design: A unified view of combined location and network design problems. *European Journal of Operational Research*, **219**(3), 680–697.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *The Journal of the Operational Research Society*, **52**(8), 928–936.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M., and Soumis, F. (2002). VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 7, pages 155–194. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Cordeau, J. F., Laporte, G., and Mercier, A. (2004). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *The Journal of the Operational Research Society*, **55**(5), 542–546.

Council of the European Communities (1992). *Council Directive 92/106/EEC of 7 December 1992 on the establishment of common rules for certain types of combined transport of goods between Member States.* Available at http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:01992L0106-20130701&from=EN. Accessed 4th February 2016.

Craig, A., Blanco, E., and Sheffi, Y. (2013). Estimating the CO_2 intensity of intermodal freight transportation. *Transportation Research Part D*, **22**, 49–53.

Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research*, **122**(2), 272–288.

Crainic, T. G. (2003). Long-haul freight transportation. In R. W. Hall, editor, *Handbook of Transportation Science*, pages 451–516. Springer, Boston, MA.

Crainic, T. G. and Laporte, G. (1997). Planning models for freight transportation. *European Journal of Operational Research*, **97**, 409–438.

Crainic, T. G., Kim, K. H., *et al.* (2006). Intermodal transportation. *Transportation*, **14**, 467–537.

Crainic, T. G., Hewitt, M., Toulouse, M., and Vu, D. M. (2015). Location and service network design. Technical Report CIRRELT-2015-45, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport.

Cuda, R., Guastaroba, G., and Speranza, M. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, **55**, 185–199.

Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, **8**(3), 250–255.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, **8**(1), 101–111.

Dekker, R., Bloemhof, J., and Mallidis, I. (2012). Operations research for green logistics – an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*, **219**(3), 671–679.

Derigs, U., Pullmann, M., and Vogel, U. (2013). Truck and trailer routing—Problems, heuristics and computational experience. *Computers & Operations Research*, **40**(2), 536–546.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, **58**(1), 179–192.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57–93. Springer, Boston, MA.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York.

Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, **42**(3), 387–404.

Desaulniers, G., Desrosiers, J., and Spoorendonk, S. (2010). The vehicle routing problem with time windows: State-of-the-art exact solution methods. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*, volume 8, pages 5742–5749. John Wiley & Sons, Inc.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). The vehicle routing problem with time windows. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016a). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, **64**(6), 1388–1405.

Desaulniers, G., Contardo, C., and Pecin, D. (2016b). Selective pricing in branch-price-and-cut algorithms for vehicle routing. Les Cahiers du GERAD G-2016-110, GERAD, Montréal, Canada.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, **40**(2), 342–354.

Desrosiers, J. and Lübbecke, M. E. (2011). Branch-price-and-cut algorithms. In *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**(1), 269–271.

Doerner, K. F. and Salazar-González, J.-J. (2014). Pickup-and-delivery problems for people transportation. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 7, pages 193–212. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Drexl, M. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Journal of Quantitative Methods for Economics and Business Administration*, **12**, 5–38.

Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research*, **5**(1), 47–63.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, **42**(5), 977–978.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pick-up and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7–22.

Economic Commission for Europe (2001). *Terminology on Combined Transport*. Available at http://www.unece.org/fileadmin/DAM/trans/wp24/documents/term.pdf. Accessed 4th February 2016.

European Commission (2014). *EU Transport in Figures – Statistical Pocketbook 2014*. Available at http://ec.europa.eu/transport/facts-fundings/statistics/doc/2014/pocketbook2014.pdf. Accessed 4th February 2016.

Feillet, D., Dejax, P., Gendreau, M., and Guéguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, **44**(3), 216–229.

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, **45**(4), 239–256.

Francis, P., Smilowitz, K., and Tzur, M. (2006). The period vehicle routing problem with service choice. *Transportation Science*, **40**(4), 439–454.

Francis, P., Smilowitz, K., and Tzur, M. (2007). Flexibility and complexity in periodic distribution problems. *Naval Research Logistics (NRL)*, **54**(2), 136–150.

Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer US, Boston, MA.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large scale aircrew rostering problems. *Operations Research*, **47**, 247–262.

Garcia, R. (2009). *Resource Constrained Shortest Paths and Extensions*. Ph.D. thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, U.S.A.

Gaudioso, M. and Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, **26**(2), 86–92.

Gehring, H. and Homberger, J. (2002). Parallelization of a Two-Phase metaheuristic for routing problems with time windows. *Journal of Heuristics*, **8**(3), 251–276.

Gerdessen, J. (1996). Vehicle routing problem with trailers. *European Journal of Operational Research*, **93**(1), 135–147.

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, **43**(1), 17–26.

Goel, A. and Irnich, S. (2017). An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*, **51**(2), 737–754.

Gomory, R. E. (1963). An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill Book Company.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, **195**(3), 791–802.

Hennig, F., Nygreen, B., and Lübbecke, M. (2012). Nested column generation applied to the crude oil tanker routing and scheduling problem with split pickup and split delivery. *Naval Research Logistics*, **59**(3–4), 298–310.

Homfeld, H. (2012). *Consolidating Car Routes in Rail Freight Service by Discrete Optimization*. Ph.D. thesis, Technische Universität Darmstadt.

Horváth, M. and Kis, T. (2016). Solving resource constrained shortest path problems with LP-based methods. *Computers & Operations Research*, **73**, 150–164.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York.

Irnich, S. and Villeneuve, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, **18**(3), 391–406.

Irnich, S., Toth, P., and Vigo, D. (2014a). The family of vehicle routing problems. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 1, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Irnich, S., Schneider, M., and Vigo, D. (2014b). Four variants of the vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing*, chapter 9, pages 241–271. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Ishfaq, R. and Sox, C. R. (2011). Hub location-allocation in intermodal logistic networks. *European Journal of Operational Research*, **210**(2), 213–230.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, **33**(1), 101–116.

Li, H. and Lim, A. (2003). A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, **12**(2), 173–186.

Lin, S., Yu, V., and Lu, C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, **38**(12), 15244–15252.

Lozano, L. and Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. *Computers & Operations Research*, **40**(1), 378–384.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Magnanti, T., Mirchandani, P., and Vachani, R. (1993). The convex hull of two core capacitated network design problems. *Mathematical Programming*, **60**(1-3), 233–250.

Mirmohammadsadeghi, S. and Ahmed, S. (2015). Memetic heuristic approach for solving truck and trailer routing problems with stochastic demands and time windows. *Networks and Spatial Economics*, **15**(4), 1093–1115.

Naddef, D. and Rinaldi, G. (2002). Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 3, pages 53–84. Society for Industrial and Applied Mathematics, Philadelphia, PA.

O'Kelly, M. E. (1986). The location of interacting hub facilities. *Transportation Science*, **20**, 92–105.

Parragh, S., Doerner, K., and Hartl, R. (2008). A survey on pickup and delivery problems, Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, **58**(2), 81–117.

Parragh, S. N. and Cordeau, J.-F. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, **83**, 28–44.

Pasha, U., Hoff, A., and Løkketangen, A. (2014). A hybrid approach for milk collection using trucks and trailers. *Annals of Management Science*, **3**(1), 87–110.

Pecin, D. (2014). *Exact Algorithms for the Capacitated Vehicle Routing Problem*. Ph.D. thesis, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, **9**(1), 61–100.

Pirkwieser, S. and Raidl, G. R. (2009a). Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ILP techniques. In S. Voss and M. Caserta, editors, *Proceedings of the 8th Metaheuristic International Conference*.

Pirkwieser, S. and Raidl, G. R. (2009b). A column generation approach for the periodic vehicle routing problem with time windows. In M. S. et al, editor, *Proceedings of the International Network Optimization Conferencee*.

Poggi, M. and Uchoa, E. (2014). New exact algorithms for the capacitated vehicle routing problem. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 3, pages 59–86. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Pugliese, L. D. P. and Guerriero, F. (2013). A reference point approach for the resource constrained shortest path problems. *Transportation Science*, **47**(2), 247–265.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, **51**(3), 155–170.

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, **43**(3), 267–286.

Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, **9**(1), 1–17.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, **33**(4), 894–909.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, **48**(4), 500–520.

Schrijver, A. (2003). *Combinatorial optimization: polyhedra and efficiency*. Springer, Berlin/Heidelberg.

Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, **41**(4), 469–488.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**(2), 254–265.

SteadieSeifi, M., Dellaert, N., Nuijten, W., Woensel, T. V., and Raoufi, R. (2014). Multimodal freight transportation planning: A literature review. *European Journal of Operational Research*, **233**(1), 1–15.

Tan, K., Chew, Y., and Lee, L. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, **172**(3), 855–885.

Tilk, C. (2016). Branch-and-price-and-cut for the vehicle routing and truck driver scheduling problem. Technical Report LM-2016-04, Chair of Logistics Management, Johannes Gutenberg University, Mainz, Germany.

Tilk, C. and Irnich, S. (2017). Dynamic programming for the minimum tour duration problem. *Transportation Science*, **51**(2), 549–565.

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., and Meisel, F. (2016). Branch-and-price for the active-passive vehicle-routing problem. *Transportation Science*. Forthcoming.

Torres, I., Cruz, C., and Verdegay, J. L. (2015). Solving the truck and trailer routing problem with fuzzy constraints. *International Journal of Computational Intelligence Systems*, **8**(4), 713–724.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Tricoire, F. (2016). Private communication.

Üster, H. and Agrahari, H. (2011). A benders decomposition approach for a distribution network design problem with consolidation and capacity considerations. *Operations Research Letters*, **39**(2), 138–143.

Veenstra, M., Cherkesly, M., Desaulniers, G., and Laporte, G. (2017). The pickup and delivery problem with time windows and handling operations. *Computers & Operations Research*, **77**, 127–140.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, **40**(1), 475–489.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., and Velasco, N. (2010). GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, **23**(5), 780–794.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., and Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, **38**(9), 1319–1334.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., and Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, **230**(2), 231–244.

Villeneuve, D. and Desaulniers, G. (2005). The shortest path problem with forbidden paths. *European Journal of Operational Research*, **165**(1), 97–107.

Wen, M., Cordeau, J.-F., Laporte, G., and Larsen, J. (2010). The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, **37**(9), 1615–1623.

Wieberneit, N. (2008). Service network design for freight transportation: A review. *OR Spectrum*, **30**(1), 77–112.

Yaghini, M. and Akhavan, R. (2012). Multicommodity network design problem in rail freight transportation planning. *Procedia – Social and Behavioral Sciences*, **43**(0), 728–739.

Yoon, M.-G. and Current, J. (2008). The hub location and network design problem with fixed and variable arc costs: Formulation and dual-based solution heuristic. *The Journal of the Operational Research Society*, **59**(1), 80–89.

Zhang, M., Wiegmans, B., and Tavasszy, L. (2013). Optimization of multimodal networks including environmental costs: A model and findings for transport policy. *Computers in Industry*, **64**(2), 136–145.

Zhu, X. and Wilhelm, W. E. (2013). Implementation of a three-stage approach for the dynamic resource-constrained shortest-path sub-problem in branch-and-price. *Computers & Operations Research*, **40**(1), 385–394.

Ann-Kathrin Rothenbächer | Vita

Chair of Logistics Management, Johannes Gutenberg University Mainz
Jakob-Welder-Weg 9 – 55128 Mainz, Germany
☎ +49 6131 39 22086 • FAX +49 6131 39 22097 • ✉ anrothen@uni-mainz.de
logistik.bwl.uni-mainz.de/index.php

Education

2017 **PhD**, *Johannes Gutenberg University*, Mainz - Germany.
- Chair of Logistics Management
- PhD Thesis: *'Contributions to Branch-and-Price-and-Cut Algorithms for Routing Problems'*

2014 **Master of Science**, *Technical University of Darmstadt*, Darmstadt - Germany.
- Studies of Mathematical Economics
- Master Thesis: *'Hub Location and Network Design in Combined Freight Transport'*

Publications

Articles in Refereed Journals

- Gschwind, T., Irnich, S., Rothenbächer, A.-K., and Tilk, C. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. *European Journal of Operational Research*, doi: https://doi.org/10.1016/j.ejor.2017.09.035, 2017.
- Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, 261(2), 530–539, 2017.
- Rothenbächer, A.-K., Drexl, M., and Irnich, S. Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, 2016. Forthcoming.
- Rothenbächer, A.-K., Drexl, M., and Irnich, S. Branch-and-price-and-cut for a service network design and hub location problem. *European Journal of Operational Research*, 255(3), 935 – 947, 2016.

Technical Reports

- Rothenbächer, A.-K. Branch-and-price-and-cut for the periodic vehicle routing problem with flexible schedule structures. Technical Report LM-2017-05, Chair of Logistics Management, Johannes Gutenberg University Mainz, Germany, 2017.

Talks

- Rothenbächer, A.-K. Branch-and-price-and-cut for the periodic vehicle routing problem with flexible schedule structures. OR. Berlin, Germany, 7th September 2017.
- Rothenbächer, A.-K., Drexl, M., and Irnich, S. Branch-price-and-cut for the truck and trailer routing problem with time windows. OR. Hamburg, Germany, 31th August 2016.
- Rothenbächer, A.-K., Drexl, M., and Irnich, S. Branch-price-and-cut for the truck and trailer routing problem with time windows. VeRoLog. Nantes, France, 7th June 2016.
- Rothenbächer, A.-K., Drexl, M., and Irnich, S. Branch-and-price-and-cut for a service network design and hub location problem. OR. Vienna, Austria, 4th September 2015.