Contributions to Exact Approaches in Combinatorial Optimization

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
Dipl.-Math. Timo Gschwind
in Mainz

im Jahr 2014

Tag der mündlichen Prüfung: 16.10.2014

Contents

List of Papers

- Timo Gschwind[1], Stefan Irnich[1] (2011). "A Note on Symmetry Reduction for Circular Traveling Tournament Problems". *European Journal of Operational Research* **210**(2), 452–456.
- Timo Gschwind[1], Stefan Irnich[1] (2014). "Effective Handling of Dynamic Time Windows and its Application to Solving the Dial-a-Ride Problem". *Transportation Science*. Forthcoming.
- Timo Gschwind[1] (2014). "A Comparison of Column-Generation Approaches to the Vehicle Routing Problem with Time Windows and Temporal Synchronized Pickup and Delivery". Technical Report LM-2014-01, Chair of Logistics Management, Johannes Gutenberg University Mainz, Mainz, Germany.
- Timo Gschwind[1], Stefan Irnich[1] (2014). "Dual Inequalities for Stabilized Column Generation Revisited". Technical Report LM-2014-03, Chair of Logistics Management, Johannes Gutenberg University Mainz, Mainz, Germany.

Additional papers not included in this thesis:

- Holger A. Stephan[2], Timo Gschwind[1], Stefan Minner[3](2010). "Manufacturing Capacity Planning and the Value of Multi-Stage Stochastic Programming under Markovian Demand". *Flexible Services and Manufacturing Journal* **22**(3–4), 143–162.

[1]Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany
[2]Daimler AG, Production Network Planning Trucks, D-70546 Stuttgart, Germany
[3]TUM School of Management, Technische Universität München, Arcisstraße 21, D-80333 München, Germany

List of Figures

List of Tables

Chapter 1

Introduction

In combinatorial optimization the task is to find an optimal object from a finite set of discrete objects (Schrijver, 2003, p. 1). Often, combinatorial optimization problems are motivated by real-life problems for which an optimal object is one that minimizes or maximizes a specific objective (e.g. profit, risk, length, ...) subject to side constraints that describe the set of feasible alternatives. Applications in logistics include cutting and packing, routing, scheduling, and network design problems.

Though finite, the set of feasible alternatives is typically very large (often growing exponentially in the size of the instance) so that well-thought-out algorithmic strategies are necessary for solving such problems. Still, many combinatorial optimization problems are $\mathcal{NP}$-hard meaning that no polynomial-time algorithm is likely to exist for their solution (Schrijver, 2003, p. 1). In other words, they are very hard to solve. Consequently, exact solution approaches for these problems, i.e., approaches guaranteeing an optimal solution, are generally not able to solve instances of practical size in acceptable time. Thus, for decision support, (meta-)heuristics that provide good, but generally not optimal solutions in reasonable time are typically indispensable.

However, there are also numerous examples of problem-specific, exact algorithms that are capable of solving large instances, sometimes even of practical size. Examples include the work of Applegate *et al.* (2006) for the traveling salesman problem, Martello *et al.* (1999) for the knapsack problem, Östergård (2002) and Held *et al.* (2012) for cliques/independent sets, Vanderbeck (1999) for the cutting stock problem, or García *et al.* (2011) for the p-median problem. This can be attributed to the continuous development of new and innovative exact approaches over the past decades, which has led to a significant improvement on the size of the instances that can be solved to optimality for many problems. Hence, apart from the theoretical interest in such exact algorithms, their enhancement is also crucial from a practical point of view. Moreover, many algorithmic concepts originating from exact approaches found their way into successful, modern heuristics.

The focus of this thesis is to contribute to the development of new exact approaches in combinatorial optimization. Thereby, we present ideas for different problems coming from a variety of applications, rather than restricting ourselves to only a specific problem and variants of it. Before we can state the contributions of this thesis, we must first give a brief description of the main problems and the key algorithmic concept considered in this thesis. For a detailed and more formal definition we refer to the respective chapters of the thesis.

The *traveling tournament problem* (TTP) is a problem in the area of sports league

scheduling abstracted from the key requirements of scheduling a major sports league in North America (Easton *et al.*, 2001). It consists of finding a distance-minimal tournament schedule for an even number of teams such that each team plays at each time slot, each team faces every other team exactly twice (once at home, once away), and no team has more than three consecutive matches at home or away. The TTP appears to be a problem that is practically very hard to solve and the largest instances that could be solved to proven optimality consist of only 10 teams.

Vehicle routing problems (VRPs) form one of the most widely studied classes of combinatorial optimization problems. Their huge success as a research topic can arguably be attributed to the fact that they are both theoretically challenging and of high relevance in transportation and logistics practice (Toth and Vigo, 2002, p. xvii). The classical capacitated VRP searches for a distance-minimal set of routes starting and ending at the depot such that each of the given customers is visited exactly once and vehicle capacities are obeyed at any time. Among many other VRP variants considered in the routing literature, a current research branch is devoted to VRPs with synchronization. The additional presence of synchronization constraints makes the solution of these problems particularly challenging (Drexl, 2012).

A prominent VRP with intra-route synchronization is the so-called *dial-a-ride problem* (DARP) in which passengers have to be transported between specific origin (=pickup) and destination (=delivery) locations. The DARP mainly arises in door-to-door transportation services for school children, handicapped persons, or the elderly and disabled. The transportation of passengers rather than goods necessitates that passenger inconvenience considerations are taken into account to guarantee a certain level of service. Therefore, maximum user ride times are often imposed to limit the time a passenger is on board of the vehicle. From a modeling point of view, the maximum ride-time constraints are intra-route synchronization constraints coupling the service times at the pickup and delivery locations of a passenger.

A straightforward generalization of the DARP also considers minimum ride-time constraints. Arguably, this kind of constraint is less reasonable in the sense of requiring a passenger or good to be on board of the vehicle for a minimum time. However, it can be used to model a generalized synchronization condition for the service times at corresponding pickup and delivery locations or, more generally, at two nodes for which a pairing and precedence relation is given. Practical applications include the planning of security guards where locations have to be inspected repeatedly within given time intervals (Bredström and Rönnqvist, 2008). Similar planning problems arise in home health care, e.g., when patients have to be monitored by a nurse several times a day (Eveborn *et al.*, 2006). We call the abstracted problem the *vehicle routing problem with time windows and temporal synchronized pickup and delivery* (VRPTWTSPD).

Column generation (CG) has been proven a successful tool for solving very large-scale linear programs (LPs) with a huge number of variables. Often, the set of variables is too large for explicitly representing the model. Embedding CG into a branch-and-bound algorithm leads to an integer CG approach commonly known as branch-and-price that allows for for the solution of mixed integer programs (Lübbecke and Desrosiers, 2005). In many applications, such large-scale models result from a Dantzig-Wolfe decomposition

(Dantzig and Wolfe, 1960), i.e., a problem reformulation using variable redefinition, of an integer compact model. The reformulation gives rise to the so-called (integer) master program that typically has an exponential number of variables.

Instead of solving the master program directly, the basic idea of CG is to work on a restricted master program that considers only a reasonably small subset of variables. Missing negative reduced-cost variables are added dynamically to the model when needed. For their identification, the so-called CG pricing problem (=subproblem) has to be solved. The solution of the LP relaxation of the master program is obtained by alternating between re-optimizing the restricted master program and solving the subproblem as long as no more variables with negative reduced cost exist. Thus, the overall CG process is driven by the values of the dual variables (Desrosiers and Lübbecke, 2005).

The strength of CG based approaches can mainly be attributed to the following aspects: (i) the reformulated model typically provides stronger LP bounds than the compact formulation, (ii) non-linearities that are present in the compact formulation may be hidden in the subproblem, and (iii) highly effective solution procedures for the subproblem are often available (Lübbecke and Desrosiers, 2005). The main drawbacks of CG algorithms are related to instability issues like, e.g., the heading-in effect (generation of irrelevant columns in the first iterations because of poor dual information), the plateau effect (no improvement of the RMP value for several iterations), the tailing-off effect (slow convergence of the CG process), and the oscillation of the duals (Vanderbeck, 2005).

1.1 Contribution

The main goal of this thesis is to contribute to the development of new, exact solution approaches to different combinatorial optimization problems. In particular, we derive dedicated solution approaches to a special class of TTPs, the DARP, and the VRPTWT-SPD. Furthermore, we extend the concept of using *dual optimal inequalities* (DOIs) for stabilized CG as proposed by Ben Amor *et al.* (2006) and detail its application to improved CG algorithms for the *cutting stock problem* (CS), the *bin packing problem* (BP), the *vertex coloring problem* (VC), and the *bin packing problem with conflicts* (BPC). In all approaches, we make use of some knowledge about the structure of the problem at hand to individualize and enhance existing algorithms. Specifically, we utilize knowledge about the input data (Chapter 2), problem-specific constraints (Chapters 3 and 4), and the dual solution space (Chapter 5).

We elaborate on the contributions of the different parts in more detail now. For the TTP, Easton *et al.* (2001) introduced the so-called circular instances, where venues of the teams are located on a circle. It is empirically proven that these instances are very hard to solve due to the inherent symmetry. We perform a detailed analysis of this symmetry and present new ideas to cut off essentially identical parts of the solution space. Furthermore, we empirically test the proposed symmetry reduction by modifying the DFS* algorithm of Uthus *et al.* (2009) and show that speedups can approximate factor $4n$.

For many VRP variants, some of the strongest approaches are based on integer CG (Baldacci *et al.*, 2012). The master programs of such approaches are typically formulated on variables representing feasible routes for the problem at hand and provide very tight LP bounds. The CG subproblems are elementary shortest-path problems with resource constraints which can often be solved effectively with dynamic-programming labeling algorithms (Irnich and Desaulniers, 2005). However, there are also variants for which certain classes of route constraints are hard to incorporate in the subproblem in combination, e.g., time windows together with maximum ride-time constraints. In these cases, traditional labeling algorithms are often not able to handle all route constraints and, thus, to price out feasible routes for the problem at hand. A common strategy to overcome this problem is to relax one or more classes of route constraints in the subproblem and handle them in the master instead (e.g. Ropke and Cordeau (2005) for the DARP or Cherkesly *et al.* (2014) for the pickup-and-delivery problem with time windows and last-in-first-out Loading). The resulting easier to solve subproblems come at the cost of weaker lower bounds and, thus, potentially larger search trees.

We propose a new branch-and-cut-and-price algorithm for the DARP that for the first time handles all route constraints in the subproblem. The key component of the approach is an effective CG pricing procedure that allows fast shortest-path computations due to new, strong dominance rules. Moreover, checking whether or not a given route obeys both time-window and maximum ride-time constraints is intricate. We provide a labeling procedure which can be used for efficient feasibility checking when a partial route is extended in a node-by-node fashion. Also, we compare the proposed branch-and-cut-and-price approach with the best known approaches from the literature and alternative CG algorithms that handle maximum ride-time constraints either as constraints of the master program or with less effective labeling procedures. Extensive computational experiments indicate the superiority of the new approach.

Next, we extend our findings for the DARP to the case of general temporal intra-route synchronization constraints, i.e., with minimum and maximum ride-time constraints. Thereto, we first introduce the VRPTWTSPD as the prototypical VRP with temporal intra-route synchronization. This problem has to the best of our knowledge not been considered before. Compared to the DARP, the additional presence of minimum ride times further complicates the VRPTWTSPD. Again, it is not clear if the additional effort of integrating them into the subproblem pays off in the overall algorithm. Therefore, we develop four exact solution approaches to the VRPTWTSPD based on CG formulations whose master programs are formulated on different sets of variables implying different subproblems. Two of these subproblems have not been considered before. One of them is the subproblem that incorporates all route constraints of the VRPTWTSPD so that time windows as well as temporal intra-route synchronization with both minimum and maximum ride-time constraints have to be dealt with. In the other one, maximum ride-times are relaxed. Based on our ideas for the subproblem of the DARP, we derive new dominance rules and labeling algorithms for their solution. Furthermore, we report extensive computational results over a large number of test instances with different characteristics regarding the number of customer requests and the tightness of capacity, time-window, and minimum and maximum ride-time constraints. They indicate that

integrating either both types of ride-time constraints or only the maximum ride-time constraints into the subproblem results in the strongest overall algorithm.

Our last contribution is related to overcoming instability issues of CG approaches. Among different alternatives for stabilizing the CG process, Ben Amor *et al.* (2006) suggest the use of DOIs in the context of CS and BP. We generalize their work in several aspects. First, we provide insights into the relations of optimal solution values and optimal solutions of the original models (primal and dual) and the extended models to which extra *dual inequalities* (DIs) have been added. Second, we define several new classes of properties for the coefficient matrix of a LP with unit costs enabling the identification of new DOIs and *deep dual optimal inequalities* (DDOIs). Herewith, we derive additional classes of DDOIs for BP and DOIs for CS and identify DOIs and DDOIs also for several new classes of problems (the *vector packing problem*, VC, BPC) and, thus, extend the DI-based stabilization to these problems. Moreover, we suggest the dynamic addition of violated DIs in a cutting-plane fashion and the use of DIs that are not necessarily DDOIs. In the latter case, a recovery procedure is needed to restore primal feasibility. Computational results proving the usefulness of the methods are presented.

1.2 Outline

This thesis by publication comprises four articles that have either been published in or submitted to scientific journals. The remainder of the thesis is organized as follows.

In Chapter 2, we detail our symmetry reduction ideas for the circular TTP instances. A formal description of the TTP and the circular instances is given in Section 2.1. Section 2.2 describes different kinds of symmetry and, if symmetry is reduced, compares the worst-case number of solutions to consider in enumerative solution approaches for the TTP. In Section 2.3, we empirically test the symmetry reduction methods on the branch-and-bound algorithm that constitutes the kernel of the DFS* algorithm of Uthus *et al.* (2009).

In Chapters 3 and 4, we address routing problems with temporal intra-route synchronization constraints deriving new branch-and-cut-and-price solution approaches to the DARP and the VRPTWTSPD. Chapter 3 focuses on the DARP and the handling of maximum ride-time constraints. A short literature review on the DARP is provided in Section 3.2. Section 3.3 gives a formal definition of the DARP, presents compact and CG formulations from the literature, and outlines the differences to the CG formulation proposed in this thesis. For the effective solution of the subproblem, we develop weak and strong dominance relations between partial routes to be used in dynamic-programming labeling algorithms. They are presented in Section 3.4. The basic components of the branch-and-cut-and-price solution algorithm are briefly discussed in Section 3.5 followed by computational experiments in Section 3.6.

The case of routing with general temporal intra-route synchronization constraints is dealt with in Chapter 4. In Section 4.2, we define the VRPTWTSPD and present different CG formulations of it. Section 4.3 describes the dominance rules and labeling

algorithms we use for solving the subproblems of our approaches. Our basic branch-and-cut-and-price algorithm is briefly described in Section 4.4. Computational results are reported in Section 4.5.

Chapter 5 presents our work on stabilized CG. In Section 5.1, we introduce the original and extended, primal and dual CG models considered in this chapter. Section 5.2 presents a proposition on the relations of optimal solution values and optimal solutions to these models. Useful properties allowing for the identification of DIs that are DOIs and/or DDOIs are derived in Section 5.3. Furthermore, the application of these properties to different problems is demonstrated. This includes the proper introduction of the CG formulations of these problems and the specification of the different new classes of DOIs and DDOIs. Section 5.4 clarifies the dynamic separation of violated DIs and presents the recovering procedure that is necessary if DIs that are not DDOIs have been used. Computational results for several problems are presented in Section 5.5.

Chapter 6 summarizes and draws final conclusions.

Chapter 2

A Note on Symmetry Reduction for Circular Traveling Tournament Problems

Timo Gschwind, Stefan Irnich

Abstract

The traveling tournament problem (TTP) consists of finding a distance-minimal double round-robin tournament where the number of consecutive breaks is bounded. Easton *et al.* (2001) introduced the so-called circular TTP instances, where venues of teams are located on a circle. The distance between neighboring venues is one, so that the distance between any pair of teams is the distance on the circle. It is empirically proven that these instances are very hard to solve due to the inherent symmetry. This note presents new ideas to cut off essentially identical parts of the solution space. Enumerative solution approaches, e.g. relying on branch-and-bound, benefit from this reduction. We exemplify this benefit by modifying the DFS* algorithm of Uthus *et al.* (2009) and show that speedups can approximate factor $4n$.

2.1 Introduction

The *traveling tournament problem* (TTP) consists of finding a distance-minimal double round-robin tournament where the number of consecutive breaks is bounded (cf. Easton *et al.*, 2001). Teams and venues are given by $T = \{1, 2, \ldots, n\}$ for an even number $n \in 2\mathbb{N}$. Throughout the paper, we use indices $t, t' \in T$ for teams and indices $i, j \in T$ for their venues. Let $D = (d_{ij})$ for $i, j \in T$ be the distance between venues i and j. Any feasible solution to the TTP can be represented by tours $p^t = (i^t_1, \ldots, i^t_{2(n-1)})$ for all teams $t \in T$. Herein, $i^t_s \in T$ describes for each time slot $s \in \{1, 2, \ldots, 2(n-1)\}$ at which venue team t plays. For $i^t_s = t$, team t is playing at *home*, otherwise t is playing *away*. Obviously, a solution is only feasible if tours are compatible, i.e. an away game $i^t_s = t' \neq t$ implies a home game $i^{t'}_s = t'$. With the definitions $i^t_0 := i^t_{2n-1} := t$ for all teams $t \in T$, the cost of a solution p is the overall distance traveled $\sum_{t \in T} \sum_{s=0}^{2(n-1)} d_{i^t_s, i^t_{s+1}}$. Additionally, *no-repeater constraints* (NRCs) require that a home game t against t' is not followed by the corresponding away game, where t' plays at home against t.

Easton *et al.* (2001) introduced the so-called *circular* TTP instances, where venues of the teams are located on a circle. The distance between neighboring venues is one so that the distance between two teams $i > j$ is $d_{ij} = \min\{i-j, n+j-i\}$. It is empirically proven that these instances are very hard to solve due to the inherent symmetry. The purpose of this note is, therefore, to fully explicate the symmetry and exploit it algorithmically.

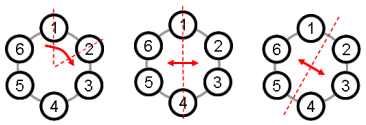
This paper is structured as follows: In Section 2.2 we describe different kinds of symmetry and, if symmetry is reduced, we compare the worst-case number of solutions to consider in enumerative solution approaches for the TTP. Section 2.3 empirically tests the symmetry reduction methods on the branch-and-bound algorithm that constitutes the kernel of the DFS* algorithm presented in (Uthus *et al.*, 2009). Final conclusions are drawn in Section 2.4.

2.2 Symmetry in Circular TTP Instances

It is possible to rotate the teams and venues of a feasible solution, i.e. replace t by $t+1$ (for $t < n$) and n by 1. The rotated solution is also feasible and has identical costs. Furthermore, reflections through opposite vertices or edges, as depicted in Figure 2.1, produce symmetric solutions with identical costs.

$$(d_{ij}) = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & 1 \\ 1 & 0 & 1 & 2 & 3 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 3 & 2 & 1 & 0 & 1 \\ 1 & 2 & 3 & 2 & 1 & 0 \end{pmatrix}$$

(a)



(b) (c) (d)

Figure 2.1: Circular instance for $n = 6$; (a) Distances, (b) Rotation by $2\pi/6$, (c) and (d) Reflections through vertices or edges

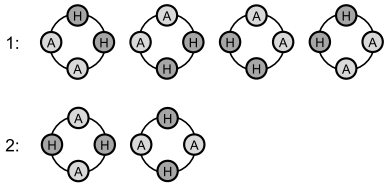
Formally, the symmetry group of an n-sided regular polygon, which is known as the dihedral group D_n, operates on the teams/venues (for an introduction to group theory we refer to Aschbacher, 2000). D_n consists of n rotations by multiples of $2\pi/n$ (including the identity) and n reflections through opposite vertices and edges. We can exploit this operation in different ways, either by considering home-away assignments or oriented 1-factors. Both will be explained in the following.

Symmetry on Home-Away Assignments A *home-away assignment (HAA)* partially characterizes a solution to the TTP. A HAA specifies, for each team and each time slot, whether the team plays at home or away. Since there are equally many teams playing at home and away in a time slot, the overall number of *slot HAAs* is $\binom{n}{n/2}$. D_n operates on the set of slot HAAs of a given slot s.

For $n = 6$, there are $\binom{6}{3} = 20$ possible slot HAAs. The slot HAAs that are symmetric to (*hhhaaa*) are defined as the *orbit*. For instance, the orbit of (*hhhaaa*) is

$\{(hhhaaa), (ahhhaa), (aahhha), (aaahhh), (haaahh), (hhaaah)\}$. The three different orbits with lengths 2, 6, and 12 are depicted in Figure 2.2(b).

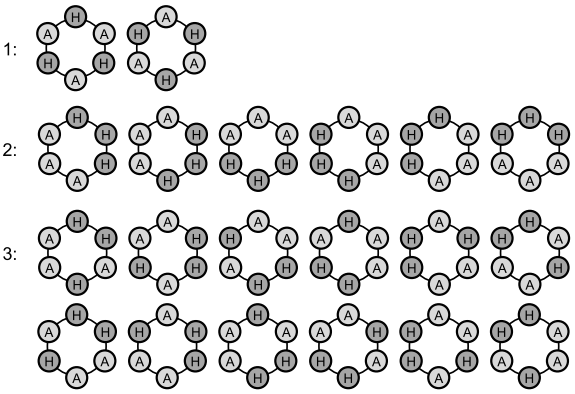
Due to symmetry in the circular instances, it suffices to consider one *representative* slot HAA from each orbit for a given time slot s. This is the key observation for symmetry reduction. Without loss of generality, we consider the first time slot $s = 1$ in the following. In branch-and-bound, symmetry reduction can be implemented as an a priori branching rule in the first level of the branch-and-bound tree. For $n = 4$, there are just two branches: The first branch can fix that teams 1 and 2 play at home, while the others play away; the second branch can fix that teams 2 and 4 play at home and 1 and 3 away, as depicted in Fig. 2.2(a). Also other enumerative solution approaches for circular TTP instances benefit from choosing representatives, e.g. the branch-and-price algorithm by Irnich (2010).



(a)

Nb. of teams n	Nb. of slot HAAs	Nb. of orbits	Ratio
4	6	2	3
6	20	3	≈6.7
8	70	8	≈8.8
10	252	16	≈15.8
12	924	50	≈18.5

(c)



(b)

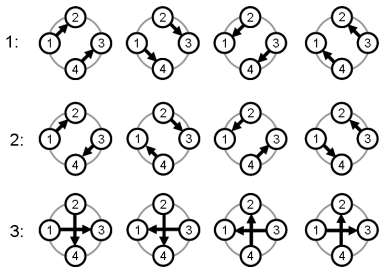
Figure 2.2: (a) Orbits of HAAs for $n = 4$, (b) for $n = 6$, and (c) Number of HAAs and orbits

Moreover, Figure 2.2(c) compares standard branching with a priori branching on slot HAA representatives. In the worst case, all $\binom{n}{n/2}$ HAAs in slot 1 are checked in a standard branch-and-bound approach. Taking representatives creates just as many branches as orbits exist. Thus, the ratio of the two numbers is an (optimistic) estimate what can be gained from selecting representatives. This ratio is 3 for $n = 4$ and increases to approximately 18 for $n = 12$. Since the length of an orbit divides the order of the symmetry group, the ratio is here bounded (from above) by $|D_n| = 2n$. The numbers in the table presented in Figure 2.2(c) were computed with a straightforward enumeration procedure.

Symmetric 1-Factors For each time slot s, a solution imposes a tournament, i.e. an *oriented 1-factor* of the complete graph with node set T (cf. de Werra, 1980). For instance, $f := \{(3, 2), (1, 6), (5, 4)\}$ means that team 3 plays at home against 2, 1 at

home against 6, and 5 at home against 4. The dihedral group D_n naturally operates on oriented 1-factors. On f, D_n produces an orbit consisting of four elements: $\{f, \{(2,1),(4,3),(6,5)\}, \{(1,2),(3,4),(5,6)\}, \{(2,3),(4,5),(6,1)\}\}$. As before, in an enumerative solution approach such as branch-and-bound, all elements of an orbit are equivalent. Therefore, it suffices to consider one *representative* oriented 1-factor for the slot $s = 1$.

Figure 2.3(a) shows for $n = 4$ the three orbits with four elements each. Reducing symmetry in branch-and-bound for $n = 4$ can a priori branch on the three representatives, e.g. the first branch assigns the games $(1,2),(4,3)$ to slot 1, the second $(1,2),(3,4)$, and the third $(1,3),(2,4)$. The table in Figure 2.3(b) has the same meaning as the one presented for the slot HAAs. In particular, the ratio shows the reduction in the number of cases to be considered when reducing symmetry with 1-factors. Interestingly, the ratio here is larger compared to the ratio for slot HAAs. For $n \geq 10$, it is approximately equal to the order $2n$ of the symmetry group D_n.



(a)

Nb. of teams n	Nb. of ori. 1-factors	Nb. of orbits	Ratio
4	12	3	4
6	120	13	≈9.2
8	1,680	121	≈13.9
10	30,240	1,538	≈19.6
12	665,280	28,010	≈23.8

(b)

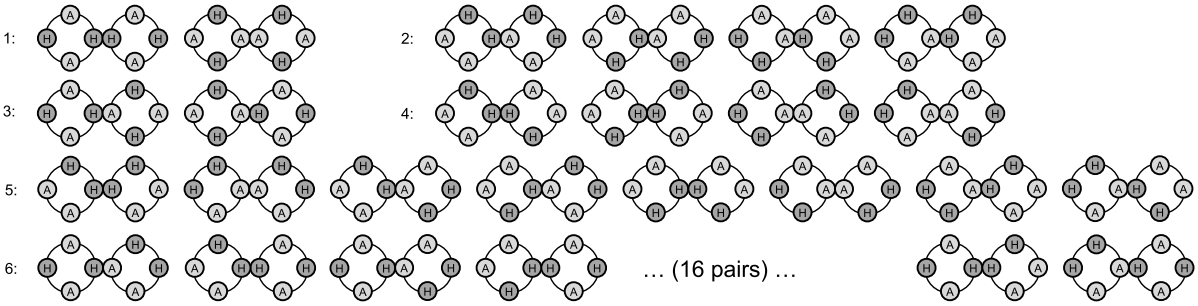
Figure 2.3: (a) Orbits of oriented 1-factors for $n = 4$, (b) Number of oriented 1-factors and orbits

Reversing Tours Another type of symmetry can be utilized if distances d_{ij} are symmetric. In this case, any solution with tours $p^t = (i_1^t, i_2^t, \ldots, i_{2(n-1)}^t)$ of the teams $t \in T$ has a corresponding *reverse solution* with reverse tours $(i_{2(n-1)}^t, \ldots, i_2^t, i_1^t)$ for all teams t. The symmetry group is the group $\mathbb{Z}_2 = \{id, rev\} \cong \{+1, -1\}$ consisting of the identity $id = +1$ and the reversal $rev = -1$. Several authors have already exploited this symmetry: In the branch-and-price algorithm of Irnich (2010), arcs are removed from the pricing network of one team (say team 1) in such a way that only one of the two symmetric tours can be generated. In their branch-and-bound approach, Uthus *et al.* (2009) require for team 1 that the first half of the schedule (i.e. games in slots 1 to $n-1$) has more home than away games (or vice versa). This is checked when the schedule of team 1 is constructed round by round. In any case, the speedup that can be gained from exploiting reverse tours is obviously bounded by factor 2.

Symmetry on Pairs of HAAs Both symmetry within time slot 1 (slot HAAs or oriented 1-factors) and symmetry induced by tour reversal can be combined. The symmetry group is $G_n = D_n \times \{+1, -1\}$ with $4n$ elements. Any group element $g = (d, \varepsilon) \in G$

operates on a solution with tours $(i_1^t, i_2^t, \ldots, i_{2(n-1)}^t)$, $t \in T$ as follows: the resulting solution is given by the tours $(i_1^{d(t)}, i_2^{d(t)}, \ldots, i_{2(n-1)}^{d(t)})$ for $\varepsilon = +1$ and $(i_{2(n-1)}^{d(t)}, \ldots, i_2^{d(t)}, i_1^{d(t)})$ for $\varepsilon = -1$.

In order to exploit this symmetry, we consider two slot HAAs simultaneously. The HAAs must refer to corresponding time slots (w.r.t reversal), say, the first time slot 1 and the last time slot $2(n-1)$. The overall number of HAA pairs is $\binom{n}{n/2}^2$. For $n = 4$, the operation of G_4 on the $\binom{4}{2}^2 = 36$ pairs of HAAs is summarized in Figure 2.4(a): G_4 produces six orbits with lengths 2, 2, 4, 4, 8, and 16. In general, the table in Figure 2.4(b) shows that the ratio approximates the order $4n$ of the symmetry group G_n. It is, from $n = 4$ on, larger than the ratios for single slot HAAs and oriented 1-factors, and it is for $n \geq 6$ larger than twice the ratio for single slot HAAs. These observations suggest that using pairs of HAAs may lead to larger reductions in computing time than the symmetry reduction techniques that only take the dihedral group D_n into account. We will empirically check this assumption in Section 2.3.



(a)

Nb. of teams n	Nb. of HAA pairs	Nb. of orbits	Ratio
4	36	6	6
6	400	26	≈15.4
8	4,900	186	≈26.3
10	63,504	1,682	≈37.8
12	853,776	18,168	≈47.0

(b)

Nb. of teams n	Nb. of 1-fact. pairs	Nb. of orbits	Ratio
4	144	17	≈8.5
6	14,400	652	≈22.1
8	2,822,400	89,093	≈31.7
10	914,457,600	?	?
12	4,425,974,784	?	?

(c)

Figure 2.4: Operation of $G_n = D_n \times \mathbb{Z}_2$ (a) Orbits of HAA pairs for $n = 4$, (b) Number of HAA pairs and orbits, (c) Number of 1-factor pairs and orbits

Now it is straightforward to also combine oriented 1-factors with reversals. Since the number of oriented 1-factors quickly increases with n, the number of pairs grows even faster, as can be seen in the table in Figure 2.4(c). Due to the huge number of pairs we were not able to compute the orbits for $n = 10$ and $n = 12$. Thus, symmetry reduction based on pairs of oriented 1-factors is not a promising approach to accelerate enumerative algorithms. We will not consider pairs of oriented 1-factors any further.

2.3 Computational Results

Algorithm To quantify the benefits of the different symmetry reductions, we implemented a modified version of the DFS* approach of Uthus *et al.* (2009). Simplifying, we apply only the kernel of their approach which is a standard depth-first brand-and-bound (DFB&B) search, where we initialize upper bounds with the best known solutions from the literature. The DFB&B uses recursive backtracking to construct the overall schedule time slot by time slot. In each step, it picks (in numerical order) the first team that has not been assigned in the respective time slot and pairs it with a feasible opponent, at home or away. For the first time slot, the integration of our symmetry reduction methods is straightforward: only games that match the given representative in the first time slot are allowed.

Lower bounds in each step are given by the sum of the distances traveled so far and the independent lower bounds (ILBs) for the remaining tours of each team individually. The ILBs for each team are determined prior to the DFB&B execution. Herein, we use all information available at the respective step except for the particular teams left to play at home. Thus, each ILB is indexed by the team t it belongs to, the time slot s of the last assigned game of team t, the venue i_s^t, the current number of breaks, and the teams against whom team t still has to play away. This contrasts to the approach of Uthus *et al.* (2009) who do not use the exact number of remaining home games resulting in a slightly lower memory usage by the cost of possibly weaker bounds in some cases. The ILBs are the second point where the symmetry on HAA pairs is applied: we allow only for those tours that match the pattern of a given representative in the last time slot, which leads to tighter ILBs in many cases. To sum up, our branch-and-bound algorithm works as follows: First all orbits and all ILBs are computed, followed by the DFB&B for one representative of each orbit. This offers a natural way to parallelize the overall optimization as the orbits can be processed independently of each other.

Computational Setup We tested our symmetry reduction methods on the different variants of the circular TTP instances, i.e. the constrained instances, where the number of consecutive home/away games is bounded by 3, as well as the unconstrained instances with and without NRCs. The TTP benchmark instances can be found on Michael Trick's website `http://mat.tepper.cmu.edu/TOURN`.

All algorithms were coded in C++, compiled in release mode with MS-Visual C++ 2008 version 9.0, .NET version 3.5 SP 1. Most runs were performed on a standard PC with Intel Core2 Duo E8400 CPU with 3.00 GHz, 4GB main memory, on Windows 7 using a single thread. For the harder instances (circ 8 unconstrained and circ 10) we distributed the computations on several computers with different performance, but normalized computation times.

Runtime Analysis The first observation is that the costs for calculating the orbits are negligible as their computation always takes less than 0,1% of the overall runtime for instances with $n \geq 8$ teams. For the smaller instances the computation times are not measurable.

Our most important finding is that all symmetry reductions cut the computation times of the branch-and-bound algorithm significantly. Table 2.1 summarizes our results regarding the constrained instances. For each instance, it shows the overall computation time and the total number of recursions. Since both times and recursions are mostly smallest for symmetry reduction based on HAA pairs, we present absolute values (=100%) for HAA pairs but relative values in all other cases. In general, the larger the symmetry group, the larger the average length of an orbit can be. Note that the ratios presented in Section 2.2 are in fact the average lengths of orbits. Now we empirically observe that the larger the orbits, the larger the speedup. Interestingly, the gains in computation times are slightly smaller than the gains in the number of recursions. This is due to the fact that in the branch-and-bound algorithm recursions are less expensive the deeper they are in the search tree, and reduction methods with more orbits generally have to consider more recursions with lower depth.

With increasing complexity of the instances we observe increasing speedups that, for circ 8, almost reach the maximal possible savings for all symmetry groups as computed in Section 2.2. For instance, we calculated a ratio of 26.3 for HAA pairs and attained a speedup of factor 20.4; for 1-factors we have a ratio of 13.9 and a speedup of factor 10.7, and for slot HAAs 8.8 and 7.4, respectively.

The results suggest that similar speedups can be gained also for larger instances (if one were able to solve these). Thus, we think that the use of HAA pairs is the most promising approach to reduce symmetry when solving constrained instances. Using HAA pairs, we also confirm the result of Uthus et al. regarding the optimal solution of the constrained circ 10 instance (cf. `http://mat.tepper.cmu.edu/TOURN`).

The results for the unconstrained instances are given in Table 2.2 which is structured analogously to Table 2.1. However, our benchmark is now the symmetry reduction using 1-factors as computation times and number of recursions are smallest here. Still, the results are less uniform compared to the constrained case. While symmetry reduction based on slot HAAs and 1-factors again perform well and produce speedups approximating the ratios presented in Section 2.2, symmetry reduction with HAA pairs is slower than expected. We suspect that this is related to the gap between upper bound (or optimal solution) and ILBs, which is small for the constrained but huge for the unconstrained instances ($\frac{UB-ILB}{ILB}$ between 50% and 60% for $n = 6, 8$). When applying reduction with HAA pairs, the fixations in the last slot lead to slightly tighter bounds being particularly effective if the gap is small. For the unconstrained case, however, the small increase of the lower bounds typically does not suffice to terminate the tree search at an early stage. As a result, fixations with slot HAAs and HAA pairs need approximately the same number of recursions for one orbit, while there are many more orbits for HAA pairs leading to longer overall computation times. This explains why it is most beneficial to use 1-factors for the unconstrained instances.

Comparing constrained and unconstrained instances, the latter are considerably harder to solve (with approaches using ILBs). However, with the symmetry reduction methods we were able to prove optimality of the solutions of the unconstrained circ 8 instances provided by Langford (cf. `http://mat.tepper.cmu.edu/TOURN`). Using symmetry reduction based on 1-factors, circ 8 unconstrained with NRCs is solved in less than 50 hours, and circ 8 unconstrained without NRCs in less than 42 hours.

Instance	Opt. sol.	Computation time					Number of recursions				
		None	Tour reversal	HAA	1-Factor	HAA pairs (100%)	None	Tour reversal	HAA	1-Factor	HAA pairs (100%)
circ 4	20	?	?	?	?	0s	229%	145%	84%	66%	38
circ 6	64	320%	200%	60%	60%	0.5s	713%	457%	127%	85%	58 713
circ 8	132	2041%	1411%	275%	189%	44s	2259%	1553%	303%	205%	8 568 856
circ 10	242	-	-	-	-	$6.0 \cdot 10^7$s	-	-	-	-	$1.2 \cdot 10^{13}$

Table 2.1: Constrained instances: Computation times and number of recursions ("?"=unmeasurable)

Instance	Opt. sol.	Computation time					Number of recursions				
		None	Tour reversal	HAA pairs	HAA	1-Factor (100%)	None	Tour reversal	HAA pairs	HAA	1-Factor (100%)
With NRCs											
circ 4	20	?	?	?	?	0s	348%	220%	152%	128%	25
circ 6	56	825%	494%	306%	119%	1.6s	887%	525%	291%	128%	428 587
circ 8	102	-	-	829%	129%	179 094s	-	-	633%	146%	41 103 371 129
Without NRCs											
circ 4	20	?	?	?	?	0s	332%	229%	123%	113%	31
circ 6	54	775%	483%	258%	108%	1.2s	883%	546%	263%	130%	283 361
circ 8	100	-	-	795%	134%	149 611s	-	-	562%	147%	34 037 268 057

Table 2.2: Unconstrained instances: Computation times and number of recursions ("?"=unmeasurable)

2.4 Conclusions

In this note we provided insights into the nature of the symmetry inherent in the circular TTP instances. The largest symmetry group is $G_n = D_n \times \mathbb{Z}_2$, which is the Cartesian product of the dihedral group of order $2n$ (n is the number of teams) and the group of order 2. We have shown that G_n operates on pairs of home-away assignments of two corresponding time slots and can reduce the search space by a factor approximating $4n$. By selecting representatives from orbits it is easy to design algorithms that exactly cut off the symmetric parts of the solution space and thus are much faster.

Using a branch-and-bound approach we demonstrated that the speedups increase with the number n of teams. For the constrained circular instance with eight teams we observed a speedup of factor 20. A full analysis for more teams is impossible due to the large computation times, but the empirical results suggest that symmetry reduction can lead to speedups that approximate $4n$ for the larger constrained instances.

For the unconstrained circular instances, only the operation of D_n on 1-factors (implied by the games of a time slot) can be exploited effectively in the branch-and-bound approach. Here, speedups seem to approximate $2n$. Finally, with the accelerated branch-and-bound we were able to solve the two unconstrained instances circ 8 to proven optimality for the first time.

References

Aschbacher, M. (2000). *Finite Group Theory*. Cambridge University Press.

de Werra, D. (1980). Geography, games and graphs. *Discrete Applied Mathematics*, **2**, 327–337.

Easton, K., Nemhauser, G., and Trick, M. (2001). The traveling tournament problem description and benchmarks. In T. Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg.

Irnich, S. (2010). A new branch-and-price algorithm for the traveling tournament problem. *European Journal of Operational Research*, **204**(2), 218–228.

Uthus, D. C., Riddle, P. J., and Guesgen, H. W. (2009). DFS* and the traveling tournament problem. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. In *6th International Conference, CPAIOR 2009*, Pittsburgh, PA.

Chapter 3

Effective Handling of Dynamic Time Windows and its Application to Solving the Dial-a-Ride Problem

Timo Gschwind, Stefan Irnich

Abstract

A dynamic time window relates to two operations that have to be executed within a given time meaning that the difference between the points in time when the two operations are performed is bounded from above. The most prevalent context of dynamic time windows is when precedences are given for the two operations so that it is a priori specified that one operation must take place before the other. A prominent vehicle routing problem with dynamic time windows and precedences is the dial-a-ride problem (DARP), where user-specified transportation requests from origin to destination points have to be serviced. The paper presents a new branch-and-cut-and-price solution approach to the DARP, the prototypical vehicle-routing problem with ordinary and dynamic time windows. For the first time, both ordinary and dynamic time windows are handled in the column-generation subproblem. For its solution, an effective column-generation pricing procedure is derived that allows fast shortest-path computations due to new dominance rules. The new approach is compared with alternative column-generation algorithms that handle dynamic time windows either as constraints of the master program or with less effective labeling procedures. Computational experiments indicate the superiority of the new approach.

3.1 Introduction

In vehicle routing and scheduling, an ordinary or static time window restricts the point in time a specific operation takes place. Such an operation can be the visit of a customer, or in a more complex setting, the execution of a single service operation that is needed to fulfill a request consisting of several working steps. In contrast, a *dynamic time window* relates to two operations: Both operations have to be executed within a given time meaning that the difference between the points in time when the two operations are

performed is bounded from above. Thus, the difference is bounded by the dynamic time window. The most prevalent context of dynamic time windows is when precedences are given for the two operations so that it is a priori specified that one operation must take place before the other. However, precedence alone cannot model that both operations are synchronized by the dynamic time window.

A prominent *vehicle routing problem* (VRP) with dynamic time windows and precedences is the *dial-a-ride problem* (DARP), where user-specified transportation requests from origin to destination points have to be serviced. Common applications of the DARP are door-to-door transportation of school children, handicapped persons, the elderly and disabled (see, e.g., Russell and Morrel, 1986; Madsen *et al.*, 1995; Toth and Vigo, 1997; Borndörfer *et al.*, 1997). In order to guarantee a certain level of service, the duration a passenger is on board the vehicle is either penalized or restricted by a so-called ride-time constraint. We consider the latter case which is, obviously, an example for a dynamic time window. It refers to pickup and delivery operations, e.g., the origin can be the person's home and the destination a school, hospital, or training workshop.

In other VRP variants, dynamic time windows are used to limit the time that a vehicle can be away from the depot (see, e.g., Ceselli *et al.*, 2009; Azi *et al.*, 2010). This can be relevant, e.g., when perishable goods are transported or when there is a limit on the total working hours of drivers. The synchronization with dynamic time windows is an intra-route synchronization opposed to inter-route synchronization conditions that couple different routes (Drexl, 2012).

Intra-route synchronization constraints have additional applications in home care, where patients have to be monitored and attended by nurses in given intervals (Eveborn *et al.*, 2006). In the service industries, a service technician might need to come back to a location to finally finish a job that he has begun before. The reason might be that a workpiece must dry, harden, rest etc. before a next working step can start. Moreover, security guards have to inspect facilities repeatedly with a guaranteed time between two inspections (Bredström and Rönnqvist, 2008). Note that in these applications lower and upper bounds on the time difference can be present. In the paper at hand we exclusively focus on the case of upper bounds given by dynamic time windows as, e.g., a maximum ride-time constraints in the DARP.

The contribution of this paper is, first and foremost, the presentation of an effective solution approach to solving vehicle-routing problems with intra-route synchronization constraints. Specifically, we establish the following three results: First, we derive a new branch-and-cut-and-price solution approach to the DARP, the prototypical vehicle-routing problem with ordinary and dynamic time windows. For the first time, both time-window and ride-time constraints are handled in the column-generation subproblem. Decisive is the development of an effective column-generation pricing procedure that allows fast shortest-path computations based on new strong dominance rules. Second, checking for a given route whether or not there exists a feasible time schedule with service times that at the same time obey the ordinary and the dynamic time window constraints is intricate. We provide a labeling procedure which can be used for efficient feasibility checking when a partial route is extended in a node-by-node fashion. Third, we compare the proposed branch-and-price approach with alternative column-generation

algorithms that handle ride-time constraints either as constraints of the master program or with less effective labeling procedures.

We elaborate on the three contributions in more detail now: First, there are numerous examples for VRP variants, for which highly successful exact solution approaches are based on integer column generation (e.g., Desrochers *et al.*, 1992; Jepsen *et al.*, 2008; Baldacci and Mingozzi, 2009). The success of integer column generation can be attributed to the stronger lower bounds (we assume a minimization problem) that a column-generation (=extensive) formulation provides compared to an original formulation, from which the former is derived by Dantzig-Wolfe decomposition (Lübbecke and Desrosiers, 2005). *No* lower bound improvement can be gained if the subproblem possesses the *integrality property* (Lübbecke and Desrosiers, 2005). In many routing and scheduling applications, however, the subproblem is an elementary shortest-path problem with resource constraints (ESPPRC) not possessing the integrality property (Desaulniers *et al.*, 1998). Interestingly, stronger lower bounds can even result when only a proper relaxation of the subproblem is solved. For ESPPRC, one can relax the elementary condition and herewith allow paths visiting nodes more than once (resulting in an SPPRC). Both ESPPRC and SPPRC are typically solved using dynamic-programming labeling algorithms (Irnich and Desaulniers, 2005). In early column-generation algorithms, the use of SPPRC was the only viable approach because pseudo-polynomial algorithms are known in this case (Desrochers and Soumis, 1988). Later on, the exploitation of the tradeoff between the hardness of the pricing problem and the quality of the lower bound has led to several other ESPPRC relaxations such as SPPRC with 2-cycle and k-cycle free paths (Houck *et al.*, 1980; Irnich and Villeneuve, 2006), partial elementary paths (Desaulniers *et al.*, 2008), and ng-paths (Baldacci *et al.*, 2011b).

Similarly, when there are groups of complicated constraints in the subproblem hard to incorporate in combination, relaxing one type of constraint might lead to a well-solvable subproblem. The column-generation approach of Ropke and Cordeau (2005) for the DARP is an example. In the presence of two potentially conflicting temporal constraints (ordinary and dynamic time windows), traditional labeling algorithms are not able to check both. Thus, instead of including all tour constraints of the DARP in the subproblem, Ropke and Cordeau (2005) chose to relax the ride-time constraints. The resulting subproblem is well studied and can be solved effectively (Dumas *et al.*, 1991; Ropke and Cordeau, 2009). The ride-time constraints are handled with infeasible-path elimination inequalities in the master program in their algorithm. Our approach, in contrast, is to integrate both time-window and ride-time constraints into the column-generation subproblem. This generally provides stronger lower bounds at the cost of a harder to solve pricing problem. We derive two new dominance criteria leading to effective labeling procedures for this before unsolvable subproblem.

Second, concerning feasibility testing of a given DARP route, Tang *et al.* (2010) presented an $\mathcal{O}(n^2)$ algorithm (where n is the length of the route), Haugland and Ho (2010) a faster $\mathcal{O}(n \log n)$ algorithm, and recently, Firat and Woeginger (2011) a linear time $\mathcal{O}(n)$ algorithm. However, when a given feasible partial route is extended by just one node, neither of these algorithms is suited to derive a simple and efficient feasibility check. Note that the extension of a partial route by one node is a fundamental algo-

rithmic step in both local search-based heuristics and exact approaches that are based on (E)SPPRC subproblems. A by-product of the approach proposed here is a $\mathcal{O}(nM)$ labeling-based feasibility check, where M denotes the maximum number of open requests along a feasible route. As M is always bounded by the vehicle capacity and typically small in real-world applications, the proposed labeling procedure is a step towards simple and efficient feasibility checking.

Third, we provide an extensive computational study comparing several exact solution approaches for the DARP including the two strongest approaches from the literature (Ropke *et al.*, 2007; Ropke and Cordeau, 2005). It is a priori not clear if the additional effort of solving a subproblem that handles all DARP routing constraints pays off in the branch-and-bound tree. Thus, we implemented different branch-and-price algorithms based on two basic formulations that handle ride-time constraints either in the subproblem or in the master program. In algorithmic variants, different classes of valid inequalities are added to the master program to strengthen the respective formulations, and different separation strategies are investigated. Moreover, for the new branch-and-cut-and-price approach that handles all routing constraints in the subproblem, two different labeling strategies for the shortest-path computations are compared. The detailed analysis shows that the new branch-and-cut-and-price algorithm outperforms all exact solution approaches from the DARP literature.

The rest of the paper is structured as follows: Section 3.2 provides a literature review on the DARP. In Section 3.3, we give a formal definition of the DARP, present compact and extensive formulations from the literature, and outline the differences to the extensive formulation proposed in this paper. The novelty of our approach is the joint handling of static and dynamic time windows in the column-generation subproblem. For its effective solution, we develop weak and strong dominance relations between partial routes to be used in dynamic-programming labeling algorithms presented in Section 3.4. The basic components of the branch-and-cut-and-price solution algorithms are briefly discussed in Section 3.5 followed by computational experiments in Section 3.6. The paper ends with final conclusions and an outlook given in Section 3.7.

3.2 The Dial-a-Ride Problem

The DARP is a pickup-and-delivery vehicle routing problem for passenger transportation where user specified transportation requests from origin to destination points have to be served by a fleet of vehicles located at a central depot. Most of the literature on the DARP is based on specific practical applications and uses problem formulations tailored to these applications. Thus, no common problem definition exists. For a comprehensive overview on DARP variants and solution approaches we refer to recent surveys (Cordeau and Laporte, 2007; Parragh *et al.*, 2008; Cordeau *et al.*, 2008).

The DARP variant we consider in the following is the basic VRP where static and dynamic time windows occur together. It was introduced by Cordeau and Laporte (2003) and seeks to minimize the total travel distance subject to pairing, precedence, capacity, time-window and ride-time constraints. The resulting DARP is very close to other VRP

variants. In fact, it differs from the *pickup-and-delivery problem with time windows* (PDPTW) only by an additional constraint imposing a maximum time L_i between a pickup i and the corresponding delivery $i+n$, i.e., the time a request is on board of the vehicle. In other words, the ride-time constraints impose a dynamic time window $[0, L_i]$ between pickup and delivery of the request i. As a generalization of many other VRPs the DARP is $\mathcal{NP}$-hard.

Heuristic approaches on the DARP with an additional constraint on the route duration include the work of Cordeau and Laporte (2003) suggesting a tabu search algorithm and Parragh *et al.* (2010) suggesting a variable neighborhood search. These and similar heuristic approaches are indispensable for solving huge instances of the DARP as they occur in practice: They often include more than 3,000 requests per day and heterogeneous fleets with small buses and dedicated taxis. For this reason, ride-time constraints are relaxed in practice and partially enforced by appropriately adjusting regular time windows.

Regarding exact solution approaches, Cordeau (2006) developed a branch-and-cut algorithm based on a three-index formulation. Maximum route duration constraints are considered in the model, but they coincide with the time windows of the depots in the benchmark instances. Thus, they are not explicitly present there. Known inequalities from the TSP, VRP(TW) and PDPTW were adapted to the DARP and used along with new inequalities valid for the DARP. The algorithm was able to solve randomly generated benchmark instances with up to four vehicles and 36 requests.

Another branch-and-cut approach based on two different two-index formulations was proposed by Ropke *et al.* (2007). These more compact formulations and different new liftings of several families of inequalities allowed the authors to solve benchmark instances with up to eight vehicles and 96 requests, outperforming the branch-and-cut algorithm of Cordeau (2006).

A branch-and-cut-and-price algorithm was developed in (Ropke and Cordeau, 2005). This paper is an earlier version of (Ropke and Cordeau, 2009) where the PDPTW is considered. The same algorithm is used to solve the DARP in (Ropke and Cordeau, 2005). Tailored to the PDPTW, their subproblem asks for feasible PDPTW routes, which may violate ride-time constraints. Infeasible-path inequalities in the master program enforce the maximum ride times if needed. Computational results showed that the branch-and-cut-and-price approach often leads to stronger lower bounds than the branch-and-cut algorithm of Ropke *et al.* (2007).

3.3 Compact and Extensive Formulations

For a formal definition of the DARP, let n be the number of customer requests. We assume a directed graph $G = (N, A)$ with node set $N = P \cup D \cup \{0, 2n+1\}$ and arc set A. Node 0 denotes the origin and node $2n+1$ the destination depot, $P = \{1, ..., n\}$ the pickup nodes, and $D = \{n+1, ..., 2n\}$ the delivery nodes. For each customer request i consisting of a pickup node $i \in P$ and a delivery node $i+n \in D$, a maximum ride time L_i is given. At the pickup node i, $d_i \in \mathbb{N}$ passengers have to be picked up altogether.

The same passengers are dropped at the delivery node $i+n$, indicated by the negative demand $d_{i+n} = -d_i$. Furthermore, a non-negative service time s_j (with $s_0 = s_{2n+1} = 0$) and a time window $[a_j, b_j]$ in which the service has to be started are given for each node $j \in N$. This means that arriving before a_j is allowed, in which case the vehicle has to wait until time a_j to begin the service. Moreover, whenever the vehicle arrives at a node j, it can always wait and delay the start of service voluntarily. We assume that there is no restriction on the waiting time, however, the service has to be started not after b_j. The possibility to delay the start of service at some nodes is crucial for the feasibility of routes in the presence of maximum ride times (see Section 3.4.1).

A routing cost c_{ij} and a travel time t_{ij} are associated with each arc and we assume that the triangle inequality and non-negativity holds for both. We assume that from the arc set A at least the obviously infeasible arcs are excluded (see Section 3.5). Thus, the arc set A is a subset of

$$\{(i,j) \in N \times N : i \neq 2n+1, j \neq 0, j \neq i-n, a_i + s_i + t_{ij} \leq b_j, |d_i + d_j| \leq C\}.$$

A homogeneous fleet K of vehicles, each with a capacity of C is available to serve the requests. The task is to find $|K|$ DARP-feasible vehicle routes starting and ending at the depot nodes 0 and $2n+1$, so that all requests are served exactly once and the total routing costs are minimal. A route is DARP-feasible, if it satisfies the following constraints:

Pairing and precedence: Pickup node i and delivery node $i+n$ of a request i have to be visited on the same route, and i has to be visited before $i+n$.

Capacity: The number of passengers on board must not exceed C at any time.

Time windows: At each node i the service has to start within the time window $[a_i, b_i]$.

Ride time: The time between the end of service at the pickup node i and the start of service at the delivery node $i+n$ (i.e., the time request i is actually on board) must not exceed L_i for each request i.

We do not impose an explicit bound on the maximum duration of a route, as e.g., Cordeau and Laporte (2003) do. Instead, we assume that route duration constraints are given implicitly by the time windows of the origin and destination depots. Note, however, that for our stronger dominance rule Dom_{strong} the integration of a maximum route duration constraint is straightforward and causes only constant additional computational effort.

To formulate the DARP as a mixed-integer program, let x_{ij}^k be binary variables indicating if vehicle $k \in K$ uses arc $(i,j) \in A$. For each vehicle $k \in K$, let L_i^k be the time request i is on board, T_i^k be the start of service at node $i \in N$ and Q_i^k the load of vehicle k after visiting node i. For any node $i \in N$, the *in-arcs* and *out-arcs* of i are defined as $\delta^-(i) = \{(h,i) \in A : h \in N\}$ and $\delta^+(i) = \{(i,j) \in A : j \in N\}$, respectively. We use a condensed notation, where for the vector $x^k \in \{0,1\}^A$ and any subset $B \subseteq A$, the term $x^k(B)$ means $\sum_{b \in B} x_b^k$.

Then, a three-index model for the DARP is as follows (Cordeau, 2006):

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \tag{3.1}$$

$$\text{s.t.} \sum_{k \in K} x^k(\delta^+(i)) = 1 \qquad \forall i \in P, \tag{3.2}$$

$$x^k(\delta^+(i)) - x^k(\delta^+(n+i)) = 0 \qquad \forall i \in P, k \in K, \tag{3.3}$$

$$x^k(\delta^+(i)) - x^k(\delta^-(i)) = \begin{cases} 1 & i = 0, \\ -1 & i = 2n+1, \\ 0 & i \in P \cup D, \end{cases} \qquad \forall i \in N, k \in K, \tag{3.4}$$

$$T_j^k \geq (T_i^k + s_i + t_{ij}) x_{ij}^k \qquad \forall (i,j) \in A, k \in K, \tag{3.5}$$

$$a_i \leq T_i^k \leq b_i \qquad \forall i \in N, k \in K, \tag{3.6}$$

$$Q_j^k \geq (Q_i^k + d_i) x_{ij}^k \qquad \forall (i,j) \in A, k \in K, \tag{3.7}$$

$$\max\{0, d_i\} \leq Q_i^k \leq \min\{C, C + d_i\} \qquad \forall i \in N, k \in K, \tag{3.8}$$

$$L_i^k = T_{n+i}^k - (T_i^k + s_i) \qquad \forall i \in P, k \in K, \tag{3.9}$$

$$t_{i,i+n} \leq L_i^k \leq L_i \qquad \forall i \in P, k \in K, \tag{3.10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K. \tag{3.11}$$

The objective is to minimize the total routing costs (3.1). Constraints (3.2) guarantee that each request is served exactly once, and (3.3) impose pairing. The flow conservation constraints (3.4) require that each route starts at the depot 0, is continued, and finally ends in the depot $2n + 1$. Constraints (3.5) and (3.7) ensure consistency of the time and load variables with the routing variables. Time variables are bounded by the time windows (3.6), and load variables by the vehicle capacity (3.8). The ride times are defined by equations (3.9) and are bounded by the maximum ride times (3.10). Due to the non-negativity of the ride times constraints, constraints (3.10) also impose the precedences. This three-index formulation (3.1)–(3.11) is non-linear because of constraints (3.5) and (3.7). A linearization, however, is straightforward (see, e.g., Cordeau (2006) or Ropke *et al.* (2007)), but useless if the subproblem is solved with dynamic programming.

Two tighter problem formulations are presented by Ropke *et al.* (2007). Using appropriate formulated precedence inequalities, they are able to ensure pairing and precedence without the use of an additional index for the vehicle and thus to formulate the DARP with a two-index model. These mixed integer programs can be solved with standard MIP-solvers or tailored branch-and-cut algorithms.

The formulation (3.1)–(3.8) and (3.11), i.e., without constraints (3.9)–(3.10) on ride times, is the standard three-index formulation for the PDPTW. The PDPTW has been subject of intensive research (Dumas *et al.*, 1991; Desaulniers *et al.*, 2002; Ropke and Cordeau, 2009), which we can rely on because the PDPTW is a relaxation of the DARP.

Another possibility to model the ride-time constraints is to replace (3.9)–(3.10) by *infeasible-path elimination constraints* (IPEC). For this purpose, let $\mathcal{I}$ be the set of all

paths that are infeasible with respect to time window and ride-time constraints. The IPEC are of the form

$$x^k(I) \leq |I| - 1 \qquad \forall I \in \mathcal{I}, k \in K, \tag{3.12}$$

where $|I|$ denotes the length of the infeasible path $I \in \mathcal{I}$, i.e., the number of its arcs. IPEC were first successfully applied for solving the traveling salesman problem with time windows (Ascheuer *et al.*, 2000), but offer an universal approach to handling (complex) constraints in routing models that solely consist of routing variables. On the downside, since generally IPEC form an exponential family of valid inequalities, they cannot be included en masse when solving the MIP, but violated IPEC have to be separated and added dynamically.

To solve the DARP with column generation, we can partition the three-index formulation in different ways into master and subproblem constraints for applying a Dantzig-Wolfe decomposition. The constraints (3.3)–(3.11) and also the IPEC (3.12) are all non-coupling constraints as they refer to each vehicle individually. A straightforward Dantzig-Wolfe decomposition has partitioning constraints (3.2) as coupling constraints and all other constraints in the subproblem. The column-generation (or extensive) formulation, therefore, uses route variables λ_r corresponding with DARP-feasible routes r. The set of all DARP-feasible routes is denoted by Ω^{DARP}.

An alternative Dantzig-Wolfe decomposition leaves the partitioning constraints (3.2) and the IPEC (3.12) in the master program. Since IPEC replace the ride-time constraints (3.9)–(3.10) in this case, the subproblem comprises only the PDPTW constraints (3.3)–(3.8). The variables λ_r of this extensive formulation model PDPTW-feasible routes, where the set of all such routes is Ω^{PDPTW}.

In the master programs, the cost of a route $r \in \Omega^{DARP}$ or $r \in \Omega^{PDPTW}$ is c_r, respectively. For each request $i \in P$ and each route r, let $a_{ir} \in \mathbb{Z}_+$ be the number of times route r performs request i. In an elementary route, a_{ir} is binary. However, we also allow relaxations where non-elementary routes may serve a request more than once. Moreover, for any infeasible path $I \in \mathcal{I}$ and any route r, the coefficient b_{Ir} indicates how many times the route traverses arcs of that path. The integer master programs (*IMP*) of both decompositions are now presented side-by-side:

(*IMP-I*)	(*IMP*)	
$\min \sum_{r \in \Omega^{PDPTW}} c_r \lambda_r$	$\min \sum_{r \in \Omega^{DARP}} c_r \lambda_r$	(3.13)
s.t. $\sum_{r \in \Omega^{PDPTW}} a_{ir} \lambda_r = 1 \quad \forall i \in P,$	s.t. $\sum_{r \in \Omega^{DARP}} a_{ir} \lambda_r = 1 \quad \forall i \in P,$	(3.14)
$\sum_{r \in \Omega^{PDPTW}} \lambda_r = \lvert K \rvert,$	$\sum_{r \in \Omega^{DARP}} \lambda_r = \lvert K \rvert,$	(3.15)
$\sum_{r \in \Omega^{PDPTW}} b_{Ir} \lambda_r \leq \lvert I \rvert - 1 \quad \forall I \in \mathcal{I},$		(3.16)
$\lambda_r \in \{0,1\} \quad \forall r \in \Omega^{PDPTW}.$	$\lambda_r \in \{0,1\} \quad \forall r \in \Omega^{DARP}.$	(3.17)

The objective (3.13) is the minimization of the total costs, the pendant to request partitioning constraints (3.2) is (3.14), and constraints (3.15) are generalized convexity constraints (resulting from aggregation by vehicles, see Desaulniers *et al.*, 1998). The reformulation of the IPEC (3.12) in the route variables is (3.16).

The *IMP* on the left-hand side is exactly the one used by Ropke and Cordeau (2005) in the following denoted by *IMP-I* (we also add the suffix *-I* to identify its components), while the approach presented in this paper relies on the right-hand side formulation. As the number of feasible routes in the masters is too large to solve them directly, one has to rely on column-generation (or Lagrangian relaxation) techniques. The linear relaxation of (3.13)–(3.17), denoted by (MP), is solved by initializing the column-generation process with a proper subset of all routes. Missing routes are then added dynamically to this restricted master program (RMP). Integrality is ensured by integrating the column-generation process into a branch-and-bound algorithm.

While *IMP-I* and *IMP* have the same set of feasible integer solutions, the linear relaxation MP is generally stronger than *MP-I*. The reason is that in *MP-I* a subset of DARP-infeasible routes can be convex-combined to form routes that do not violate the IPEC. In *MP*, however, DARP-infeasible routes are excluded so that *MP*'s lower bound is always not smaller than *MP-I*'s lower bound. This advantage of a stronger model comes at the price of a harder to solve subproblem. In the following, we develop an effective solution procedure for the MP subproblem. We also show empirically that, with the proposed subproblem algorithm, the trade-off (between bounds and effort to gain the bounds) clearly turns towards favoring MP.

3.4 Column-Generation Subproblem and Labeling Algorithms

The task of the column-generation subproblem a.k.a. pricing problem (*PP*) is to identify negative reduced-cost routes or to prove that no such routes exist. Let $\pi = (\pi_i)_{i \in P}$, μ, and $\rho = (\rho_I)_{I \in \mathcal{I}}$ be the values of the dual variables to the *RMP* constraints (3.14), (3.15), and (3.16), respectively. Moreover, let $b_{I,ij}$ be the number of times that arc $(i,j) \in A$ is present in the path $I \in \mathcal{I}$. To identify feasible routes with negative reduced costs, one first has to compute the reduced costs of the routing variables x_{ij} as

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \pi_i - \sum_{I \in \mathcal{I}} b_{I,ij} \rho_I & \text{if } i \in P, \\ c_{ij} - \sum_{I \in \mathcal{I}} b_{I,ij} \rho_I & \text{otherwise,} \end{cases} \qquad \tilde{c}_{ij} = \begin{cases} c_{ij} - \pi_i & \text{if } i \in P, \\ c_{ij} & \text{otherwise,} \end{cases} \tag{3.18}$$

and then the following subproblem has to be solved:

$$\begin{aligned} &(PP\text{-}I) = \text{(SPPPDPTW)} && (PP) = \text{(SPPDARP)} \\ &\min \sum_{(i,j) \in A} \tilde{c}_{ij} x_{ij} - \mu && \min \sum_{(i,j) \in A} \tilde{c}_{ij} x_{ij} - \mu \end{aligned} \tag{3.19}$$

s.t. (3.3)–(3.8) and (3.11), s.t. (3.3)–(3.11),

where in the constraints (3.3)–(3.11) the index k for the specific vehicle is dropped.

A final remark about *IMP-I* subproblem for the generation of PDPTW routes seems appropriate here. The advantage of replacing (3.9)–(3.10) by IPEC, as suggested by Ropke and Cordeau, is that the reduced costs (3.18) in *PP-I* solely depend on the chosen arcs, but not on a route's schedule, i.e., values T_i^k in (3.3)–(3.11). If schedules were involved, Desaulniers *et al.* (1998) have explained that highly complex variants of SPPRC would result, where linear node costs and profits have to be included (Ioachim *et al.*, 1998).

Subproblem Solution by Dynamic-Programming Labeling Algorithms The basic principle of a labeling algorithm is the following: Starting at a distinct source node, partial paths are iteratively extended along arcs of the underlying graph until the sink node is reached and the path is complete. The partial paths are represented by labels in which attributes such as the accumulated cost or time consumption along the partial path are stored. To be able to solve such shortest-path problems, it is crucial to identify and discard useless labels so that not all possible paths are enumerated. Dominance rules accomplish this task. A more detailed discussion of general dominance principles can be found in (Desaulniers *et al.*, 1998; Irnich and Desaulniers, 2005) and in the introductory paragraph of the proofs in Section 3.A of the Appendix.

In the following, we briefly summarize the labeling procedure and dominance rules applicable to the PDPTW. Moreover, we clarify that in the additional presence of ride-time constraints, however, the standard relation to ensure dominance for the time resource is not valid anymore. Basically, it is no longer correct that serving earlier is better than serving later. Thus, there seem to be no simple attributes to model feasible DARP routes with non-decreasing resource extension functions (REFs).

PDPTW Labeling Procedure and Dominance Rules Ropke and Cordeau (2005) use an ESPPRC tailored to the PDPTW (ESPPPDPTW) to solve the DARP with formulation *IMP-I*.

In the SPPPDPTW, a feasible path must satisfy time-window, capacity, pairing and precedence constraints. For a non-elementary path, pairing and precedence means that a request is allowed to be served again, once it has been picked up *and* delivered. Hence, several pickup-and-delivery pairs of the same request can be present in a single path. For ease of notation, we assume for the rest of the paper, that all (partial) paths are elementary. All arguments, however, are similar for non-elementary paths.

Feasibility of a partial path means that time-window, capacity and precedence constraints must be respected, whereas pairing constraints need not to be satisfied for all requests. If for some pickup-and-delivery pair $(i, i+n)$ only the pickup node i is visited on the partial path, the request i is said to be *open*. It is then necessary for a feasible completion to visit the delivery nodes of those open requests. Conversely, however, partial paths visiting only the delivery node $i+n$ are not allowed.

A general prerequisite for the following dominance rule is that the reduced costs $\tilde{c}$ fulfill $\tilde{c}_{ij} \leq \tilde{c}_{ik} + \tilde{c}_{kj}$ for all arcs $(i,j),(i,k),(k,j) \in A$ with $k \in D$. Ropke and Cordeau (2009) call this property the *delivery triangle inequality* (DTI). It may be fulfilled by definition (3.18) and permits a more efficient solution of the PDPTW pricing problem (Dumas *et al.*, 1991). However, adding additional cuts to the *RMP* formulated in the x_{ij} variables may lead to reduced costs that do not satisfy the DTI. Ropke and Cordeau (2009) show how to transform the reduced-cost matrix into an equivalent one that satisfies the DTI. From now on, we assume that the DTI holds. In particular, the validity of the following and all other dominance rules in this paper is based on this assumption.

The standard dominance criterion for SPPPDPTW (Dumas *et al.*, 1991; Ropke and Cordeau, 2009) uses the following attributes for each label ℓ: the node η_ℓ the label belongs to, its (reduced) cost $\tilde{c}_\ell$, the earliest start of service t_ℓ at η_ℓ, and the set of open requests O_ℓ. Dumas *et al.* (1991) and Ropke and Cordeau (2009) have shown:

Proposition 3.1. (Dom_{PDPTW}) *A feasible label ℓ_1 dominates a label ℓ_2 if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad \textit{and} \quad O_{\ell_1} \subseteq O_{\ell_2}.$$

Ropke and Cordeau (2009) also discussed three additional aspects of Dom_{PDPTW}. First, an extension to the elementary case one just needs to keep track of the set of serviced requests U for each label and additionally require $U_{\ell_1} \subseteq U_{\ell_2}$ in the dominance rules. Second, if the DTI does not hold dominance is only possible between labels with identical sets $O_{\ell_1} = O_{\ell_2}$ (and $U_{\ell_1} = U_{\ell_2}$ in the elementary case). Third, when column generation is embedded into branch-and-bound, the use of Dom_{PDPTW} as dominance criterion limits the choice of branching rules. In particular, branching on arcs of the original problem formulation (3.1)–(3.11) is critical because it is not possible to remove an arc (i,j) when there exists a delivery node $k \in D$ such that (i,k,j) is a feasible sub-path. In this case, it is not possible to meet the DTI.

PDPTW and Ride-Time Constraints We now show that ride-time constraints are not compatible with the standard SPPPDPTW dominance rule. The main difficulty is to deal with the trade-off between serving all nodes as early as possible (promoting feasibility regarding time-window constraints) and serving pickups as late as possible (promoting feasibility regarding ride-time constraints). In general, it is no longer sufficient to solely keep track of the earliest possible start of service. An example to illustrate this is given next.

Example 3.1. Consider two labels ℓ_1 and ℓ_2 representing the respective partial paths $\mathcal{P}(\ell_1) = (0,j,i,j+n,\eta)$ and $\mathcal{P}(\ell_2) = (0,h,i,h+n,\eta)$ as shown in Table 3.1. Assume that the travel times t_{ij} are 10 between all nodes and that service times s_i are zero for all nodes. At node η, only request i is open for both labels, thus they are comparable in the SPPPDPTW sense. When considering the time-related resources of the SPPPDPTW, which are only the earliest start of service t_{ℓ_1} and t_{ℓ_2} for both labels, then ℓ_1 seems to dominate ℓ_2. This is, however, not true for the SPPDARP. If the maximum ride time

of requests i is $35 \leq L_i < 40$, then ℓ_2 can be feasibly extended to delivery node $i + n$, while ℓ_1 cannot, since it violates the ride-time constraint of requests i.

3.4.1 Weak Dominance for SPPDARP

One may ask why we present a weak dominance relation if a stronger dominance is also available (Section 3.4.3). On the one hand, the weak dominance provides some useful insights about basic information needed for applying any dominance rule, later used for the development of the strong dominance rules. On the other hand, there is a trade-off between the strength of the rule and the effort needed to implement and perform the comparison of labels: The weak dominance rule is easy to understand, simple to implement, and computationally cheap. Section 3.6, therefore, compares branch-and-price algorithms based on both weak and strong dominance rules.

Example 3.1 demonstrated that for labels with open requests, their ride-time constraints affect the feasibility of the partial paths (and possible extensions), and Dom_{PDPTW} cannot guarantee dominance for the SPPDARP. This is not an issue if there are no *active* ride-time constraints at a label. Consequently, whenever a label ℓ is feasible and represents an empty vehicle at the current node $\eta_\ell \in D$, i.e., $O_\ell = \varnothing$, it can dominate other labels according to the rule Dom_{PDPTW}.

When extending a label with $O_{\ell'} = \varnothing$ to a customer node, this needs to be a pickup node $\eta_\ell \in \mathcal{P}$. The only active ride-time constraint for the resulting label ℓ is that of the request picked up at the current node η_ℓ. Thus, there are no ride times connecting the preceding part of the path with η_ℓ, which allows to delay the start of service at η_ℓ for ℓ to at least the same time as it is possible for any other label at η_ℓ. As a result, ℓ can dominate other labels according to the rule Dom_{PDPTW} leading to the following dominance rule for the SPPDARP:

Proposition 3.2. (Dom_{weak}) *A feasible label ℓ_1 dominates a label ℓ_2, if*

$$\begin{aligned} &\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \text{ and} \\ &|O_{\ell_1}| = \begin{cases} 0 & \text{if } \eta_{\ell_1} \in D, \\ 1 & \text{if } \eta_{\ell_1} \in P. \end{cases} \end{aligned} \tag{3.20}$$

Note that condition (3.20) implies $O_{\ell_1} \subseteq O_{\ell_2}$. In contrast to the SPPPDPTW, it is not obvious whether a label ℓ is feasible or not in the SPPDARP sense. As in Example 3.1, there may exist labels ℓ_2' with a later earliest start of service representing a feasible path, while a stronger label ℓ_1' in the SPPPDPTW sense does not. Thus, for correct domination, it is necessary to ensure the feasibility of the dominating label. The SPPPDPTW attributes of a label, however, are not sufficient to decide on the label's feasibility.

Example 3.2. Consider label ℓ_2' from Example 3.1. When arriving at node $i + n$, it is not clear if the ride-time constraint of request i is satisfied or not (nor is it clear for h). Even storing the start of service at node i in ℓ_2' does not suffice to decide on the feasibility

Labels							
ℓ_1 for path $(0, ..., \eta)$	Nodes in $\mathcal{P}(\ell_1)$, $\mathcal{P}(\ell'_1)$	0	j	i	$j+n$	η	$i+n$
ℓ'_1 for path $(0, .., i+n)$	Time window $[a_i, b_i]$	$[0, 100]$	$[0, 15]$	$[20, 35]$	$[15, 30]$	$[50, 65]$	$[60, 75]$
	Earliest time t_{ℓ_1}	0	10	20	30	50	60
Labels	Earliest time t_{ℓ_2}	0	20	30	45	55	65
ℓ_2 for path $(0, ..., \eta)$	Time window $[a_i, b_i]$	$[0, 100]$	$[20, 35]$	$[20, 35]$	$[45, 60]$	$[50, 65]$	$[60, 75]$
ℓ'_2 for path $(0, .., i+n)$	Nodes in $\mathcal{P}(\ell_2)$, $\mathcal{P}(\ell'_2)$	0	h	i	$h+n$	η	$i+n$

Table 3.1: Label ℓ_1 dominates label ℓ_2 in the SPPPDPTW sense, but is inferior regarding the ride-time constraint of request i

of ℓ'_2. Suppose that $L_i = 30$, then ℓ'_2 appears to be infeasible as the actual ride time for i is $65 - 30 = 35$. Yet, it is possible to voluntarily delay the start of service at node i for 5 units. This reduces the actual ride time of request i from 35 to 30 (without affecting the service times of the succeeding nodes), which means that ℓ'_2 does indeed represent a feasible path.

Additional interdependencies between several requests and the corresponding time-window and ride-time constraints further complicate feasibility testing.

3.4.2 Labeling Algorithm with Weak Dominance

SPPDARP can be solved by a labeling algorithm that uses the new dominance rule Dom_{weak}, but apart from that is identical to the one of Ropke and Cordeau (2009) for the SPPPDPTW. The resources that need to be stored within each label ℓ are $\eta_\ell, \tilde{c}_\ell, t_\ell$ and O_ℓ as defined in Proposition 3.1. For convenience, we also include a resource for the load l_ℓ that the vehicle carries when departing from the respective node. For extending a label ℓ to a node x along an arc $(\eta_\ell, x) \in A$, one first needs to check whether this extension is feasible. Consistency with the pairing and precedence constraints for ℓ and x result from $x \notin O_\ell$ if $x \in P$, $x - n \in O_\ell$ if $x \in D$, and $O_\ell = \varnothing$ if $x = 2n + 1$. Feasibility regarding capacity and time-window constraints is guaranteed by $t_\ell + t_{\eta_\ell, x} \leq b_x$ and $l_\ell + d_x \leq C$.

Ride-Time Feasibility If a path $\mathcal{P} = (h_1, \ldots, h_q)$ is feasible in the SPPDARP sense, then there exists a time schedule $T_\mathcal{P} = (\tau_1, \ldots, \tau_q)$ satisfying

$$\tau_i \in [a_{h_i}, b_{h_i}] \quad \forall i = 1, \ldots, q, \tag{3.21}$$

$$\tau_i + s_{h_i} + t_{h_i, h_{i+1}} \leq \tau_{i+1} \quad \forall i = 1, \ldots, q-1, \tag{3.22}$$

$$\tau_i + s_{h_i} + L_{h_i} \geq \tau_j \quad \text{if } h_i + n = h_j. \tag{3.23}$$

The time schedule $T_\mathcal{P}$ is then said to be feasible. Note again, that elementarity of $\mathcal{P}$ is assumed. Otherwise, inequalities (3.23) have to be satisfied for every pair of corresponding pickup and delivery nodes.

Hence, when solving the SPPDARP, an explicit feasibility test is required if $O_{\ell'} = \varnothing$ holds for a dominating label ℓ'. Only the feasibility of a label with $\eta_{\ell'} \in P$ and $|O_{\ell'}| = 1$ follows immediately from the feasibility of its parent label ℓ together with $t_\ell + t_{\eta_\ell, \eta_{\ell'}} \leq b_{\eta_{\ell'}}$. To check if the partial path represented by ℓ' is DARP-feasible, we use a revised procedure of the feasibility test of Tang *et al.* (2010) with a worst-case time of $\mathcal{O}(n^2)$. The average time, however, should be significantly better because only the part of the path between the node where the vehicle was empty for the last time and $\eta_{\ell'}$ needs to be checked. In computational tests we observed that paths of only a few nodes need to be tested. Therefore, it seems unlikely that feasibility tests with a superior worst-case time are practically beneficial. Besides, the algorithm of Tang *et al.* (2010) can use some of the data already computed during the labeling process.

Label Generation and Elimination If the extension of ℓ along the arc (η_ℓ, x) is feasible, the new label ℓ' is created and the resources of the new label are set according to the following REFs:

$$\eta_{\ell'} = x, \quad \tilde{c}_{\ell'} = \tilde{c}_\ell + \tilde{c}_{\eta_\ell,x}, \quad t_{\ell'} = \max\{a_x, t_\ell + s_{\eta_\ell} + t_{\eta_\ell,x}\}, \quad l_{\ell'} = l_\ell + d_x, \tag{3.24}$$

$$O_{\ell'} = \begin{cases} O_\ell \cup \{x\} & \text{if } x \in P, \\ O_\ell \backslash \{x - n\} & \text{if } x \in D. \end{cases} \tag{3.25}$$

The pairing constraints enable the elimination of labels whenever there exists no feasible extension to the destination depot $2n+1$. Rules for label elimination were formulated in Dumas *et al.* (1991) in the PDPTW context. Their basic idea is that every feasible completion has to deliver all open requests $i \in O_\ell$ of a label ℓ before reaching the node $2n+1$. If there exists no path starting at η_ℓ at time t_ℓ and ending in $2n+1$ that visits at least all delivery nodes of the open requests satisfying the time-window constraints, then ℓ can be eliminated. When travel times satisfy the triangle inequality, this comes down to solving a traveling salesman problem with time windows over the nodes $\{i+n \in D : i \in O_\ell\} \cup \{\eta_\ell, 2n+1\}$, which is $\mathcal{NP}$-hard. Thus, Dumas *et al.* (1991) consider only subsets of O_ℓ with not more than two requests. We follow the same approach, but additionally make use of the fact that here a path can also be infeasible because of ride-time constraints. When using Dom_{weak}, however, no additional information about the open requests can be used as, e.g., the order in which they have been picked up. This means that, except for the earliest possible start of service at the current node t_ℓ, no label-specific information can be used to decide whether or not a ride-time feasible extension to the node $2n+1$ exists. Label elimination based on ride-time constraints can therefore only be performed due to bounds on the minimum time the requests are on board. This time depends on the time windows, travel times, and the start of service t_ℓ at the current node. Still, speedups are obtainable when accounting for maximum ride times in the label elimination.

3.4.3 Strong Dominance for SPPDARP

Proposition 3.2 has shown that when restricting the set of dominating labels according to (3.20) then the ride-time constraints can be ignored in the dominance relation between two labels. In general, however, the ride times of open requests have to be considered in the dominance rule.

To decide if in the presence of ride-time constraints a label ℓ_1 with open requests can dominate another label ℓ_2, the times in which these requests can be delivered have to be known. In particular, whenever the open requests $i \in O_{\ell_1}$ can be ride-time feasibly delivered in a completion Q of ℓ_2 it must also be possible to feasibly deliver them for ℓ_1 using the completion Q' that results from Q by leaving out those deliveries that belong to non-open requests of $\mathcal{P}(\ell_1)$ (i.e., open in $\mathcal{P}(\ell_2)$, but not open in $\mathcal{P}(\ell_1)$). As the ride-time constraints only bound the maximum time on board, delivering early is never a problem, but delivering late can be. Thus, the latest possible delivery times still

respecting the maximum ride times of the open requests are of importance. As seen in Example 3.2, the possibility to delay the start of service at pickup nodes has to be incorporated. Consequently, we always have to consider those latest possible delivery times, where the start of service at the respective pickup nodes has been delayed as much as possible without violating any other constraints, i.e., time windows of successive nodes or maximum ride times of other requests. This usually requires adapting the starts of service of other nodes on the path as well. The idea of delaying the service at pickup nodes as much as possible in order to maximize ride-time feasibility is similar to the idea of using the *forward time slack* that was originally introduced by Savelsbergh (1992) for the TSPTW and generalized by Cordeau and Laporte (2003) for the DARP.

One difficulty of this approach in a labeling algorithm is that a label ℓ represents only a partial path up to node η_ℓ, and the completion of this path is not yet known. As a result, not all constraints are known that may limit the possibility to delay the service at certain pickup nodes. To be more precise, the time windows of the nodes succeeding η_ℓ as well as the ride times of the requests that are open at node η_ℓ clearly depend on the respective extension of ℓ and are not known for ℓ. Consequently, not even the actual start of service t_ℓ at the current node η_ℓ can be specified for sure. While for some extension it may be necessary to choose t_ℓ as early as possible due to the strict time window of a succeeding node, this may not be needed for another extension where delaying the start of service t_ℓ is beneficial in order to maximize the latest delivery times of open requests. However, we need to ensure that it is always, i.e., for any feasible time the current node η_ℓ can be serviced, possible that a dominating label ℓ_1 can deliver its open requests if a dominated label ℓ_2 can do also. We show (in Proposition 3.3) that this property is, along with Dom_{PDPTW}, in fact sufficient to guarantee dominance in the presence of time-window and ride-time constraints.

For a strong dominance criterion, we therefore store, in addition to the earliest start of service t_ℓ at the current node η_ℓ, for all open request $i \in O_\ell$ the *latest possible delivery time* $ld^i_\ell(t)$ as a function of the start of service t at η_ℓ. In other words, $ld^i_\ell(t)$ is the latest delivery time for request i, i.e., the latest ride-time feasible start of service at the delivery node $i+n$, where the start of service at its pickup node i has been delayed as much as possible. No constraints that are already known for the partial path up to η_ℓ are violated, and the start of service at η_ℓ is not delayed beyond t.

To formalize $ld^i_\ell(t)$, let $\mathcal{P}(\ell) = (h_1, ..., h_q)$ with $h_q = \eta_\ell$ be the path represented by label ℓ. For any path $\mathcal{P}$, let $\mathscr{T}_\mathcal{P}(t)$ be the *set of all feasible time schedules* $(\tau_1, ..., \tau_q)$ with $\tau_q \leq t$. Then, $ld^{h_i}_\ell(t), t \geq t_\ell, h_i \in O_\ell$ can be defined as

$$ld^{h_i}_\ell(t) = \min\left\{b_{h_i+n}, \max_{T_{\mathcal{P}(\ell)}=(\tau_1,...,\tau_q)\in\mathscr{T}_{\mathcal{P}(\ell)}(t)} \{\tau_i\} + s_{h_i} + L_{h_i}\right\}. \tag{3.26}$$

To maximize τ_i, generally, the starts of service τ_j at several nodes h_j of the path $\mathcal{P}$ need to be changed compared to the respective earliest starts of service. However, this does not lead to any conflicts when the latest delivery times have to be computed for more than one request. In fact, maximizing τ_i for some h_i has no restricting effect on the maximization of τ_j for all other nodes $h_j \neq h_i$ on path $\mathcal{P}$. The following lemma shows

that, for any path $\mathcal{P}$, there exists a unique time schedule where simultaneously the start of service is maximal for all nodes.

Lemma 3.1. *Let $\mathcal{P} = (h_1, ..., h_q)$ be a feasible partial path, and let $\tau_i^*(t) = \max_{T_\mathcal{P}=(\tau_1,...,\tau_q)\in\mathscr{T}_\mathcal{P}(t)} \{\tau_i\}$ be the maximum value for the start of service at node h_i with $t \leq \tau_q$. Then, $T_\mathcal{P}^*(t) = (\tau_1^*(t), ..., \tau_q^*(t)) \in \mathscr{T}_\mathcal{P}(t)$.*

Proofs of this and all other lemmas and propositions can be found in Section 3.A of the Appendix.

Integrating the information $ld_\ell^i(t)$ on the latest possible delivery times for each open request $i \in O_\ell$ into the dominance relation leads to the following dominance rule for the SPPDARP:

Proposition 3.3. *(Dom^*_{strong}) A feasible label ℓ_1 dominates a label ℓ_2, if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \text{ and} \tag{3.27}$$

$$ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t) \quad \forall i \in O_{\ell_1}, t \in [t_{\ell_2}, b_{\eta_{\ell_2}}]. \tag{3.28}$$

Equations (3.28) ensure that for all open requests $i \in O_{\ell_1}$ of the dominating label ℓ_1 and for each time t that may represent a feasible start of service at η_{ℓ_2} for ℓ_2, it is feasible to deliver request i for ℓ_1 whenever it is feasible for ℓ_2. Note that the functions $ld_\ell^i(t)$ give the latest possible delivery time of request i for ℓ when *allowing* the start of service at η_ℓ to be delayed up to time t. In particular, this does not require that t actually *is* a feasible start of service at η_ℓ. The implication for Dom^*_{strong} and equations (3.28) is that it is not necessary for the start of service of the dominating label ℓ_1 to actually reach all feasible times t for the start of service of ℓ_2, as long as $ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t)$ holds for all $i \in O_{\ell_1}, t \in [t_{\ell_2}, b_{\eta_{\ell_2}}]$.

Note that it is straightforward to extend Dom^*_{strong} to a valid dominance rule for the subproblem of the DARP with an additional constraint on the route duration. Introducing a dummy request with origin and destination depots as pickup and delivery nodes and setting its maximum ride time equal to the maximum route duration guarantees dominance regarding the maximum route duration.

Since the dominance rule of Proposition 3.3 requires the comparison of several functions in (3.28), the dominance relation is practically not yet applicable in a labeling algorithm. We therefore characterize the shape of the functions $ld_\ell^i(t)$ next enabling a version of Dom^*_{strong} that is easy to handle.

Proposition 3.4. *Let ℓ be a feasible label at node η_ℓ with earliest start of service t_ℓ and open requests O_ℓ. The functions $ld_\ell^i(t), t \geq t_\ell$ are of the form $\min\{k_1^i + t, k_2^i\}$ for all $i \in O_\ell$, where k_1^i and k_2^i are constants.*

The visualization of the result of Proposition 3.4 together with notation is given in Figure 3.1. Using Proposition 3.4, Dom^*_{strong} can be simplified to a dominance rule where instead of having to compute and compare the entire functions $ld_\ell^i(t)$ it is sufficient to store and check the values of $ld_\ell^i(t)$ only at two distinct points of time. We have chosen

Resource	Description
η_ℓ	The node of the label
$\tilde{c}_\ell$	The current reduced cost
t_ℓ	The earliest start of service at the current node η_ℓ
l_ℓ	The current load
O_ℓ	The set of open requests
$ld^i_\ell(t)$	The latest possible delivery time of request $i \in O_\ell$ as a function of the start of service t at node η_ℓ
B^i_ℓ	The point of time when $ld^i_\ell(t)$ becomes constant
$\tilde{b}_{\eta_\ell}$	The latest feasible start of service at node η_ℓ

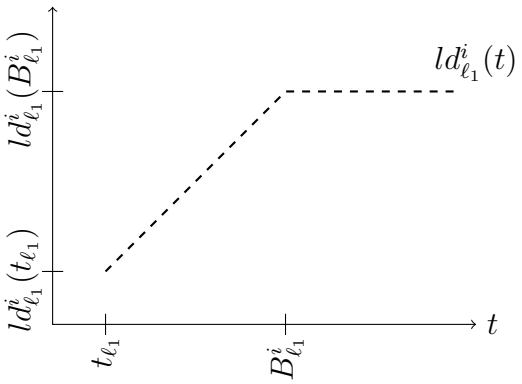


Figure 3.1: Resources of a label ℓ used within the labeling algorithms.

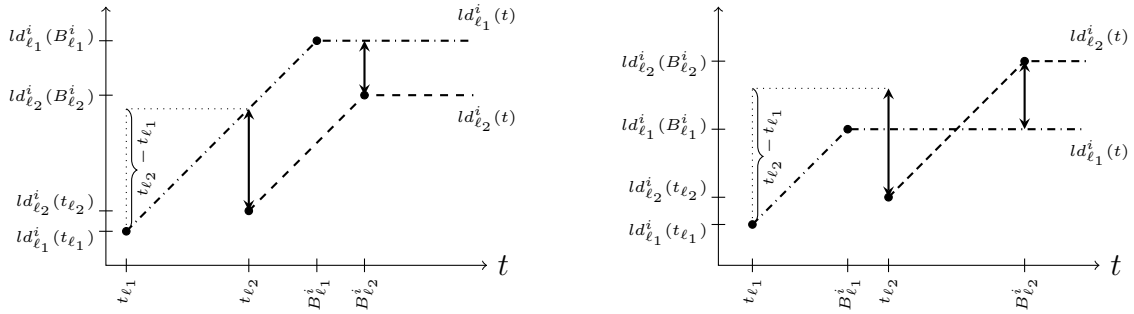


Figure 3.2: Dominance condition for ld^i fulfilled (left hand side) and not fulfilled (right hand side).

the earliest possible start of service t_ℓ and time B^i_ℓ where the latter is the time when $ld^i_\ell(t)$ becomes constant. This choice also enables an elegant way to compute the respective values of $ld^i_\ell(t)$ using REFs (see next subsection). Then a simplified version of Dom^*_{strong} is as follows (see Figure 3.2):

Proposition 3.5. (Dom_{strong}) *A label ℓ_1 dominates a label ℓ_2, if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \text{ and}$$
$$ld^i_{\ell_1}(t_{\ell_1}) + (t_{\ell_2} - t_{\ell_1}) \geq ld^i_{\ell_2}(t_{\ell_2}) \text{ and } ld^i_{\ell_1}(B^i_{\ell_1}) \geq ld^i_{\ell_2}(B^i_{\ell_2}) \quad \forall i \in O_{\ell_1}.$$

3.4.4 Labeling Algorithm with Strong Dominance

The basic course of the labeling algorithm with dominance rule Dom_{strong} is the same as for Dom_{weak}. In what follows, we only describe additional aspects. According to Proposition 3.5, the two additional resources $ld^i_\ell(t_\ell)$ and $ld^i_\ell(B^i_\ell)$ for each open request $i \in O_\ell$ have to be stored within each label ℓ. Furthermore, the computation of $ld^i_\ell(B^i_\ell)$ requires carrying along B^i_ℓ for each $i \in O_\ell$ as an additional resource.

With $ld^i_\ell(B^i_\ell)$, precise information on the actual ride times of the open requests $i \in O_\ell$ and hence on the feasibility of a label ℓ is available. Namely, the extension of ℓ along an arc (η_ℓ, x) can only be feasible if $t_\ell + s_{\eta_\ell} + t_{\eta_\ell,x} \leq ld^i_\ell(B^i_\ell)$ holds for all $i \in O_\ell$. Otherwise the ride-time constraint of at least one open request cannot be satisfied. When applying the label elimination rules described later, this consistency check on the ride-time constraints is redundant.

Proposition 3.6. *Let the label ℓ' result from the extension of a label ℓ along the arc (η_ℓ, x). Then, the resources $ld^i_\ell(t_\ell)$, $ld^i_\ell(B^i_\ell)$, and B^i_ℓ for all open request $i \in O_\ell$ are either initialized or updated according to the following REFs:*

$$B^i_{\ell'} = \begin{cases} \min\{\tilde{b}_x, b_{x+n} - s_x - L_x\} & \text{if} \quad i = x, \\ \max\{t_{\ell'}, \min\{B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x}, \tilde{b}_x\}\} & \text{otherwise}, \end{cases} \tag{3.29}$$

$$ld^i_{\ell'}(t_{\ell'}) = \begin{cases} \min\{t_{\ell'} + s_x + L_x, b_{x+n}\} & \text{if} \quad i = x, \\ ld^i_\ell(t_\ell) + (\min\{t_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x}, B^i_\ell\} - t_\ell) & \text{otherwise}, \end{cases} \tag{3.30}$$

$$ld^i_{\ell'}(B^i_{\ell'}) = \begin{cases} B^i_{\ell'} + s_x + L_x & \text{if} \quad i = x, \\ ld^i_\ell(B^i_\ell) - \max\{0, B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x} - \tilde{b}_x\} & \text{otherwise}, \end{cases} \tag{3.31}$$

where

$$\tilde{b}_x = \begin{cases} b_x & \text{if } x \in P, \\ \min\{b_x, ld^{x-n}_\ell(B^{x-n}_\ell)\} & \text{if } x \in D. \end{cases} \tag{3.32}$$

A detailed example of applying the labeling algorithm with the strong dominance rule is given in Section 3.B of the Appendix.

Label elimination is performed in the same way as for Dom_{weak}. When using Dom_{strong}, however, valuable information on the actual ride times of the open requests is available through the resources $ld^i_\ell(B^i_\ell), i \in O_\ell$. With this information, label elimination based on maximum ride-time constraints is very effective.

3.4.5 A Pseudo-Linear Feasibility Test for the DARP

A by-product of having the information $ld^i_\ell(B^i_\ell)$ on the latest possible delivery times is that we can decide on the feasibility of (partial) paths. Thus, the labeling algorithm with Dom_{strong} also serves as a feasibility test for a given DARP route.

The worst-case complexity of this test is easy to analyze: If the number of open requests can be bounded by a number M, then the number of resources at each node

and the complexity of the updates (3.24)–(3.25) and (3.29)–(3.31) is $\mathcal{O}(M)$. Hence, the feasibility test requires $\mathcal{O}(nM)$ time and space, where n is the length of the route. Assuming integer values for node demands d and vehicle capacity C (which seems natural for passenger transportation), then, e.g., $M \leq C$ holds. The travel and service times in combination with the time windows allow similar estimations. Thus, feasibility testing with the labeling procedure is pseudo-linear.

Considering other work on feasibility testing for the DARP, Cordeau and Laporte (2003) proposed the eight-step route evaluation procedure that runs in $\mathcal{O}(n^2)$. Neither this algorithm nor the tests of Firat and Woeginger (2011), Haugland and Ho (2010) and Tang *et al.* (2010) are suited to derive a simple and efficient check in the SPPRC context, when a given feasible partial route is extended node by node. For example, the test by Firat and Woeginger (2011) is based on a reformulation of the problem into a cycle-detection problem on an appropriate graph that structurally differs for every new path. Our labeling algorithm, however, allows to decide on the feasibility of such an extension in $\mathcal{O}(M)$.

3.4.6 Refinements of the Strong Dominance

We briefly discuss some refinements of the dominance rule for the strong labeling algorithm that are helpful to speedup the pricing process.

The only point where the labeling algorithm with Dom_{strong} of Section 3.4.4 utilizes the information on the actual ride times is for label elimination. We now show that actual ride times are useful for the determination of the resources $B^i_{\ell'}$ and $ld^i_{\ell'}(B^i_{\ell'})$, and therewith to improve the dominance Dom_{strong}. More precisely, let ℓ' result from the extension of a label ℓ to node x so that $\mathcal{P}(\ell') = (h_1, ..., h_q = x)$. When computing the values $B^{h_i}_{\ell'}$ and $ld^i_{\ell'}(B^{h_i}_{\ell'})$, we consider inequalities (3.21) and (3.22), and inequalities (3.23) for requests already picked up and delivered. No information on the ride-time constraints of the open requests is included. Instead of ignoring these maximum ride times, they can be used to impose additional constraints on the service time τ_q of the last node $x = h_q$. Consequently, each feasible partial schedule must respect $\tau_q + s_x + t_{x,h_i+n} \leq ld^i_\ell(B^{h_i}_\ell)$ for all $h_i \in O_{\ell'}$ in order to be feasibly completed. These constraints bound τ_q. Using this bound in equation (3.32), a tighter upper bound $\hat{b}_x$ instead of $\tilde{b}_x$ can be computed. As a result, the values $B^i_{\ell'}$ and $ld^i_{\ell'}(B^i_{\ell'})$ may also decrease.

Obviously, when there are two or more open requests, a feasible completion must successively visit the delivery nodes of all the open requests. Depending on the delivery order of the open requests different bounds on τ_q result. Therefore, the computational effort to find the strongest bound on τ_q can become prohibitively large because all permutations have to be considered. In order to reduce the effort, we take a similar approach as in the label elimination strategy and consider only subsets of $O_{\ell'}$ of not more than two requests.

Summarizing, the labeling algorithm is then altered in the following way: Prior to the propagation of the resources $B^i_{\ell'}$, $ld^i_{\ell'}(t_{\ell'})$, and $ld^i_{\ell'}(B^i_{\ell'})$ an upper bound $\hat{b}_x$ on the latest feasible start of service at the current node x is computed. In the REFs (3.29)–(3.31), the value $\tilde{b}_x$ is then replaced by $\hat{b}_x$. Algorithm 1 outlines the label-extension and

label-elimination step of the refined labeling algorithm with strong dominance.

Algorithm 1: Label-extension and label-elimination step of the refined labeling algorithm with strong dominance

input : Label ℓ, arc (η_ℓ, x)
output: New label ℓ' resulting from the extension of ℓ along arc (η_ℓ, x), if feasible

```
if Extension of ℓ along (η_ℓ, x) is feasible then
    Create label ℓ'
    Compute η_ℓ', c̃_ℓ', t_ℓ', l_ℓ', and O_ℓ' according to (3.24)–(3.25)
    Determine refined bound b̂_x on the latest start of service at x
    for all i ∈ O_ℓ' do
        Compute B^i_ℓ', ld^i_ℓ'(t_ℓ'), and ld^i_ℓ'(B^i_ℓ') according to (3.29)–(3.31)
    Perform label elimination using information ld^i_ℓ'(B^i_ℓ') of open requests i ∈ O_ℓ'
```

Bounding the start of service at the current node has two positive effects on the dominance rule: First, the dominance relation is less restrictive, since it is a relaxed condition to require $ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t)$ for all $t \in [t_{\ell_2}, \hat{b}_{\eta_{\ell_2}}]$ rather than for all $t \in [t_{\ell_2}, \tilde{b}_{\eta_{\ell_2}}]$. Second, the possible bounding effect on B^i_ℓ and $ld^i_\ell(B^i_\ell)$ gets stronger the more requests are open. This increases the possibility that $ld^i_\ell(B^i_{\ell_1}) \geq ld^i_\ell(B^i_{\ell_2})$ holds for two labels ℓ_1 and ℓ_2 with $O_{\ell_1} \subset O_{\ell_2}$, compared to not using the additional bounds.

3.5 Branch-and-Price Algorithms

We now briefly describe the main components of our implementations of the branch-and-price and branch-and-cut-and-price algorithms that are compared in Section 3.6. Mainly, we analyze the difference between using the two formulations *IMP-I* and *IMP*. For a fair comparison, we do not only compare with the results of Ropke and Cordeau (2005) directly, but we implemented versions for solving *IMP-I*, i.e., where the pricing problem is an SPPPDPTW and ride-time constraints are enforced on the master problem level. The use of formulation *IMP* requires that a SPPDARP subproblem is solved, and we also compare the two new labeling algorithms of Sections 3.4.2 and 3.4.4. Apart from the different handling of the maximum ride-times, all approaches follow the same basic algorithm.

Preprocessing As preprocessing steps, we use time-window tightening and arc-elimination rules proposed for the VRPTW and tailored to the PDPTW and the DARP (Desrochers *et al.*, 1992; Dumas *et al.*, 1991; Cordeau, 2006). A comprehensive description of these rules can be found in (Cordeau, 2006; Ropke and Cordeau, 2005), where the authors also comment on additional difficulties that arise when applying preprocessing techniques to the DARP.

Pricing Problem In each column-generation iteration, an SPPRC pricing problem has to be solved to generate routes with negative reduced costs. For accelerating the

column-generation process, it is often beneficial to solve the pricing problem heuristically, instead of always using an exact method. When the heuristics fail to find negative reduced-cost routes, an exact method has to be invoked such as those described in Section 3.4. Our implementation comes with two straightforward pricing heuristics: The first is solving the pricing problem on a reduced network only. The other is for the SPPDARP only and solves the SPPPDPTW, i.e., it ignores the ride-time constraints and drops all ride-time infeasible routes.

Preliminary computational tests indicated that the benefits from using these pricing heuristics appear to be rather high for the SPPDARP labeling algorithm with Dom_{weak}. Speedups are small, however, when using Dom_{strong} for SPPDARP or when solving the SPPPDPTW.

Cutting Planes Cordeau (2006), Ropke *et al.* (2007) and Ropke and Cordeau (2009) have proposed several valid inequalities for the PDPTW and the DARP. These valid inequalities can be used in branch-and-cut and also in branch-and-price (resulting in a branch-and-cut-and-price algorithm), using the correspondence for arc flows

$$x_{ij} = \sum_{k \in K} x_{ij}^k = \sum_{r \in \Omega} a_{ij,r} \lambda_r \qquad \forall (i,j) \in A \tag{3.33}$$

between two-index and three-index compact formulations and extensive formulations (the coefficient $a_{ij,r}$ is the number of times route r traverses arc (i,j)). For the sake of clarity, we distinguish between variables and their actual values, denoted by $\bar{x}_{ij}$ for all arcs $(i,j) \in A$.

We use the following classes of valid inequalities in our branch-and-cut-and-price algorithm: tournament constraints and another lifting of the IPEC, rounded capacity inequalities, 2-path cuts, and fork inequalities. Section 3.C of the Appendix gives further details on these inequalities and their separation. For the computational studies, we distinguish between approaches that solely use IPEC and their liftings and those that make use of all the valid inequalities. The latter case is indicated with an index ^{cut}.

Moreover, some separation procedures rely on checking ride-time constraints. Those that utilize a SPPDARP-based separation procedure, i.e., ride-time constraints are taken into account in a comprehensive way, are indicated with an index $_{sepRT}$. Details on the different separation procedures are provided in Section 3.C of the Appendix.

Branching Strategy and Node Selection If the solution of *MP* or *MP-I* is fractional, we use two different branching rules to obtain integer solutions. First, using (3.33) we branch on the number $\bar{x}(\delta^+(0))$ of vehicles if fractional, and create the two branches $x(\delta^+(0)) \leq \lfloor \bar{x}(\delta^+(0)) \rfloor$ and $x(\delta^+(0)) \geq \lceil \bar{x}(\delta^+(0)) \rceil$. This rule was proposed by Desrochers *et al.* (1992). Second, we branch on the outflow of a node set S as proposed by Naddef and Rinaldi (2002) where we restrict the choice of S to sets of size two. A set S with outflow $\bar{x}(\delta^+(S))$ closest to 1.5 is chosen and the two branches $x(\delta^+(S)) \leq 1$ and $x(\delta^+(S)) \geq 2$ are created. For each branch of either branching decision, an additional

linear constraint is added to the master program. No structural changes have to be made on the subproblem.

The node selection strategy is best first, and no upper bounds are given to the algorithms.

3.6 Computational Results

To compare the proposed branch-and-cut-and-price algorithms and existing exact solution approaches from the literature, we use the benchmark instances for the DARP introduced by Cordeau (2006) and larger instances with the same characteristics later added by Ropke *et al.* (2007). There are two sets of randomly generated Euclidean instances with tighter (type `a`) and less tight (type `b`) maximum ride times L_i. The instances are labeled in the form `aK-n` and `bK-n`, where `K` indicates the number of available vehicles and `n` the number of requests. The largest instance for both types consists of 8 vehicles and 96 requests. A detailed description of the instances can be found in Cordeau (2006) and the entire benchmark is available at `http://www.hec.ca/chairedistributique/data/darp`.

In our computational study, we compare two basic approaches, i.e., handling the ride-time constraints in the master problem based on formulation *IMP* and in the pricing problem based on formulation *IMP-I*. The respective linear relaxations are *MP* and *MP-I*. Both approaches are varied using different algorithmic components explained in the following paragraph. Moreover, we compare our approaches to the branch-and-cut algorithm of Ropke *et al.* (2007) denoted by *B&C*, and the branch-and-cut-and-price algorithm of Ropke and Cordeau (2005) denoted by *B&P-RC*.

The pure set-partitioning formulation *IMP-I* uses the lifted IPEC (see Section 3.C of the Appendix, eqs. (3.34) and (3.35)) to ensure the ride-time constraints, but no additional cuts. Since the ride-time constraints are not handled in the pricing problem, the resulting subproblem is a SPPPDPTW which we solve using a labeling algorithm with Dom_{PDPTW}. In this case, IPEC are always needed to enforce the maximum ride times in the master program. In contrast, *IMP-*I^{cut} refers to the version where 2-path cuts, rounded capacity inequalities, and fork inequalities are separated (see Section 3.C of the Appendix). Following the notation of Section 3.5, *IMP-*I_{sepRT} and *IMP-*I^{cut}_{sepRT} integrate the ride-time constraints in all separation procedures, while *IMP-I* and *IMP-*I^{cut} do not.

The approaches based on *IMP* where the subproblem is an SPPDARP solved using the proposed labeling algorithm with Dom_{weak} and Dom_{strong} are denoted by IMP_{weak} and IMP_{strg}, respectively. As before, a superscript ^{cut} indicates that cuts are used, giving rise to *IMP* and IMP^{cut}. All algorithms are summarized in Table 3.2.

In preceding experiments, we found that there are only small improvements in the root lower bounds when the elementary variants of the pricing problems replace the non-elementary. Due to the hardness of the elementary subproblem the labeling algorithms are slower. Overall, this results in slightly longer computation times for the entire branch-and-price algorithms. These findings coincide with those of Ropke and Cordeau

Algorithm	*Acronym*	*B&C* (Ropke et al., 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*IMP-I*	*IMP-I*	*IMP-I*	*IMP-I*	*IMP-I*	*IMP*	*IMP*	*IMP*	*IMP*
Formulation		2-Index	*IMP-I*	*IMP-I*	$IMP\text{-}I^{cut}$	$IMP\text{-}I_{sepRT}$	$IMP\text{-}I^{cut}_{sepRT}$	IMP_{weak}	IMP^{cut}_{weak}	IMP_{strg}	IMP^{cut}_{strg}
Subproblem											
SPPPDPTW			•	•	•	•	•	•	•	•	•
SPPDARP											
–weak dominance	*weak*							•	•	•	•
–strong dominance	*strg*								•		•
Cutting planes											
lifted IPEC	(⌣)	•	•	•	•	•	•		•		•
rounded cap. ineq.	*cut*	•	•	•	•		•		•		•
2-path cuts	*cut*	•	•	•	•		•		•		
fork ineq.	*cut*	•	•	•	•		•				
other cuts	(⌣)	•	•	•							
Ride-time check in											
separation basic	(⌣)	?	?	•	•		•				
sets & sequences	*sepRT*	?	?			•	•	•	•	•	•

Table 3.2: Overview of DARP solution approaches used for comparison

(2005). As a consequence, we decided to not report any computational results for the elementary case.

All of our algorithms were coded in C++ using the callable library of CPLEX 12.2 to re-optimize the RMP. Arc costs and travel times are computed with double precision. For stability reasons, we replaced partitioning constraints (3.14) by covering constraints (du Merle *et al.*, 1999). The computations were carried out on a standard PC with an Intel(R) Core(TM)2 Duo E8400 at 3.0 GHz with 4.0 GB main memory using a single thread only.

3.6.1 Linear Relaxation Results

We start the analysis with results on the linear relaxation of the master program (3.13)–(3.17). Tables 3.3 and 3.4 show details for the lower bound values lb and summarize the computation times. The notation in the tables has the following meaning:

opt	The value of an optimal solution
# opt	The number of optimal solutions obtained by respective algorithm
# best	The number of times the respective algorithm provides the best lb of all considered algorithms
% gap	The average percentage integrality gap $(opt - lb)/opt$
avg. time	The average computation time (in seconds) per instance

More detailed tables with individual computation times per instance can be found in Section 3.D of the Appendix. Note that for *B&C* and *B&P-RC* the computation times were reported for different computers so that a direct comparison is critical. According to `http://www.spec.org/cpu2006/results` our computer is approximately twice as fast as an AMD Opteron 254 which is in turn about 15% faster (`http://www.spec.org/cpu2000/#Results`) than the AMD Opteron 250 used by Ropke *et al.* (2007) and Ropke and Cordeau (2005).

Lower Bounds The computed lower bound values lb for linear relaxations show that integrating ride-time constraints into the subproblem yields significantly stronger lower bounds compared to handling ride times in the master program. This holds for both approaches with and without cuts. For the type `a` instances, lower bounds obtained with *MP* are even stronger than those obtained by *MP-I*-based approaches with additional cuts. In general, the differences in the lower bound values lb between approaches with and without ride-time constraints in the subproblem are larger for the `a` than for the `b` instances. This results mainly from the fact that the integrality gaps of formulations *MP-I* are larger for the type `a` than for the `b` instances. Compared to the branch-and-cut algorithm *B&C*, the column-generation approaches are typically superior in terms of lower bound values for the type `b` instances, while for the type `a` instances only the *MP*-based formulations, *MP-*I^{cut}_{sepRT}, and *B&P-RC* yield better lower bounds than *B&C*.

MP^{cut}_{strg} produces for 16 out of 21 instances of the more constrained type `a` instances integer optimal solutions in the root node. The version without cuts, i.e., MP_{strg}, still

Instance	*opt*	*B&C* (Ropke *et al.*, 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*MP-I*	*MP-I*^{cut}	*MP-I*$_{sepRT}$	*MP-I*$^{cut}_{sepRT}$	*MP*$_{weak}$	*MP*$_{strg}$	*MP*$^{cut}_{weak}$	*MP*$^{cut}_{strg}$
a2-16	294.2	*	*	294.0	294.0	294.0	*	*		*	
a2-20	344.8	*	*	343.7	*	*	*	*		*	
a2-24	431.1	430.3	430.4	430.6	430.6	430.6	430.6	*		*	
a3-24	344.8	*	*	339.4	343.5	341.7	344.5	*		*	
a3-30	494.8	*	*	490.5	493.5	490.9	*	*		*	
a3-36	583.2	579.0	579.0	576.0	576.1	576.7	578.7	579.0		579.0	
a4-32	485.5	*	*	484.0	485.3	484.1	*	*		*	
a4-40	557.7	553.9	556.6	553.7	553.7	553.7	556.9	*		*	
a4-48	668.8	666.5	668.1	663.2	664.1	664.1	*	667.4		*	
a5-40	498.4	*	*	497.4	498.3	497.4	497.9	*		*	
a5-50	686.6	680.0	680.8	671.9	675.8	673.3	681.8	684.0		686.3	
a5-60	808.4	804.1	*	805.5	806.1	806.0	806.9	808.0		*	
a6-48	604.1	*	*	601.9	602.2	602.2	*	*		*	
a6-60	819.2	816.2	819.1	814.2	814.9	815.5	*	819.2		*	
a6-72	916.0	910.1	913.6	904.5	906.0	905.7	913.4	913.9		914.5	
a7-56	724.0	718.5	720.9	717.1	718.4	717.3	718.3	721.8		721.8	
a7-70	889.1	886.7	888.8	883.8	885.7	884.7	886.5	*		*	
a7-84	1033.4	1025.2	1028.6	1022.9	1029.2	1024.4	1032.0	1029.8		*	
a8-64	747.5	743.7	747.3	741.9	742.8	743.1	745.7	*		*	
a8-80	945.7	938.1	940.3	925.7	930.2	928.0	942.2	944.6		945.1	
a8-96	1229.7	1213.4	1224.5	1205.3	1212.8	1209.1	1225.1	1228.8		*	
# opt		7	8	0	1	1	7	11		16	
# best		7	8	0	1	1	7	13		21	
% gap		0.42	0.20	0.88	0.60	0.74	0.22	0.12		0.06	
avg. time		?	?	3.7	19.9	5.6	60.4	9.0	1.4	33.7	2.1

Table 3.3: Lower bound values *lb* for type a instances, * if lower bound = opt.

solves eleven of the type a instances in the root node. For the type b instances, these numbers decrease to ten and eight, respectively. For the type a instances, the maximum number of solved instances over all other approaches is eight. In contrast, *B&P-RC* solves the same type b instances as MP^{cut}_{strg}. Here, the strength of the *B&P-RC* algorithm results from the use of additional families of valid inequalities.

Both the formulation *MP* and *MP-I* are significantly strengthened when using 2-path cuts, rounded capacity cuts, and fork inequalities. Moreover, integrating the ride-time information into all separation procedures (algorithms with suffix $_{sepRT}$) raises the lower bounds. This is particularly beneficial for the type a instances.

Computation Times We now compare the computation times for solving the linear relaxations *MP-I* and *MP*. The most important finding is that MP_{strg} and MP^{cut}_{strg} algorithms have consistently smaller (not longer) computation times compared to the corre-

Instance	*opt*	*B&C* (Ropke *et al.*, 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*MP-I*	*MP-I*^{cut}	*MP-I*$_{sepRT}$	*MP-I*$^{cut}_{sepRT}$	*MP*$_{weak}$	*MP*$_{strg}$	*MP*$^{cut}_{weak}$	*MP*$^{cut}_{strg}$
b2-16	309.4	308.1	*	309.3	309.3	309.4	*	*		*	
b2-20	332.6	*	*	*	*	*	*	*		*	
b2-24	444.7	444.5	444.5	444.6	444.6	444.6	444.6	444.5		444.6	
b3-24	394.5	392.9	392.2	392.2	393.9	392.2	393.9	392.2		393.9	
b3-30	531.4	*	*	*	*	*	*	*		*	
b3-36	603.8	*	*	*	*	*	*	*		*	
b4-32	494.8	*	*	*	*	*	*	*		*	
b4-40	656.6	*	*	*	*	*	*	656.6		*	
b4-48	673.8	671.9	673.2	672.9	673.0	673.0	673.0	672.9		673.2	
b5-40	613.7	611.1	*	613.5	*	613.5	*	613.5		*	
b5-50	761.4	756.2	*	*	*	*	*	*		*	
b5-60	902.0	893.9	898.3	895.9	896.9	896.5	897.9	896.7		898.9	
b6-48	714.8	*	*	714.7	714.7	714.7	714.7	*		*	
b6-60	860.1	*	*	*	*	*	*	*		*	
b6-72	978.5	963.1	975.7	974.1	975.3	974.1	975.4	975.7		977.0	
b7-56	824.0	808.3	822.2	821.7	822.0	821.7	822.0	820.3		822.2	
b7-70	912.6	907.2	911.1	906.5	911.4	906.5	911.4	906.6		911.7	
b7-84	1203.4	1193.2	1202.0	1201.9	1202.0	1201.9	1202.0	1201.3		1202.0	
b8-64	839.9	834.7	836.9	836.4	837.4	836.4	838.0	836.6		838.1	
b8-80	1036.3	1032.6	1036.2	1035.7	1035.7	1035.7	1036.2	1035.9		1036.2	
b8-96	1185.6	1165.1	1181.5	1181.4	1181.9	1181.4	1182.2	1181.3		1183.8	
# opt		7	10	7	8	7	9	8		10	
# best		7	11	8	11	8	13	8		20	
% gap		0.51	0.12	0.18	0.11	0.18	0.10	0.18		0.07	
avg. time		?	?	1.8	6.2	1.9	4.3	5.0	1.2	35.4	3.4

Table 3.4: Lower bound values *lb* for type b instances, * if lower bound = opt.

sponding *MP-I*-based approaches, even though the solution of a single pricing problem is much harder for the SPPDARP compared to the SPPPDPTW. Since the *MP-I*-based approaches generate routes that generally do not satisfy the ride-time constraints, many more violated valid inequalities, in particular IPEC, need to be added here. Even more, the necessity to repeatedly solve pricing problems and separation problems to achieve an optimal ride-time feasible solution imposes that many more pricing and separation problems have to be solved. This complicates and slows down the re-optimization of the master program. As a result, the overall computation times of the *MP-I*-based approaches exceed those that are *MP*-based.

The analysis of the different dominance rules Dom_{weak} and Dom_{strong} for the SPPDARP yields the following result: weak dominance based algorithms MP_{weak} and MP^{cut}_{weak} are slower than strong dominance based algorithms MP_{strg} and MP^{cut}_{strg}. However, in both cases computation times are comparable for most instances and seem still acceptable.

Next we compare computation times for $MP\text{-}I_{sepRT}$ and $MP\text{-}I^{cut}_{sepRT}$, i.e., approaches with full consideration of ride-time feasibility, with $MP\text{-}I$ and $MP\text{-}I^{cut}$. Recall that for the type a instances, full ride-time consideration produces significantly stronger lower bounds. With an increasing number of requests, the computation times also become longer, in particular for $MP\text{-}I^{cut}_{sepRT}$. The reasons are that the separation procedures with full consideration of ride-time feasibility are much harder to solve, better bounds generally require more time, and more valid inequalities are separated in the variants $MP\text{-}I_{sepRT}$ and $MP\text{-}I^{cut}_{sepRT}$. Interestingly, for the type b instances, the opposite is true for the solution times of $MP\text{-}I^{cut}_{sepRT}$ and $MP\text{-}I^{cut}$. Our interpretation is that with full consideration of ride-times the dominating effect is that the separated cuts are in fact stronger so that overall less pricing problems have to be solved.

Extended Instances The standard instances from the literature are characterized by narrow time windows and small capacities. Consequently, solutions comprise relatively short routes with only a small number of requests that are open at the same time so that the new approach either solves the instances in the root node or with a small integrality gap. In this paragraph, we want to analyze the behavior of the two most powerful approaches, $MP\text{-}I^{cut}_{sepRT}$ and MP^{cut}_{strg}, on instances with less tight constraints.

We introduce three new benchmark sets, constructed from the a and b problems from Cordeau (2006) in the following way: The time window $[a_i, b_i]$ of each node $i \in N$ is extended by adding 5, 10, and 15 time units to the latest start of service b_i so that the length of the time window is multiplied by factor 4/3, 5/3, and 6/3, respectively. Similarly, the capacity C is enlarged by the same factor. Overall the three new benchmark sets comprise 126 instances.

Section 3.E of the Appendix presents detailed linear relaxation results summarized in Table 3.5. Note that for some instances produced with factors 5/3 and 6/3 optimal solutions are unknown so that integrality gaps cannot be specified. Expectedly, with increasing time windows and capacities instances become harder to be solved for both algorithms $MP\text{-}I^{cut}_{sepRT}$ and MP^{cut}_{strg}, i.e., integrality gaps and computations times increase. Comparing average computation times for $MP\text{-}I^{cut}_{sepRT}$ and MP^{cut}_{strg}, the latter approach is by factor $30, 50, 77$, and 112 faster for the type a instances and by factor $1.2, 2.0, 2.6$, and 3.4 faster for the type b instances, respectively. We interpret the outcome in the sense that the new algorithm MP^{cut}_{strg} can better handle the harder instances than $MP\text{-}I^{cut}_{sepRT}$.

3.6.2 Integer Solution Results

Results for the computation times of optimal integer solutions are given in Tables 3.6 and 3.7. The entry *1h* (*2h*) indicates that the respective algorithm was unable to solve the instance within the time limit of one hour (two hours for *B&P-RC*). There was no time limit for *B&C*.

Our approach IMP^{cut}_{strg} with the SPPDARP subproblem and cuts was able to solve all 42 instances from the benchmark set. Without using the cuts, IMP_{strg} fails on instance b8-96. The only other approach to solve all the instances is *B&C*. All branch-and-price algorithms based on *IMP-I* using SPPPDPTW as a subproblem (including *B&P-RC*)

	orig.		4/3		5/3		6/3	
	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}
Type a instances								
# *opt*	7	16	6	12	5	8	4	6
# *best*	7	21	6	21	5	21	6	21
% *gap*	0.22	0.06	0.37	0.05	0.71	0.13	1.00	0.16
avg. time	60.4	2.1	171.7	3.3	301.0	3.9	610.5	5.4
Type b instances								
# *opt*	9	10	5	5	3	7	1	5
# *best*	13	20	9	21	3	21	1	21
% *gap*	0.10	0.07	0.46	0.18	?	?	?	?
avg. time	4.3	3.4	8.0	3.9	19.3	7.2	45.3	13.2

Table 3.5: Linear relaxation results for extended instances.

fail at least in solving the two biggest type a instances a8-80 and a8-96. The approach without ride-time information in the separation procedures $IMP\text{-}I^{cut}$ fails on three more instances, the approaches without any cuts (except those necessary to handle ride-time constraints) *IMP-I* and $IMP\text{-}I_{sepRT}$ on four additional instances.

For the instance a8-96, we computed an optimal integer solution with value 1229.66 that differs from the value 1232.61 reported in (Ropke *et al.*, 2007).

Dominance Rules in SPPDARP For solving SPPDARP, we derived the weak and strong dominance rules and associated labeling algorithms. With Dom_{weak}, we are unable to solve ten instances with IMP^{cut}_{weak} and twelve instances with IMP_{weak}. Apparently, only one of the larger instances, a8-80, that is not already solved to optimality in the root node can be solved with IMP^{cut}_{weak}. No additional instance is solved to optimality with IMP_{weak}. In these unsuccessful cases, we observed that often a single pricing problem could not be solved within the time limit. It seems that additional dual values resulting either from adding valid inequalities or, more often, from branching complicate the pricing so much that the labeling algorithm with Dom_{weak} cannot solve it. Obviously, the weak dominance rule is too weak to solve larger instances.

Computation Times Computation times for integer solutions are now discussed separately for type a and type b instances. The type a instances are generally more constraining w.r.t. ride-times. Here, the algorithms IMP_{strg} and IMP^{cut}_{strg} clearly outperform all other approaches. Computation times are always below 100 seconds, while all other approaches either fail or need at least one hour. Ride-time constraints in the subproblem in combination with an effective labeling algorithm makes solving these instances seemingly easy. For example, with IMP^{cut}_{strg} all but three instances can be solved within

10 seconds (exceptions are a6-72: 24.4s, a8-80: 43.0s, a8-96: 11.1s). All *MP-I*-based approaches and *B&P-RC* fail on the last two instances. For the larger instances with at least 50 requests, IMP^{cut}_{strg} is by at least a factor of 10 faster than these algorithms.

For the type b instances, where the maximum ride times are less constraining, the picture is less clear. Still, IMP^{cut}_{strg} is always among the fastest approaches for all instances. With respect to computation times, the branch-and-cut algorithm *B&C* seems inferior to all *IMP* and *IMP-I*-based approaches.

As for the linear relaxation, the weak dominance rule is clearly inferior to the strong dominance rule. For both benchmark sets, IMP_{weak} and IMP^{cut}_{weak} are only competitive on instances that are already solved in the root node.

Concluding, using cutting planes is beneficial for all approaches. With very few exceptions, the solution times of the algorithms with cuts are faster than without cuts. Despite the harder separation, the integration of the ride-time information in all separation procedures leads to overall faster computations. This is particularly true for the type a instances. Note that the best strategy for all approaches is to separate violated inequalities only at the root node.

Extended Instances We use the same three sets of extended benchmark instances as for the linear relaxation results in Section 3.6.1. Section 3.E of the Appendix presents detailed integer computational results for these instances that we have summarized in Table 3.8. Again, the more the constraints are relaxed, the harder the instances. The new algorithm MP^{cut}_{strg} can solve all except for one type a instances, while $MP\text{-}I^{cut}_{sepRT}$ fails for 28 instances. For the type b instances, the former fails on ten and the latter on 27 instances. It seems that the new approach MP^{cut}_{strg} is more advantageous for the type a instances, for which the ride-time constraints are more binding.

3.7 Conclusions and Outlook

In this paper, we presented dynamic time windows as an example of intra-tour synchronization constraints that are relevant for applications in passenger transportation, and service industries such as for the routing and scheduling of service personal (technicians, security guards etc.), and in home care. In a recent survey, Drexl (2012) outlined that there is a growing interest in vehicle routing with synchronization constraints. The work presented here can be seen as the central building block for handling dynamic time windows of the form $[0, L_i]$ to synchronize two operations i and $i + n$.

The DARP, in the variant where the service level is controlled by means of ride-time constraints, is the prototypical VRP with dynamic time windows. We proposed a new column-generation based solution approach where for the first time both time-window and ride-time constraints are handled in the subproblem. The detailed computational study has shown that this approach outperforms all other exact solution techniques either based on branch-and-cut or branch-and-cut-and-price, but with ride-time constraints handled in the column-generation master program. The superiority of the newly proposed branch-and-cut-and-price algorithm can be attributed to the following facts:

Instance	B&C (Ropke *et al.*, 2007)	B&P-RC (Ropke and Cordeau, 2005)	IMP-I	IMP-I^{cut}	IMP-I$_{sepRT}$	IMP-I$^{cut}_{sepRT}$	IMP$_{weak}$	IMP$^{cut}_{weak}$	IMP$_{strg}$	IMP$^{cut}_{strg}$
a2-16	0.6	0.3	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1
a2-20	1.2	0.9	0.4	0.3	0.1	0.3	0.1	0.1	0.1	0.1
a2-24	2.4	3.0	0.4	1.9	0.4	0.9	0.2	0.2	0.1	0.1
a3-24	1.8	1.3	2.0	2.9	2.0	1.3	0.1	0.1	0.1	0.1
a3-30	4.8	3.0	2.7	1.5	2.1	0.9	0.2	0.2	0.2	0.2
a3-36	10.8	12.3	9.9	9.4	7.6	5.2	2.1	2.2	0.9	1.0
a4-32	5.4	3.8	1.5	3.4	3.0	2.5	0.3	0.3	0.1	0.1
a4-40	19.2	12.2	4.2	5.6	4.4	6.8	1.0	1.0	0.5	0.5
a4-48	33.6	45.7	10.2	23.4	16.8	9.1	4.8	2.3	1.9	1.5
a5-40	10.2	6.4	1.2	1.7	4.3	1.3	0.5	0.5	0.3	0.3
a5-50	62.4	544.9	1h	1h	1h	194.4	298.4	32.7	15.9	3.2
a5-60	93.0	63.4	71.2	34.3	94.7	36.6	41.5	7.2	5.1	2.6
a6-48	26.4	17.4	22.9	16.4	7.9	9.9	1.6	1.6	0.5	0.5
a6-60	101.4	54.9	787.1	810.6	168.6	24.4	15.8	5.7	2.8	2.4
a6-72	198.6	1721.6	1h	1h	1h	686.0	1h	1h	97.7	24.4
a7-56	103.2	63.0	198.9	171.3	152.6	94.8	262.5	13.7	2.6	4.8
a7-70	209.4	135.1	840.1	361.2	130.0	128.7	5.1	5.1	1.9	1.9
a7-84	493.8	1436.5	1h	2686.7	1h	146.9	1h	51.7	86.5	7.8
a8-64	216.6	53.2	925.6	623.5	278.0	65.6	4.7	4.7	1.2	1.2
a8-80	733.2	2h	1h	1h	1h	1h	1h	395.7	67.4	43.0
a8-96	4233.0	2h	1h	1h	1h	1h	1h	178.4	20.5	11.1
# opt	21	19	16	17	16	19	17	20	21	21
avg. time	312.4	884.7	994.2	912.1	898.7	410.3	716.2	204.9	14.6	5.1

Table 3.6: Computation times for optimal integer solutions of type a instances in seconds.

First, a column-generation formulation with all intra-route constraints in the subproblem provides better lower bounds when solving its linear relaxation. Second, the key for the success of such stronger bounds is that we can compute them in relatively short time due to the new dynamic-programming labeling algorithm. Its heart is an effective dominance rule for comparing partial paths represented by labels. As obvious when looking back on Section 3.4, the dominance rules were non-trivial to derive, but their application boils down to some simple updates of attributes when constructing a new label. Third, with the considerable work on valid inequalities for the PDPTW and DARP by Cordeau (2006) and (Ropke *et al.*, 2007) it was simple to further improve the lower bounds. As a result, the proposed branch-and-cut-and-price algorithm with ride-time and time-windows constraints in the subproblem can solve all instances from the standard benchmark set and is about one order of magnitude faster than previous approaches.

We see several promising avenues for future research building on the techniques pre-

Instance	B&C (Ropke et al., 2007)	B&P-RC (Ropke and Cordeau, 2005)	IMP-I	IMP-I^{cut}	IMP-I_{sepRT}	IMP-I^{cut}_{sepRT}	IMP_{weak}	IMP^{cut}_{weak}	IMP_{strg}	IMP^{cut}_{strg}
b2-16	0.6	0.3	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1
b2-20	0.6	0.4	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
b2-24	1.8	1.7	0.4	1.0	0.4	0.9	0.4	0.7	0.3	0.6
b3-24	1.8	2.0	0.4	0.7	0.4	0.6	0.8	0.7	0.5	0.5
b3-30	3.0	1.9	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2
b3-36	4.8	3.7	0.3	0.4	0.4	0.4	0.4	0.4	0.3	0.3
b4-32	3.0	2.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
b4-40	8.4	6.3	0.6	1.3	0.6	1.0	2.1	1.4	1.0	0.7
b4-48	30.0	23.1	7.6	24.7	17.3	16.3	674.3	1h	2.5	5.5
b5-40	16.8	4.8	0.5	0.8	0.5	0.7	0.8	0.7	0.7	0.6
b5-50	39.6	11.6	0.7	0.7	0.7	0.7	0.9	0.9	0.7	0.7
b5-60	108.6	192.8	146.5	716.6	845.9	227.1	1h	1h	25.7	22.9
b6-48	14.4	7.2	0.6	1.3	0.8	0.8	0.4	0.4	0.2	0.2
b6-60	40.8	18.4	0.8	0.9	0.9	0.8	1.1	1.1	1.0	1.0
b6-72	1024.2	402.7	468.8	230.8	600.3	212.9	1h	1h	136.1	31.4
b7-56	807.6	66.2	32.1	10.8	26.7	9.3	1h	1h	30.4	50.3
b7-70	127.2	65.1	42.9	12.7	158.6	9.4	1h	1h	163.7	13.0
b7-84	396.6	237.6	30.4	45.4	75.4	36.8	1h	1h	42.0	71.7
b8-64	150.0	105.1	92.1	43.4	31.3	63.7	1h	1h	36.7	23.1
b8-80	183.0	73.1	17.7	20.8	18.7	9.6	1h	1h	14.4	10.3
b8-96	7205.4	3403.5	1h	1h	1h	3031.6	1h	1h	1h	898.8
# opt	21	21	20	20	20	21	13	13	20	21
avg. time	484.2	220.5	211.6	224.4	256.1	172.5	1403.9	1543.2	193.2	53.9

Table 3.7: Computation times for optimal integer solution of type `b` instances in seconds.

sented in this paper. First, more general intra-route synchronization with constraints of the form $K_i \leq T_{i+n} - T_i \leq L_i$ with an additional parameter $K_i > 0$ should be considered. It means that after performing operation i at least K_i time units must elapse before operation $i + n$ can be executed. The modeling of these *minimum ride-times* (in DARP vocabulary) is trivial in a two-index or three-index compact formulation. However, based on attempts to generalize our results, we suspect that the simultaneous handling of regular time windows and both minimum and maximum ride times in a dynamic programming labeling algorithm is not straightforward, but highly intricate. However, a possible way to tackle such a problem is the handling of *minimum* ride times using IPEC in the column-generation master program. The subproblem is then identical to the DARP subproblem, for which the actual implementation can be used.

Even more challenging types of VRP result from a mix of intra-route and inter-route synchronization constraints. It is clear that inter-route synchronization leads to additional constraints in the column-generation master program (Desaulniers *et al.*, 1998;

	orig.		4/3		5/3		6/3	
	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}
Type a instances								
# opt	19	21	15	21	13	20	9	21
avg. time	410.3	5.1	1121.1	28.1	1549.2	220.1	2064.4	187.2
Type b instances								
# opt	21	21	16	21	11	17	9	15
avg. time	172.5	53.9	917.8	237.9	1860.7	699.7	2206.8	1197.3

Table 3.8: Integer solution results for extended instances.

Ioachim *et al.*, 1999), resulting in costs and profits per node depending on the time of service (Ioachim *et al.*, 1998). The inter-route synchronization constraints alone are very difficult to be handled effectively, and it seems to be completely unclear how VRP combining intra-route and inter-route synchronization constraints can be modeled and solved. This relates to exact as well as heuristic approaches.

References

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, **36**(2), 69–79.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, **202**(3), 756–763.

Baldacci, R. and Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, **120**(2), 347–380.

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

Borndörfer, R., Klostermeier, F., Grötschel, M., and Küttner, C. (1997). Telebus Berlin: vehicle scheduling in a dial-a-ride system. Technical report SC 97-23, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany.

Bredström, D. and Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, **191**, 19–29.

Ceselli, A., Righini, G., and Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, **43**(1), 56–69.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, **54**(3), 573–586.

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B*, **37**(6), 579–594.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, **153**(1), 29–46.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, Boston, MA.

Desaulniers, G., Desroisiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constraint vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer, Boston, MA.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., and Soumis, F. (2002). VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*. SIAM, Philadelphia, PA.

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.

Desrochers, M. and Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, **26**(3), 191–212.

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.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, **46**(3), 297–316.

du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, **194**(1–3), 229–237.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**(1), 7–22.

Eveborn, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care – an operational system for staff planning of home care. *European Journal of Operational Research*, **171**(3), 962–976.

Firat, M. and Woeginger, G. J. (2011). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, **39**(1), 32–35.

Haugland, D. and Ho, S. C. (2010). Feasibility testing for dial-a-ride problems. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and B. Chen, editors, *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg.

Houck, D. J., Picard, J. C., Queyranne, M., and Vemuganti, R. R. (1980). The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch*, **17**, 93–109.

Ioachim, I., Gélinas, S., Desrosiers, J., and Soumis, F. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, **31**, 193–204.

Ioachim, I., Desrosiers, J., Soumis, F., and Bélanger, N. (1999). Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, **119**(1), 75–90.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. 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.

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

Madsen, O., Ravn, H., and Rygaard, J. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*, **60**, 193–208.

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*. SIAM, Philadelphia, PA.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (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., Doerner, K. F., and Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, **37**(6), 1129–1138.

Ropke, S. and Cordeau, J.-F. (2005). Branch and cut and price for the pickup and delivery problem with time windows. Chapter 9 of S. Ropke's cumulative Ph.D. dissertation, Heuristic and exact algorithms for vehicle routing problems, University of Copenhagen, Copenhagen, Denmark.

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.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, **49**(4), 258–272.

Russell, R. A. and Morrel, R. B. (1986). Routing special-education school buses. *Interfaces*, **16**(5), 56–64.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, **4**(2), 146–154.

Tang, J., Kong, Y., Lau, H., and Ip, A. W. (2010). A note on "efficient feasibility testing for dial-a-ride problems". *Operations Research Letters*, **38**(5), 405–407.

Toth, P. and Vigo, D. (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, **31**(1), 60–71.

Appendix

3.A Proofs

Before we present the proofs, we recall the basic dominance principles for (E)SPPRC and introduce some useful notation. Additional useful material can be found in (Desaulniers *et al.*, 1998; Irnich and Desaulniers, 2005; Ropke and Cordeau, 2009).

The *standard dominance principle* says that a label ℓ_1 representing the partial path $\mathcal{P}(\ell_1)$ dominates a label ℓ_2 representing the partial path $\mathcal{P}(\ell_2)$ if the following three conditions hold:

1. $\mathcal{P}(\ell_1)$ and $\mathcal{P}(\ell_2)$ must end at the same node.
2. Every completion Q of ℓ_2 to the sink node that gives a feasible path $\mathcal{P}_2 = (\mathcal{P}(\ell_2), Q)$ is a completion of ℓ_1 that must also result in a feasible path $\mathcal{P}_1 = (\mathcal{P}(\ell_1), Q)$.
3. The (reduced) cost of $\mathcal{P}_1$ must not exceed the (reduced) cost of $\mathcal{P}_2$.

In this case, ℓ_2 can never lead to a path with smaller cost than ℓ_1, and can thus be discarded.

The condition 2. requires that $\mathcal{P}(\ell_1)$ must be feasible. In this case, the label ℓ is called *feasible*. Also, a completion Q of a label ℓ that leads to a feasible (partial) path $(\mathcal{P}(\ell), Q)$ is called *feasible*. Typically, considering all possible feasible completions is not tractable so that the above dominance relation is hardly directly applicable. Instead, condition 2. is often proven indirectly comparing attributes of the labels ℓ_1 and ℓ_2, i.e., accumulated costs and consumed resources. If ℓ_1 has 'better' attributes than ℓ_2, the same holds for all extended partial paths, whenever resource consumptions are monotone. The importance of non-decreasing REFs for the validity of dominance rules was stressed and exemplified in (Desaulniers *et al.*, 1998; Irnich, 2008).

We recall Proposition 3.1 from the article (see also (Dumas *et al.*, 1991; Ropke and Cordeau, 2009)):

Proposition 3.1: (Dom_{PDPTW}) A feasible label ℓ_1 dominates a label ℓ_2 if

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \text{ and } \quad O_{\ell_1} \subseteq O_{\ell_2}.$$

The dominance rule of Proposition 3.1 slightly differs from the standard dominance principle presented before. Indeed, if $O_{\ell_1} \neq O_{\ell_2}$, then a feasible completion Q of ℓ_2 cannot be a feasible completion of ℓ_1 due to pairing constraints. We introduce some notation helpful for describing completions resolving the complication with pairing constraints. For any number n and any set of numbers M, let $n + M = \{n + m : m \in M\}$. Furthermore, for any sequence (path or schedule) $\mathcal{P} = (h_1, h_2, \ldots, h_p)$ and any set M of numbers, the term $\mathcal{P} \setminus M$ denotes the sub-sequence of $\mathcal{P}$ where h_i is removed if $h_i = m$ is the first occurrence of $m \in M$ in the sequence $\mathcal{P}$.

The proof of Proposition 3.1 results from the generalization of the second condition of the standard dominance. Condition 2. can be replaced by

2'. For every completion Q_2 of ℓ_2 to the sink node that is a feasible path $\mathcal{P}_2 = (\mathcal{P}(\ell_2), Q_2)$ there exists a completion Q_1 of ℓ_1 so that $\mathcal{P}_1 = (\mathcal{P}(\ell_1), Q_1)$ is a feasible path.

Note that Q_1 may or may not be identical or resulting from Q_2. Now consider the completion $Q' = Q \backslash \{n + (O_{\ell_2} \backslash O_{\ell_1})\}$ of ℓ_1. Q' is equal to Q except for skipping the delivery nodes for the additional open requests $O_{\ell_2} \backslash O_{\ell_1}$. The path $\mathcal{P}_1 = (\mathcal{P}(\ell_1), Q')$ satisfies pairing and precedence constraints. The relations in Proposition 3.1 ensure that $\mathcal{P}_1$ also respects capacity and time-window constraints if Q is a feasible completion for ℓ_2, and that the total costs of $\mathcal{P}_2 = (\mathcal{P}(\ell_2), Q)$ cannot be smaller than that of $\mathcal{P}_1$. The latter is true as visiting an additional delivery node is never beneficial when the DTI holds.

With the presentation of the proof of Proposition 3.1 and the additional notation, we able to give the proof of all other lemmas and propositions.

Proof of Lemma 3.1: The schedule $T^*_{\mathcal{P}}(t) = (\tau^*_1(t), ..., \tau^*_q(t))$ is feasible, i.e., $T^*_{\mathcal{P}}(t) \in \mathscr{T}_{\mathcal{P}}(t)$, if it satisfies conditions (3.21)–(3.23) and $\tau^*_q(t) \leq t$. By definition of $\tau^*_i(t)$, there exists a feasible schedule $T^i_{\mathcal{P}} = (\tau^i_1, ..., \tau^*_i(t), ..., \tau^i_q)$ for each $i = 1, ..., q$, with $\tau^i_q \leq t$ and $\tau^i_j \leq \tau^*_j(t)$ for all $j \neq i$. Hence, $\tau^*_i(t) \in [a_{h_i}, b_{h_i}]$ for all $i = 1, ..., q$ and $\tau^*_q(t) \leq t$ obviously hold. Since $\tau^*_i(t) + s_{h_i} + t_{h_i,h_{i+1}} \leq \tau^i_{i+1}$ and $\tau^i_{i+1} \leq \tau^*_{i+1}(t)$ hold for $i = 1, ..., q-1$, $T^*_{\mathcal{P}}(t)$ satisfies inequalities (3.22). Finally, using $\tau^j_i + s_{h_i} + L_{h_i} \geq \tau^*_j(t)$ and $\tau^*_i(t) \geq \tau^j_i$ for each pair i, j with $h_i + n = h_j$, $T^*_{\mathcal{P}}(t)$ also satisfies inequalities (3.23) and thus $T^*_{\mathcal{P}}(t) \in \mathscr{T}_{\mathcal{P}}(t)$. □

Proof of Proposition 3.3: Let Q be a feasible completion of ℓ_2 and denote by $\mathcal{P}_2 = (\mathcal{P}(\ell_2), Q)$ the corresponding path. The completion $Q' = Q \backslash \{n + O_{\ell_2} \backslash O_{\ell_1}\}$ of ℓ_1 leads to a path $\mathcal{P}_1 = (\mathcal{P}(\ell_1), Q')$ with $\mathcal{P}(\ell_1) = (h_1, ..., h_q = \eta_{\ell_1})$. It is known from Proposition 3.1 that $\mathcal{P}_1$ satisfies pairing, precedence, and capacity constraints and has smaller cost than $\mathcal{P}_2$. To show that $\mathcal{P}_1$ is feasible, it remains to show that there exists a feasible time schedule $T_{\mathcal{P}_1}$ for $\mathcal{P}_1$.

Let $T_{\mathcal{P}_2} = (T_{\mathcal{P}(\ell_2)}, T_Q)$ be a feasible schedule for $\mathcal{P}_2$ with $T_Q = (\tau_{q+1}, ..., \tau_r)$ and start of service $\tau^{\ell_2}_q$ at the current node $h_q = \eta_{\ell_2}$. Denote by $T^*_{\mathcal{P}(\ell_1)}(\tau^{\ell_2}_q) = (\tau^*_1(\tau^{\ell_2}_q), ..., \tau^*_q(\tau^{\ell_2}_q)) \in \mathscr{T}_{\mathcal{P}(\ell_1)}(\tau^{\ell_2}_q)$ the time schedule for $\mathcal{P}(\ell_1)$ that maximizes the start of service τ_i for all nodes $h_i, i = 1, ..., q$ while $\tau_q \leq \tau^{\ell_2}_q$. Lemma 3.1 guarantees the existence and feasibility of this time schedule. Moreover, denote by $T_{Q'} = T_Q \backslash \{\tau_i : h_i - n \in O_{\ell_2} \backslash O_{\ell_1}\}$ the schedule for Q' that assigns each node h_i of Q' the same start of service τ_i as in T_Q. Then, using that $T^*_{\mathcal{P}(\ell_1)}$ and $T_{\mathcal{P}_2}$ are feasible, $\tau^*_q(\tau^{\ell_2}_q) \leq \tau^{\ell_2}_q$, and $\tau_i \leq ld^{h_i-n}_{\ell_2}(\tau^{\ell_2}_q) \leq ld^{h_i-n}_{\ell_1}(\tau^{\ell_2}_q)$ for all nodes h_i of Q' with $h_i - n \in O_{\ell_1}$, it follows that the schedule $T_{\mathcal{P}_1} = (T^*_{\mathcal{P}(\ell_1)}, T_{Q'})$ is feasible. □

Proof of Proposition 3.4: Let $\mathcal{P}(\ell) = (h_1, ..., h_q)$ with $h_q = \eta_\ell$ be the path represented by label ℓ. We first show that the latest feasible start of service $\tau^*_q(t), t \geq t_\ell$ at the last node h_q is of the form $\tau^*_q(t) = \min\{t, k\}$, where k is a constant. By definition, $\tau^*_q(t) = \max_{T_{\mathcal{P}(\ell)} = (\tau_1, ..., \tau_q) \in \mathscr{T}_{\mathcal{P}(\ell)}(t)} \{\tau_q\}$ with $\tau_q \leq t$. If $\eta_\ell \in P$, then clearly $\tau^*_q(t) = \min\{t, b_{\eta_\ell}\}$.

If $\eta_\ell \in D$, then $\tau_q^*(t)$ is given by $\min\{t, b_{\eta_\ell}, \tau_j^*(t) + s_{h_j} + L_{h_j}\}$ where $\tau_j^*(t)$ is the latest feasible start of service at node h_j with $\tau_q \leq t$ and $h_j = \eta_\ell - n$ the pickup node of η_ℓ. Consider the case $\tau_j^*(t) + s_{h_j} + L_{h_j} < \min\{t, b_{\eta_\ell}\}$, i.e., $\tau_q^*(t)$ is strictly smaller than t. Then, $\tau_j^*(t)$ is independent of t and hence both $\tau_q^*(t)$ and $\tau_j^*(t)$ are constant for all such t. As a result, we have that $\tau_q^*(t)$ is of the form $\min\{t, k\}$, with a constant k.

Let $h_i \in O_\ell$ be an open request, and let $m = q - i$ be the length of the sub-path of $\mathcal{P}(\ell)$ starting at h_i. We can now show by induction over the number m that $ld_\ell^{h_i}(t) = \min\{k_1^{h_i} + t, k_2^{h_i}\}, t \geq t_\ell$ for all $h_i \in O_\ell$: Recall that $ld_\ell^{h_i}(t) = \min\{b_{h_i+n}, \tau_i^*(t) + s_{h_i} + L_{h_i}\}$. Thus, it is sufficient to show that $\tau_i^*(t) = \min\{\tilde{k}_1^{h_i} + t, \tilde{k}_2^{h_i}\}$ with constants $\tilde{k}_1^{h_i}$ and $\tilde{k}_2^{h_i}$.

The case $m = 0$ means that the request is just being picked up, i.e., $h_i = h_q = \eta_\ell$. The property follows immediately from $\tau_q^*(t) = \min\{t, k\}$ as shown above.

For the case $m + 1$, let ℓ be a label representing a path $\mathcal{P}(\ell) = (h_1, ...h_q = \eta_\ell)$ where h_i is on board for m nodes and $\tau_i^*(t) = \min\{\tilde{k}_1^{h_i} + t, \tilde{k}_2^{h_i}\}$. If the extension of ℓ along the arc (η_ℓ, x) with $x \neq h_i + n$ is feasible, a new feasible label ℓ' where h_i is on board for $m + 1$ nodes is created. The case $x = h_i + n$ can be neglected, as then $h_i \notin O_{\ell'}$. Denote by $T^{*,\ell'}_{\mathcal{P}(\ell')}(t) = (\tau_1^{*,\ell'}(t), ..., \tau_q^{*,\ell'}(t), \tau_x^{*,\ell'}(t)) \in \mathscr{T}_{\mathcal{P}(\ell')}(t)$ the feasible schedule for $\mathcal{P}(\ell')$ maximizing the start of service at all nodes with $\tau_x^{*,\ell'} \leq t$, and by $T'_{\mathcal{P}(\ell')\setminus\{x\}}(t') = (\tau'_1(t'), ..., \tau'_q(t')) \in \mathscr{T}_{\mathcal{P}(\ell')\setminus\{x\}}(t')$ the feasible schedule for $\mathcal{P}(\ell')\setminus\{x\}$ maximizing the start of service at all nodes with $\tau'_q \leq t'$. Comparing the defining maximization problems for $T^{*,\ell'}_{\mathcal{P}(\ell')}(t)$ and $T'_{\mathcal{P}(\ell')\setminus\{x\}}(t')$ we have that $T^{*,\ell'}_{\mathcal{P}(\ell')}(t) = (T'_{\mathcal{P}(\ell')\setminus\{x\}}(t'), \tau_x^{*,\ell'}(t))$ for $t' = \tau_x^{*,\ell'}(t) - s_{h_q} - t_{h_q,x}$. Since $T'_{\mathcal{P}(\ell')\setminus\{x\}}(t')$ is by definition equal to the schedule that maximizes all τ_i with $\tau_q \leq t'$ for the path represented by ℓ, it follows that $\tau'_i(t') = \tau_i^{*,\ell'}(t)$. By assumption $\tau'_i(t') = \min\{\tilde{k}_1^{h_i} + t', \tilde{k}_2^{h_i}\}$ holds and thus $\tau_i^{*,\ell'}(t) = \min\{\tilde{k}_1^{h_i} + \tau_x^{*,\ell'}(t) - s_{h_q} - t_{h_q,x}, \tilde{k}_2^{h_i}\}$. Using the property that $\tau_x^{*,\ell'}(t) = \min\{t, k\}$ with a constant k proves the proposition. □

Proof of Proposition 3.6: We need to show that $ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t)$ holds for all $t \in [t_{\ell_2}, b_{\eta_{\ell_2}}], i \in O_{\ell_1}$, whenever $ld^i_{\ell_1}(t_{\ell_1}) + (t_{\ell_2} - t_{\ell_1}) \geq ld^i_{\ell_2}(t_{\ell_2})$ and $ld^i_{\ell_1}(B^i_{\ell_1}) \geq ld^i_{\ell_2}(B^i_{\ell_2})$ hold. Using Proposition 3.4, we have that $ld^i_{\ell_1}(t) = \min\{k^i_{\ell_1,1} + t, k^i_{\ell_1,2}\}$ and $ld^i_{\ell_2}(t) = \min\{k^i_{\ell_2,1} + t, k^i_{\ell_2,2}\}$. Thus, four cases have to be differentiated:

1. For all t with $ld^i_{\ell_1}(t) = k^i_{\ell_1,2}$ and $ld^i_{\ell_2}(t) = k^i_{\ell_2,2}$ we have that $ld^i_{\ell_1}(t) = k^i_{\ell_1,2} = ld^i_{\ell_1}(B^i_{\ell_1}) \geq ld^i_{\ell_2}(B^i_{\ell_2}) = k^i_{\ell_2,2} = ld^i_{\ell_2}(t)$.

2. For all t with $ld^i_{\ell_1}(t) = k^i_{\ell_1,2}$ and $ld^i_{\ell_2}(t) = k^i_{\ell_2,1} + t$ we have that $ld^i_{\ell_1}(t) = k^i_{\ell_1,2} \geq k^i_{\ell_2,2} \geq k^i_{\ell_2,1} + t = ld^i_{\ell_2}(t)$.

3. For all t with $ld^i_{\ell_1}(t) = k^i_{\ell_1,1} + t$ and $ld^i_{\ell_2}(t) = k^i_{\ell_2,1} + t$ we have that $ld^i_{\ell_1}(t) = ld^i_{\ell_1}(t_{\ell_1}) + t - t_{\ell_1}$ and $ld^i_{\ell_2}(t) = ld^i_{\ell_2}(t_{\ell_2}) + t - t_{\ell_2}$. Using $ld^i_{\ell_1}(t_{\ell_1}) + t_{\ell_2} - t_{\ell_1} \geq ld^i_{\ell_2}(t_{\ell_2})$ it follows that $ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t)$.

4. For all t with $ld^i_{\ell_1}(t) = k^i_{\ell_1,1} + t$ and $ld^i_{\ell_2}(t) = k^i_{\ell_2,2}$ we have that $ld^i_{\ell_1}(t) = k^i_{\ell_1,1} + t \geq k^i_{\ell_2,1} + t \geq k^i_{\ell_2,2} = ld^i_{\ell_2}(t)$.

□

Proof of Proposition 3.6: Consider first the case $i = x$. We have shown in the proof of Proposition 3.4 that the latest possible delivery times for i in this case are $ld^i_{\ell'}(t) = \min\{\min\{t, b_x\} + s_x + L_i, b_{i+n}\}$. The point of time when $ld^i_{\ell'}(t)$ becomes constant is then clearly given by $B^i_{\ell'} = \min\{b_x, b_{i+n} - s_x - L_i\}$. The formulas for $ld^i_{\ell'}(t_{\ell'})$ and $ld^i_{\ell'}(B^i_{\ell'})$ are obvious in this case.

For $i \neq x$, it is again known from the proof of Proposition 3.4 that $T^{*,\ell'}_{\mathcal{P}(\ell')}(t) = (T^{*,\ell}_{\mathcal{P}(\ell)}(t'), \tau^{*,\ell'}_x(t))$ and thus $ld^i_{\ell'}(t) = ld^i_\ell(t')$ holds for $t \geq t_{\ell'}$ and with $t' = \tau^{*,\ell'}_x(t) - s_{\eta_\ell} - t_{\eta_\ell,x}$ and $\tau^{*,\ell'}_x(t) = \min\{t, \tilde{b}_x\}$. If $t \geq \tilde{b}_x$ then $\tau^{*,\ell'}_x(t)$ is constant and hence $ld^i_{\ell'}(t)$ is constant for all such t. For all $t < \tilde{b}_x$, $\tau^{*,\ell'}_x(t)$ increases linear in t and so does $ld^i_\ell(t')$ as long as $B^i_\ell \geq t'$. If $B^i_\ell < t'$, then $ld^i_\ell(t')$ is constant and so is $ld^i_{\ell'}(t)$. As a result, the time when $ld^i_{\ell'}(t)$ becomes constant is $B^i_{\ell'} = \max\{t_{\ell'}, \min\{B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x}, \tilde{b}_x\}\}$.

For $ld^i_{\ell'}(t_{\ell'})$ and $ld^i_{\ell'}(B^i_{\ell'})$ note first that $\tau^{*,\ell'}_x(t_{\ell'}) = t_{\ell'}$ and $\tau^{*,\ell'}_x(B^i_{\ell'}) = B^i_{\ell'}$. Then, we have $ld^i_{\ell'}(t_{\ell'}) = ld^i_\ell(t_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x}) = ld^i_\ell(t_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x} + t_\ell - t_\ell)$ and $ld^i_{\ell'}(B^i_{\ell'}) = ld^i_\ell(B^i_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x}) = ld^i_\ell(B^i_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x} + B^i_\ell - B^i_\ell)$. Using the property that $ld^i_\ell(t)$ increases linearly for all $t \in [t_\ell, B^i_\ell]$ and is constant for all $t > B^i_\ell$ we have that $ld^i_{\ell'}(t_{\ell'}) = ld^i_\ell(t_\ell) + \min\{(t_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x}) - t_\ell, B^i_\ell - t_\ell\}$ and $ld^i_{\ell'}(B^i_{\ell'}) = ld^i_\ell(B^i_\ell) - \max\{0, B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x} - B^i_{\ell'}\}$. □

3.B Example of Applying the Labeling Procedure with Strong Dominance

A small example to illustrate the REFs and the underlying intuition for B^i_ℓ, $ld^i_\ell(t_\ell)$ and $ld^i_\ell(B^i_\ell)$ is given in Table 3.9. Consider the labels ℓ_0, ℓ_i, ℓ_j and ℓ_k which are successively generated along the path $\mathcal{P} = (0, i, j, k)$, see Figure 3.3. Assume that the travel times between all nodes are 5, the maximum ride time of request i is $L_i = 20$, and there are no service times at the nodes. When extending the initial label ℓ_0 along the arc $(0, i)$ label ℓ_i is created. As request i is being picked up $B^i_{\ell_i}, ld^i_{\ell_i}(t_{\ell_i})$ and $ld^i_{\ell_i}(B^i_{\ell_i})$ need to be initialized. Assuming that the end of the time window b_{i+n} of the delivery node $i + n$ is not binding, the latest possible delivery times are clearly given by the start of service at i plus the maximum ride time, i.e., $ld^i_{\ell_i}(t_{\ell_i}) = 5 + 20 = 25$ and $ld^i_{\ell_i}(B^i_{\ell_i}) = 15 + 20 = 35$, and $B^i_{\ell_i}$ is equal to $b_i = 15$.

The extension of ℓ_i along the arc (i, j) leads to the label ℓ_j with $t_{\ell_j} = t_{\ell_i} + 5 = 10$. Regarding the latest delivery times for request i, starting the service at j at time t_{ℓ_j} means that the predecessor node i has to be serviced at the earliest possible time t_{ℓ_i}, and consequently $ld^i_{\ell_i}(t_{\ell_i}) = ld^i_{\ell_j}(t_{\ell_j}) = 25$. To maximize the latest delivery of i, we want to delay the start of service at i until $B^i_{\ell_i}$. This means that one would arrive at node j at time 20 which is too late, as the latest feasible start of service at j is at time 18. As a result, within the partial path $(0, i, j)$ the service at i can at latest be delayed up to time $b_j - 5 = 13 = B^i_{\ell_i} - 2$. The corresponding latest delivery time of i for the latest feasible service at j is $ld^i_{\ell_j}(B^i_{\ell_j}) = ld^i_{\ell_i}(B^i_{\ell_i} - 2) = ld^i_{\ell_i}(B^i_{\ell_i}) - 2 = 33$ and $B^i_{\ell_j} = b_j = 18$.

Considering label ℓ_k, the earliest start of service is $t_{\ell_k} = 17$ implying that there is a waiting time of 2 when serving all nodes as early as possible. In terms of maximizing the latest possible delivery of i, this waiting time should be shifted before the service

	ℓ_0	ℓ_i	ℓ_j	ℓ_k
η_ℓ	0	i	j	k
$[a_{\eta_\ell}, b_{\eta_\ell}]$	$[0, 100]$	$[5, 15]$	$[10, 18]$	$[17, 25]$
t_ℓ	0	5	10	17
B^i_ℓ	-	15	18	23
$ld^i_\ell(t_\ell)$	-	25	25	27
$ld^i_\ell(B^i_\ell)$	-	35	33	33

Table 3.9: REFs for B^i, $ld^i(t)$ and $ld^i(B^i)$ along the path $\mathcal{P} = (0, i, j, k)$ assuming a maximum ride time of $L_i = 20$

of node i. In other words, when serving k at time $t_{\ell_k} = 17$ the service of i should not be started at t_{ℓ_i} but instead be delayed until $t_{\ell_i} + 2 = 7$. This also implies that j is serviced at time $t_{\ell_j} + 2 = 12$. The corresponding latest delivery times of i for the earliest feasible service at k are then given by $ld^i_{\ell_k}(t_{\ell_k}) = ld^i_{\ell_j}(t_{\ell_i} + 2) = ld^i_{\ell_j}(t_{\ell_i}) + 2 = 27$. To maximize the latest delivery of i we again want to delay the start of service at j until $B^i_{\ell_j}$ which results in a feasible time for the start of service at k. Even more, serving node k later than $B^i_{\ell_j} + 5 = 23$ does not increase the latest delivery time of i, as it is already constrained by the time window at node j. To satisfy the time window of j, service at node i cannot start later than at time 13 implying a start of service at j of $B^i_{\ell_j} = 18$ and a start of service at k of $B^i_{\ell_k} = 23$. Consequently $ld^i_{\ell_k}(B^i_{\ell_k}) = ld^i_{\ell_j}(B^i_{\ell_j}) = 33$.

3.C Valid Inequalities

To describe valid inequalities of the PDPTW and DARP some additional notation is necessary. For each subset $S \subseteq N$ of nodes, we denote by $\pi(S) = \{i \in P : i + n \in S, i \notin S\}$ the sets of predecessors, and by $\sigma(S) = \{i + n \in D : i \in S, i + n \notin S\}$ the sets of successors of S. Let $\delta^+(S) = \{(i, j) : i \in S, j \notin S\}$ be the set of arcs leaving S and $d(S) = \sum_{i \in S} d_i$ be the total demand of set S.

The first class results from lifting IPEC and is known as *tournament constraints* (Ascheuer *et al.*, 2000). Suppose the path $I = (h_1, ..., h_q)$ is infeasible (with respect to any constraint), then the associated tournament constraint is

$$x([I]) \leq |I| - 1 \quad (= q - 2), \tag{3.34}$$

where $[I] := \{(h_i, h_j) \in A : 1 \leq i < j \leq q\}$ is the transitive closure of the path I.

In the context of pickup-and-delivery problems, another strengthening of IPEC is possible. If $h_1 = i$ and $h_q = i + n$, i.e., the path I connects a pickup node i with its corresponding delivery node $i+n$, then Cordeau (2006) suggested the use of the following inequality:

$$x(I) \leq |I| - 2 \quad (= q - 3). \tag{3.35}$$

For the separation of both types of infeasible path inequalities, we use a straightforward

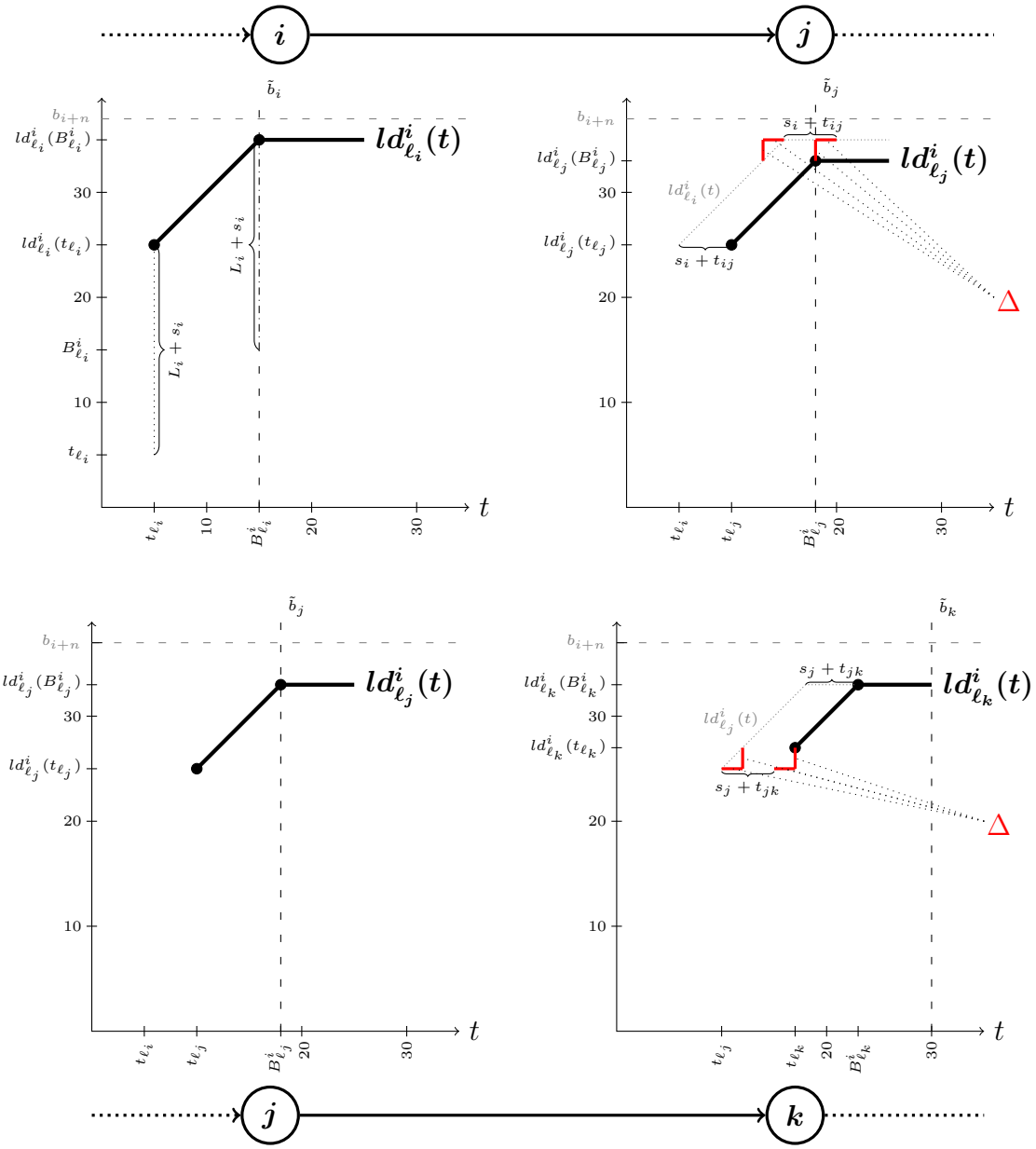


Figure 3.3: Extension of labels: Latest delivery times $ld^i_\ell(t)$ for an open request i for the extension along arc (i, j) (upper part) and along arc (j, k) (lower part).

enumeration procedure (see Ascheuer *et al.*, 2000).

The second family of inequalities are *rounded capacity inequalities*. Capacity inequalities impose a lower bound on the number of vehicles that must leave (and enter) a set of nodes $S \subset N$ due to capacity limitations of the vehicles. Such inequalities have been applied successfully in approaches for the CVRP. In the context of pickup-and-delivery problems, where there is negative demand at the delivery nodes, Ropke and Cordeau (2009) proposed rounded capacity inequalities of the following form:

$$x(\delta^+(S)) \geq \max\left\{1, \left\lceil \frac{d(\pi(S))}{C} \right\rceil, \left\lceil \frac{-d(\sigma(S))}{C} \right\rceil \right\}.$$

The right-hand side takes into account bounds on the number of vehicles that must leave S by considering the demands of the predecessor nodes $\pi(S)$ and the demands of the successor nodes $\sigma(S)$ separately. Clearly, a feasible solution must satisfy both bounds. For the separation of the rounded capacity inequalities, we use the heuristic separation procedure of Ropke and Cordeau (2009): It starts from a singleton set $S = \{h\}$, node subsets $S \subset P \cup D$ are iteratively enlarged maximizing a parameterized objective that seeks finding a violated rounded capacity inequality. This procedure is repeated several times for each possible initial node h and randomly chosen parameters of the objective.

In the DARP, a set of nodes $S \subseteq P \cup D$ may need to be served by several vehicles not only due to capacity limitations of the vehicles, but also because of all other route constraints. This is the key idea of the 2-path cuts that were introduced by Kohl *et al.* (1999) for the VRPTW. If the nodes S cannot be feasibly served by just one vehicle, then the following inequality is valid:

$$x(\delta^+(S)) \geq 2.$$

The separation of the 2-path cuts is performed using another heuristic proposed by Ropke and Cordeau (2009): Also here, the node subsets S for constructing tentative sets S are initialized with a single node and additional nodes are iteratively added so that $\bar{x}(\delta^+(S))$ is minimized. Whenever $\bar{x}(\delta^+(S)) < 2$, it is checked whether or not S can be served by a single vehicle. The procedure is started several times from each node and is randomized by adding a random value to each x_{ij} when selecting a node to add.

The last class of inequalities available are *fork inequalities*, which were proposed by Ropke *et al.* (2007) for the PDPTW. They can be differentiated into *outfork* and *infork inequalities*. Their basic idea is to consider groups of infeasible paths that are built around an inner path that is feasible (see Ropke *et al.*, 2007, for illustrative examples). Consider a feasible path $\mathcal{P} = (h_1, ..., h_q)$ and node subsets $S, T_1, ..., T_q \subset P \cup D$ satisfying $h_j \notin T_{j-1}$ for $j = 2, ..., q$. If the path $(i, h_1, ..., h_m, j)$ is infeasible for each integer $m \leq q$ and any two nodes $i \in S$ and $j \in T_m$, then the following outfork inequality is valid for the DARP:

$$\sum_{\substack{i \in S: \\ (i,h_1) \in A}} x_{i,h_1} + \sum_{m=1}^{q-1} x_{h_m,h_{m+1}} + \sum_{m=1}^{q} \sum_{\substack{j \in T_m: \\ (h_m,j) \in A}} x_{h_m,j} \leq q.$$

In analogy, the infork inequalities are defined for a feasible path $\mathcal{P} = (h_1, ..., h_q)$ and node subsets $S_1, ..., S_q, T \subset P \cup D$ satisfying $h_j \notin S_{j+1}$ for $j = 1, ..., q-1$. If the path $(i, h_m, ..., h_q, j)$ is infeasible for each integer $m \leq q$ and any two nodes $i \in S_m$ and $j \in T$, then the following infork inequality is valid:

$$\sum_{m=1}^{q} \sum_{\substack{i \in S_m: \\ (i,h_m) \in A}} x_{i,h_m} + \sum_{m=1}^{q-1} x_{h_m,h_{m+1}} + \sum_{\substack{j \in T: \\ (h_q,j) \in A}} x_{h_q,j} \leq q.$$

Ropke and Cordeau (2009) showed that both types of fork inequalities are in fact implied by a set-partitioning formulation that uses elementary routes only. As mentioned before, even when solving the non-elementary SPPDARP subproblem, hardly any non-elementary routes are generated. Therefore, we do not use fork constraints in our implementations for *MP* and *IMP* that handle the ride-time constraints in the subproblem. With SPPPDPTW subproblems, however, the generated routes may not satisfy the maximum ride-times. In this case, the fork constraints are not implied and we use them to strengthen the lower bounds.

The lifted IPEC, 2-path cuts, and fork inequalities all rely on the existence of a feasible path visiting either a given sequence $\mathcal{P}$ of nodes, or visiting all nodes of a given subset S in any order. In a pickup-and-delivery context, to decide whether or not there exists such a feasible path, pairing and precedence constraints can be taken into account. Herewith, stronger conditions for feasibility can be derived (see Ropke and Cordeau, 2009) because for any subset $S \subseteq N$ of nodes, the feasible path over S has to visit all predecessor nodes $\pi(S)$ before and all successor nodes $\sigma(S)$ after visiting S. At least one of the possible sequences over $(\pi(S), S, \sigma(S))$ still has to satisfy the constraints of the DARP. The same is true for a feasible path over a given sequence $\mathcal{P}$. Consequently, a huge number of possible paths might have to be checked for a given sequence $\mathcal{P}$ or subset S.

The enumeration of all paths that possibly need to be checked is not promising to attain efficient separation procedures. Instead, we can use an adaptation of our labeling algorithm for SPPDARP with Dom_{strong}. We present computational results for two different versions of the separation procedures when using SPPPDPTW as subproblem. It seems fair not to include such a labeling algorithm for separation if the pricing problem cannot handle ride-times. Here, ride-time constraints are considered only in feasibility checks that do *not* incorporate the predecessor nodes $\pi(S)$ and successor nodes $\sigma(S)$. Algorithms where the separation procedures fully integrate the ride-times are marked with the subscript $_{sepRT}$ in the computational results. In the following, we describe the adaptation of our labeling algorithm when used for feasibility checking in separation procedures: First, all costs can be disregarded and therefore be set to zero. Second, for a given subset S, we build a network with three layers in which the layers solely contain the nodes $\pi(S)$, S, and $\sigma(S)$, respectively. Third, for a subpath $\mathcal{P}$, we build the same type of layered network where, additionally, the order of the nodes in the middle layer is fixed. Fourth, additional resources (i.e., visiting counters) ensure that every node in every layer is visited exactly once. A path $\mathcal{P}$ or subset S is feasible if and only if there exists at least one path in the layered network.

Care must be taken when including the predecessor and successor information to identify infeasible paths in the case of the lifted IPEC (3.35). These inequalities are not valid if the respective path I is only infeasible because of a violation of the ride-time constraint of a request that has only its pickup or its delivery node in I but not both. As a simple example consider the sequence $\mathcal{P} = (i, j, i+n)$ with a pickup $j \in P$. Any feasible path for $\mathcal{P}$ needs to visit node $j+n$ after the nodes of the sequence $\mathcal{P}$ implied by $\sigma(\{i, j, i+n\}) = \{j+n\}$ and $\pi(\{i, j, i+n\}) = \varnothing$. In this case, the only possibility is $\tilde{\mathcal{P}} = (i, j, i+n, j+n)$. If $\tilde{\mathcal{P}}$ is infeasible only because of a violation of the maximum ride-time of request j, then the path $\hat{\mathcal{P}} = (i, j, j+n, i+n)$ may still be feasible and be part of an integer solution. This shows the possible usage of the arc (i, j) in an integer solution and, hence, that the respective inequality (3.35), i.e., $x(\mathcal{P}) \leq |\mathcal{P}| - 2 = 0$, is not valid in this case. With the same argumentation it can be shown that the demands of requests where only the pickup or only the delivery node is in $\mathcal{P}$ must not be included when checking feasibility for inequalities (3.35).

3.D Detailed Computational Results for Standard Instances

The Tables 3.10 and 3.11, 3.12 and 3.13, and 3.14 and 3.15, show the detailed computation times for the root node lower bounds, the total number of nodes explored within the branch-and-cut-and-price algorithms, and the total number of cuts generated within the algorithms, respectively.

Instance	$MP\text{-}I$	$MP\text{-}I^{cut}$	$MP\text{-}I_{sepRT}$	$MP\text{-}I^{cut}_{sepRT}$	MP_{weak}	MP^{cut}_{weak}	MP_{strg}	MP^{cut}_{strg}
a2-16	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
a2-20	0.1	0.3	0.1	0.3	0.1	0.1	0.1	0.1
a2-24	0.3	1.8	0.3	0.7	0.2	0.2	0.1	0.1
a3-24	0.2	1.8	0.2	1.2	0.1	0.1	0.1	0.1
a3-30	0.3	1.1	0.3	0.9	0.2	0.2	0.2	0.2
a3-36	0.8	1.9	1.0	2.4	0.7	0.8	0.3	0.5
a4-32	0.5	3.2	0.6	2.5	0.3	0.3	0.1	0.1
a4-40	0.5	1.5	0.5	5.7	1.0	1.0	0.5	0.5
a4-48	2.4	8.2	2.7	9.1	1.5	2.3	0.8	1.5
a5-40	0.5	1.2	0.4	1.0	0.5	0.5	0.3	0.3
a5-50	1.5	13.3	1.8	11.5	1.7	3.5	0.6	1.4
a5-60	4.7	19.6	5.7	16.6	4.6	7.2	2.0	2.6
a6-48	1.3	8.2	1.6	9.9	1.6	1.6	0.5	0.5
a6-60	5.5	17.1	7.8	24.4	3.9	5.7	1.3	2.4
a6-72	8.6	49.2	12.9	123.0	32.2	46.9	3.2	5.6
a7-56	1.5	8.3	1.4	6.2	1.9	2.2	0.7	1.0
a7-70	4.8	27.4	7.7	33.2	5.1	5.1	1.9	1.9
a7-84	11.4	55.5	16.4	82.4	9.8	51.7	3.6	7.8
a8-64	2.1	11.7	3.6	12.0	4.7	4.7	1.2	1.2
a8-80	11.4	79.3	16.7	178.7	9.5	395.7	3.9	6.2
a8-96	19.7	106.8	35.5	746.8	108.7	178.4	7.8	11.1
avg. time	3.7	19.9	5.6	60.4	9.0	33.7	1.4	2.1

Table 3.10: Computation times for root lower bounds and type a instances in seconds.

Instance	$MP\text{-}I$	$MP\text{-}I^{cut}$	$MP\text{-}I_{sepRT}$	$MP\text{-}I^{cut}_{sepRT}$	MP_{weak}	MP^{cut}_{weak}	MP_{strg}	MP^{cut}_{strg}
b2-16	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1
b2-20	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
b2-24	0.2	0.8	0.2	0.7	0.1	0.5	0.1	0.4
b3-24	0.1	0.5	0.1	0.5	0.1	0.4	0.1	0.4
b3-30	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2
b3-36	0.3	0.4	0.4	0.3	0.4	0.4	0.3	0.3
b4-32	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
b4-40	0.6	1.3	0.6	1.0	0.8	1.4	0.3	0.7
b4-48	2.1	7.4	2.9	4.9	2.3	5.1	0.9	3.2
b5-40	0.2	0.8	0.2	0.7	0.5	0.7	0.3	0.6
b5-50	0.7	0.7	0.7	0.7	0.9	0.9	0.7	0.7
b5-60	2.8	10.1	2.8	8.2	3.6	8.9	1.5	5.6
b6-48	0.4	1.0	0.4	0.7	0.4	0.4	0.2	0.2
b6-60	0.8	0.9	0.9	0.8	1.1	1.1	1.0	1.0
b6-72	4.1	14.9	4.5	13.3	9.6	21.5	2.8	10.5
b7-56	1.4	3.5	1.4	3.2	3.0	566.5	1.0	2.9
b7-70	1.6	10.6	1.8	6.9	1.7	11.2	1.4	6.8
b7-84	5.7	10.7	5.0	9.5	51.3	59.4	3.5	9.6
b8-64	1.2	5.6	1.3	7.3	1.7	6.5	1.0	4.3
b8-80	4.0	9.7	4.0	5.5	3.9	7.9	2.1	6.3
b8-96	10.4	50.1	12.2	26.4	23.3	50.8	7.6	16.8
avg. time	1.8	6.2	1.9	4.3	5.0	35.4	1.2	3.4

Table 3.11: Computation times for root lower bounds and type b instances in seconds.

Instance	*B&C* (Ropke *et al.*, 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*IMP-I*	*IMP-I*^{cut}	*IMP-I*$_{sepRT}$	*IMP-I*$^{cut}_{sepRT}$	*IMP*$_{weak}$	*IMP*$^{cut}_{weak}$	*IMP*$_{strg}$	*IMP*$^{cut}_{strg}$
a2-16	1	1	5	5	3	1	1	1	1	1
a2-20	1	1	7	1	1	1	1	1	1	1
a2-24	4	3	3	3	3	3	1	1	1	1
a3-24	1	1	27	8	23	3	1	1	1	1
a3-30	1	1	22	4	14	1	1	1	1	1
a3-36	10	7	31	22	19	9	3	3	3	3
a4-32	1	1	14	3	25	1	1	1	1	1
a4-40	9	3	18	18	17	2	1	1	1	1
a4-48	5	5	13	13	17	1	5	1	3	1
a5-40	1	1	5	3	13	3	1	1	1	1
a5-50	60	101	1714	996	1271	133	51	7	35	7
a5-60	10	1	47	14	62	13	3	1	3	1
a6-48	1	1	71	25	29	1	1	1	1	1
a6-60	14	3	370	250	92	1	3	1	3	1
a6-72	26	31	686	341	537	29	2	2	27	9
a7-56	68	17	291	201	201	115	13	13	7	15
a7-70	21	7	393	102	70	27	1	1	1	1
a7-84	29	57	606	300	495	7	1	1	25	1
a8-64	87	3	692	456	279	52	1	1	1	1
a8-80	309	270	738	388	621	118	2	1	35	21
a8-96	1653	31	500	277	382	42	1	1	5	1
avg. number	110	26	298	163	199	27	5	2	8	3

Table 3.12: Number of nodes explored within branch-and-cut-and-price algorithms for type a instances.

Instance	*B&C* (Ropke *et al.*, 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*IMP-I*	*IMP-I*^{cut}	*IMP-I*$_{sepRT}$	*IMP-I*$^{cut}_{sepRT}$	*IMP*$_{weak}$	*IMP*$^{cut}_{weak}$	*IMP*$_{strg}$	*IMP*$^{cut}_{strg}$
b2-16	5	1	5	3	3	1	1	1	1	1
b2-20	1	1	1	1	1	1	1	1	1	1
b2-24	2	3	3	3	3	4	3	3	3	3
b3-24	4	17	13	6	13	6	13	5	11	5
b3-30	1	1	1	1	1	1	1	1	1	1
b3-36	1	1	1	1	1	1	1	1	1	1
b4-32	1	1	1	1	1	1	1	1	1	1
b4-40	1	1	1	1	1	1	3	1	3	1
b4-48	16	3	7	15	11	13	3	4	3	5
b5-40	9	1	3	1	3	1	3	1	3	1
b5-50	18	1	1	1	1	1	1	1	1	1
b5-60	98	33	129	307	400	138	1	1	21	17
b6-48	1	1	3	3	3	3	1	1	1	1
b6-60	1	1	1	1	1	1	1	1	1	1
b6-72	990	41	198	89	217	90	1	1	55	13
b7-56	1121	39	60	16	54	18	1	2	33	57
b7-70	18	9	50	2	113	5	1	1	117	11
b7-84	59	9	11	10	22	10	1	1	9	11
b8-64	67	57	199	81	65	122	1	1	73	44
b8-80	9	3	9	7	9	3	2	2	7	3
b8-96	2747	225	680	505	625	505	1	1	415	153
avg. number	246	21	66	50	74	44	2	2	36	16

Table 3.13: Number of nodes explored within branch-and-cut-and-price algorithms for type `b` instances.

Instance	*B&C* (Ropke *et al.*, 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*IMP-I*	*IMP-I*^{cut}	*IMP-I*$_{sepRT}$	*IMP-I*$^{cut}_{sepRT}$	*IMP*$_{weak}$	*IMP*$^{cut}_{weak}$	*IMP*$_{strg}$	*IMP*$^{cut}_{strg}$
a2-16	99	13	4	4	7	11	0	0	0	0
a2-20	183	27	5	11	8	34	0	0	0	0
a2-24	315	21	10	18	8	22	0	0	0	0
a3-24	330	52	42	75	51	57	0	0	0	0
a3-30	564	24	23	25	33	43	0	0	0	0
a3-36	604	64	32	22	28	34	0	0	0	0
a4-32	759	73	39	86	49	104	0	0	0	0
a4-40	896	77	24	27	41	95	0	0	0	0
a4-48	1848	85	27	69	41	98	0	5	0	5
a5-40	937	3	1	4	5	4	0	0	0	0
a5-50	1875	250	226	306	307	163	0	8	0	8
a5-60	2203	102	72	79	79	110	0	11	0	11
a6-48	1893	121	66	83	55	125	0	0	0	0
a6-60	2793	110	126	141	135	174	0	1	0	1
a6-72	3144	337	249	319	252	297	0	8	0	8
a7-56	2338	186	135	118	150	132	0	13	0	0
a7-70	3655	137	159	166	123	210	0	0	0	0
a7-84	3970	374	259	232	297	261	0	38	0	37
a8-64	2995	146	175	177	162	167	0	0	0	0
a8-80	4360	495	248	337	356	410	0	15	0	15
a8-96	7432	780	287	387	399	573	0	6	0	6
avg. number	2057	166	105	128	123	149	0	12	0	11

Table 3.14: Number of cuts generated within branch-and-cut-and-price algorithms for type a instances.

Instance	*B&C* (Ropke *et al.*, 2007)	*B&P-RC* (Ropke and Cordeau, 2005)	*IMP-I*	*IMP-I*^{cut}	*IMP-I*$_{sepRT}$	*IMP-I*$^{cut}_{sepRT}$	*IMP*$_{weak}$	*IMP*$^{cut}_{weak}$	*IMP*$_{strg}$	*IMP*$^{cut}_{strg}$
b2-16	170	11	4	6	6	13	0	0	0	0
b2-20	80	0	0	0	0	0	0	0	0	0
b2-24	225	5	5	11	6	16	0	2	0	2
b3-24	291	11	10	2	10	2	0	1	0	1
b3-30	313	0	0	0	0	0	0	0	0	0
b3-36	341	0	0	0	0	0	0	0	0	0
b4-32	253	0	0	0	0	0	0	0	0	0
b4-40	609	10	5	7	6	8	0	5	0	5
b4-48	1072	31	17	25	16	37	0	9	0	7
b5-40	1243	5	0	3	0	3	0	3	0	3
b5-50	1326	0	0	0	0	0	0	0	0	0
b5-60	1913	78	39	60	77	64	0	10	0	10
b6-48	613	7	2	2	3	3	0	0	0	0
b6-60	1237	0	0	0	0	0	0	0	0	0
b6-72	4278	66	36	65	73	70	0	30	0	28
b7-56	5148	34	20	21	20	24	0	11	0	11
b7-70	1892	22	19	32	34	28	0	35	0	41
b7-84	3463	18	6	5	5	5	0	4	0	4
b8-64	2165	29	42	37	23	60	0	25	0	33
b8-80	1994	6	6	7	13	11	0	4	0	4
b8-96	8431	112	76	90	86	106	0	40	0	35
avg. number	1765	30	21	25	27	30	0	14	0	14

Table 3.15: Number of cuts generated within branch-and-cut-and-price algorithms for type `b` instances.

3.E Detailed Computational Results for Extended Instances

The Tables 3.16-3.19 show detailed computational results for the original instances by Cordeau (2006) and extended instances the were constructed as follows: The time window $[a_i, b_i]$ of each node $i \in N$ is extended by adding 5, 10, and 15 time units to the latest start of service b_i so that the length of the time window is multiplied by factor 4/3, 5/3, and 6/3, respectively. Similarly, the capacity C is enlarged by the same factor. Overall the three new benchmark sets comprise 126 instances.

	orig.			4/3			5/3			6/3		
Instance	*opt*	*MP-I*$^{cut}_{sepRT}$	*MP*$^{cut}_{strg}$	*opt*	*MP-I*$^{cut}_{sepRT}$	*MP*$^{cut}_{strg}$	*opt*	*MP-I*$^{cut}_{sepRT}$	*MP*$^{cut}_{strg}$	*opt*	*MP-I*$^{cut}_{sepRT}$	*MP*$^{cut}_{strg}$
a2-16	294.2	*	*	279.5	*	*	278.2	*	*	278.2	*	*
a2-20	344.8	*	*	344.3	*	*	343.2	*	*	330.7	*	*
a2-24	431.1	430.6	*	418.5	*	*	415.6	*	*	389.1	*	*
a3-24	344.8	344.5	*	337.6	334.3	*	292.4	287.0	292.1	289.6	286.4	289.2
a3-30	494.8	*	*	473.0	*	*	455.3	*	*	452.8	*	*
a3-36	583.2	578.7	579.0	526.3	*	*	503.4	503.4	*	501.0	499.3	499.3
a4-32	485.5	*	*	477.8	477.5	*	466.5	*	*	447.3	446.5	446.5
a4-40	557.7	556.9	*	551.3	*	*	517.0	514.5	516.1	509.0	506.8	*
a4-48	668.8	*	*	667.2	662.9	667.1	634.7	629.2	631.9	619.0	612.6	616.5
a5-40	498.4	497.9	*	498.1	496.6	497.7	482.5	479.9	*	464.1	462.6	*
a5-50	686.6	681.8	686.3	645.2	645.0	*	635.9	633.1	635.7	622.0	616.2	621.4
a5-60	808.4	806.9	*	785.2	783.2	785.1	761.2	759.4	*	745.4	737.6	745.0
a6-48	604.1	*	*	601.8	598.9	601.6	587.0	582.4	586.7	572.5	564.8	572.2
a6-60	819.2	*	*	798.7	798.1	798.6	776.4	773.0	775.4	757.9	750.7	757.8
a6-72	916.0	913.4	914.5	899.3	893.5	*	881.0	863.0	876.6	868.3	844.6	866.1
a7-56	724.0	718.3	721.8	715.4	705.5	712.1	696.4	686.2	694.7	663.4	651.0	659.7
a7-70	889.1	886.5	*	870.5	865.4	*	844.4	835.6	842.9	824.4	811.1	822.6
a7-84	1033.4	1032.0	*	1003.8	1001.5	1003.0	970.1	962.9	967.0	950.6	935.0	947.6
a8-64	747.5	745.7	*	725.1	722.4	*	713.5	704.8	712.4	701.2	689.1	699.6
a8-80	945.7	942.2	945.1	928.1	916.2	926.1	902.2	887.2	898.9	880.3	864.4	878.5
a8-96	1229.7	1225.1	*	1190.0	1184.5	1188.8	1133.1	1120.7	1131.4	1115.1	1094.4	1110.8
# *opt*		7	16		6	12		5	8		4	6
# *best*		7	21		6	21		5	21		6	21
% *gap*		0.22	0.06		0.37	0.05		0.71	0.13		1.00	0.16
avg. time		60.4	2.1		171.7	3.3		301.0	3.9		610.5	5.4

Table 3.16: Lower bound values *lb* for extended type a instances, * if lower bound = opt.

	orig.			4/3			5/3			6/3		
Instance	*opt*	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	*opt*	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$\overline{opt}$ or $\overline{lb}$	$MP\text{-}I^{cut}_{tncsepRT}$	$MP^{cut}_{tncstrg}$	$\overline{opt}$ or $\overline{lb}$	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}
b2-16	309.4	*	*	270.3	265.5	267.7	255.1	254.3	*	244.3	242.2	*
b2-20	332.6	*	*	320.2	*	*	296.6	294.2	*	280.2	*	*
b2-24	444.7	444.6	444.6	417.8	417.2	417.2	385.1	380.7	384.2	372.6	370.2	*
b3-24	394.5	393.9	393.9	350.1	*	*	342.4	341.0	*	321.2	319.0	319.8
b3-30	531.4	*	*	483.7	*	*	445.4	*	*	433.3	431.9	433.2
b3-36	603.8	*	*	557.2	*	*	524.8	*	*	486.0	486.0	*
b4-32	494.8	*	*	450.1	*	*	434.3	*	*	410.1	408.2	409.5
b4-40	656.6	*	*	583.6	583.5	583.5	533.8	533.0	533.3	478.7	478.6	*
b4-48	673.8	673.0	673.2	580.5	580.4	580.4	568.0	562.7	567.4	543.3	537.4	541.9
b5-40	613.7	*	*	571.2	568.3	569.7	524.9	511.1	522.7	494.8	485.0	492.7
b5-50	761.4	*	*	685.3	675.8	683.1	631.4	625.2	*	606.8	586.9	601.4
b5-60	902.0	897.9	898.9	822.1	817.0	821.4	732.1	728.1	732.0	704.5	687.2	701.3
b6-48	714.8	714.7	*	665.8	665.3	665.3	607.2	601.8	606.6	$\overline{577.8}$	558.0	571.8
b6-60	860.1	*	*	790.0	788.0	789.3	743.5	728.8	742.2	$\overline{703.3}$	677.4	697.1
b6-72	978.5	975.4	977.0	901.2	893.2	897.6	842.4	828.5	835.6	789.1	775.7	786.9
b7-56	824.0	822.0	822.2	728.5	726.9	727.5	660.3	656.2	659.7	$\overline{603.9}$	595.1	602.1
b7-70	912.6	911.4	911.7	820.5	814.0	819.2	776.1	769.5	775.1	$\overline{740.9}$	723.2	734.9
b7-84	1203.4	1202.0	1202.0	1086.1	1075.0	1083.6	?	991.0	1004.1	?	930.6	949.9
b8-64	839.9	838.0	838.1	745.8	742.3	745.1	$\overline{664.8}$	658.4	663.1	?	624.0	629.7
b8-80	1036.3	1036.2	1036.2	956.8	955.2	956.6	$\overline{897.4}$	886.0	892.7	$\overline{822.2}$	806.7	815.4
b8-96	1185.6	1182.2	1183.8	1063.4	1053.9	1058.4	?	972.6	988.1	?	915.7	939.8
# *opt*		9	10		5	5		3	7		1	5
# *best*		13	20		9	21		3	21		1	21
% *gap*		0.10	0.07		0.46	0.18		?	?		?	?
avg. time		4.3	3.4		8.0	3.9		19.3	7.2		45.3	13.2

Table 3.17: Lower bound values *lb* for extended type `b` instances, * if lower bound = opt.

	orig.		4/3		5/3		6/3	
Instance	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}
a2-16	0.1	0.1	0.1	0.1	0.2	0.1	0.2	0.1
a2-20	0.3	0.1	0.5	0.1	0.7	0.2	0.1	0.1
a2-24	0.9	0.1	1.4	0.8	3.6	1.3	1.0	0.2
a3-24	1.3	0.1	6.3	0.3	18.0	0.7	37.5	0.9
a3-30	0.9	0.2	4.2	0.2	2.5	0.2	9.8	0.6
a3-36	5.2	1.0	2.8	1.1	13.3	0.5	19.0	2.4
a4-32	2.5	0.1	11.5	0.2	4.5	0.4	19.6	0.8
a4-40	6.8	0.5	4.3	0.5	36.0	2.5	42.3	1.3
a4-48	9.1	1.5	352.8	6.8	1223.6	23.9	1h	10.4
a5-40	1.3	0.3	3.2	1.9	11.8	0.4	22.6	1.4
a5-50	194.4	3.2	15.5	1.5	247.1	6.5	1h	4.2
a5-60	36.6	2.6	695.3	7.1	148.1	2.5	1h	83.8
a6-48	9.9	0.6	167.7	2.7	2024.1	3.6	1h	8.9
a6-60	24.4	2.4	308.6	4.5	1h	18.0	1h	5.8
a6-72	686.0	24.4	1h	10.2	1h	1h	1h	284.1
a7-56	94.8	4.8	1h	22.4	1h	5.8	1h	17.2
a7-70	128.7	1.9	1h	5.1	1h	54.9	1h	59.4
a7-84	146.9	7.8	1h	23.7	1h	259.5	1h	1422.2
a8-64	65.6	1.2	367.9	1.7	1h	9.5	1h	75.3
a8-80	1h	43.0	1h	252.6	1h	520.3	1h	386.9
a8-96	1h	11.1	1h	246.7	1h	111.0	1h	1564.7
# opt	19	21	15	21	13	20	9	21
avg. time	410.3	5.1	1121.1	28.1	1549.2	220.1	2064.4	187.2

Table 3.18: Computation times for optimal integer solutions of extended type a instances in seconds.

	orig.		4/3		5/3		6/3	
Instance	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}	$MP\text{-}I^{cut}_{sepRT}$	MP^{cut}_{strg}
b2-16	0.1	0.1	14.3	0.3	0.7	0.1	1.8	0.1
b2-20	0.1	0.1	0.1	0.1	2.8	0.1	0.2	0.1
b2-24	0.9	0.6	1.1	1.3	8.5	3.6	4.2	0.7
b3-24	0.6	0.5	0.5	0.1	3.0	0.3	2.5	2.9
b3-30	0.1	0.2	0.2	0.2	0.2	0.3	4.9	1.7
b3-36	0.4	0.5	1.5	0.5	1.7	1.0	2.7	1.1
b4-32	0.1	0.2	0.3	0.3	0.8	0.6	6.5	1.6
b4-40	1.0	0.7	2.3	1.6	10.9	3.0	8.4	2.2
b4-48	16.3	5.5	6.6	4.9	1h	16.2	3112.6	25.1
b5-40	0.7	0.6	10.3	5.0	1h	22.3	1h	18.9
b5-50	0.7	0.7	920.4	3.7	1h	2.3	1h	801.5
b5-60	227.1	22.9	1h	25.0	240.0	15.6	1h	489.6
b6-48	0.8	0.3	2.9	2.4	1430.1	6.1	1h	1153.4
b6-60	0.8	1.0	29.6	7.4	1h	62.7	1h	1h
b6-72	212.9	31.4	1h	1691.4	1h	1h	1h	696.8
b7-56	9.3	50.3	44.8	26.8	1376.7	38.5	1h	347.4
b7-70	9.4	13.0	1h	50.0	1h	47.8	1h	1h
b7-84	36.8	71.7	1h	518.6	1h	1h	1h	1h
b8-64	63.7	23.1	187.1	9.3	1h	73.5	1h	1h
b8-80	9.6	10.3	51.6	15.4	1h	1h	1h	1h
b8-96	3031.6	898.8	1h	2135.6	1h	1h	1h	1h
# opt	21	21	16	21	11	17	9	15
avg. time	172.5	53.9	917.8	237.9	1860.7	699.7	2206.8	1197.3

Table 3.19: Computation times for optimal integer solutions of extended type `b` instances in seconds.

References

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, **36**(2), 69–79.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, **54**(3), 573–586.

Desaulniers, G., Desroisiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constraint vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer, Boston, MA.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**(1), 7–22.

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*. Springer, New York, NY.

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.

Ropke, S. and Cordeau, J.-F. (2005). Branch and cut and price for the pickup and delivery problem with time windows. Chapter 9 of S. Ropke's cumulative Ph.D. dissertation, Heuristic and exact algorithms for vehicle routing problems, University of Copenhagen, Copenhagen, Denmark.

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.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, **49**(4), 258–272.

Chapter 4

A Comparison of Column-Generation Approaches to the Vehicle Routing Problem with Time Windows and Temporal Synchronized Pickup and Delivery

Timo Gschwind

Abstract

In the vehicle routing problem with time windows and temporal synchronized pickup and delivery (VRPTWTSPD), user-specified transportation requests from origin to destination points have to be serviced by a fleet of homogeneous vehicles. The task is to find a set of minimum-cost routes satisfying pairing and precedence, capacities, and time windows. Additionally, temporal synchronization constraints couple the service times at the pickup and delivery locations of the customer requests in the following way: A request has to be delivered within prespecified minimum and maximum time lags (called ride times) after it has been picked up. The presence of these ride-time constraints severely complicates the subproblem of the natural column-generation formulation of the VRPTWTSPD. It is not clear if their integration into the subproblem pays off in an integer column-generation approach. We develop four exact approaches to the VRPTWTSPD based on column-generation formulations with differing subproblems. Two of these subproblems are considered for the first time in this paper. We derive new dominance rules and labeling algorithms for their effective solution. Extensive computational results indicate that integrating either both types of ride-time constraints or only the maximum ride-time constraints into the subproblem results in the strongest overall approach.

4.1 Introduction

In the family of one-to-one *pickup-and-delivery problems* (PDPs), customer requests consist of transporting goods or people between paired origin and destination points: for each request a specific good or person has to be picked up at one location and to be transported to the corresponding delivery location. Typically, the task is to design

a set of minimum-cost routes satisfying all customer requests subject to pairing and precedence, and other problem-specific constraints. For details on different PDP-variants we refer to the recent surveys (Berbeglia *et al.*, 2007; Cordeau *et al.*, 2008; Parragh *et al.*, 2008). The qualitative dependence of the visits of paired customer locations, i.e., pick first deliver second, is a key characteristic of PDPs.

A well-studied one-to-one PDP is the *pickup-and-delivery problem with time windows* (PDPTW) (e.g., Dumas *et al.*, 1991; Ropke and Cordeau, 2009; Baldacci *et al.*, 2011a) in which vehicle routes must respect pairing and precedence, capacities, and time windows. In this article, we introduce the *vehicle routing problem with time windows and temporal synchronized pickup and delivery* (VRPTWTSPD). It extends the PDPTW by imposing additional constraints that couple the service times at the pickup and delivery locations of the customer requests in the following way: A delivery node has to be serviced within prespecified *minimum* and *maximum* time lags (called *ride times*) after the service at the corresponding pickup node has been completed. Because both pickup and delivery are performed by the same vehicle, these additional constraints are temporal *intra-route synchronization* constraints. Compared to other PDP-variants, the synchronization imposes that there is not only a qualitative (pickup before delivery) but also a quantitative dependence of the visiting times of paired customer locations in the VRPTWTSPD. As a generalization of the PDPTW the VRPTWTSPD is clearly $\mathcal{NP}$-hard.

As pointed out, e.g., by Dohn *et al.* (2011) or Drexl (2012), synchronization aspects are highly relevant in routing practice and there is a growing interest on *vehicle routing problems* (VRPs) with synchronization constraints in the research community. We see the VRPTWTSPD as the prototypical VRP with intra-route synchronization in the sense that synchronization takes place only within disjunctive pairs of nodes and that there are no other non-standard constraints present. In this respect, the development of an effective algorithm for solving the VRPTWTSPD constitutes a central building block for the solution of richer VRPs with synchronization constraints.

A special case of the VRPTWTSPD is the so-called *dial-a-ride problem* (DARP) in which only a maximum ride time is specified for each pickup-and-delivery pair. The DARP has been subject to extensive research. For a review on different modeling variants and algorithmic approaches to the DARP we refer to the survey of Cordeau and Laporte (2007). The DARP mainly arises in door-to-door transportation services for school children, handicapped persons, or the elderly and disabled (see, e.g., Russell and Morrel, 1986; Madsen *et al.*, 1995; Toth and Vigo, 1997; Borndörfer *et al.*, 1997). In this context, maximum ride times are used to guarantee a certain service level by limiting the time a passenger is on board of the vehicle. A similar service-related use of maximum ride-time constraints is described by Plum *et al.* (2014) in the context of liner shipping service design.

Other applications of temporal intra-route synchronization in which also a minimum ride time is relevant include the planning of security guards where locations have to be inspected repeatedly within given time intervals (Bredström and Rönnqvist, 2008). There, no actual pickup at one location followed by a delivery at another location takes place. Instead, just a pairing and precedence relation between the services at the nodes forming a customer request is given. Similar planning problems arise in home health care,

e.g., when patients have to be monitored by a nurse several times a day (Eveborn *et al.*, 2006). In the service industries, some jobs may require several working steps that cannot be performed in direct succession. A reason might be that after completing one task a workpiece must dry or harden before it can be further processed. Thus, a technician may have to visit a certain location several times with some given time between each visit. Furthermore, when there is a limit on the total working hours of drivers (Ceselli *et al.*, 2009) or when transporting perishable goods (Azi *et al.*, 2010), the time a vehicle is away from the depot has to be restricted. This can be modeled by imposing a maximum ride-time constraint on a dummy request originating and destinating at the depot. Similarly, one might want to have a limit on both the minimum and maximum duration of the routes in order to achieve an even work-distribution of the drivers.

The contributions of this paper are the following: First, we introduce the VRPTWT-SPD as the prototypical VRP with temporal intra-route synchronization. This problem has to the best of our knowledge not been considered before. Second, we develop four exact solution approaches to the VRPTWTSPD based on column-generation formulations whose master programs are formulated on different sets of variables implying different subproblems. Two of these subproblems are considered for the first time in this paper. We derive new dominance rules and labeling algorithms for their solution. One of them is the natural subproblem of the VRPTWTSPD, in which time windows as well as temporal intra-route synchronization with both minimum and maximum ride times have to be dealt with. Finally, to compare the strength of the different solution approaches, we report extensive computational results over a large number of test instances with different characteristics regarding the number of customer requests and the tightness of capacity, time-window, and minimum and maximum ride-time constraints. The analysis shows that integrating either both types of ride-time constraints or only the maximum ride-time constraints into the subproblem results in the strongest overall approach.

Integer column-generation methods have proven to be very successful in solving many VRP-variants including PDPs (e.g., Dumas *et al.*, 1991; Ropke and Cordeau, 2009; Baldacci *et al.*, 2011a). The column-generation master program of such approaches typically is an extended set-partitioning model formulated on variables representing feasible routes for the problem at hand. These formulations generally provide stronger bounds compared to other formulations like, e.g., arc-flow formulations or extended set-partitioning models formulated on a relaxed set of variables. However, the overall success of an integer column-generation approach to VRP-variants relies not only on strong bounds but also on the effective solution of the subproblem.

This is the main challenge when synchronization comes into play (Drexl, 2012). In the case of inter-route synchronization, additional constraints have to be included in the master programs (Desaulniers *et al.*, 1998). Because of the dual variables associated with these constraints, the resulting subproblems are highly complex (e.g., Christiansen and Nygreen, 1998; Ioachim *et al.*, 1999; Dohn *et al.*, 2011) and cannot be solved by standard dynamic-programming labeling algorithms. This is also true for intra-route synchronization where no additional linking constraints are necessary. There, the increased complexity of the subproblems is not caused by additional duals but by the synchronization constraints themselves, which may be hard to incorporate into the sub-

problem. For the DARP, e.g., Hunsaker and Savelsbergh (2002) have demonstrated that in the presence of time windows and maximum ride times checking the feasibility of a given route is intricate. Clearly, the effective generation of such routes within a column-generation approach is even more challenging.

In the case of intra-route synchronization, the complexity of the subproblems can be reduced by relaxing one or more types of constraints in the subproblem and handling them in the master programs instead. The resulting easier-to-solve subproblems come at the cost of weaker lower bounds and, thus, larger branch-and-bound trees. Ropke and Cordeau (2005) follow this approach to solve the DARP with column generation. They relax the maximum ride times in the subproblem and enforce them using *infeasible path elimination constraints* (IPEC) in the master. The resulting subproblem is well studied and effective algorithms for its solution exist (Dumas *et al.*, 1991; Ropke and Cordeau, 2009).

The opposite approach was taken in Chapter 3 which proposed two branch-and-cut-and-price algorithms for the DARP handling all route constraints in the subproblem. A weak and a strong dominance rule resulting in labeling algorithms with differing effectiveness for the solution of the subproblem were derived. Computational results indicated that the algorithm based on the weak dominance rule is inferior to the approach of Ropke and Cordeau (2005), while the one using the strong dominance criterion is superior. Thus, it is a priori not clear if the integration of certain route constraints in the subproblem pays off in the overall solution algorithm for a problem. We, therefore, consider and compare the efficiency of four column-generation algorithms to the VRPTWTSPD. Each algorithm uses a different subproblem: One that handles all route constraints of the VRPTWTSPD, one that relaxes the minimum ride times, one that relaxes the maximum ride times, and one that relaxes both types of ride-time constraints.

The remainder of the paper is organized as follows. Section 4.2 defines the VRPTWTSPD and presents column-generation formulations of it. Section 4.3 describes the dominance rules and labeling algorithms we use for solving the different subproblems. Our basic branch-and-cut-and-price algorithm is briefly described in Section 4.4. Computational results are reported in Section 4.5. Section 4.6 concludes the chapter.

4.2 Problem Definition and Column-Generation Formulations

In this section, we give a formal definition of the VRPTWTSPD and describe different column-generation formulations of it.

4.2.1 Definition of the VRPTWTSPD

The VRPTWTSPD is defined on a directed graph $G = (N, A)$ with node set $N = P \cup D \cup \{0, 2n+1\}$ and arc set A. The subsets $P = \{1, ..., n\}$ and $D = \{n+1, ..., 2n\}$ contain the pickup and delivery nodes of n transportation requests, respectively. Node 0 denotes the origin depot and node $2n+1$ the destination depot. For each request

$i = 1, ..., n$, a minimum ride time $\underline{L}_i$ and a maximum ride time $\overline{L}_i$ are specified, coupling the service times at the pickup node i and the delivery node $i + n$.

With each node $i \in N$, a non-negative service duration s_i and a demand d_i such that $d_i = -d_{i+n}$ for all $i = 1, ..., n$ are associated. We assume $d_0 = d_{2n+1} = 0$. Furthermore, a time window $[a_i, b_i]$ in which the service has to be started is associated with each node $i \in N$. When arriving at node i prior to a_i, the vehicle has to wait until time a_i before starting its service. It is also allowed to delay the start of service voluntarily at any node. We assume that there is no restriction on the length of the waiting times. The possibility of delaying the start of service at some nodes is crucial for the feasibility of routes in the presence of ride times and time windows (see Hunsaker and Savelsbergh, 2002; Chapter 3).

With each arc $(i, j) \in A$, a routing cost c_{ij} and a travel time t_{ij} are associated. We assume that both routing costs and travel times are non-negative and satisfy the triangle inequality. To serve the n transportation requests, a fleet K of identical vehicles with capacity C is located at the depot 0.

The VRPTWTSPD consists in finding $|K|$ vehicle routes starting and ending at the depot nodes 0 and $2n+1$, respectively, such that each request is served exactly once and the total routing costs are minimal. Thereby, the routes have to satisfy the following conditions:

Pairing and precedence: For each request i, pickup node i and delivery node $i+n$ are visited on the same route, and the pickup node i is visited first.

Capacity: The load of the vehicle must not exceed C at any time.

Time windows: For each node i, the start of service must lie within the time window $[a_i, b_i]$.

Ride times: The service at a delivery node $i + n$ has to start at least $\underline{L}_i$ and at most $\overline{L}_i$ units of time after the service at the corresponding pickup node i has been completed.

Note that it is not straightforward to decide on the feasibility of a route in the VRPTWTSPD sense due to the presence of different types of potentially contrasting temporal constraints. More precisely, to verify the feasibility of a given route $r = (h_1, ..., h_q)$ with $h_1 = 0$ and $h_q = 2n + 1$ one has to find a time schedule $T_r = (\tau_1, ..., \tau_q)$ satisfying

$$\tau_i + s_{h_i} + t_{h_i h_{i+1}} \leq \tau_{i+1} \quad \forall i = 1, ..., q-1, \tag{4.1}$$

$$a_{h_i} \leq \tau_i \leq b_{h_i} \quad \forall i = 1, ..., q, \tag{4.2}$$

$$\tau_i + s_{h_i} + \underline{L}_{h_i} \leq \tau_j \quad \text{if } h_i + n = h_j, \tag{4.3}$$

$$\tau_i + s_{h_i} + \overline{L}_{h_i} \geq \tau_j \quad \text{if } h_i + n = h_j, \tag{4.4}$$

where τ_i denotes the start of service at node h_i. Constraints (4.1) ensure consistency of the service times along the route. Inequalities (4.2) impose time windows, while (4.3) and (4.4) are minimum and maximum ride-time constraints, respectively. A schedule

satisfying (4.1)–(4.4) is called feasible. We denote by $\mathscr{T}_r$ the set of all feasible schedules for a route r. Furthermore, let $\mathscr{T}_r(t) = \{T_r \in \mathscr{T}_r : \tau_q \leq t\}$ be the set of feasible schedules with start of service τ_q at the last node h_q not later than t. All definitions and notations for routes and schedules are also used for partial routes, i.e., with $h_q \neq 2n+1$, and corresponding partial schedules. Note that for partial routes it is possible to visit only the pickup node of a request i. In this case, no ride-time constraints (4.3) or (4.4) have to be respected for this request in a partial schedule.

4.2.2 Column-Generation Formulations of the VRPTWTSPD

To formulate the VRPTWTSPD as a set-partitioning problem, let Ω be the set of all VRPTWTSPD-feasible routes. The cost of a route $r \in \Omega$ is denoted by c_r. Moreover, for each route r and each request $i \in P$ denote by $a_{ir} \in \mathbb{Z}$ the number of times request i is performed by route r. Let λ_r be binary variables indicating if route r is used in the solution. The VRPTWTSPD can then be formulated as follows:

$$(IMP) \quad \min \sum_{r\in\Omega} c_r\lambda_r \tag{4.5}$$

$$\text{s.t.} \sum_{r\in\Omega} a_{ir}\lambda_r = 1 \quad \forall i \in P, \tag{4.6}$$

$$\sum_{r\in\Omega} \lambda_r = |K|, \tag{4.7}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \Omega. \tag{4.8}$$

The objective function (4.5) minimizes the total routing costs. Partitioning constraints (4.6) ensure that all requests are served exactly once. Equality (4.7) imposes the number of routes in the solution, while (4.8) are binary conditions for route variables.

Typically, the number $|\Omega|$ of feasible routes is very large so that model *IMP* cannot be solved directly. We, therefore, use an integer column-generation approach to solve it. The linear relaxation of the so-called integer master program *IMP* is initialized with a proper subset of routes and missing routes with negative reduced cost are added dynamically. Integrality is ensured by integrating this process into a branch-and-bound algorithm.

To identify negative reduced-cost routes the column-generation subproblem has to be solved. Let $\pi_i, i \in P$ and μ be the dual variables associated with constraints (4.6) and (4.7), respectively. The reduced cost of arc $(i,j) \in A$ is defined as

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \pi_i & \text{if } i \in P, \\ c_{ij} & \text{otherwise.} \end{cases} \tag{4.9}$$

The reduced cost $\tilde{c}_r$ of a route $r \in \Omega$ is $\tilde{c}_r = \sum_{(i,j)\in A(r)} \tilde{c}_{ij} - \mu$ where $A(r)$ denotes the

sequence of arcs traversed by route r. The subproblem is then given by

$$\min_{r\in\Omega}\{\tilde{c}_r\}. \tag{4.10}$$

The set-partitioning model *IMP* is the most natural formulation for column-generation based approaches to VRP-variants including the VRPTWTSPD in the following sense: The variable set Ω consists of all routes that are feasible for the problem at hand, i.e., the subproblem takes care of all constraints relating to single routes, while the master program comprises only coupling constraints. Decisive for the success of approaches based on such a formulation is that an effective solution procedure for generating feasible routes with negative reduced cost is available. Because of the simultaneous presence of time-window and ride-time constraints, however, the natural subproblem of the VRPTWTSPD is intricate.

An alternative approach is to formulate the master program in relaxed routing variables $r \in \Omega' \supseteq \Omega$ that may violate one or several types of constraints relating to a single route. This can be a promising strategy when generating routes $r \in \Omega$ is complex, while working with the relaxed set Ω' results in a well-solvable subproblem. A commonly used relaxation is to drop the elementarity condition of routes. In this case, the partitioning constraints (4.6) ensure that non-elementary routes can never be part of an integer solution. Thus, *IMP* with variable set Ω' has the same set of optimal solutions as *IMP* with variable set Ω.

This property, however, does not hold for all relaxations Ω' of Ω and a route $r \in \Omega' \setminus \Omega$ that is infeasible for the original problem might then be part of an integer solution. Consequently, the constraints that have been relaxed in the subproblem must be enforced in the master program to ensure feasibility of the solutions. Adding IPEC is one way of doing this. Let $\mathcal{I}$ be the set of all paths that are infeasible with respect to the constraints that have been relaxed in the subproblem. Moreover, for any infeasible path $I \in \mathcal{I}$ and any route r let b_{Ir} be the number of times route r traverses arcs of path I. The IPEC can then be written as

$$\sum_{r\in\Omega'} b_{Ir}\lambda_r \leq |I| - 1 \quad \forall I \in \mathcal{I}, \tag{4.11}$$

where $|I|$ denotes the length of the infeasible path I, i.e., the number of its arcs. We denote by *IMP-I* a master program that incorporates the set-partitioning model (4.5)–(4.8) formulated on a relaxed variable set Ω' together with the IPEC (4.11) to handle the remaining route constraints.

Obviously, approaches based on formulation *IMP* benefit from stronger LP-bounds compared to approaches using formulation *IMP-I*. The reason is that VRPTWTSPD-infeasible routes, which are excluded in the former, may be convex-combined to form routes that do not violate the IPEC in the latter. Typically, the tighter LP-bounds lead to smaller search trees for *IMP*-based approaches. This comes at the cost of a harder to solve subproblem and it is a priori not clear which formulation enables the overall strongest algorithm.

In the following sections, we consider branch-and-price algorithms for the VRPTWTSPD based on four different column-generation formulations. We denote by IMP^{max}_{min} the approach working on variable set Ω in the master program. The addition of IPEC is not necessary in this case and the master program comprises only coupling constraints. The corresponding subproblem SP^{max}_{min} has to generate VRPTWTSPD-feasible routes.

The other approaches formulate their master programs in routing variables $r \in \Omega'$ that relax either the minimum ride times, the maximum ride times, or both. By *IMP-I* we denote the algorithm that ignores both minimum and maximum ride times in the subproblem (denoted *SP*) and handles them using IPEC in the master program. *SP* is the natural subproblem of the PDPTW and generates routes that respect pairing and precedence, capacity, and time-window constraints, i.e., a time schedule satisfying constraints (4.1) and (4.2) exists for such routes.

The approach that handles only the maximum ride times in the subproblem and uses routing variables where the minimum ride times have been relaxed is denoted by *IMP-I*^{max}. The corresponding subproblem is SP^{max}. It is the natural subproblem of the DARP. Routes generated by SP^{max} satisfy pairing and precedence, and capacity constraints. Moreover, these routes can be assigned a time schedule respecting constraints (4.1), (4.2), and (4.4). *IMP-I*$_{min}$ and SP_{min} are the analog to *IMP-I*^{max} and SP^{max} where minimum ride rimes are handled in the subproblem.

4.3 Column-Generation Subproblems

In this section, we describe solution algorithms for the different subproblems *SP*, SP^{max}, SP_{min}, and SP^{max}_{min}. All four subproblems are Elementary Shortest Path Problems with Resource Constraints (ESPPRC) which are typically solved using dynamic-programming labeling algorithms (Irnich and Desaulniers, 2005). In a labeling algorithm, partial paths are gradually extended in a graph G seeking to find a minimum-cost path from the source node to the sink node. The partial paths are represented by labels that store the accumulated cost and resource consumption along the path. We denote by $\mathcal{P}_\ell$ the partial path corresponding to label ℓ. Decisive for the effectiveness of a labeling algorithm is the use of strong dominance rules to eliminate unpromising labels. A more detailed discussion on ESPPRC and labeling algorithms can be found, e.g., in (Irnich and Desaulniers, 2005).

Note that for the rest of this paper we consider the non-elementary versions of the four subproblems for the following two reasons: First, preliminary computational results indicated that the linear-relaxation lower bounds of the master programs obtained by subproblems with the elementarity conditions were rarely stronger compared to the corresponding non-elementary subproblems. This resulted in slightly weaker overall algorithms for the former. Second, the extension of all dominance rules and labeling algorithms to the elementary case is straightforward (see Ropke and Cordeau, 2009; Chapter 3). In the presence of pairing and precedence, non-elementarity means that a request can be picked up again, after it has been picked up and delivered. Hence, several pickup-and-delivery pairs of the same request can be present in a path. For

ease of notation, however, we assume for the rest of the paper that all partial paths are elementary. Furthermore, we assume that the service duration is zero for all nodes. All proofs and arguments are analog when considering non-elementary partial paths and non-zero service durations.

Also, we assume that the reduced-cost matrix satisfies $\tilde{c}_{ij} \leq \tilde{c}_{ik} + \tilde{c}_{kj}$ for all $(i,j) \in A, k \in D$. Ropke and Cordeau (2009) call this property the *delivery triangle inequality* (DTI). It enables the use of stronger dominance rules for all considered subproblems. Roughly speaking, the DTI ensures that visiting an additional delivery node is never beneficial. Ropke and Cordeau (2009) also show how to transform a reduced-cost matrix that does not satisfy the DTI into one that does, while maintaining the cost of each route unchanged. Hence, working with this assumption is no restriction.

All notation previously introduced for (partial) routes is also used for (partial) paths in the following. Moreover, we use the same notation for all subproblems. The meaning should be clear from the context. The set of feasible schedules $\mathcal{T}_\mathcal{P}$ for a path $\mathcal{P}$, e.g., always refers to feasibility regarding the temporal constraints that are present in the considered subproblem.

4.3.1 *SP* - Subproblem without Ride-Time Constraints

SP is an elementary shortest-path problem with pairing and precedence, capacities, and time windows. It is the natural subproblem of the PDPTW. In this context, it has been subject to prior research (Dumas *et al.*, 1991; Ropke and Cordeau, 2009; Baldacci *et al.*, 2011a) and strong dominance rules exist for its solution by a labeling algorithm.

In what follows, we summarize the main concepts of Dumas *et al.* (1991) and Ropke and Cordeau (2009) for solving *SP*. Both the dominance rule and the labeling strategy for *SP* also serve as basis for the solution approaches to the other subproblems in Sections 4.3.2–4.3.4. Table 4.1 summarizes all resources that are needed in the solution algorithms for the different subproblems and indicates which resource is relevant for which subproblem.

Dominance Rule for *SP* Within each label ℓ, the following information has to be stored: the node η_ℓ the label belongs to, its reduced cost $\tilde{c}_\ell$, the earliest start of service t_ℓ at node η_ℓ, and the set of open requests O_ℓ (requests that have been picked up but not yet delivered). Then, the following dominance rule is valid for *SP* (Dumas *et al.*, 1991):

Proposition 4.1. *(Dom-SP) A feasible label ℓ_1 dominates a label ℓ_2 if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \text{ and } \quad O_{\ell_1} \subseteq O_{\ell_2}. \tag{4.12}$$

Note that if the DTI does not hold, dominance is only possible between labels with identical sets of open requests $O_{\ell_1} = O_{\ell_2}$.

Labeling Algorithm for *SP* We now briefly describe the labeling algorithm of Dumas *et al.* (1991) and Ropke and Cordeau (2009) for solving *SP*. In addition to the

Resource	Description	SP	SP^{max}	SP_{min}	SP^{max}_{min}
η_ℓ	The node of the label	•	•	•	•
$\tilde{c}_\ell$	The reduced cost	•	•	•	•
t_ℓ	The earliest start of service at the node η_ℓ	•	•	•	•
l_ℓ	The current load	•	•	•	•
O_ℓ	The set of open requests	•	•	•	•
$\tilde{b}_\ell$	The latest feasible start of service at node η_ℓ		•		•
$ld^i_\ell(t)$	The latest possible delivery time of request $i \in O_\ell$		•		•
B^i_ℓ	The point of time when $ld^i_\ell(t)$ becomes constant		•		•
$\underline{ld}^i_\ell(t)$	The latest possible delivery time of request $i \in O_\ell$ such that all other open requests $j \in O_\ell \setminus \{i\}$ are picked up as early as possible				•
$\underline{B}^i_\ell$	The point of time when $\underline{ld}^i_\ell(t)$ becomes constant				•
ed^i_ℓ	The earliest feasible start of service at node η_ℓ			•	•
$\overline{ed}^i_\ell(t)$	The earliest possible delivery time of request $i \in O_\ell$ such that all other open requests $j \in O_\ell \setminus \{i\}$ are picked up as late as possible				•
$\overline{B}^i_\ell$	The point of time when $\overline{ed}^i_\ell(t)$ becomes constant				•

Table 4.1: Resources of a label ℓ. A bullet indicates that the resource is relevant for the respective subproblem.

resources needed for dominance in Proposition 4.1, they store at each label ℓ the load l_ℓ of the vehicle when leaving η_ℓ, enabling a fast consistency check regarding capacity. The extension of a label ℓ along arc $(\eta_\ell, x) \in A$ is only allowed if either $x \notin O_\ell$ if $x \in P$, or $x - n \in O_\ell$ if $x \in D$, or $O_\ell = \varnothing$ if $x = 2n + 1$ holds. Otherwise, pairing and precedence are not satisfied resulting in an infeasible label. Furthermore, consistency with respect to time-window and capacity constraints is ensured by requiring $t_\ell + t_{\eta_\ell,x} \leq b_x$ and $l_\ell + d_x \leq C$, respectively.

If extending label ℓ along arc $(\eta_\ell, x) \in A$ is feasible, a new label ℓ' is created. Its resources are determined according to the following resource extension functions (REFs):

$$\eta_{\ell'} = x, \quad \tilde{c}_{\ell'} = \tilde{c}_\ell + \tilde{c}_{\eta_\ell,x}, \quad t_{\ell'} = \max\{a_x, t_\ell + t_{\eta_\ell,x}\}, \quad l_{\ell'} = l_\ell + d_x, \tag{4.13}$$

$$O_{\ell'} = \begin{cases} O_\ell \cup \{x\} & \text{if } x \in P, \\ O_\ell \setminus \{x - n\} & \text{if } x \in D. \end{cases} \tag{4.14}$$

To reduce the number of labels that have to be processed in the algorithm, unpromising labels are eliminated using dominance rule *Dom-SP*. Moreover, labels that cannot be feasibly completed to node $2n + 1$ can be discarded. For SP, pairing constraints require that each feasible completion to a label ℓ must visit the delivery nodes $i + n$ of all open requests $i \in O_\ell$ and thereby obey all time-window constraints. If no such completion exists, then label ℓ can be eliminated. When travel times satisfy the triangle inequality,

testing if such a completion exists reduces to solving a traveling salesman problem with time windows (TSPTW) over the nodes $\{i+n : i \in O_\ell\} \cup \{\eta_\ell, 2n+1\}$, which is known to be $\mathcal{NP}$-hard. Thus, we consider only subsets of O_ℓ of at most two requests, as proposed by Dumas *et al.* (1991).

4.3.2 SP^{max} - Subproblem with Maximum Ride-Time Constraints

In SP^{max}, the natural subproblem of the DARP, paths have to respect pairing and precedence, capacities, time windows, and maximum ride times. The latter two impose that for each feasible path a time schedule satisfying inequalities (4.1), (4.2), and (4.4) must exist. The main difficulty for solution approaches to SP^{max} is to deal with these partially contrasting temporal constraints. In fact, they impose a trade-off between servicing all nodes as early as possible and servicing pickup nodes as late as possible. The implication for labeling algorithms is as follows. Considering only the earliest start of service (as in *Dom-SP*) is not sufficient to guarantee dominance with respect to the temporal constraints of SP^{max} (see Example 1 of Chapter 3). Thus, one either has to include additional time-related resources in a dominance rule based on *Dom-SP* or come up with a different strategy to deal with the temporal constraints of SP^{max}.

Chapter 3 proposed an effective labeling algorithm for solving SP^{max} that uses an extended version of *Dom-SP* as dominance rule. The basic idea is the following: Let ℓ be a label with $O_\ell \neq \varnothing$. For each open request $i \in O_\ell$, the corresponding maximum ride-time constraint (4.4) imposes an upper bound on the start of service at delivery node $i+n$ restricting the set of feasible completions to ℓ. Clearly, a larger value for this bound is preferable. As a result, dominance between two labels is only possible if for each of its open requests the dominating label has a larger upper bound value for the start of service at the respective delivery node. Determining these bounds, however, is not straightforward. They obviously depend on the actual service times at the corresponding pickup nodes within the path $\mathcal{P}_\ell$. Thereby, the possibility to delay the start of service at some nodes has to be incorporated.

Dominance Rule for SP^{max} To formalize the approach in a dominance rule, Chapter 3 first defines the *latest possible delivery time* ld_ℓ^i, i.e., the latest feasible start of service at the delivery node, of an open request $i \in O_\ell$ as a function in the start of service $t \geq t_\ell$ at the current node η_ℓ. Let $\mathcal{P}_\ell = (h_1, ..., h_q = \eta_\ell)$ be the path corresponding to label ℓ. Then, $ld_\ell^{h_i}(t)$ with $t \geq t_\ell$ and $h_i \in O_\ell$ is given by

$$ld_\ell^{h_i}(t) = \min\{b_{h_i+n}, \overline{\tau}_i(t) + \overline{L}_{h_i}\}, \tag{4.15}$$

where $\overline{\tau}_i(t) = \max_{T_{\mathcal{P}_\ell} \in \mathcal{T}_{\mathcal{P}_\ell}(t)}\{\tau_i\}$ is the latest feasible start of service at the pickup node h_i while $\tau_q \leq t$.

Moreover, two important properties of $ld_\ell^{h_i}(t)$ are proven. First, Chapter 3 shows that the schedule $\overline{T}_{\mathcal{P}_\ell}(t) = (\overline{\tau}_1(t), ..., \overline{\tau}_q(t))$ assigning each node $h_i, i = 1, ..., q$ its latest start of service $\overline{\tau}_i$ is feasible. This means that all open requests and the associated

latest delivery times can be treated independently in the dominance criterion. Second, Chapter 3 shows that the functions $ld_\ell^{h_i}(t)$ are of the form $ld_\ell^{h_i}(t) = \min\{k_1^i + t, k_2^i\}$ with constants k_1^i and k_2^i. With this property, the comparison of two such functions can be simplified to comparing them at two distinct points of time.

Let B_ℓ^i be the point of time when $ld_\ell^i(t)$ becomes constant. The following proposition describes a valid dominance rule for SP^{max} (Chapter 3):

Proposition 4.2. *(Dom-SP^{max}) A feasible label ℓ_1 dominates a label ℓ_2 if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \text{ and} \tag{4.16}$$

$$ld^i_{\ell_1}(t_{\ell_1}) + (t_{\ell_2} - t_{\ell_1}) \geq ld^i_{\ell_2}(t_{\ell_2}) \text{ and } ld^i_{\ell_1}(B^i_{\ell_1}) \geq ld^i_{\ell_2}(B^i_{\ell_2}) \quad \forall i \in O_{\ell_1}. \tag{4.17}$$

Labeling Algorithm for SP^{max} The labeling algorithm of Chapter 3 for solving SP^{max} is analog to that of Dumas *et al.* (1991) and Ropke and Cordeau (2009) for SP sketched in Section 4.3.1. The additional presence of maximum ride times and the use of dominance rule *Dom-SP^{max}* involve only some minor modifications. First, the extension of a label ℓ along an arc $(\eta_\ell, x) \in A$ is only feasible if $t_\ell + t_{\eta_\ell,x} \leq ld^i_\ell(B^i_\ell)$ holds for all $i \in O_\ell$. Second, the resources $B^i_{\ell'}$, $ld^i_{\ell'}(t_{\ell'})$, and $ld^i_{\ell'}(B^i_{\ell'})$ have to be determined for all $i \in O_{\ell'}$ when creating a new label ℓ' resulting from the extension of ℓ along arc (η_ℓ, x). Chapter 3 devised the following simple REFs:

$$B^i_{\ell'} = \begin{cases} \min\{\tilde{b}_x, b_{x+n} - \overline{L}_x\} & \text{if } i = x, \\ \max\{t_{\ell'}, \min\{\tilde{b}_x, B^i_\ell + t_{\eta_\ell,x}\}\} & \text{otherwise,} \end{cases} \tag{4.18}$$

$$ld^i_{\ell'}(t_{\ell'}) = \begin{cases} \min\{b_{x+n}, t_{\ell'} + \overline{L}_x\} & \text{if } i = x, \\ ld^i_\ell(t_\ell) + (\min\{t_{\ell'} - t_{\eta_\ell,x}, B^i_\ell\} - t_\ell) & \text{otherwise,} \end{cases} \tag{4.19}$$

$$ld^i_{\ell'}(B^i_{\ell'}) = \begin{cases} B^i_{\ell'} + \overline{L}_x & \text{if } i = x, \\ ld^i_\ell(B^i_\ell) - \max\{0, B^i_\ell + t_{\eta_\ell,x} - \tilde{b}_x\} & \text{otherwise,} \end{cases} \tag{4.20}$$

where

$$\tilde{b}_x = \begin{cases} b_x & \text{if } x \in P, \\ \min\{b_x, ld^{x-n}_\ell(B^{x-n}_\ell)\} & \text{if } x \in D, \end{cases} \tag{4.21}$$

is the latest feasible start of service at the current node η_ℓ. Third, the information on the latest possible delivery times $ld^i_\ell, i \in O_\ell$ can also be used for the elimination of labels that cannot be feasibly completed to node $2n + 1$.

4.3.3 SP_{min} - Subproblem with Minimum Ride-Time Constraints

Subproblem SP_{min} is an elementary shortest path problem with pairing and precedence, capacity, time-window, and minimum ride-time constraints. To the best of our knowledge, SP_{min} has not been considered before and an effective labeling algorithm for its solution is presented here for the first time.

Similar to SP^{max}, different types of temporal constraints are present in SP_{min}. More precisely, a schedule satisfying inequalities (4.1)–(4.3) must be assignable to each feasible path. The main task for a labeling approach to SP_{min} based on *Dom-SP* is to ensure consistency of the dominance rule with these constraints.

In contrast to SP^{max}, however, the temporal constraint system of SP_{min} is rather straightforward to handle in a labeling algorithm. Both types of constraints that couple the service times at two different nodes are less or equal constraints (from front to back of the path). Consequently, the optimal strategy regarding time-window constraints, i.e., servicing all nodes as early as possible, is also an optimal strategy in the additional presence of minimum ride-time constraints. This implies that waiting and delaying the service at some node is never beneficial and the possibility to do so can be neglected. Still, inequalities (4.3) induce that a time schedule in SP_{min} is linked not only between consecutive nodes. Thus, for a label ℓ not only the service time at the current node η_ℓ, but also the service times at all open requests $i \in O_\ell$ are important.

Dominance Rule for SP_{min} To obtain a formal dominance criterion for SP_{min}, we follow the approach of Chapter 3 for SP^{max}. For each open request $i \in O_\ell$ of label ℓ, the minimum ride-time constraints impose a lower bound on the start of service at the delivery node $i+n$. We define this *earliest possible delivery time* $ed_\ell^{h_i}$ for request $h_i \in O_\ell$ as

$$ed_\ell^{h_i} = \max\{a_{h_i+n}, t_\ell + t_{\eta_\ell,h_i+n}, \underline{\tau}_i + \underline{L}_{h_i}\}, \tag{4.22}$$

where $\underline{\tau}_i = \min_{T_{\mathcal{P}_\ell} \in \mathscr{T}_{\mathcal{P}_\ell}}\{\tau_i\}$ with $\mathcal{P}_\ell = (h_1, ..., h_q = \eta_\ell)$ is the earliest feasible start of service at the pickup node h_i. Regarding the set of feasible completions to ℓ, a small value $ed_\ell^{h_i}$ is obviously less restrictive than a larger one. Furthermore, the following lemma shows that for each feasible path $\mathcal{P}$ the time schedule $\underline{T}_\mathcal{P}$ which assigns each node its earliest feasible start of service is feasible. Thus, the values $ed_\ell^{h_i}$ can be treated independently in a dominance rule.

Lemma 4.1. *Let $\mathcal{P} = (h_1, ..., h_q)$ be a feasible partial path. Then, $\underline{T}_\mathcal{P} = (\underline{\tau}_1, ..., \underline{\tau}_q) \in \mathscr{T}_\mathcal{P}$.*

The proofs of all lemmas and propositions are presented in Section 4.A of the Appendix.

Using the values ed_ℓ^i, we obtain the following extension to *Dom-SP* that is a valid dominance criterion for SP_{min}:

Proposition 4.3. *(Dom-SP_{min}) A feasible label ℓ_1 dominates a label ℓ_2 if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \text{ and } \quad ed_{\ell_1}^i \leq ed_{\ell_2}^i \ \forall i \in O_{\ell_1}. \tag{4.23}$$

Labeling Algorithm for SP_{min} SP_{min} can be solved using the labeling algorithm of Section 4.3.1 for solving *SP*. Let ℓ' be the label resulting from the extension of label ℓ

along arc (η_ℓ, x). The REFs for the additional resources $ed^i_{\ell'}, i \in O_{\ell'}$ are

$$ed^i_{\ell'} = \begin{cases} \max\{a_{x+n}, t_{\ell'} + t_{x,x+n}, t_{\ell'} + \underline{L}_x\} & \text{if } i = x, \\ \max\{ed^i_\ell, t_{\ell'} + t_{x,i+n}\} & \text{otherwise.} \end{cases} \tag{4.24}$$

Moreover, the REF (4.13) for the earliest start of service t_ℓ has to be replaced by

$$t_{\ell'} = \begin{cases} \max\{ed^{x-n}_\ell, t_\ell + t_{\eta_\ell,x}\} & \text{if } x \in D, \\ \max\{a_x, t_\ell + t_{\eta_\ell,x}\} & \text{otherwise.} \end{cases} \tag{4.25}$$

Again, the information ed^i_ℓ is also used for eliminating labels that cannot be completed to feasible $0-(2n+1)$-paths.

4.3.4 SP^{max}_{min} - Subproblem with Minimum and Maximum Ride-Time Constraints

Subproblem SP^{max}_{min} is the natural subproblem of the VRPTWTSPD in which generated paths represent VRPTWTSPD-feasible routes, i.e., they have to respect pairing and precedence, capacities, time windows, and minimum and maximum ride times. The implied scheduling problem is (4.1)–(4.4). It simultaneously includes both minimum and maximum ride times which significantly complicates SP^{max}_{min} compared to SP^{max} and SP_{min}. The key problem is the interference of different types of ride-time constraints of different requests so that a straightforward combination of the approaches of Sections 4.3.2 and 4.3.3 is not possible. We demonstrate this in more detail in the following.

Generalizing the idea of Chapter 3, the minimum and maximum ride times of an open request $i \in O_\ell$ impose a lower bound (ed^i_ℓ) and an upper bound (ld^i_ℓ) on the start of service at the delivery node $i+n$. Again, a small value ed^i_ℓ and a large value ld^i_ℓ are preferable. The implied optimal strategies for the start of service at the pickup node i, i.e., an early-as-possible service to minimize ed^i_ℓ and a late-as-possible service to maximize ld^i_ℓ, are clearly opposing. Even more, different strategies for the pickup times of different open requests may interfere. More precisely, servicing one node late may imply that another one cannot be serviced early, and vice versa. As a result, there is generally no feasible time schedule that minimizes the values ed^i_ℓ for some $i \in O_\ell$ and at the same time maximizes the values ld^j_ℓ for some other $j \in O_\ell$. Thus in SP^{max}_{min}, the open requests and the associated earliest and latest delivery times cannot be treated independently in a dominance rule. Table 4.2 gives a small example to illustrate this.

Let ℓ_1 and ℓ_2 be two labels representing the paths $\mathcal{P}_{\ell_1} = (0, i, j, k)$ and $\mathcal{P}_{\ell_2} = (0, j, i, k)$. Assume identical travel times of 10 between all nodes. Furthermore, let the minimum and maximum ride times for all requests be 40 and 50, respectively. The time windows of nodes 0, i, j, and k are specified in Table 4.2, while the time windows at the corresponding delivery nodes are assumed to be not binding ($[0, \infty]$). Then, the earliest possible delivery times of requests i and j for label ℓ_1 (as defined in Section 4.3.3) are $ed^i_{\ell_1} = \max\{0, 50+10, 10+40\} = 60$ and $ed^j_{\ell_1} = \max\{0, 50+10, 20+40\} = 60$. The

	Nodes in $\mathcal{P}(\ell_1)$	0	i	j	k
Label ℓ_1 for path $(0,i,j,k)$	Time window $[a.,b.]$	$[0,100]$	$[0,30]$	$[20,50]$	$[50,50]$
	Earliest start of service t_{ℓ_1}	0	10	20	50
	Earliest possible delivery ed_{ℓ_1}	–	50	60	90
	Latest possible delivery ld_{ℓ_1}	–	80	90	100
	Nodes in $\mathcal{P}(\ell_2)$	0	j	i	k
Label ℓ_2 for path $(0,j,i,k)$	Time window $[a.,b.]$	$[0,100]$	$[20,50]$	$[0,30]$	$[50,50]$
	Earliest start of service t_{ℓ_2}	0	20	30	50
	Earliest possible delivery ed_{ℓ_2}	–	60	70	90
	Latest possible delivery ld_{ℓ_2}	–	70	80	100

Table 4.2: Label ℓ_1 dominates label ℓ_2 in the sense of $Dom-SP^{max}$ and $Dom-SP_{min}$. This does not imply a valid dominance relation for SP^{max}_{min}.

latest possible delivery times $ld^i_{\ell_1}$ and $ld^j_{\ell_1}$ as defined in Section 4.3.2 are, in general, functions in the start of service at the current node $\eta_{\ell_1} = k$. Here, the only feasible start of service at node k is at time 50. Thus, we only need to consider the values $ld^i_{\ell_1}(50) = \min\{\infty, 30+50\} = 80$ and $ld^i_{\ell_1}(50) = \min\{\infty, 40+50\} = 90$, which imply delaying the start of service at node i until time 30 and at node j until time 40. It is easy to see, however, that there is no feasible time schedule $T_{\mathcal{P}_{\ell_1}}$ that at the same time allows a latest possible delivery time of 80 for request i and an earliest possible delivery time of 60 for request j. Note that the former implies a service time not smaller than 30 at node i while the latter implies a service time not larger than 20 at node j.

As a result, simply combining the relations for ed^i_ℓ and ld^i_ℓ of dominance rules *Dom-SP*$_{min}$ and *Dom-SP*^{max} does not lead to a valid dominance criterion for SP^{max}_{min}. The completion $Q = (j+n, i+n, k+n, 2n+1)$, e.g., is feasible for label ℓ_2 but infeasible for ℓ_1, although $ed^x_{\ell_1} < ed^x_{\ell_2}$ and $ld^x_{\ell_1}(50) > ld^x_{\ell_2}(50)$ hold for $x = i, j, k$ (see Table 4.2).

Dominance Rule for SP^{max}_{min} The example above has shown that the interdependence of different open requests has to be incorporated when trying to dominate labels in SP^{max}_{min}. Roughly speaking, this means that one has to be careful when determining the earliest and latest delivery times of the open requests of a label.

On the one hand, there are completions where one or more requests can only be delivered at the earliest or latest possible time. Consequently, for a dominated label ℓ we have to consider the best possible values for feasible delivery times of all open requests $i \in O_\ell$, i.e., we consider ed^i_ℓ and $ld^i_\ell(t), t \geq t_\ell$ as defined in Sections 4.3.3 and 4.3.2, respectively. Note again that here $\mathscr{T}_{\mathcal{P}_\ell}$ refers to the set of all schedules satisfying constraint system (4.1)–(4.4).

On the other hand, a completion might generally require picking up some open requests early and some other open requests late. For a dominating label ℓ with $\mathcal{P}_\ell = (h_1, ..., h_q = \eta_\ell)$ we, therefore, determine two bounds on the service time of an open request $h_i \in O_\ell$

that are independent of the service times at the other open requests $h_j \in O_\ell \setminus \{h_i\}$.

First, we use the following upper bound $\underline{\tau}_i^{\overline{O}_\ell}(t)$ for an early-as-possible service at a pickup node h_i within path $\mathcal{P}_\ell$. Practically speaking, $\underline{\tau}_i^{\overline{O}_\ell}(t)$ gives the earliest service time at h_i that can be attained without restricting the pickup times of the other open requests. Or from the opposite perspective, when scheduling all other open requests $h_j \in O_\ell \setminus \{h_i\}$ in the most unfavorable way for picking up h_i early, i.e., as late as possible, then the earliest possible service time at h_i that is still feasible is $\underline{\tau}_i^{\overline{O}_\ell}(t)$. Formally, $\underline{\tau}_i^{\overline{O}_\ell}(t) = \min_{T_{\mathcal{P}_\ell} \in \mathscr{T}_{\mathcal{P}_\ell}^{i,\overline{O}_\ell}(t)} \{\tau_i\}$ with $\mathscr{T}_{\mathcal{P}_\ell}^{i,\overline{O}_\ell}(t) = \{T_{\mathcal{P}_\ell} \in \mathscr{T}_{\mathcal{P}_\ell}(t) : \tau_j \geq \overline{\tau}_j(t)\ \forall h_j \in O_\ell \setminus \{h_i\}\}$. Note that $\underline{\tau}_i^{\overline{O}_\ell}(t)$ is a function in t, as the times $\overline{\tau}_j(t)$ depend on t.

Second and analog to $\underline{\tau}_i^{\overline{O}_\ell}(t)$, a lower bound for a late-as-possible service at node h_i is denoted by $\overline{\tau}_i^{O_\ell}(t)$. It gives the latest feasible start of service at the pickup node h_i such that the start of service at all other open requests $h_j \in O_\ell \setminus \{h_i\}$ takes its minimal value $\underline{\tau}_j$ and $\tau_q \leq t$, i.e., $\overline{\tau}_i^{O_\ell}(t) = \max_{T_{\mathcal{P}_\ell} \in \mathscr{T}_{\mathcal{P}_\ell}^{i,\underline{O}_\ell}(t)} \{\tau_i\}$ with $\mathscr{T}_{\mathcal{P}_\ell}^{i,\underline{O}_\ell}(t) = \{T_{\mathcal{P}_\ell} \in \mathscr{T}_{\mathcal{P}_\ell}(t) : \tau_j \leq \underline{\tau}_j\ \forall h_j \in O_\ell \setminus \{h_i\}\}$. Maximizing the service at h_i may delay the start of service τ_q at the current node η_ℓ. Thus, $\overline{\tau}_i^{O_\ell}(t)$ is also a function in t.

Using $\underline{\tau}_i^{\overline{O}_\ell}(t)$ and $\overline{\tau}_i^{O_\ell}(t)$ we have the following upper and lower bounds for the earliest and latest delivery times of an open request $h_i \in O$, respectively:

$$\overline{ed}_\ell^{h_i}(t) = \max\{a_{h_i+n}, t_\ell + t_{\eta_\ell, h_i+n}, \underline{\tau}_i^{\overline{O}_\ell}(t) + \underline{L}_{h_i}\}, \tag{4.26}$$

$$\underline{ld}_\ell^{h_i}(t) = \min\{b_{h_i+n}, \overline{\tau}_i^{O_\ell}(t) + \overline{L}_{h_i}\}. \tag{4.27}$$

A valid dominance rule for SP_{min}^{max} is then given by:

Proposition 4.4. *(Dom^*-SP_{min}^{max}) A feasible label ℓ_1 dominates a label ℓ_2 if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \text{ and} \tag{4.28}$$

$$\overline{ed}_{\ell_1}^i(t) \leq ed_{\ell_2}^i \text{ and } \underline{ld}_{\ell_1}^i(t) \geq ld_{\ell_2}^i(t) \quad \forall i \in O_{\ell_1}, t \in [t_{\ell_2}, \tilde{b}_{\ell_2}]. \tag{4.29}$$

Applying dominance rule Dom^*-SP_{min}^{max} requires the comparison of different functions in inequalities (4.29) which is clearly not practicable for general functions within a labeling algorithm. The following lemma characterizes the shape of the functions $\overline{ed}_\ell^i(t)$ and $\underline{ld}_\ell^i(t)$ allowing for a simplified version of Dom^*-SP_{min}^{max}.

Lemma 4.2. *Let $\mathcal{X}$ be the set of all $X = (x_1, ..., x_q) \in \mathbb{R}^q$ satisfying*

$$a_i \leq x_i \leq b_i \quad \forall i = 1, ..., q, \tag{4.30}$$

$$x_i + c_{ij} \leq x_j \quad \forall j = 2, ..., q; i < j, \tag{4.31}$$

$$x_i + d_{ij} \geq x_j \quad \forall j = 2, ..., q; i < j, \tag{4.32}$$

with real-valued constants a_i, b_i, c_{ij}, and d_{ij}. Denote $\mathcal{X}(t) = \{X \in \mathcal{X} : x_q \leq t\}, t \in \mathbb{R}$. Let also be $\overline{x}_i = \max_{X \in \mathcal{X}}\{x_i\}$, $\underline{x}_i = \min_{X \in \mathcal{X}}\{x_i\}$, $\overline{x}_i(t) = \max_{X \in \mathcal{X}(t)}\{x_i\}$, and $\underline{x}_i(t) =$

$\min_{X\in\mathcal{X}(t)}\{x_i\}$. *Furthermore, define by* $\mathcal{X}^{\underline{S}}(t) = \{X \in \mathcal{X}(t) : x_i \leq \underline{x}_i(t)\ \forall i \in \underline{S}\}$ *and by* $\mathcal{X}^{\overline{S}}(t) = \{X \in \mathcal{X}(t) : x_i \geq \overline{x}_i(t)\ \forall i \in \overline{S}\}$ *with* $\underline{S}, \overline{S} \subseteq \{1, ..., q\}$. *Denote* $\overline{x}_i^{\underline{S}}(t) = \max_{X\in\mathcal{X}^{\underline{S}}(t)}\{x_i\}$ *and* $\underline{x}_i^{\overline{S}}(t) = \min_{X\in\mathcal{X}^{\overline{S}}(t)}\{x_i\}$. *Finally, let* t^* *be the smallest* t *with* $\mathcal{X}(t) \neq \varnothing$. *Then, the following properties hold:*

1. $\overline{x}_i(t) = \min\{k_i^1, k_i^2 + t\}$ *for all* $i = 1, ..., q, t \geq t^*$ *with constants* k_i^1 *and* k_i^2.
2. $\underline{x}_i(t) = \underline{x}_i$ *for all* $i = 1, ..., q, t \geq t^*$.
3. $\overline{x}_i^{\underline{S}}(t) = \min\{k_i^1, k_i^2 + t\}$ *for all* $i = 1, ..., q, t \geq t^*$ *with constants* k_i^1 *and* k_i^2.
4. $\underline{x}_i^{\overline{S}}(t) = \max\{k_i^1, \min\{k_i^2, k_i^3 + t\}\}$ *for all* $i = 1, ..., q, t \geq t^*$ *with constants* k_i^1, k_i^2, *and* k_i^3.

Clearly, the scheduling problem (4.1)–(4.4) of SP_{min}^{max} is a special case of the constraint system (4.30)–(4.32) considered in Lemma 4.2. Thus, the functions $\overline{ed}_\ell^i(t)$, $ld_\ell^i(t)$, and $\underline{ld}_\ell^i(t)$ are of the forms $\overline{ed}_\ell^i(t) = \max\{k_i^1, \min\{k_i^2, k_i^3 + t\}\}$, $ld_\ell^i(t) = \min\{k_i^4, k_i^5 + t\}$, and $\underline{ld}_\ell^i(t) = \min\{k_i^6, k_i^7 + t\}$, where all $k_i^\cdot$ are constants. This result is similar to that of Chapter 3 for the latest possible delivery times in SP^{max}. Herewith, the comparison of the functions in *Dom**-SP_{min}^{max} can be reduced to comparing them at two distinct points of time. Denote by $\underline{B}_\ell^i$ and $\overline{B}_\ell^i$ the points of time such that $\underline{ld}_\ell^i(t) = \underline{ld}_\ell^i(\underline{B}_\ell^i)$ and $\overline{ed}_\ell^i(t') = \overline{ed}_\ell^i(\overline{B}_\ell^i)$ holds for all $t \geq \underline{B}_\ell^i$ and $t' \geq \overline{B}_\ell^i$, respectively. Then, the following dominance rule for SP_{min}^{max} results:

Proposition 4.5. *(Dom-*SP_{min}^{max}*) A feasible label* ℓ_1 *dominates a label* ℓ_2 *if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \tag{4.33}$$

$$\overline{ed}_{\ell_1}^i(t_{\ell_1}) \leq ed_{\ell_2}^i \text{ and } \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) - \max\{0, \overline{B}_{\ell_1}^i - \tilde{b}_{\ell_2}\} \leq ed_{\ell_2}^i \quad \forall i \in O_{\ell_1}, \text{ and} \tag{4.34}$$

$$\underline{ld}_{\ell_1}^i(t_{\ell_1}) + (t_{\ell_2} - t_{\ell_1}) \geq ld_{\ell_2}^i(t_{\ell_2}) \text{ and } \underline{ld}_{\ell_2}^i(\underline{B}_{\ell_1}^i) \geq ld_{\ell_2}^i(B_{\ell_2}^i) \quad \forall i \in O_{\ell_1}. \tag{4.35}$$

Note that for determining valid bounds $\overline{ed}_\ell^{h_i}$ and $\underline{ld}_\ell^{h_i}$ on the earliest and latest delivery times of request h_i that do not restrict the pickup times of other open requests $h_j \in O_\ell \setminus \{h_i\}$, it is generally not necessary to consider the maximum and minimum values $\overline{\tau}_j$ and $\underline{\tau}_j$ for the starts of service at the nodes h_j. Instead, it is sufficient to ensure that all h_j can be delivered at their earliest or latest possible delivery times $ed_\ell^{h_j}$ and $ld_\ell^{h_j}$. These times induce starting the service not later than $ed_\ell^{h_j} - \underline{L}_{h_j} \geq \underline{\tau}_j$ and not earlier than $ld_\ell^{h_j} - \underline{L}_{h_j} \leq \overline{\tau}_j$. Hence, bounds that are stronger than $\overline{ed}_\ell^i$ and $\underline{ld}_\ell^i$ can be obtained enabling more dominance when used in *Dom-*SP_{min}^{max}. For simplicity of notation and exposition this has been disregarded in the derivation of *Dom-*SP_{min}^{max}. All proofs, however, are analog.

Dom-SP_{min}^{max} can further be strengthened by using a concept proposed in Chapter 3 for *Dom-*SP^{max}. Let ℓ be the parent label of ℓ'. The information ed_ℓ^i and ld_ℓ^i on the feasible delivery times of open requests $i \in O_\ell$ can be used to determine an upper bound on the

start of service at node $\eta_{\ell'}$ for which $\mathcal{P}_{\ell'}$ can be completed to a feasible $0-(2n+1)$-path. This bound, which is generally smaller than $\tilde{b}_{\ell'}$, strengthens the dominance relation in $Dom\text{-}SP^{max}_{min}$ (see Section 3.4.6 for details).

Labeling Algorithm for SP^{max}_{min} The basic course of our labeling algorithm with $Dom\text{-}SP^{max}_{min}$ for solving SP^{max}_{min} is identical to those in Sections 4.3.1-4.3.3. When creating a new label ℓ', the resources $\eta_{\ell'}$, $\tilde{c}_{\ell'}$, $l_{\ell'}$, and $O_{\ell'}$ are updated using the REFs (4.13) and (4.14). The earliest start of service $t_{\ell'}$ is set according to the adapted REF (4.25).

Determining the values of the resources related to feasible delivery times of open requests $i \in O_{\ell'}$ is intricate. Because of the simultaneous handling of minimum and maximum ride-time constraints, the information on the earliest and latest delivery is interdependent. As a consequence, the determination of these values is much more complex than in the isolated cases in SP_{min} and SP^{max}.

The key problems are the following: When creating a new label ℓ', the implied scheduling problem has additional constraints compared to the scheduling problem implied by the parent label ℓ. These constraints impose bounds on the start of service at the current node $\eta_{\ell'}$ that may restrict other service times within the schedule. The impact on these service times may further propagate throughout the constraint system (see proofs of Proposition 4.4 and Lemma 4.2) so that their effect on the earliest and latest pickup times at the open requests $i \in O_{\ell'}$ is non-trivial to identify. Moreover, if the extended label ℓ' ends at a delivery node $\eta_{\ell'} \in D$, then the corresponding request $\eta_{\ell}-n$ is no longer open. For all open requests $h_i \in O_{\ell'}$, this reduces the set of requests whose latest and earliest service times have to be taken into account when determining the bounds $\underline{\tau}_i^{\overline{O}_{\ell'}}$ and $\overline{\tau}_i^{\underline{O}_{\ell'}}$, respectively. Thus, the relation between the resource values $\overline{ed}^i_{\ell'}(t_{\ell'})$, $\overline{ed}^i_{\ell'}(\overline{B}^i_{\ell'})$, $\underline{ld}^i_{\ell'}(t_{\ell'})$, and $\underline{ld}^i_{\ell'}(\underline{B}^i_{\ell'})$ with $i \in O_{\ell'}$ and the corresponding values $\overline{ed}^i_{\ell}(t_{\ell})$, $\overline{ed}^i_{\ell}(\overline{B}^i_{\ell})$, $\underline{ld}^i_{\ell}(t_{\ell})$, and $\underline{ld}^i_{\ell}(\underline{B}^i_{\ell})$ of the parent label ℓ is highly complex.

As a result, we were not able to derive simple update formulas for the resources related to feasible delivery times of open requests. We suspect that if there are REFs for these resources carrying along several auxiliary resources needed for the calculations is necessary. It seems also mandatory for the computation of these resources to know the actual node sequence $\mathcal{P}_{\ell'}$ represented by label ℓ'.

In our algorithm, the earliest and latest delivery times are computed from scratch within each label. To do so, we use a generalization of the procedure of Tang *et al.* (2010) to obtain a feasible schedule with early-as-possible service times for all nodes which provides the values $ed^i_{\ell'}, i \in O_{\ell'}$. Starting from this schedule, we repeatedly delay the service at distinct nodes to obtain the remaining delivery times. This is done using an adapted version of the forward time slack originally introduced by Savelsbergh (1992) for the TSPTW and later generalized by Cordeau and Laporte (2003) for the DARP. Note that it is not necessary to consider the complete path $\mathcal{P}_{\ell'}$ for these computations. Instead, it is sufficient to take into account the subpath between the node at which the vehicle was empty for the last time and the current node $\eta_{\ell'}$.

The elimination of labels with no feasible completion to node $2n+1$ makes use of both the earliest possible delivery times ed^i_{ℓ} and the latest possible delivery times ld^i_{ℓ}. With

this information, the label elimination strategy is very effective.

4.4 Branch-and-Cut-and-Price Algorithm

This section briefly describes the main components of our basic branch-and-cut-and-price algorithm. Based on this algorithm, we devise four different integer column-generation approaches to the VRPTWTSPD. Each of the approaches formulates the master problem on a different variable set implying subproblem SP, SP_{min}, SP^{max}, or SP^{max}_{min}. The subproblems are solved using the respective labeling algorithms of Sections 4.3.1 - 4.3.4.

Preprocessing Time-window tightening and arc elimination is performed according to the rules proposed by Desrochers *et al.* (1992), Dumas *et al.* (1991), and Cordeau (2006) for the PDPTW or tailored to the PDPTW or the DARP. The integration of minimum and maximum ride-time constraints into these rules is straightforward. Note that all preprocessing steps have to be consistent with the dominance rule used to solve the subproblem. This means that one has to be careful when excluding arcs that cannot be part of a feasible VRPTWTSPD-route and at the same time using a formulation with a subproblem other than SP^{max}_{min}.

Assume, e.g., that it is infeasible in the presence of minimum and maximum ride times to have requests i and k on board of the vehicle at the same time. Thus, arc $(k, i+n)$ could be eliminated during preprocessing. Consider now an approach using SP as subproblem. Let ℓ_1 with $\mathcal{P}_{\ell_1} = (0, i, k)$ and ℓ_2 with $\mathcal{P}_{\ell_2} = (0, i, j, k)$ be two labels and suppose that ℓ_1 dominates ℓ_2 in the sense of *Dom-SP*. Although visiting node $i+n$ after node k always leads to an infeasible path in the VRPTWTSPD-sense, it may still be feasible for SP. Thus, it might be possible to feasibly complete ℓ_2 to path $\mathcal{P}_2 = (0, i, j, k, j+n, i+n, k+n, 2n+1)$ while the corresponding completed path $\mathcal{P}_1 = (0, i, k, i+n, k+n, 2n+1)$ associated with ℓ_1 cannot be generated because arc $(k, i+n)$ has been eliminated. As a result, dominance rule $Dom-SP$ is not valid anymore in this case.

Consequently, dominance rules other than $Dom-SP$, $Dom-SP^{max}$, and $Dom-SP_{min}$ have to be used when utilizing all information on VRPTWTSPD-feasibility of routes for preprocessing. The alternative is to exclude the information on VRPTWTSPD-feasibility during preprocessing that is also relaxed within the respective subproblem.

Pricing Problem Heuristics To speed up the column-generation process, heuristics can be used to identify negative reduced-cost columns fast. When the heuristics are unable to find additional columns, one has to resort to an exact method to solve the subproblem. In our algorithms, we use two straightforward pricing problem heuristics. The first is to solve a more relaxed subproblem, e.g., solving SP when actually having to solve SP^{max}, and to drop all routes that are infeasible for the actual subproblem. The other is to solve the subproblem on a reduced network only. Preliminary computational tests indicated that the benefits from using these heuristics were rather limited for all algorithms.

A different strategy related to pricing problem heuristics is the following: For the approaches based on the relaxed subproblems SP, SP_{min}, and SP^{max}, we perform full preprocessing, i.e., we include full information on VRPTWTSPD-feasibility during preprocessing, which means that the respective dominance rules $Dom - SP$, $Dom - SP_{min}$, and $Dom - SP^{max}$ are not valid for solving the resulting subproblems. However, algorithms with these dominance criteria can still be used as very effective pricing heuristics. Corresponding exact methods are obtained by using the weaker dominance rules in which the condition $O_{\ell_1} \subseteq O_{\ell_2}$ is replaced by $O_{\ell_1} = O_{\ell_2}$. In preliminary computational tests, this strategy has proven to be significantly superior to the one applying only preprocessing steps that are consistent with the stronger dominance rules $Dom - SP$, $Dom - SP_{min}$, and $Dom - SP^{max}$. Therefore, we use this strategy in our algorithms.

Cutting Planes In our branch-and-cut-and-price algorithms, we use the following types of valid inequalities: 2-path inequalities (Kohl *et al.*, 1999), rounded capacity inequalities in a form proposed by Ropke and Cordeau (2009) for the PDPTW, fork inequalities (Ropke *et al.*, 2007), and two different liftings of IPEC introduced by Ascheuer *et al.* (2000) for the TSPTW and Cordeau (2006) for the DARP. Heuristic separation procedures proposed by Ropke and Cordeau (2009) are used to separate 2-path inequalities, rounded capacity inequalities, and fork inequalities. For the exact separation of the lifted IPEC we use a straightforward enumeration procedure (see Ascheuer *et al.*, 2000). VRPTWTSPD-feasibility of an integer solution obtained by approaches using a relaxed variable set Ω' is, thus, guaranteed by the lifted IPEC.

Branching Strategy and Node Selection A hierarchical branching scheme is used to obtain integer solutions in our algorithms. We first branch on the number of vehicles, if fractional. We then branch on the outflow of a node set of cardinality two. Both branching rules are enforced by adding a single linear constraint to the master problem. The structure of the subproblems remains unchanged.

The branch-and-bound tree is explored with a best-first strategy and no upper bounds are given to the algorithm.

4.5 Computational Results

This section summarizes the computational experiments that we have conducted to compare the performance of the four different branch-and-cut-and-price approaches to the VRPTWTSPD. All algorithms described in this paper were implemented in C++ using CPLEX 12.2 as LP-solver. Arc costs and travel times are computed with double precision. The experiments were performed on a standard PC with an Intel(R) Core(TM)2 Duo E8400 at 3.0 GHz with 4.0 GB main memory using a single thread only. The time limit was set to one hour.

The test instances used in the computational study are all based on the benchmark set for the DARP originally introduced by Cordeau (2006) and later extended by instances

with larger problem sizes by Ropke *et al.* (2007). For a detailed description of these instances and their generation we refer to (Cordeau, 2006).

The DARP benchmark set consists of random Euclidean instances with problem sizes reaching from two vehicles and 16 customer requests to eight vehicles and 96 customer requests. All instances are characterized by small vehicle capacities and narrow time windows. To also consider harder instances in which these constraints are less restrictive, three new instances were constructed from each original instance by enlarging both capacities and time-window lengths by factors of 4/3, 5/3, and 6/3.

In the original benchmark set, there are two subsets of instances (type `a` and type `b`) with different characteristics regarding customer demand and vehicle capacity. Moreover, the maximum ride times are specified by a fixed number for each subset and are identical for all requests and all instances of the given subset. With fixed ride times, however, either the minimum ride-time constraints for several requests are trivially satisfied as they are smaller than the direct travel time between the respective pickup and delivery locations. Or with a minimum ride time that is larger than all those direct travel times, the instance is likely to become infeasible. We, therefore, chose not to use fixed ride times for our test instances. Instead, we modeled both ride times proportional to the direct travel time between pickup and delivery node of a request so that none of the ride time constraints is redundant a priori. More precisely, the maximum ride time of a request is equal to the product of the direct travel time and a random number chosen according to a uniform distribution over a given interval. To generate instances with different characteristics regarding the tightness of the ride-time constraints, we considered the two intervals $[2.25, 2.75]$ (for more restrictive maximum ride times) and $[2.75, 3.25]$ (for less restrictive maximum ride times). The minimum ride times were specified in a similar fashion using the intervals $[1.75, 2.25]$ and $[1.25, 1.75]$ for generating instances with more restrictive and less restrictive minimum ride times, respectively.

The complete benchmark comprises 672 instances labeled in the form `RT-TW-iK-n`, where `n` denotes the number of requests, `K` denotes the number of vehicles, and $\texttt{i} \in \{\texttt{a}, \texttt{b}\}$ denotes the subset the instance originates from. Moreover, $\texttt{TW} = \texttt{A}$ refers to the original instances with small vehicle capacities and time-window lengths, while $\texttt{TW} = \texttt{B}$, $\texttt{TW} = \texttt{C}$ and $\texttt{TW} = \texttt{D}$ denote the instances in which these values have been enlarged by a factor of 4/3, 5/3, and 6/3, respectively. The characteristics regarding ride times are specified by $\texttt{RT} \in \{\texttt{MM}, \texttt{ML}, \texttt{LM}, \texttt{LL}\}$, where `M` and `L` indicate the more restrictive and less restrictive cases, respectively, while the first character refers to minimum ride times and the second character refers to maximum ride times. Note that some of the small instances are infeasible in the presence of minimum and maximum ride-time constraints. We allowed the use of additional vehicles to obtain well-defined instances in these cases. All instances are available at `http://logistik.bwl.uni-mainz.de/Dateien/vrptwtspd.zip`.

Table 4.3 summarizes our results averaged over all benchmark instances. Tables 4.4 and 4.5 present averaged results for the subclasses `A`, `B`, `C`, `D` and `MM`, `LM`, `ML`, `MM`, respectively. More detailed results can be found in Table 4.7 in Section 4.B of the Appendix. The columns of the tables report the following:

opt^{tree} The number of optimal solutions obtained by respective algorithm

	opt^{tree}	$time$	opt^{root}	gap^{root}	$cuts$	$nodes$
IMP^{max}_{min}	653	204	345	0.12	8	19
$IMP\text{-}I^{max}$	652	194	298	0.17	42	43
$IMP\text{-}I_{min}$	552	781	296	0.29	193	35
$IMP\text{-}I$	540	852	269	0.33	214	35

Table 4.3: Summary results aggregated over all 672 instances

$time$	The average computation time in seconds
opt^{root}	The number of instances solved to optimality in the root node
gap^{root}	The average percentage integrality gap in the root node
$cuts$	The average number of cuts generated
$nodes$	The average number of nodes solved

The results in Table 4.3 indicate that algorithms IMP^{max}_{min} and $IMP\text{-}I^{max}$ are clearly superior to $IMP\text{-}I_{min}$ and $IMP\text{-}I$. The overall performance of the two stronger approaches IMP^{max}_{min} and $IMP\text{-}I^{max}$ is comparable. In total, IMP^{max}_{min} is able to solve 653 out of the 672 instances to optimality, one more than $IMP\text{-}I^{max}$. Regarding computation times, $IMP\text{-}I^{max}$ is on average slightly faster than IMP^{max}_{min}. Algorithm $IMP\text{-}I_{min}$ is inferior to both of the former approaches regarding both computation times and number of solved instances. $IMP\text{-}I$ performs even worse on both numbers.

The superiority of IMP^{max}_{min} and $IMP\text{-}I^{max}$ over $IMP\text{-}I_{min}$ and $IMP\text{-}I$ can be attributed to the following reasons: First, the root node lower bounds of IMP^{max}_{min} and $IMP\text{-}I^{max}$ are significantly stronger resulting in smaller search trees for these approaches. Second, for $IMP\text{-}I_{min}$ and $IMP\text{-}I$ substantially more cuts are added to the master programs severely complicating their reoptimization. This is also the reason why the average number of solved nodes is smaller for approaches $IMP\text{-}I_{min}$ and $IMP\text{-}I$ compared to approach $IMP\text{-}I^{max}$. While $IMP\text{-}I^{max}$ explores a huge number of nodes when solving difficult instances, algorithms $IMP\text{-}I_{min}$ and $IMP\text{-}I$ spend a lot of time reoptimizing the master programs and, thus, can solve only few nodes within the time limit. When comparing instances solved by all approaches, the number of explored nodes is indeed much higher for approaches $IMP\text{-}I_{min}$ and $IMP\text{-}I$.

The more disaggregated results in Tables 4.4 and 4.5 indicate that all findings from the overall results regarding the performance of the different approaches do also hold for all subclasses of instances. This means that the characteristics of the ride-time constraints have only limited influence on the relation of the strengths of the considered algorithms. This is also true for vehicle capacity, customer demands, and time-window lengths.

Another interesting result of our experiments is that handling the maximum ride-time constraints in the subproblem seems to be more important than integrating the minimum ride times into the subproblem. Our interpretation is that the minimum ride-time constraints are often satisfied without explicitly considering them for the following reasons: When the time windows are narrow, many customer requests are picked up at their origin node i at time a_i. In these cases, the time-window tightening rules ensure

		opt^{tree}	*time*	opt^{root}	gap^{root}	*cuts*	*nodes*
A (orig.)	IMP^{max}_{min}	166	122	111	0.06	5	13
	IMP-I^{max}	165	109	102	0.08	23	23
	IMP-I_{min}	163	168	107	0.09	70	14
	IMP-I	162	192	99	0.10	82	15
B (4/3)	IMP^{max}_{min}	166	109	92	0.08	6	12
	IMP-I^{max}	165	122	81	0.10	32	26
	IMP-I_{min}	154	401	77	0.15	137	23
	IMP-I	152	451	70	0.17	157	28
C (5/3)	IMP^{max}_{min}	164	193	83	0.13	8	21
	IMP-I^{max}	163	163	65	0.18	47	38
	IMP-I_{min}	135	903	63	0.31	224	41
	IMP-I	131	1034	54	0.35	248	46
D (6/3)	IMP^{max}_{min}	157	391	59	0.20	12	35
	IMP-I^{max}	159	381	50	0.30	68	93
	IMP-I_{min}	100	1653	49	0.61	342	58
	IMP-I	95	1730	46	0.68	371	57

Table 4.4: Aggregated results for subclasses A, B, C, and D

		opt^{tree}	*time*	opt^{root}	gap^{root}	*cuts*	*nodes*
MM	IMP^{max}_{min}	167	67	103	0.08	7	11
	IMP-I^{max}	167	70	85	0.11	46	21
	IMP-I_{min}	148	584	91	0.15	185	18
	IMP-I	145	644	80	0.19	208	18
LM	IMP^{max}_{min}	166	127	92	0.08	9	19
	IMP-I^{max}	168	79	85	0.11	31	27
	IMP-I_{min}	143	688	79	0.22	207	18
	IMP-I	139	748	77	0.25	223	18
ML	IMP^{max}_{min}	161	249	72	0.15	7	19
	IMP-I^{max}	156	339	61	0.25	57	70
	IMP-I_{min}	131	913	61	0.38	192	49
	IMP-I	128	1012	53	0.45	221	49
LL	IMP^{max}_{min}	159	371	78	0.16	9	27
	IMP-I^{max}	161	287	67	0.19	35	55
	IMP-I_{min}	130	939	65	0.40	189	56
	IMP-I	128	1003	59	0.42	206	56

Table 4.5: Aggregated results for subclasses MM, LM, ML, and LL

that the minimum ride times are respected. With wide time windows, on the other hand, several other customer nodes are often visited in between the pickup and delivery of a request. This increases the ride times of the respective request compared to the direct travel times so that the minimum ride-time constraints might already be satisfied.

Table 4.6 reports detailed results for the stronger approaches IMP^{max}_{min} and $IMP\text{-}I^{max}$ on all 24 instances that are solved to optimality by at most one of the algorithms. Optimal solution values for all instances together with the computation times of the approaches IMP^{max}_{min} and $IMP\text{-}I^{max}$ can be found in Tables 4.8-4.11 in Section 4.B of the Appendix. In Table 4.6, the following notation is used:

inst	The name of the instance
$opt/\overline{ub}$	The value of the optimal solution or best known upper bound (overlined)
lb^{tree}	The lower bound provided by the respective algorithm within the time limit; * if instance is solved to optimality within the time limit
gap^{tree}	The percentage integrality gap when reaching the time limit
lb^{root}	The root node lower bound
gap^{root}	The percentage integrality gap in the root node
time	The computation time in seconds; *1h* if unable to solve instance within time limit
cuts	The number of cuts generated
nodes	The number of nodes solved

In total, 15 instances were solved by neither of the algorithms within the time limit. We were able to close the integrality gap for ten of these instances using different algorithmic settings and/or allowing more computation time. For the remaining five instances, only an upper bound is reported.

4.6 Conclusions

In this paper, we introduced the Vehicle Routing Problem with Time Windows and Temporal Synchronized Pickup and Delivery (VRPTWTSPD) as the prototypical VRP with temporal intra-route synchronization. In the VRPTWTSPD, vehicle routes have to satisfy pairing and precedence, capacities, and time windows. Additionally, temporal synchronization constraints couple the service times at the pickup and delivery locations of the customer requests in the following way: A delivery node has to be serviced within prespecified minimum and maximum time lags (called ride times) after the service at the corresponding pickup node has been completed.

The minimum and maximum ride-time constraints severely complicate the subproblem of the natural column-generation formulation of the VRPTWTSPD and it is not clear if their integration into the subproblem pays off in an integer column-generation approach. We, therefore, developed four solution approaches to the VRPTWTSPD based on column-generation formulations with differing subproblems. Two of these subproblems, the natural subproblem of the VRPTWTSPD that integrates all constraints

		IMP^{max}_{min}							$IMP\text{-}I^{max}$						
inst	$opt/\overline{ub}$	lb^{tree}	gap^{tree}	lb^{root}	gap^{root}	*time*	*cuts*	*nodes*	lb^{tree}	gap^{tree}	lb^{root}	gap^{root}	*time*	*cuts*	*nodes*
ML-A-a6-72	908.54	*	0.00	904.84	0.41	1837	10	264	906.50	0.23	900.95	0.84	*1h*	191	681
ML-A-a8-96	1149.95	1149.26	0.06	1146.05	0.34	*1h*	32	214	1149.67	0.02	1145.16	0.42	*1h*	152	500
LL-A-a7-84	981.31	979.87	0.15	974.14	0.74	*1h*	15	348	980.70	0.06	972.07	0.95	*1h*	167	634
ML-B-a8-96	1114.48	*	0.00	1112.50	0.18	938	18	27	1113.08	0.13	1108.86	0.51	*1h*	186	345
LL-B-a7-84	952.52	951.16	0.14	946.37	0.65	*1h*	47	264	951.45	0.11	945.13	0.78	*1h*	178	563
LL-B-b8-96	$\overline{1107.70}$	1106.38	0.12	1102.17	0.50	*1h*	14	288	1106.17	0.14	1101.27	0.58	*1h*	72	635
MM-C-b8-80	943.70	943.50	0.02	939.65	0.43	*1h*	19	522	942.99	0.08	938.64	0.54	*1h*	204	626
ML-C-a8-96	$\overline{1101.72}$	1095.80	0.54	1093.28	0.77	*1h*	25	70	1093.84	0.72	1089.58	1.11	*1h*	223	316
ML-C-b8-80	925.53	*	0.00	921.44	0.44	1000	14	134	925.50	0.00	920.56	0.54	*1h*	187	883
ML-C-b8-96	1047.67	1046.33	0.13	1043.78	0.37	*1h*	1	77	1046.61	0.10	1042.98	0.45	*1h*	163	362
LL-C-a8-96	1087.91	1085.35	0.24	1083.48	0.41	*1h*	11	42	1087.46	0.04	1083.14	0.44	*1h*	134	291
LM-D-a7-84	953.92	952.88	0.11	948.74	0.55	*1h*	54	292	*	0.00	947.89	0.64	2681	162	476
LM-D-b8-64	695.72	695.36	0.05	689.27	0.94	*1h*	20	842	*	0.00	688.65	1.03	3406	197	1365
ML-D-a6-72	853.62	852.59	0.12	847.08	0.77	*1h*	33	222	852.57	0.12	845.67	0.94	*1h*	208	573
ML-D-a7-84	$\overline{927.15}$	925.07	0.22	921.71	0.59	*1h*	34	78	923.07	0.44	917.60	1.04	*1h*	220	424
ML-D-a8-96	$\overline{1070.58}$	1068.11	0.23	1066.54	0.38	*1h*	9	17	1068.11	0.23	1064.06	0.61	*1h*	118	283
ML-D-b7-84	1016.84	1015.89	0.09	1009.86	0.69	*1h*	12	101	1014.04	0.28	1005.67	1.11	*1h*	211	403
ML-D-b8-80	869.68	*	0.00	866.71	0.34	1690	40	114	868.50	0.14	862.14	0.88	*1h*	287	680
ML-D-b8-96	1021.89	*	0.00	1021.08	0.08	486	3	10	1020.47	0.14	1015.45	0.63	*1h*	194	303
LL-D-a7-84	909.82	907.67	0.24	904.34	0.61	*1h*	47	81	907.81	0.22	902.27	0.84	*1h*	134	369
LL-D-a8-96	1057.65	1056.94	0.07	1055.32	0.22	*1h*	12	11	1057.65	0.00	1054.36	0.31	*1h*	59	155
LL-D-b6-60	726.76	725.11	0.23	719.90	0.95	*1h*	4	523	*	0.00	719.24	1.05	2497	133	893
LL-D-b7-84	1012.45	1011.80	0.06	1008.15	0.43	*1h*	23	132	*	0.00	1007.48	0.49	2776	144	292
LL-D-b8-80	$\overline{861.25}$	858.88	0.28	854.37	0.81	*1h*	20	148	857.08	0.49	849.67	1.36	*1h*	227	710
∅			0.13		0.52		22	201		0.15		0.75		173	532

Table 4.6: Results for IMP^{max}_{min} and $IMP\text{-}I^{max}$ on instances solved by at most one algorithm

relating to single routes and the one in which the maximum ride-time constraints are relaxed, were considered for the first time in this paper. New dominance rules and labeling algorithms for their solution have been derived. Extensive computational experiments demonstrate the applicability of these labeling algorithms in the sense that they are capable of solving subproblems arising in state-of-the-art benchmark instances in reasonable time.

The computational results also indicate a clear ranking of the four presented algorithms for solving the VRPTWTSPD. The strongest approaches are the approach based on the natural column-generation formulation and the one that handles only the maximum ride times in the subproblem. They performed comparably well and were consistently significantly stronger than the approach with only the minimum ride times in the subproblem. This was in turn slightly stronger than the approach in which both ride-time constraints are relaxed in the subproblem. We conclude that the integration of temporal intra-route synchronization constraints into the column-generation subproblem is beneficial for the VRPTWTSPD and that it is particularly rewarding for the maximum ride-times.

References

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, **36**(2), 69–79.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, **202**(3), 756–763.

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

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, **15**(1), 1–31.

Borndörfer, R., Klostermeier, F., Grötschel, M., and Küttner, C. (1997). Telebus Berlin: vehicle scheduling in a dial-a-ride system. Technical report SC 97-23, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany.

Bredström, D. and Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, **191**, 19–29.

Ceselli, A., Righini, G., and Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, **43**(1), 56–69.

Christiansen, M. and Nygreen, B. (1998). Modelling path flows for a combined ship routing and inventory management problem. *Annals of Operations Research*, **82**, 391–413.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, **54**(3), 573–586.

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B*, **37**(6), 579–594.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, **153**(1), 29–46.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, Boston, MA.

Desaulniers, G., Desroisiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constraint vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer, Boston, MA.

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.

Dohn, A., Rasmussen, M. S., and Larsen, J. (2011). The vehicle routing problem with time windows and temporal dependencies. *Networks*, **58**(4), 273–289.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, **46**(3), 297–316.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**(1), 7–22.

Eveborn, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care – an operational system for staff planning of home care. *European Journal of Operational Research*, **171**(3), 962–976.

Hunsaker, B. and Savelsbergh, M. (2002). Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters*, **30**(3), 169–173.

Ioachim, I., Desrosiers, J., Soumis, F., and Bélanger, N. (1999). Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, **119**(1), 75–90.

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

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.

Madsen, O., Ravn, H., and Rygaard, J. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*, **60**, 193–208.

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

Plum, C. E., Pisinger, D., Salazar-González, J.-J., and Sigurd, M. M. (2014). Single liner shipping service design. *Computers & Operations Research*, **45**, 1–6.

Ropke, S. and Cordeau, J.-F. (2005). Branch and cut and price for the pickup and delivery problem with time windows. Chapter 9 of S. Ropke's cumulative Ph.D. dissertation, Heuristic and exact algorithms for vehicle routing problems, University of Copenhagen, Copenhagen, Denmark.

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.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, **49**(4), 258–272.

Russell, R. A. and Morrel, R. B. (1986). Routing special-education school buses. *Interfaces*, **16**(5), 56–64.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, **4**(2), 146–154.

Tang, J., Kong, Y., Lau, H., and Ip, A. W. (2010). A note on "efficient feasibility testing for dial-a-ride problems". *Operations Research Letters*, **38**(5), 405–407.

Toth, P. and Vigo, D. (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, **31**(1), 60–71.

Appendix

4.A Proofs

For the presentation of the proofs some additional notation is necessary. For any set of numbers M and any number n, let $n + M = \{n + m : m \in M\}$. Moreover, given a sequence $\mathcal{P} = (h_1, ..., h_q)$ (path or schedule) and a set of numbers M, let $\mathcal{P} \setminus M$ be the sub-sequence of $\mathcal{P}$ where h_i is removed if $h_i = m$ for all $m \in M$.

We first sketch the proof of Proposition 4.1 (see Dumas *et al.*, 1991) which is the basis for Propositions 4.3 and 4.4.

Proof of Proposition 4.1: The basic idea of the proof is as follows. Let Q_2 be a feasible completion to ℓ_2, i.e., Q_2 is a extension of $\mathcal{P}_{\ell_2}$ to node $2n+1$ such that the path $\mathcal{P}_2 = (\mathcal{P}_{\ell_2}, Q_2)$ is feasible. Consider the path $\mathcal{P}_1 = (\mathcal{P}_{\ell_1}, Q_1)$ where $Q_1 = Q_2 \setminus \{n+(O_{\ell_2} \setminus O_{\ell_1})\}$ is the completion to ℓ_1 resulting from Q_2 by skipping the delivery nodes of all additional open requests of ℓ_2. Clearly, pairing and precedence constraints are then satisfied by $\mathcal{P}_1$. Using that $\mathcal{P}_2$ is feasible, the relations in Proposition 4.1 also ensure that $\mathcal{P}_1$ is feasible with respect to capacity and time windows. Furthermore, when the DTI holds the cost of $\mathcal{P}_1$ is always not higher than that of $\mathcal{P}_2$. □

Proof of Lemma 4.1: $\underline{T}_{\mathcal{P}}$ is feasible if it satisfies inequalities (4.1)–(4.3). By definition of $\underline{\tau}_i$, there exists for each $i = 1, ..., q$ a feasible schedule $T^i_{\mathcal{P}} = (\tau^i_1, ..., \underline{\tau}_i, ..., \tau^i_q)$ with $\tau^i_j \geq \underline{\tau}_j$ for all $j \neq i$. It follows immediately that $\underline{T}_{\mathcal{P}}$ satisfies (4.2) for all $i = 1, ..., q$. Using $\tau^{i+1}_i + t_{h_i h_{i+1}} \leq \underline{\tau}_{i+1}$ and $\underline{\tau}_i \leq \tau^{i+1}_i$ it follows that $\underline{T}_{\mathcal{P}}$ satisfies (4.1) for all $i = 1, ..., q-1$. Also, using $\tau^j_i + \underline{L}_{h_i} \leq \underline{\tau}_j$ and $\underline{\tau}_i \leq \tau^j_i$ it follows that $\underline{T}_{\mathcal{P}}$ satisfies (4.3) for all $h_i = h_j - n$. □

Proof of Proposition 4.3: The proof is similar to the proof of Proposition 4.1. Following the same argumentation and using the same notation, it remains to show that $\mathcal{P}_1$ also respects minimum ride-time constraints, i.e., we have to show that there exists a feasible time schedule $T_{\mathcal{P}_1}$ for $\mathcal{P}_1$.

Let $T_{\mathcal{P}_2} = (T_{\mathcal{P}_{\ell_2}}, T_{Q_2})$ be a feasible schedule for $\mathcal{P}_2$ with $T_{Q_2} = (\tau_{q+1}, ..., \tau_r)$. Denote by $\underline{T}_{\mathcal{P}_{\ell_1}} = (\underline{\tau}_1, ..., \underline{\tau}_q) \in \mathscr{T}_{\mathcal{P}_{\ell_1}}$ the time schedule for $\mathcal{P}_{\ell_1}$ that minimizes the start of service at all nodes and by $T_{Q_1} = T_{Q_2} \setminus \{\tau_i : h_i - n \in O_{\ell_2} \setminus O_{\ell_1}\}$ the schedule for Q_1 that assigns each node h_i of Q_1 the same start of service τ_i as in T_{Q_2}. Then, using that $\underline{T}_{\mathcal{P}_{\ell_1}}$ and $T_{\mathcal{P}_2}$ are feasible, $\underline{\tau}_q = t_{\ell_1} \leq t_{\ell_2}$, and $ed^{h_i}_{\ell_1} \leq ed^{h_i}_{\ell_2} \leq \tau_i$ for all nodes h_i of Q_1 with $h_i - n \in O_{\ell_1}$ it follows that the schedule $T_{\mathcal{P}_1} = (\underline{T}_{\mathcal{P}_{\ell_1}}, T_{Q_1})$ is feasible. □

Proof of Proposition 4.4: The basic course of the proof is similar to that in the proof of Proposition 4.3. With the same argumentation and notation, it again remains to show that there exists a feasible time schedule $T_{\mathcal{P}_1}$ for $\mathcal{P}_1$.

Let $\tau^{\ell_2}_q$ be the start of service at the current node $h_q = \eta_{\ell_1}$ within the schedule $T_{\mathcal{P}_2}$. Denote by $\overline{\tau}'_i(\tau^{\ell_2}_q) = \max_{T_{\mathcal{P}_{\ell_1}} \in \mathscr{T}_{\mathcal{P}_{\ell_1}}(\tau^{\ell_2}_q), \tau_j = \tau'_j \forall h_j \in O_{\ell_1} \setminus \{h_i\}} \{\tau_i\}$ the latest feasible start of service at node $h_i \in O_{\ell_1}$ given a feasible choice τ'_j for the values $\tau_j, h_j \in O_{\ell_1} \setminus \{h_i\}$, i.e., a

choice such that a feasible schedule $T_{\mathcal{P}_{\ell_1}} = (\tau_1, ..., \tau_q) \in \mathscr{T}_{\mathcal{P}_{\ell_1}}(\tau_q^{\ell_2})$ with $\tau_j = \tau'_j$ for all $h_j \in O_{\ell_1} \setminus \{h_i\}$ exists. We first show that $\overline{\tau}'_i(\tau_q^{\ell_2}) \geq \overline{\tau}_i^{O_{\ell_1}}(\tau_q^{\ell_2})$ holds.

The proof is constructive. Denote by $\overline{\tau}'_k(\tau_q^{\ell_2})$ and $\overline{\tau}_k^{O_{\ell_1}}(\tau_q^{\ell_2})$ also the latest start of service at all other nodes $h_k \neq h_i$ subject to the choices τ'_j and $\underline{\tau}_j$ for all $h_j \in O_{\ell_1} \setminus \{h_i\}$, respectively. Moreover for ease of notation, the explicit functional dependence on the start of service at the current node is omitted in the following, as all considered values relate to schedules with $\tau_q \leq \tau_q^{\ell_2}$. We start from the schedule $\overline{T}'_{\mathcal{P}_{\ell_1}} = (\overline{\tau}'_1, ..., \overline{\tau}'_q) \in \mathscr{T}_{\mathcal{P}_{\ell_1}}$. Feasibility of $\overline{T}'_{\mathcal{P}_{\ell_1}}$ follows analog to the proof of Lemma 4.1. Thus, $\overline{T}'_{\mathcal{P}_{\ell_1}}$ satisfies the constraint system (4.1)–(4.4), $\tau_q \leq \tau_q^{\ell_2}$, and $\tau_j = \tau'_j, h_j \in O_{\ell_1} \setminus \{h_i\}$ of the maximization problems of $\overline{\tau}'_k, k = 1, ..., q$. Replacing equalities $\tau_j = \tau'_j$ by $\tau_j = \underline{\tau}_j$ for all $h_j \in O_{\ell_1} \setminus \{h_i\}$ leads to the constraints system for the maximization problems of $\overline{\tau}_i^{O_{\ell_1}}, k = 1, ..., q$. The modified equalities directly impose stronger bounds on all values τ_{j-1} because of inequalities (4.1). The case $h_{j-1} \in O_{\ell_1} \setminus \{h_i\}$ can be neglected, as the value of τ_{j-1} is then fixed to $\underline{\tau}_{j-1}$ and consistency of $\underline{\tau}_{j-1}$ and $\underline{\tau}_j$ follows directly from feasibility of the choice $\underline{\tau}_k, h_k \in O_{\ell_1}$ which can again be shown analog to the proof of Lemma 4.1.

In the case $h_{j-1} \notin O_{\ell_1} \setminus \{h_i\}$, the maximal value at position $j-1$ decreases from $\overline{\tau}'_{j-1}$ to $\overline{\tau}_{j-1}^{O_{\ell_1}} = \min\{\overline{\tau}'_{j-1}, \underline{\tau}_j - t_{h_{j-1}h_j}\} = \overline{\tau}'_{j-1} + \Delta_{j-1}$ with $\Delta_{j-1} \leq 0$. A reduced value $\overline{\tau}_{j-1}^{O_{\ell_1}}$ compared to $\overline{\tau}'_{j-1}$ may in turn influence the maximal values at positions $j-2$, k with $h_k = h_{j-1} - n$, and l with $h_l = h_{j-1} + n$ because of inequalities (4.1), (4.3), and (4.4), respectively. W.l.o.g. consider the impact on the maximal value at position $j-2$ which decreases from $\overline{\tau}'_{j-2}$ to

$$\begin{aligned}
\overline{\tau}_{j-2}^{O_{\ell_1}} &= \min\{\overline{\tau}'_{j-2}, \overline{\tau}_{j-1}^{O_{\ell_1}} - t_{h_{j-2}h_{j-1}}\} \\
&= \overline{\tau}'_{j-2} + \min\{0, \overline{\tau}_{j-1}^{O_{\ell_1}} - t_{h_{j-2}h_{j-1}} - \overline{\tau}'_{j-2}\} \\
&= \overline{\tau}'_{j-2} + \min\{0, \overline{\tau}'_{j-1} + \Delta_{j-1} - t_{h_{j-2}h_{j-1}} - \overline{\tau}'_{j-2}\} \\
&= \overline{\tau}'_{j-2} + \Delta_{j-2},
\end{aligned}$$

with $\Delta_{j-2} \leq 0$. As $\overline{\tau}'_{j-1} - t_{h_{j-2}h_{j-1}} - \overline{\tau}'_{j-2} \geq 0$ it follows that $0 \leq \Delta_{j-2} \leq \Delta_{j-1}$. Thus, the value by which $\overline{\tau}_{j-2}^{O_{\ell_1}}$ is decreased compared to $\overline{\tau}'_{j-2}$ is strictly not larger than the value by which $\overline{\tau}_{j-1}^{O_{\ell_1}}$ is decreased compared to $\overline{\tau}'_{j-1}$. The same result is obtained for the maximal values at positions k with $h_k = h_{j-1} - n$ and l with $h_l = h_{j-1} + n$.

Decreased values $\overline{\tau}_{j-2}^{O_{\ell_1}}$, $\overline{\tau}_k^{O_{\ell_1}}$ with $h_k = h_{j-1} - n$, and $\overline{\tau}_l^{O_{\ell_1}}$ with $h_l = h_{j-1} + n$ in turn constrain the maximal values at several other positions m within the path in an analog fashion as $\overline{\tau}_{j-1}^{O_{\ell_1}}$ constrained themselves as shown above. As a result, we get $\overline{\tau}_m^{O_{\ell_1}} = \overline{\tau}'_m + \Delta_m$ with $\Delta_m \leq 0$. Moreover, $0 \leq \Delta_m \leq \Delta_{j-2}, \Delta_k, \Delta_l$ also holds, i.e., again the decreasing effects on values $\overline{\tau}_m^{O_{\ell_1}}$ compared to $\overline{\tau}'_m$ are strictly not larger than the decreasing effect on $\overline{\tau}_{j-2}^{O_{\ell_1}}$, $\overline{\tau}_k^{O_{\ell_1}}$, and $\overline{\tau}_l^{O_{\ell_1}}$ compared to $\overline{\tau}'_{j-2}$, $\overline{\tau}'_k$, and $\overline{\tau}'_l$, respectively.

Iteratively propagating the stronger conditions resulting from decreased values $\overline{\tau}_k^{O_{\ell_1}}, k = 1, ..., q$ compared to $\overline{\tau}'_k$ in the constraint system allows to construct $\overline{T}^{O_{\ell_1}} = (\overline{\tau}_1^{O_{\ell_1}}, ..., \overline{\tau}_q^{O_{\ell_1}})$

with $\overline{\tau}_k^{O_{\ell_1}} = \overline{\tau}'_k + \Delta_k$ and $\Delta_k \leq 0$ for all $k = 1, ..., q$. If there are more than just one decreasing effects that have not been propagated yet, they are processed in non-increasing absolute values. Using the property that the decreasing effects Δ_k are non-increasing from propagation step to propagation step, there exists a unique value by which the maximal start of service at each position is decreased and no cycle effects can occur. As a result, $\overline{\tau}'_i(\tau_q^{\ell_2}) \geq \overline{\tau}_i^{O_{\ell_1}}(\tau_q^{\ell_2})$ holds for any feasible choice τ'_j of the values $\tau_j, h_j \in O_{\ell_1} \setminus \{h_i\}$.

In an analog fashion, we can show that $\underline{\tau}'_i(\tau_q^{\ell_2}) \leq \underline{\tau}_i^{\overline{O}_{\ell_1}}(\tau_q^{\ell_2})$ holds. Clearly, we also have that $\underline{\tau}'_i(\tau_q^{\ell_2}) \leq \overline{\tau}'_i(\tau_q^{\ell_2})$. Moreover, it is easy to see that any convex combination of two feasible schedules is also a feasible schedule. Thus, given any feasible choice τ'_j for the values $\tau_j, h_j \in O_{\ell_1} \setminus \{h_i\}$ there exists for each $\tau_i^* \in \left[\underline{\tau}'_i(\tau_q^{\ell_2}), \overline{\tau}'_i(\tau_q^{\ell_2})\right]$ with $\underline{\tau}'_i(\tau_q^{\ell_2}) \leq \underline{\tau}_i^{\overline{O}_{\ell_1}}(\tau_q^{\ell_2})$ and $\overline{\tau}'_i(\tau_q^{\ell_2}) \geq \overline{\tau}_i^{O_{\ell_1}}(\tau_q^{\ell_2})$ a feasible schedule with start of service τ_i^* at node $h_i \in O_{\ell_1}$.

Using this property, we can construct a feasible schedule $T_{\mathcal{P}_1} = (T^*_{\mathcal{P}_{\ell_1}}, T_{Q_1})$ as follows. As defined in the proof of Proposition 4.3, fix $T_{Q_1} = T_{Q_2} \setminus \{\tau_i : h_i - n \in O_{\ell_2} \setminus O_{\ell_1}\}$ to the schedule for Q_1 that assigns each node h_i of Q_1 the same start of service τ_i as in T_{Q_2}. Let $T_{\mathcal{P}_{\ell_1}} = (\tau_1, ..., \tau_q) \in \mathscr{T}_{\mathcal{P}_{\ell_1}}(\tau_q^{\ell_2})$. For $T^*_{\mathcal{P}_{\ell_1}} = T_{\mathcal{P}_{\ell_1}}$, schedule $T_{\mathcal{P}_1}$ clearly satisfies inequalities (4.1) and (4.2). Also, it satisfies inequalities (4.3) and (4.4) for each request $h_i \notin O_{\ell_1}$.

If $T_{\mathcal{P}_1}$ also respects inequalities (4.3) and (4.4) for each request $h_i \in O_{\ell_1}$, $T_{\mathcal{P}_1}$ is feasible and the proof is complete. Otherwise, consider any request h_i for which either constraint (4.3) or (4.4) is violated. Denote by τ_k the start of service at the delivery node $h_k = h_i + n$ as fixed in schedule T_{Q_1}. To satisfy inequalities (4.3) and (4.4), $\tau_i \leq \tau_k - \underline{L}_{h_i}$ and $\tau_i \geq \tau_k - \overline{L}_{h_i}$ must hold, respectively, for the start of service τ_i at pickup node h_i within schedule $T^*_{\mathcal{P}_{\ell_1}}$. Let $\underline{\tau}'_i(\tau_q^{\ell_2})$ and $\overline{\tau}'_i(\tau_q^{\ell_2})$ be the minimal and maximal starts of service at node h_i, respectively, given the values $\tau_j, h_j \in O_{\ell_1} \setminus \{h_i\}$ as fixed within schedule $T_{\mathcal{P}_{\ell_1}}$. Then, $\tau_k - \underline{L}_{h_i} \geq ed_{\ell_2}^{h_i} - \underline{L}_{h_i} \geq \overline{ed}_{\ell_1}^{h_i}(\tau_q^{\ell_2}) - \underline{L}_{h_i} \geq \underline{\tau}_i^{\overline{O}_{\ell_1}}(\tau_q^{\ell_2}) \geq \underline{\tau}'_i(\tau_q^{\ell_2})$ and $\tau_k - \overline{L}_{h_i} \leq ld_{\ell_2}^{h_i}(\tau_q^{\ell_2}) - \overline{L}_{h_i} \leq \underline{ld}_{\ell_1}^{h_i}(\tau_q^{\ell_2}) - \overline{L}_{h_i} \leq \overline{\tau}_i^{O_{\ell_1}}(\tau_q^{\ell_2}) \leq \overline{\tau}'_i(\tau_q^{\ell_2})$ hold. Using the property shown above, there exists a feasible schedule $T'_{\mathcal{P}_{\ell_1}} = (\tau'_1, ..., \tau'_q)$ with $\tau'_j = \tau_j$ for all $h_j \in O_{\ell_1} \setminus \{h_i\}$ and any $\tau'_i = \tau_i^* \in \left[\underline{\tau}'_i(\tau_q^{\ell_2}), \overline{\tau}'_i(\tau_q^{\ell_2})\right]$. Thus for $T^*_{\mathcal{P}_{\ell_1}} = T'_{\mathcal{P}_{\ell_1}}$, $T_{\mathcal{P}_1}$ satisfies constraints (4.3) and (4.4) for request h_i. Moreover, all other constraints that have been respected in the case $T^*_{\mathcal{P}_{\ell_1}} = T_{\mathcal{P}_{\ell_1}}$ are still respected.

If for $T_{\mathcal{P}_1}$ inequalities (4.3) and (4.4) are still violated for some requests, we can iteratively repeat the same procedure just described. In each iteration one constraint that was previously violated gets satisfied. Thus, a feasible schedule $T_{\mathcal{P}_1}$ for path $\mathcal{P}_1$ eventually results which completes the proof. □

Proof of Lemma 4.2: Note first that $\mathcal{X}(t) \notin \varnothing$ for all $t \geq t^*$. Also note that $\overline{X} = (\overline{x}_1, ..., \overline{x}_q) \in \mathcal{X}$, $\underline{X} = (\underline{x}_1, ..., \underline{x}_q) \in \mathcal{X}$, $\overline{X}(t) = (\overline{x}_1(t), ..., \overline{x}_q(t)) \in \mathcal{X}(t)$, $\underline{X}(t) = (\underline{x}_1(t), ..., \underline{x}_q(t)) \in \mathcal{X}(t)$, $\overline{X}^{\underline{S}}(t) = (\overline{x}_1^{\underline{S}}(t), ..., \overline{x}_q^{\underline{S}}(t)) \in \mathcal{X}^{\underline{S}}(t)$, and $\underline{X}^{\overline{S}}(t) = (\underline{x}_1^{\overline{S}}(t), ..., \underline{x}_q^{\overline{S}}(t)) \in \mathcal{X}^{\overline{S}}(t)$. The proofs are analog to the proof of Lemma 4.1.

1. The proof is constructive. We start from $\overline{X} \in \mathcal{X}$, i.e., the q-tuple with maximal values for all $\overline{x}_i, i = 1, ..., q$ satisfying (4.30)–(4.32). When considering $\mathcal{X}(t), t \geq t^*$ instead of $\mathcal{X}$, we have the additional constraint $x_q \leq t$. The case $\overline{x}_q \leq t$ is trivial. If $\overline{x}_q > t$ for some $t \geq t^*$, the maximal value at position q decreases to $\overline{x}_q(t) = \min\{\overline{x}_q, t\} = \overline{x}_q + \min\{0, t - \overline{x}_q\} = \overline{x}_q + \Delta_q(t)$ with $\Delta_q(t) = \min\{k_q^1, k_q^2 + t\} \leq 0$. If inequalities (4.31) are satisfied for the decreased value $\overline{x}_q(t)$ and all $\overline{x}_i, i < q$, then clearly $\overline{x}_j(t) = \overline{x}_j = \min\{k_i^1, k_i^2 + t\}$ for all $j = 1, ..., q$ with constants k_i^1 and k_i^2.

   If $\overline{x}_i + c_{iq} > \overline{x}_q(t)$ holds for some $i < q$ and $t \geq t^*$, then $\overline{x}_i(t) = \min\{\overline{x}_i, \overline{x}_q(t) - c_{iq}\}$ which can be rewritten in the forms

$$\begin{aligned}
\overline{x}_i(t) &= \min\{\overline{x}_i, \overline{x}_q(t) - c_{iq}\} \\
&= \min\{k_i^1, k_i^2 + t\} \\
&= \overline{x}_i + \min\{0, \overline{x}_q(t) - c_{iq} - \overline{x}_i\} \\
&= \overline{x}_i + \min\{0, \overline{x}_q - c_{iq} - \overline{x}_i + \Delta_q(t)\} \\
&= \overline{x}_i + \Delta_i(t),
\end{aligned}$$

   where $\Delta_i(t) = \min\{\tilde{k}_i^1, \tilde{k}_i^2 + t\} \leq 0$ with constants $\tilde{k}_i^1$ and $\tilde{k}_i^2$. As $\overline{x}_q - c_{iq} - \overline{x}_i \geq 0$, it follows that $0 \leq \Delta_i(t) \leq \Delta_q(t)$. Thus, the value by which $\overline{x}_i(t)$ is decreased compared to $\overline{x}_i$ is strictly not larger than the value by which $\overline{x}_q(t)$ is decreased compared to $\overline{x}_q$.

   Decreased values $\overline{x}_i(t)$ in turn impose stronger bounds compared to $\overline{x}_i$ on the maximal values at all positions $j < i$ and $k > i$ because of inequalities (4.31) and (4.32), respectively. Thus, the decreasing effects propagate through the constraint system in the same fashion as in the proof of Proposition 4.4. A schedule $\overline{X}(t) = (\overline{x}_1(t), ..., \overline{x}_q(t)) \in \mathcal{X}(t)$ with $\overline{x}_i(t) = \max\{k_j^1, k_j^2 + t\}\}$ for all $i = 1, ..., q, t \geq t^*$ can then be constructed analog to there.

2. It is straightforward to verify that $\underline{X} \in \mathcal{X}(t)$ and $\mathcal{X}(t) \subseteq \mathcal{X}$ for all $t \geq t^*$. Thus, $\underline{x}_i(t) = \underline{x}_i$ for all $i = 1, ..., q, t \geq t^*$.

3. $\overline{x}_i^{\underline{S}}(t)$ is given by $\max\{x_i\}$, s.t. (4.30)–(4.32), $x_q \leq t$, and $x_j \leq \underline{x}_j, j \in \underline{S}$. Obviously, the latter maximization problem is of the same structure as the one for $\overline{x}_i(t)$ and, thus, $\overline{x}_i^{\underline{S}}(t) = \max\{k_i^1, k_i^2 + t\}$ with constants k_i^1 and k_i^2 for all $i = 1, ..., q, t \geq t^*$.

4. Comparing the minimization problems for $\underline{x}_i(t)$ and $\underline{x}_i^{\overline{S}}(t)$, there are additional constraints $x_j \geq \overline{x}_j(t), j \in \overline{S}$ in the latter. When constructing $\underline{X}^{\overline{S}}(t) = (\underline{x}_1^{\overline{S}}(t), ..., \underline{x}_q^{\overline{S}}(t)) \in \mathcal{X}^{\overline{S}}(t)$ starting from $\underline{X}(t) = (\underline{x}_1, ..., \underline{x}_q) \in \mathcal{X}(t)$, these con-

straints may force $\underline{x}_j^{\overline{S}}(t)$ to be increased compared to $\underline{x}_j$ for $j \in \overline{S}$ and we have

$$\begin{aligned}
\underline{x}_j^{\overline{S}}(t) &= \max\{\underline{x}_j, \overline{x}_j(t)\} && (4.36)\\
&= \max\{\underline{x}_j, \min\{k_j^1, k_j^2 + t\}\} && (4.37)\\
&= \underline{x}_j + \max\{0, \min\{k_j^1, k_j^2 + t\} - \underline{x}_j\} && (4.38)\\
&= \underline{x}_j + \Delta_j(t), && (4.39)
\end{aligned}$$

with $\Delta_j(t) = \max\{\tilde{k}_j^1, \min\{\tilde{k}_j^2, \tilde{k}_j^3 + t\}\} \geq 0$. Increased values $\underline{x}_j^{\overline{S}}(t)$ compared to $\underline{x}_j$ might in turn influence other values and the increasing effects propagate throughout the constraint system.

A schedule $\underline{X}^{\overline{S}}(t) = (\underline{x}_1^{\overline{S}}(t), ..., \underline{x}_q^{\overline{S}}(t)) \in \mathcal{X}^{\overline{S}}(t)$ with $\underline{x}_i^{\overline{S}}(t) = \max\{k_j^1, \min\{k_j^2, k_j^3 + t\}\}$ for all $i = 1, ..., q, t \geq t^*$ can then be constructed in the same fashion as $\overline{X}(t)$ was constructed in the proof of the first property.

□

Proof of Proposition 4.5: We need to show that (4.34) implies $\overline{ed}_{\ell_1}^i(t) \leq ed_{\ell_2}^i, t \in [t_{\ell_2}, \tilde{b}_{\ell_2}]$ and that (4.35) implies $\underline{ld}_{\ell_1}^i(t) \geq ld_{\ell_2}^i(t), t \in [t_{\ell_2}, \tilde{b}_{\ell_2}]$. The latter follows directly from Proposition 3.5 of Chapter 3. For the former, recall that $\overline{ed}_{\ell_1}^i(t) = \max\{k_i^1, \min\{k_i^2, k_i^3 + t\}\}$ and note that $\overline{ed}_{\ell_1}^i(t)$ is clearly non-decreasing in t.

Consider first the case $\overline{B}_{\ell_1}^i \leq \tilde{b}_{\ell_2}$. Using that $\overline{ed}_{\ell_1}^i(t)$ is constant for all $t \geq \overline{B}_{\ell_1}^i$, we have that $\overline{ed}_{\ell_1}^i(t) = \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) \leq ed_{\ell_2}^i$ for all $\overline{B}_{\ell_1}^i \leq t \leq \tilde{b}_{\ell_2}$. For all $t_{\ell_2} \leq t \leq \overline{B}_{\ell_1}^i$, clearly $\overline{ed}_{\ell_1}^i(t) \leq \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) \leq ed_{\ell_2}^i$ holds.

Consider now the case $\overline{B}_{\ell_1}^i > \tilde{b}_{\ell_2}$. If $\overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) \leq \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) - (\overline{B}_{\ell_1}^i - \tilde{b}_{\ell_2})$, then $\overline{ed}_{\ell_1}^i(t) \leq \overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) \leq \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) - (\overline{B}_{\ell_1}^i - \tilde{b}_{\ell_2}) \leq ed_{\ell_2}^i$ holds for all $t_{\ell_2} \leq t \leq \tilde{b}_{\ell_2}$. If $\overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) > \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) - (\overline{B}_{\ell_1}^i - \tilde{b}_{\ell_2})$ we differentiate two cases. First, if $\overline{B}_{\ell_1}^i = t_{\ell_1}$ it follows directly that $\overline{ed}_{\ell_1}^i(t) = \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) = \overline{ed}_{\ell_1}^i(t_{\ell_1}) \leq ed_{\ell_2}^i$ for all $t_{\ell_2} \leq t \leq \tilde{b}_{\ell_2}$. Second, if $\overline{B}_{\ell_1}^i > t_{\ell_1}$ we have that $\overline{ed}_{\ell_1}^i(t) < \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i)$ for all $t < \overline{B}_{\ell_1}^i$ and consequently $\overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) = k_i^2 = k_i^3 + \overline{B}_{\ell_1}^i$ must hold. Using $\overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) > \overline{ed}_{\ell_1}^i(\overline{B}_{\ell_1}^i) - (\overline{B}_{\ell_1}^i - \tilde{b}_{\ell_2})$ it follows that $\overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) > k_i^3 + \tilde{b}_{\ell_2}$ and, thus, $\overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) = k_i^1$. As a result, we have $\overline{ed}_{\ell_1}^i(t) = \overline{ed}_{\ell_1}^i(\tilde{b}_{\ell_2}) = k_i^1 = \overline{ed}_{\ell_1}^i(t_{\ell_1}) \leq ed_{\ell_2}^i$ for all $t_{\ell_2} \leq t \leq \tilde{b}_{\ell_2}$. □

4.B Detailed Computational Results

Table 4.7 shows aggregated results for all algorithms and subclasses. It reports the following columns:

opt^{tree}	The number of optimal solutions obtained by respective algorithm
$time$	The average computation time in seconds
opt^{root}	The number of optimal solutions obtained in the root node
gap^{root}	The average percentage integrality gap in the root node
$cuts$	The average number of cuts generated
$nodes$	The average number of nodes solved

Tables 4.8-4.11 present optimal solution/upper bound values for all instances together with the respective computation times of algorithms IMP^{max}_{min} and $IMP\text{-}I^{max}$. The meaning of the table entries are as follows:

$inst$	The name of the instance
$opt/\overline{ub}$	The value of optimal solution or best known upper bound (overlined)
IMP^{max}_{min}	The computation time in seconds of algorithm IMP^{max}_{min}; *1h* if unable to solve instance within time limit
$IMP\text{-}I^{max}$	The computation time in seconds of algorithm $IMP\text{-}I^{max}$; *1h* if unable to solve instance within time limit

		MM				LM				ML				LL				∅			
		opt^{tree}	time	opt^{root}	gap^{root}	cuts	nodes	opt^{tree}	time	opt^{root}	gap^{root}	cuts	nodes	opt^{tree}	time	opt^{root}	gap^{root}	cuts	nodes	opt^{tree}	time	opt^{root}	gap^{root}	cuts	nodes
A (orig.)	IMP^{max}_{min}	42	23	30	0.06	4	5	42	20	27	0.04	6	5	41	157	26	0.08	5	19	41	289	28	0.08	6	24	166	122	111	0.06	5	13
	$IMP\text{-}I^{max}$	42	25	30	0.07	21	10	42	15	25	0.06	17	7	40	192	23	0.11	31	38	41	205	24	0.08	22	39	165	109	102	0.08	23	23
	$IMP\text{-}I_{min}$	42	116	30	0.06	63	11	42	66	25	0.06	74	5	40	201	26	0.11	72	17	39	288	26	0.12	73	23	163	168	107	0.09	70	14
	$IMP\text{-}I$	42	103	30	0.07	76	10	41	138	24	0.07	79	8	40	242	23	0.13	91	20	39	286	22	0.12	82	24	162	192	99	0.10	82	15
B (4/3)	IMP^{max}_{min}	42	48	30	0.04	6	5	42	79	24	0.05	7	8	42	78	20	0.09	5	10	40	229	18	0.13	6	23	166	109	92	0.08	6	12
	$IMP\text{-}I^{max}$	42	29	23	0.06	35	9	42	39	23	0.07	24	12	41	155	18	0.13	41	36	40	267	17	0.15	28	46	165	122	81	0.10	32	26
	$IMP\text{-}I_{min}$	41	164	25	0.06	119	7	40	224	20	0.10	146	10	37	543	17	0.17	139	31	36	672	15	0.25	143	45	154	401	77	0.15	137	23
	$IMP\text{-}I$	40	218	20	0.08	144	10	40	257	20	0.12	161	13	37	599	15	0.21	166	31	35	728	15	0.27	156	56	152	451	70	0.17	157	28
C (5/3)	IMP^{max}_{min}	41	145	25	0.08	7	24	42	174	24	0.07	9	24	40	240	16	0.20	7	17	41	211	18	0.16	10	16	164	193	83	0.13	8	21
	$IMP\text{-}I^{max}$	41	134	18	0.14	55	33	42	75	21	0.10	31	25	39	308	12	0.26	65	62	41	133	14	0.19	35	30	163	163	65	0.18	47	38
	$IMP\text{-}I_{min}$	36	743	21	0.16	212	36	35	839	18	0.26	242	23	31	1048	11	0.40	221	50	33	981	13	0.41	222	52	135	903	63	0.31	224	41
	$IMP\text{-}I$	34	910	17	0.23	239	32	35	855	18	0.27	258	22	30	1237	8	0.48	253	64	32	1136	11	0.43	244	63	131	1034	54	0.35	248	46
D (6/3)	IMP^{max}_{min}	42	53	18	0.13	10	12	40	234	17	0.16	13	41	38	520	10	0.26	11	29	37	757	14	0.26	12	47	157	391	59	0.20	12	35
	$IMP\text{-}I^{max}$	42	92	14	0.17	73	32	42	185	16	0.20	53	63	36	701	8	0.50	91	145	39	544	12	0.35	55	108	159	381	50	0.30	68	93
	$IMP\text{-}I_{min}$	29	1313	15	0.31	345	24	26	1623	16	0.49	367	32	23	1862	7	0.82	336	80	22	1814	11	0.81	319	80	100	1653	49	0.61	342	58
	$IMP\text{-}I$	29	1346	13	0.36	372	22	23	1741	15	0.53	395	31	21	1971	7	0.97	372	80	22	1864	11	0.85	343	80	95	1730	46	0.68	371	57
∅	IMP^{max}_{min}	167	67	103	0.08	7	11	166	127	92	0.08	9	19	161	249	72	0.15	7	19	159	371	78	0.16	9	27	653	204	345	0.12	8	19
	$IMP\text{-}I^{max}$	167	70	85	0.11	46	21	168	79	85	0.11	31	27	156	339	61	0.25	57	70	161	287	67	0.19	35	55	652	194	298	0.17	42	43
	$IMP\text{-}I_{min}$	148	584	91	0.15	185	18	143	688	79	0.22	207	18	131	913	61	0.38	192	49	130	939	65	0.40	189	56	552	781	296	0.29	193	35
	$IMP\text{-}I$	145	644	80	0.19	208	18	139	748	77	0.25	223	18	128	1012	53	0.45	221	49	128	1003	59	0.42	206	56	540	852	269	0.33	214	35

Table 4.7: Results aggregated by subclass

MM				LM				ML				LL			
inst	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}
MM-A-a2-16	325.23	0.2	0.1	LM-A-a2-16	296.36	0.1	0.1	ML-A-a2-16	293.42	0.1	0.1	LL-A-a2-16	291.68	0.1	0.1
MM-A-a2-20	362.07	0.2	0.1	LM-A-a2-20	360.67	0.1	0.1	ML-A-a2-20	334.18	0.2	0.1	LL-A-a2-20	332.60	0.1	0.3
MM-A-a2-24	423.93	0.3	0.1	LM-A-a2-24	423.19	0.2	0.1	ML-A-a2-24	414.74	0.8	1.0	LL-A-a2-24	410.76	0.6	0.5
MM-A-a3-24	325.83	0.1	0.1	LM-A-a3-24	318.90	0.5	0.3	ML-A-a3-24	319.04	0.6	0.4	LL-A-a3-24	313.40	0.2	0.1
MM-A-a3-30	510.18	1.3	1.8	LM-A-a3-30	489.27	1.5	2.5	ML-A-a3-30	465.11	0.5	0.3	LL-A-a3-30	453.34	2.9	2.2
MM-A-a3-36	558.50	1.0	0.3	LM-A-a3-36	558.50	0.7	0.3	ML-A-a3-36	542.29	0.8	0.5	LL-A-a3-36	537.75	1.4	0.3
MM-A-a4-32	485.00	0.3	0.1	LM-A-a4-32	477.43	0.3	0.1	ML-A-a4-32	477.00	0.4	1.4	LL-A-a4-32	474.40	0.4	0.2
MM-A-a4-40	590.17	1.5	1.0	LM-A-a4-40	568.82	2.1	1.0	ML-A-a4-40	571.25	1.1	1.4	LL-A-a4-40	560.69	1.6	1.9
MM-A-a4-48	677.29	1.3	3.6	LM-A-a4-48	656.98	3.3	1.9	ML-A-a4-48	641.36	2.4	1.3	LL-A-a4-48	635.65	15.0	5.9
MM-A-a5-40	506.32	0.5	0.2	LM-A-a5-40	496.97	0.5	0.2	ML-A-a5-40	494.08	1.6	0.8	LL-A-a5-40	494.08	1.4	0.9
MM-A-a5-50	673.28	19.8	62.0	LM-A-a5-50	664.64	9.8	6.0	ML-A-a5-50	646.73	4.0	9.1	LL-A-a5-50	641.46	5.7	4.1
MM-A-a5-60	817.39	3.4	4.3	LM-A-a5-60	783.41	4.6	4.5	ML-A-a5-60	754.29	378.5	12.0	LL-A-a5-60	752.62	21.6	11.8
MM-A-a6-48	637.86	0.9	1.5	LM-A-a6-48	621.48	1.0	1.7	ML-A-a6-48	611.63	1.6	2.8	LL-A-a6-48	600.99	6.4	4.1
MM-A-a6-60	786.57	7.7	4.9	LM-A-a6-60	778.77	6.1	3.5	ML-A-a6-60	750.63	9.0	6.6	LL-A-a6-60	748.16	9.7	6.8
MM-A-a6-72	930.84	32.0	19.7	LM-A-a6-72	918.23	30.1	25.7	ML-A-a6-72	908.54	1836.7	*1h*	LL-A-a6-72	899.35	1876.7	2551.6
MM-A-a7-56	716.87	4.7	3.7	LM-A-a7-56	703.54	1.2	0.5	ML-A-a7-56	680.13	49.0	35.6	LL-A-a7-56	673.97	14.3	8.3
MM-A-a7-70	887.53	8.4	13.9	LM-A-a7-70	874.70	41.6	26.8	ML-A-a7-70	838.01	19.9	21.2	LL-A-a7-70	828.52	33.3	16.3
MM-A-a7-84	1031.31	332.2	588.4	LM-A-a7-84	1016.73	249.5	136.3	ML-A-a7-84	985.21	124.5	454.6	LL-A-a7-84	981.31	*1h*	*1h*
MM-A-a8-64	756.95	9.1	6.7	LM-A-a8-64	746.76	7.7	7.8	ML-A-a8-64	728.34	64.2	20.9	LL-A-a8-64	717.33	7.2	7.2
MM-A-a8-80	972.74	11.7	10.7	LM-A-a8-80	959.03	15.1	8.6	ML-A-a8-80	933.73	6.9	11.8	LL-A-a8-80	908.28	125.8	34.7
MM-A-a8-96	1186.95	307.0	125.6	LM-A-a8-96	1176.43	83.8	63.5	ML-A-a8-96	1149.95	*1h*	*1h*	LL-A-a8-96	1138.79	3050.3	2142.6
MM-A-b2-16	302.25	0.2	0.1	LM-A-b2-16	302.23	0.1	0.1	ML-A-b2-16	298.25	0.1	0.1	LL-A-b2-16	298.23	0.1	0.1
MM-A-b2-20	319.75	0.2	0.2	LM-A-b2-20	319.75	0.2	0.2	ML-A-b2-20	337.65	0.1	0.1	LL-A-b2-20	337.65	0.1	0.1
MM-A-b2-24	436.05	0.1	0.1	LM-A-b2-24	436.05	0.1	0.1	ML-A-b2-24	432.63	0.1	0.1	LL-A-b2-24	429.89	0.1	0.1
MM-A-b3-24	382.78	0.1	0.1	LM-A-b3-24	382.78	0.1	0.1	ML-A-b3-24	382.78	0.1	0.1	LL-A-b3-24	382.78	0.1	0.1
MM-A-b3-30	537.01	0.2	0.1	LM-A-b3-30	533.03	0.2	0.1	ML-A-b3-30	517.37	0.3	0.3	LL-A-b3-30	509.46	0.2	0.1
MM-A-b3-36	584.43	0.3	0.1	LM-A-b3-36	593.13	0.5	0.2	ML-A-b3-36	583.97	1.2	1.0	LL-A-b3-36	588.75	0.5	0.2
MM-A-b4-32	543.40	0.3	0.4	LM-A-b4-32	539.80	0.2	0.1	ML-A-b4-32	537.68	0.2	0.1	LL-A-b4-32	534.18	0.2	0.1
MM-A-b4-40	666.72	1.2	1.0	LM-A-b4-40	676.38	3.3	4.7	ML-A-b4-40	657.24	0.4	0.2	LL-A-b4-40	667.04	0.6	0.2
MM-A-b4-48	693.59	2.6	1.4	LM-A-b4-48	700.44	3.6	2.2	ML-A-b4-48	691.15	2.1	1.4	LL-A-b4-48	697.87	6.1	2.4
MM-A-b5-40	633.24	3.8	2.5	LM-A-b5-40	627.71	2.8	2.0	ML-A-b5-40	623.83	1.6	1.3	LL-A-b5-40	618.32	0.5	0.2
MM-A-b5-50	805.14	1.1	1.9	LM-A-b5-50	798.52	1.7	4.4	ML-A-b5-50	788.38	1.7	2.5	LL-A-b5-50	781.30	1.9	3.8
MM-A-b5-60	972.70	12.1	9.6	LM-A-b5-60	941.00	11.9	38.6	ML-A-b5-60	937.18	13.8	15.1	LL-A-b5-60	930.51	15.8	12.0
MM-A-b6-48	747.34	1.3	1.6	LM-A-b6-48	729.66	0.7	0.5	ML-A-b6-48	733.57	1.3	1.6	LL-A-b6-48	718.86	0.6	0.2
MM-A-b6-60	898.04	2.0	3.7	LM-A-b6-60	889.33	1.7	0.7	ML-A-b6-60	895.06	14.3	15.4	LL-A-b6-60	884.08	2.0	0.8
MM-A-b6-72	1015.55	48.2	41.6	LM-A-b6-72	1000.44	203.7	113.6	ML-A-b6-72	999.64	53.2	49.9	LL-A-b6-72	990.79	110.1	47.7
MM-A-b7-56	865.07	1.1	0.4	LM-A-b7-56	853.32	3.8	2.6	ML-A-b7-56	843.58	3.1	4.3	LL-A-b7-56	834.60	16.1	13.5
MM-A-b7-70	982.59	10.7	16.6	LM-A-b7-70	950.48	5.7	9.4	ML-A-b7-70	973.08	6.7	15.9	LL-A-b7-70	946.36	8.1	17.1
MM-A-b7-84	1283.07	106.1	77.2	LM-A-b7-84	1272.57	43.8	34.9	ML-A-b7-84	1261.89	326.5	119.0	LL-A-b7-84	1240.59	3162.5	59.6
MM-A-b8-64	888.80	7.8	5.2	LM-A-b8-64	869.05	6.1	4.2	ML-A-b8-64	880.97	9.1	8.1	LL-A-b8-64	861.49	5.5	4.1
MM-A-b8-80	1105.35	3.7	13.1	LM-A-b8-80	1088.24	10.5	18.2	ML-A-b8-80	1090.82	3.9	14.3	LL-A-b8-80	1074.06	9.5	14.7
MM-A-b8-96	1240.16	22.7	11.7	LM-A-b8-96	1236.96	90.6	108.2	ML-A-b8-96	1227.98	39.6	30.6	LL-A-b8-96	1215.04	15.3	31.8

Table 4.8: Results for class A (orig.) instances

MM				LM				ML				LL			
inst	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-T*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-T*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-T*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-T*^{max}
MM-B-a2-16	302.60	0.2	0.1	LM-B-a2-16	296.36	0.1	0.1	ML-B-a2-16	293.42	0.1	0.1	LL-B-a2-16	291.68	0.1	0.1
MM-B-a2-20	358.82	0.1	0.1	LM-B-a2-20	358.82	0.1	0.1	ML-B-a2-20	333.61	0.1	0.1	LL-B-a2-20	332.02	0.5	0.5
MM-B-a2-24	414.96	0.3	0.5	LM-B-a2-24	413.00	0.9	0.7	ML-B-a2-24	410.60	1.6	1.5	LL-B-a2-24	406.61	1.3	0.8
MM-B-a3-24	303.23	0.2	0.1	LM-B-a3-24	303.23	0.2	0.1	ML-B-a3-24	296.29	0.3	0.1	LL-B-a3-24	296.29	0.2	0.1
MM-B-a3-30	490.31	2.1	1.6	LM-B-a3-30	481.28	0.4	1.5	ML-B-a3-30	455.16	4.3	3.2	LL-B-a3-30	449.74	6.0	2.6
MM-B-a3-36	544.06	1.6	0.9	LM-B-a3-36	534.09	2.4	1.0	ML-B-a3-36	529.88	2.8	2.4	LL-B-a3-36	520.14	2.2	2.1
MM-B-a4-32	484.55	0.4	1.4	LM-B-a4-32	470.24	0.3	0.1	ML-B-a4-32	461.66	0.6	0.2	LL-B-a4-32	454.77	1.7	1.3
MM-B-a4-40	571.27	0.9	0.4	LM-B-a4-40	568.82	4.6	2.1	ML-B-a4-40	558.78	1.5	2.5	LL-B-a4-40	556.46	25.3	31.9
MM-B-a4-48	667.10	22.9	5.4	LM-B-a4-48	653.31	22.0	4.4	ML-B-a4-48	627.75	3.8	1.9	LL-B-a4-48	624.02	25.4	5.7
MM-B-a5-40	498.33	1.6	1.2	LM-B-a5-40	496.79	1.5	0.8	ML-B-a5-40	483.97	1.7	1.1	LL-B-a5-40	483.24	4.1	2.6
MM-B-a5-50	655.30	3.9	2.2	LM-B-a5-50	646.39	7.7	5.4	ML-B-a5-50	633.65	12.1	25.6	LL-B-a5-50	627.99	78.1	24.6
MM-B-a5-60	778.20	11.3	10.1	LM-B-a5-60	751.73	7.3	4.8	ML-B-a5-60	731.86	7.0	6.7	LL-B-a5-60	730.95	59.1	32.6
MM-B-a6-48	612.30	1.5	2.0	LM-B-a6-48	610.23	3.2	1.5	ML-B-a6-48	585.35	6.0	4.9	LL-B-a6-48	576.07	3.0	2.8
MM-B-a6-60	768.84	11.7	10.1	LM-B-a6-60	755.64	11.1	7.7	ML-B-a6-60	728.62	38.0	28.0	LL-B-a6-60	714.67	14.6	6.8
MM-B-a6-72	907.44	26.3	19.0	LM-B-a6-72	897.02	18.6	14.6	ML-B-a6-72	876.11	407.5	298.1	LL-B-a6-72	865.49	42.3	40.1
MM-B-a7-56	692.13	4.1	5.1	LM-B-a7-56	691.11	7.7	17.8	ML-B-a7-56	666.67	18.1	38.2	LL-B-a7-56	639.71	15.2	7.8
MM-B-a7-70	866.44	12.2	15.4	LM-B-a7-70	853.72	16.5	6.2	ML-B-a7-70	824.92	29.5	36.0	LL-B-a7-70	813.35	70.8	49.4
MM-B-a7-84	1007.64	79.9	94.2	LM-B-a7-84	989.92	73.5	41.4	ML-B-a7-84	958.41	664.6	764.4	LL-B-a7-84	952.52	*1h*	*1h*
MM-B-a8-64	737.25	7.7	8.4	LM-B-a8-64	726.35	27.6	14.8	ML-B-a8-64	703.74	22.1	15.7	LL-B-a8-64	700.10	32.8	28.8
MM-B-a8-80	958.43	38.4	18.8	LM-B-a8-80	933.74	27.4	20.7	ML-B-a8-80	900.45	111.1	708.4	LL-B-a8-80	881.77	85.9	130.2
MM-B-a8-96	1162.61	1576.2	780.8	LM-B-a8-96	1154.52	2851.8	1212.3	ML-B-a8-96	1114.48	938.3	*1h*	LL-B-a8-96	1103.35	1597.5	3429.5
MM-B-b2-16	289.24	0.1	0.1	LM-B-b2-16	289.24	0.2	0.2	ML-B-b2-16	289.22	0.4	0.3	LL-B-b2-16	289.22	0.3	0.3
MM-B-b2-20	323.71	0.2	0.1	LM-B-b2-20	323.71	0.1	0.1	ML-B-b2-20	323.71	0.2	0.1	LL-B-b2-20	323.71	0.2	0.1
MM-B-b2-24	458.96	0.3	0.1	LM-B-b2-24	458.69	0.2	0.1	ML-B-b2-24	454.60	0.2	0.1	LL-B-b2-24	451.81	0.3	0.1
MM-B-b3-24	361.08	0.1	0.1	LM-B-b3-24	360.24	0.1	0.1	ML-B-b3-24	361.08	0.1	0.1	LL-B-b3-24	360.24	0.1	0.1
MM-B-b3-30	527.64	0.4	0.2	LM-B-b3-30	521.42	0.4	0.1	ML-B-b3-30	504.37	0.3	0.5	LL-B-b3-30	501.27	0.3	0.1
MM-B-b3-36	570.20	0.6	0.3	LM-B-b3-36	553.04	0.6	0.2	ML-B-b3-36	570.20	0.5	0.3	LL-B-b3-36	553.04	0.6	0.3
MM-B-b4-32	487.52	0.3	0.8	LM-B-b4-32	481.45	0.3	0.2	ML-B-b4-32	456.45	0.8	0.7	LL-B-b4-32	456.26	0.9	0.6
MM-B-b4-40	618.08	1.5	3.0	LM-B-b4-40	611.16	1.7	1.7	ML-B-b4-40	609.67	1.9	3.7	LL-B-b4-40	600.16	1.4	0.9
MM-B-b4-48	656.77	6.8	8.0	LM-B-b4-48	637.10	3.7	2.1	ML-B-b4-48	643.64	8.5	7.1	LL-B-b4-48	629.47	2.9	0.9
MM-B-b5-40	603.96	1.7	0.9	LM-B-b5-40	599.88	1.3	0.8	ML-B-b5-40	579.84	2.1	2.9	LL-B-b5-40	579.40	2.3	2.0
MM-B-b5-50	755.61	5.3	4.3	LM-B-b5-50	736.76	14.3	25.0	ML-B-b5-50	719.21	48.0	121.2	LL-B-b5-50	715.82	139.0	47.3
MM-B-b5-60	871.92	6.8	11.6	LM-B-b5-60	866.01	6.9	30.6	ML-B-b5-60	857.14	139.7	306.1	LL-B-b5-60	846.67	9.4	26.7
MM-B-b6-48	706.54	1.7	3.5	LM-B-b6-48	696.19	1.8	1.6	ML-B-b6-48	680.92	0.9	1.8	LL-B-b6-48	677.45	3.0	3.4
MM-B-b6-60	844.48	2.1	4.6	LM-B-b6-60	839.92	36.1	17.6	ML-B-b6-60	829.15	13.4	7.1	LL-B-b6-60	824.66	29.8	9.1
MM-B-b6-72	964.04	68.3	94.1	LM-B-b6-72	954.51	18.6	17.5	ML-B-b6-72	930.21	32.3	50.3	LL-B-b6-72	925.96	51.9	51.1
MM-B-b7-56	764.61	3.6	3.9	LM-B-b7-56	758.26	5.7	3.9	ML-B-b7-56	736.85	4.8	2.9	LL-B-b7-56	736.77	6.3	3.6
MM-B-b7-70	880.91	4.7	4.6	LM-B-b7-70	861.72	10.7	9.0	ML-B-b7-70	870.40	22.1	14.6	LL-B-b7-70	850.36	56.9	27.1
MM-B-b7-84	1173.74	20.8	11.2	LM-B-b7-84	1157.16	54.2	40.1	ML-B-b7-84	1137.84	185.2	183.6	LL-B-b7-84	1125.09	31.1	21.5
MM-B-b8-64	791.96	22.8	23.4	LM-B-b8-64	781.98	11.6	15.1	ML-B-b8-64	773.99	3.7	3.9	LL-B-b8-64	761.51	16.0	12.4
MM-B-b8-80	1011.77	30.1	18.1	LM-B-b8-80	997.54	30.1	22.9	ML-B-b8-80	995.44	17.7	21.5	LL-B-b8-80	986.23	14.4	13.1
MM-B-b8-96	1151.96	18.4	57.8	LM-B-b8-96	1147.03	52.3	80.2	ML-B-b8-96	1111.63	540.2	233.4	LL-B-b8-96	1107.70	*1h*	*1h*

Table 4.9: Results for class B (4/3) instances

MM				LM				ML				LL			
inst	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-1*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-1*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-1*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-1*^{max}
MM-C-a2-16	300.86	0.1	0.1	LM-C-a2-16	296.36	0.1	0.1	ML-C-a2-16	289.40	0.2	0.1	LL-C-a2-16	289.40	0.2	0.1
MM-C-a2-20	354.84	0.3	1.1	LM-C-a2-20	351.49	0.2	0.1	ML-C-a2-20	319.13	0.5	0.5	LL-C-a2-20	315.59	0.6	0.5
MM-C-a2-24	409.20	1.4	1.1	LM-C-a2-24	407.46	2.4	1.7	ML-C-a2-24	405.76	4.1	5.9	LL-C-a2-24	403.94	7.0	4.3
MM-C-a3-24	303.10	0.2	1.7	LM-C-a3-24	299.26	0.2	0.4	ML-C-a3-24	296.02	0.3	1.3	LL-C-a3-24	295.43	0.3	0.7
MM-C-a3-30	471.48	0.4	0.7	LM-C-a3-30	462.89	1.8	2.2	ML-C-a3-30	443.25	3.7	1.9	LL-C-a3-30	436.71	0.9	1.2
MM-C-a3-36	535.36	2.0	1.6	LM-C-a3-36	501.05	2.2	1.1	ML-C-a3-36	523.09	3.6	2.0	LL-C-a3-36	476.85	1.4	0.9
MM-C-a4-32	472.33	0.8	2.2	LM-C-a4-32	462.41	1.0	4.3	ML-C-a4-32	456.97	3.8	6.0	LL-C-a4-32	449.06	3.1	2.7
MM-C-a4-40	557.31	2.6	2.0	LM-C-a4-40	540.58	2.6	2.4	ML-C-a4-40	527.18	35.5	53.9	LL-C-a4-40	520.10	7.9	3.9
MM-C-a4-48	651.53	3.8	3.1	LM-C-a4-48	641.98	7.8	4.5	ML-C-a4-48	621.44	95.8	26.4	LL-C-a4-48	612.24	68.3	23.4
MM-C-a5-40	493.34	0.6	0.2	LM-C-a5-40	491.02	2.0	1.3	ML-C-a5-40	468.99	2.0	1.4	LL-C-a5-40	468.99	2.4	1.4
MM-C-a5-50	647.08	11.5	17.1	LM-C-a5-50	635.09	13.8	11.9	ML-C-a5-50	623.51	100.1	130.3	LL-C-a5-50	614.56	128.2	281.4
MM-C-a5-60	770.48	14.5	7.1	LM-C-a5-60	742.39	6.0	5.9	ML-C-a5-60	716.03	347.4	78.6	LL-C-a5-60	713.79	47.1	20.3
MM-C-a6-48	609.34	1.6	2.8	LM-C-a6-48	606.39	4.4	2.5	ML-C-a6-48	571.69	101.0	30.1	LL-C-a6-48	570.19	236.0	31.9
MM-C-a6-60	754.46	65.9	25.1	LM-C-a6-60	744.35	31.6	43.4	ML-C-a6-60	713.59	19.8	17.8	LL-C-a6-60	707.32	27.1	24.5
MM-C-a6-72	895.60	25.1	28.8	LM-C-a6-72	882.09	20.2	87.0	ML-C-a6-72	861.28	132.8	712.0	LL-C-a6-72	852.35	815.3	136.4
MM-C-a7-56	684.15	42.5	54.3	LM-C-a7-56	668.72	8.3	7.0	ML-C-a7-56	646.31	17.9	13.4	LL-C-a7-56	630.31	28.2	23.9
MM-C-a7-70	857.34	111.2	449.0	LM-C-a7-70	842.63	26.5	31.0	ML-C-a7-70	807.59	24.9	50.9	LL-C-a7-70	794.65	33.1	20.2
MM-C-a7-84	995.37	1102.6	691.5	LM-C-a7-84	975.07	2356.4	569.3	ML-C-a7-84	932.83	65.8	128.1	LL-C-a7-84	926.54	463.2	141.5
MM-C-a8-64	716.43	3.6	17.3	LM-C-a8-64	699.27	17.2	10.2	ML-C-a8-64	695.34	488.9	259.5	LL-C-a8-64	687.57	389.3	82.4
MM-C-a8-80	924.19	119.5	63.7	LM-C-a8-80	912.41	21.6	34.4	ML-C-a8-80	880.66	39.5	83.4	LL-C-a8-80	867.23	394.9	448.7
MM-C-a8-96	1144.32	60.8	77.8	LM-C-a8-96	1133.72	968.7	180.9	ML-C-a8-96	1101.72	*1h*	*1h*	LL-C-a8-96	1087.91	*1h*	*1h*
MM-C-b2-16	279.95	0.3	0.2	LM-C-b2-16	279.64	0.5	0.6	ML-C-b2-16	278.01	0.4	0.6	LL-C-b2-16	274.19	0.1	0.1
MM-C-b2-20	315.00	0.1	0.1	LM-C-b2-20	315.00	0.1	0.1	ML-C-b2-20	315.00	0.4	0.5	LL-C-b2-20	314.13	0.4	0.3
MM-C-b2-24	417.66	0.2	0.1	LM-C-b2-24	417.52	0.2	0.1	ML-C-b2-24	412.49	0.6	0.5	LL-C-b2-24	410.13	0.2	0.2
MM-C-b3-24	353.81	0.3	0.5	LM-C-b3-24	351.64	0.1	0.3	ML-C-b3-24	353.81	0.3	0.8	LL-C-b3-24	351.52	0.1	0.3
MM-C-b3-30	467.01	0.7	0.4	LM-C-b3-30	467.01	0.8	0.5	ML-C-b3-30	456.89	0.4	0.5	LL-C-b3-30	455.59	0.5	0.2
MM-C-b3-36	549.85	0.5	0.2	LM-C-b3-36	542.24	0.6	0.3	ML-C-b3-36	533.92	0.6	0.3	LL-C-b3-36	525.43	0.6	0.2
MM-C-b4-32	465.99	0.8	1.0	LM-C-b4-32	465.99	1.1	0.9	ML-C-b4-32	450.17	7.8	6.0	LL-C-b4-32	449.79	11.2	6.6
MM-C-b4-40	595.06	1.9	6.1	LM-C-b4-40	580.10	1.6	1.2	ML-C-b4-40	568.19	2.0	1.5	LL-C-b4-40	555.57	1.9	1.3
MM-C-b4-48	633.02	2.7	3.7	LM-C-b4-48	619.27	2.2	4.5	ML-C-b4-48	624.58	15.7	12.0	LL-C-b4-48	600.50	6.3	4.9
MM-C-b5-40	563.53	3.1	4.4	LM-C-b5-40	560.44	0.9	2.4	ML-C-b5-40	552.38	3.4	8.4	LL-C-b5-40	550.34	7.0	7.5
MM-C-b5-50	688.43	11.2	8.4	LM-C-b5-50	686.27	12.6	6.6	ML-C-b5-50	665.24	46.7	21.2	LL-C-b5-50	662.70	47.9	14.6
MM-C-b5-60	815.30	9.4	9.8	LM-C-b5-60	811.53	8.2	17.4	ML-C-b5-60	788.81	53.0	22.6	LL-C-b5-60	780.49	7.3	37.4
MM-C-b6-48	650.88	0.9	2.5	LM-C-b6-48	641.55	4.4	3.6	ML-C-b6-48	635.37	4.5	17.2	LL-C-b6-48	626.08	16.1	10.6
MM-C-b6-60	801.09	2.2	6.0	LM-C-b6-60	780.45	2.9	1.6	ML-C-b6-60	773.95	6.9	12.7	LL-C-b6-60	759.93	14.4	11.3
MM-C-b6-72	919.55	296.1	253.0	LM-C-b6-72	914.27	25.6	17.2	ML-C-b6-72	881.75	62.8	59.8	LL-C-b6-72	879.38	89.9	23.9
MM-C-b7-56	705.29	12.0	6.1	LM-C-b7-56	697.02	26.3	14.2	ML-C-b7-56	676.10	7.5	8.0	LL-C-b7-56	674.76	13.4	12.2
MM-C-b7-70	852.49	8.3	10.1	LM-C-b7-70	832.61	42.6	58.6	ML-C-b7-70	829.00	8.5	9.6	LL-C-b7-70	810.92	27.6	21.7
MM-C-b7-84	1103.57	445.8	192.4	LM-C-b7-84	1086.43	37.7	48.6	ML-C-b7-84	1061.32	146.6	329.5	LL-C-b7-84	1054.14	221.5	107.4
MM-C-b8-64	717.73	3.8	9.8	LM-C-b8-64	709.66	7.2	4.6	ML-C-b8-64	705.86	22.2	26.9	LL-C-b8-64	690.12	36.5	19.2
MM-C-b8-80	943.70	*1h*	*1h*	LM-C-b8-80	937.16	3595.3	1960.3	ML-C-b8-80	925.53	999.9	*1h*	LL-C-b8-80	915.44	363.9	170.9
MM-C-b8-96	1096.43	113.8	56.1	LM-C-b8-96	1087.88	62.7	24.6	ML-C-b8-96	1047.67	*1h*	*1h*	LL-C-b8-96	1034.18	1731.5	300.6

Table 4.10: Results for class C (5/3) instances

MM				LM				ML				LL			
inst	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}	*inst*	$opt/\overline{ub}$	IMP^{max}_{min}	*IMP-I*^{max}
MM-D-a2-16	296.46	0.1	0.1	LM-D-a2-16	285.89	0.1	0.1	ML-D-a2-16	280.61	0.1	0.2	LL-D-a2-16	271.94	0.1	0.1
MM-D-a2-20	351.49	0.2	0.2	LM-D-a2-20	351.49	0.6	0.5	ML-D-a2-20	308.81	0.2	0.4	LL-D-a2-20	304.57	0.5	0.4
MM-D-a2-24	407.54	2.0	2.9	LM-D-a2-24	407.46	3.5	2.4	ML-D-a2-24	401.32	4.7	4.3	LL-D-a2-24	382.74	3.9	2.6
MM-D-a3-24	299.26	0.2	0.9	LM-D-a3-24	297.77	0.3	0.4	ML-D-a3-24	293.19	1.9	1.4	LL-D-a3-24	289.57	1.6	0.7
MM-D-a3-30	461.19	3.6	8.1	LM-D-a3-30	453.45	4.0	4.3	ML-D-a3-30	432.15	1.2	5.8	LL-D-a3-30	417.82	1.7	0.4
MM-D-a3-36	502.16	2.5	1.6	LM-D-a3-36	500.19	2.2	1.2	ML-D-a3-36	475.71	19.8	0.7	LL-D-a3-36	471.16	1.5	0.5
MM-D-a4-32	458.70	3.7	5.0	LM-D-a4-32	457.19	9.8	6.1	ML-D-a4-32	439.19	5.3	8.5	LL-D-a4-32	435.87	3.3	3.8
MM-D-a4-40	549.36	3.4	3.7	LM-D-a4-40	518.20	8.8	4.2	ML-D-a4-40	521.95	46.1	60.3	LL-D-a4-40	503.98	31.9	7.5
MM-D-a4-48	638.93	86.7	45.9	LM-D-a4-48	615.97	5.0	2.8	ML-D-a4-48	603.55	414.8	137.1	LL-D-a4-48	599.43	1606.6	471.1
MM-D-a5-40	471.86	1.9	1.1	LM-D-a5-40	467.48	1.0	0.3	ML-D-a5-40	450.48	11.5	3.8	LL-D-a5-40	443.19	1.8	0.5
MM-D-a5-50	629.65	17.9	10.8	LM-D-a5-50	619.63	42.1	10.4	ML-D-a5-50	595.16	22.8	10.1	LL-D-a5-50	585.90	1655.6	329.9
MM-D-a5-60	759.70	36.2	14.5	LM-D-a5-60	735.26	45.3	15.3	ML-D-a5-60	699.20	109.4	37.8	LL-D-a5-60	689.85	135.7	44.9
MM-D-a6-48	589.17	9.6	9.7	LM-D-a6-48	575.93	2.8	3.9	ML-D-a6-48	559.60	404.2	19.0	LL-D-a6-48	558.59	203.8	38.0
MM-D-a6-60	744.79	58.9	61.8	LM-D-a6-60	734.92	44.7	31.4	ML-D-a6-60	704.73	53.3	122.8	LL-D-a6-60	697.59	75.1	47.8
MM-D-a6-72	871.71	32.8	32.6	LM-D-a6-72	864.78	38.6	19.0	ML-D-a6-72	853.62	*1h*	*1h*	LL-D-a6-72	839.37	3333.2	1974.1
MM-D-a7-56	664.87	5.4	3.6	LM-D-a7-56	659.15	31.5	18.7	ML-D-a7-56	632.79	32.9	25.1	LL-D-a7-56	619.50	94.9	50.5
MM-D-a7-70	836.38	29.6	125.6	LM-D-a7-70	827.30	77.0	39.4	ML-D-a7-70	792.07	52.3	85.6	LL-D-a7-70	775.68	47.6	56.6
MM-D-a7-84	968.87	232.9	1150.2	LM-D-a7-84	953.92	*1h*	2680.8	ML-D-a7-84	$\overline{927.15}$	*1h*	*1h*	LL-D-a7-84	909.82	*1h*	*1h*
MM-D-a8-64	707.89	126.3	153.6	LM-D-a8-64	693.22	32.6	29.3	ML-D-a8-64	685.05	304.5	153.2	LL-D-a8-64	677.88	669.2	1197.0
MM-D-a8-80	910.29	46.1	73.7	LM-D-a8-80	896.52	38.7	39.2	ML-D-a8-80	862.07	481.5	477.3	LL-D-a8-80	847.78	1142.3	666.1
MM-D-a8-96	1117.93	67.2	34.5	LM-D-a8-96	1108.70	430.6	87.5	ML-D-a8-96	$\overline{1070.58}$	*1h*	*1h*	LL-D-a8-96	1057.65	*1h*	*1h*
MM-D-b2-16	279.95	0.2	0.3	LM-D-b2-16	274.50	0.1	0.2	ML-D-b2-16	274.50	0.1	0.1	LL-D-b2-16	270.99	0.2	0.1
MM-D-b2-20	315.00	0.2	0.1	LM-D-b2-20	313.11	0.5	0.8	ML-D-b2-20	313.31	0.6	0.6	LL-D-b2-20	306.23	0.2	0.5
MM-D-b2-24	406.83	0.3	0.6	LM-D-b2-24	404.87	0.2	0.1	ML-D-b2-24	394.86	2.4	8.3	LL-D-b2-24	384.27	0.6	0.2
MM-D-b3-24	334.94	0.1	0.1	LM-D-b3-24	329.80	0.1	0.3	ML-D-b3-24	329.68	0.5	0.3	LL-D-b3-24	328.50	0.5	0.3
MM-D-b3-30	465.34	0.4	1.5	LM-D-b3-30	464.84	0.5	0.2	ML-D-b3-30	447.15	1.7	1.1	LL-D-b3-30	446.76	2.8	1.7
MM-D-b3-36	542.92	1.1	0.8	LM-D-b3-36	542.92	1.2	1.0	ML-D-b3-36	520.12	1.3	0.7	LL-D-b3-36	519.72	1.5	0.6
MM-D-b4-32	439.65	0.3	0.6	LM-D-b4-32	435.90	1.4	0.9	ML-D-b4-32	419.33	1.5	2.5	LL-D-b4-32	419.21	2.4	2.5
MM-D-b4-40	556.70	1.7	1.3	LM-D-b4-40	522.69	2.1	1.5	ML-D-b4-40	528.30	2.4	2.3	LL-D-b4-40	502.92	5.1	2.7
MM-D-b4-48	628.44	5.1	7.4	LM-D-b4-48	615.54	4.7	5.9	ML-D-b4-48	606.67	76.7	288.6	LL-D-b4-48	584.89	7.7	13.6
MM-D-b5-40	554.12	30.1	13.8	LM-D-b5-40	549.60	5.4	3.5	ML-D-b5-40	534.30	22.3	26.3	LL-D-b5-40	530.71	12.0	11.1
MM-D-b5-50	668.62	13.3	22.0	LM-D-b5-50	664.22	58.7	60.8	ML-D-b5-50	642.21	246.3	2364.5	LL-D-b5-50	629.16	46.6	23.3
MM-D-b5-60	787.38	7.9	9.1	LM-D-b5-60	785.21	10.4	38.3	ML-D-b5-60	745.49	23.3	49.8	LL-D-b5-60	742.25	304.6	414.9
MM-D-b6-48	627.30	14.6	21.6	LM-D-b6-48	616.86	24.1	10.7	ML-D-b6-48	606.24	194.4	1631.5	LL-D-b6-48	590.15	72.2	46.8
MM-D-b6-60	748.29	3.5	12.9	LM-D-b6-60	742.45	69.4	43.8	ML-D-b6-60	733.29	59.9	163.9	LL-D-b6-60	726.76	*1h*	2496.8
MM-D-b6-72	895.50	114.5	184.3	LM-D-b6-72	889.12	470.1	134.6	ML-D-b6-72	851.35	2051.7	1623.9	LL-D-b6-72	840.78	2653.0	688.6
MM-D-b7-56	682.37	12.6	41.2	LM-D-b7-56	674.81	154.2	41.0	ML-D-b7-56	635.15	125.8	64.3	LL-D-b7-56	630.27	69.9	32.0
MM-D-b7-70	823.95	59.5	158.7	LM-D-b7-70	805.16	172.8	104.9	ML-D-b7-70	777.62	44.1	35.4	LL-D-b7-70	760.55	42.4	18.7
MM-D-b7-84	1057.05	773.2	773.4	LM-D-b7-84	1045.84	396.1	70.1	ML-D-b7-84	1016.84	*1h*	*1h*	LL-D-b7-84	1012.45	*1h*	2775.8
MM-D-b8-64	704.70	29.9	103.9	LM-D-b8-64	695.72	*1h*	3405.6	ML-D-b8-64	683.44	439.5	406.0	LL-D-b8-64	$\overline{669.07}$	764.4	268.1
MM-D-b8-80	906.13	42.2	447.5	LM-D-b8-80	897.14	106.5	568.6	ML-D-b8-80	869.68	1690.1	*1h*	LL-D-b8-80	$\overline{861.25}$	*1h*	*1h*
MM-D-b8-96	1079.80	355.9	323.1	LM-D-b8-96	1068.61	341.8	294.0	ML-D-b8-96	1021.89	486.0	*1h*	LL-D-b8-96	1001.47	779.8	365.1

Table 4.11: Results for class D (6/3) instances

References

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**(1), 7–22.

Chapter 5

Dual Inequalities for Stabilized Column Generation Revisited

Timo Gschwind, Stefan Irnich

Abstract

Column generation (CG) models have several advantages over compact formulations, namely, they provide better LP bounds, may eliminate symmetry, and can hide non-linearities in their subproblems. However, users also encounter drawbacks in the form of slow convergency a.k.a. the tailing-off effect and the oscillation of the dual variables. Among different alternatives for stabilizing the CG process, Ben Amor *et al.* [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] suggest the use of dual-optimal inequalities (DOIs) in the context of cutting stock and bin packing problems. We generalize their results, provide new classes of (deep) DOIs, and show the applicability to other problems (vector packing, vertex coloring, bin packing with conflicts). We also suggest the dynamic addition of violated dual inequalities in a cutting-plane fashion and the use of dual inequalities that are not necessarily (deep) DOIs. In the latter case, a recovery procedure is needed to restore primal feasibility. Computational results proving the usefulness of the methods are presented.

5.1 Introduction

Column generation (CG) has been proven a versatile tool for solving large-scale linear programs (LPs) with often more variables than explicitly representable by considering only a selected subset of them at a time. Such huge models may arise naturally or may result from a reformulation of an integer compact model using a Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960), which then leads to an integer CG approach also known as branch-and-price (Barnhart *et al.*, 1998; Lübbecke and Desrosiers, 2005). The resulting so-called extensive formulation has an enormous number of variables corresponding to extreme points and extreme rays of the domain chosen as the subproblem in the decomposition. While the huge number of variables in the extensive formulation let the model seem unattractive, it also contributes with some profound advantages:

The extensive formulation has a tighter LP-bound if the subproblem does not posses the integrality property. Moreover, if non-linearities are present in the compact formulation, they may be well managed using the right definition of the subproblems. Additionally, some very successful CG-based algorithms have been invented for applications in, e.g., vehicle and crew routing and scheduling (Desaulniers *et al.*, 1998), cutting and packing (Ben Amor and Valério de Carvalho, 2005), and graph coloring (Held *et al.*, 2012). Their success can be attributed to the availability of very powerful subproblem solution algorithms that have reached a high degree of maturity, see (Irnich and Desaulniers, 2005) for constrained shortest path problems, (Kellerer *et al.*, 2004) for knapsack problems, and (Östergård, 2002; Carraghan and Pardalos, 1990) for cliques/stable sets.

However, CG approaches in practice often suffer from instability problems. The dual variables, which drive the generation process, may oscillate heavily before they finally converge to some optimal dual values. On the primal side, CG formulations often tend to be degenerated: In many applications the master program is some (generalized) set-partitioning or set-covering problem, in which each variable represents a crew schedule, vehicle route, packing, or partition, which is often a dense column so that a few columns comprise a solution. However, a primal basis must include several other variables that are then at value zero. As a consequence, primal degeneracy leads to rather small and often non-improving LP pivots, known as the tailing-off effect (Gilmore and Gomory, 1961; Vanderbeck, 2005). The process of variable generation then continues over many iterations to produce nearly no improvement in the LP objective. In order to explicitly stabilize the dual values, algorithmic techniques like the box step method (Marsten *et al.*, 1975), bundle methods (Hiriart-Urruty and Lemaréchal, 1993), and tailored CG stabilization approaches have been proposed, e.g., by du Merle *et al.* (1999); Rousseau *et al.* (2007); Lee and Park (2011). Furthermore, a branch of the recent literature is especially devoted to tools that can help overcome or even benefit from primal degeneracy when solving huge LPs (Gauthier *et al.*, 2014; Desrosiers *et al.*, 2014).

In the paper at hand, we continue the path originally followed by Valério de Carvalho (2005) and Ben Amor *et al.* (2006). For the *cutting stock problem* (CS) and the *bin packing problem* (BP), they have shown that the knowledge of the domain of optimal dual values can be used for stabilizing the CG process. In particular, any valid inequality known for the optimal dual space can be added as an additional variable to the corresponding primal CG formulation.

We introduce the concept of dual inequalities in a more formal way now following the notation of Ben Amor *et al.* (2006): Consider a pair of primal and dual LPs of the forms

$$
\begin{array}{lll}
 & z_P = \min c^\top \lambda & \\
(P) \qquad & \text{s.t.} \quad A\lambda = b, & \\
 & \qquad\ \ \lambda \geq 0. &
\end{array}
\qquad\qquad
\begin{array}{lll}
 & z_D = \max b^\top \pi & \\
(D) \qquad & \text{s.t.} \quad A^\top \pi \leq c. &
\end{array}
$$

with a matrix $A = (a_{ij}) \in \mathbb{R}^{I \times J}$, $c = (c_i) \in \mathbb{R}^I$, and $b = (b_i) \in \mathbb{R}^I$ with row indices $i \in I$ and column indices $j \in J$. Let $m = |I|$ denote the number of rows. We refer to the jth column of A as $a_j \in \mathbb{R}^I$ such that $A = (a_j)_{j \in J}$. For any vector $a \in \mathbb{R}^I$, we write $a \in A$

if and only if $a = a_j$ for some $j \in J$. Rows of A are denoted by $a_{i*} \in \mathbb{R}^J$ for $i \in I$.

In the following, we always assume that the primal formulation P is the extensive formulation to which a CG algorithm is applied. Therefore, we do not solve P directly, but work with a restricted version of P, the *restricted master program* (RMP), for which re-optimization of the LP and the solution of the pricing problem (=subproblem) alternate. The pricing problem asks for the determination of at least one column $a_j \in A$, for which $c_j - \pi^\top a_j < 0$ holds, or the guarantee that no such column exists. In the latter case, P has been solved to optimality and the CG process terminates.

The idea of Ben Amor *et al.* (2006) was to add additional constraints $E^\top \pi \leq e$ to the dual D. Additional constraints in the dual D correspond with additional variables in the primal P, denoted by y in the following. The extended primal and dual models are:

$$
\begin{array}{lll}
 & z_{\tilde{P}} = \min c^\top \lambda + e^\top y & \\
(\tilde{P}) \quad & \text{s.t.} \quad A\lambda + Ey = b, & \\
 & \lambda \geq 0, y \geq 0. &
\end{array}
\qquad
\begin{array}{lll}
 & z_{\tilde{D}} = \max b^\top \pi & \\
(\tilde{D}) \quad & \text{s.t.} \quad A^\top \pi \leq c, & \\
 & E^\top \pi \leq e. &
\end{array}
$$

The set of additional *dual inequalities* (DIs) $E^\top \pi \leq e$ cuts part of the dual solution space. Thus, $\tilde{D}$ is a restriction of D, whereas the corresponding primal model $\tilde{P}$ is a relaxation of P. We denote by D^* the set of optimal solutions to the model D, i.e., the dual-optimal space. Throughout the text, we heavily use acronyms; for convenience, Table 5.1 lists those that are most frequently used.

One can further differentiate two special classes of DIs depending on which part of the dual solution space they cut off (see Ben Amor *et al.*, 2006, p. 455): DIs $E^\top \pi \leq e$ are called *dual-optimal inequalities* (DOIs) if *all* dual-optimal solutions $\pi^* \in D^*$ also satisfy the DIs, i.e., $D^* \subseteq \{\pi : E^\top \pi \leq e\}$. If DIs $E^\top \pi \leq e$ are satisfied by at least one dual-optimal solution $\pi^* \in D^*$, i.e., $D^* \cap \{E^\top \pi \leq e\} \neq \varnothing$, they are called a set of *deep dual-optimal inequalities* (DDOIs). Note that deep dual-optimal is a property referring to a set of DIs, meaning that DDOIs cannot be tested individually for each inequality in $E^\top \pi \leq e$. In contrast, for a set of DOIs, every single DI $E_{j*}\pi \leq e_j$, where E_{j*} refers to the jth row of E, must qualify as a DOI. This is fulfilled if all dual-optimal solutions $\pi^* \in D^*$ respect $E_{j*}\pi^* \leq e_j$.

The contribution of our paper consists in at least five new findings: First, we generalize Propositions 1 and 2 of Ben Amor *et al.* (2006) and Proposition 1 of Valério de Carvalho (2005) providing insights into the relations of optimal solution values and optimal solutions to models P, D, $\tilde{P}$, and $\tilde{D}$ when $E^\top \pi \leq e$ are DOIs or DDOIs. Second, we define several new classes of properties for integer valued matrices A and unit costs $c = \mathbf{1}$ that enable the identification of new DOIs and DDOIs. Herewith, we derive additional classes of DDOIs for BP and DOIs for CS. The latter can be extended also to the *vector packing problem* (VP) which can be seen as the multi-dimensional generalization of CS.

The third aspect is the extension of the DI-based stabilization to several new classes of problems: We address the *vertex coloring problem* (VC) in which the task is to color the vertices of a given undirected graph with a minimum number of colors so that adjacent vertices receive different colors. The synthesis of VC and BP is the *bin packing problem*

Acronym	Full Name
CG	Column generation
RMP	Restricted master program
DI	Dual inequality; any inequality in the variables of D
DOI	Dual-optimal inequality; fulfilled by every dual-optimal solution of D
DDOI	(Set of) deep dual-optimal inequality/ies
WSI	Weighted subset inequality
SI	Subset inequality
CS	Cutting stock problem
BP	Bin packing problem
BPC	Bin packing problem with conflicts
VP	Vector packing problem
VC	Vertex coloring problem
KP	Binary knapsack problem
UKP	Unbounded knapsack problem
KPC	Binary knapsack problem with conflicts
DKP	Unbounded version of the d-dimensional knapsack problem
MWIS	Maximum weight independent set

Table 5.1: Some acronyms used in the paper

with conflicts (BPC). In all cases the derivation of DOIs and DDOIs is based on showing that the above mentioned new matrix properties are valid.

Fourth, we show that a kind of over-stabilization of CG procedures can result in an overall faster convergency. There exist cases in which one can suspect that $E^\top \pi \leq e$ are DDOIs, but this may actually not be true. The property may be fulfilled just for some instances of the problem or just for a proper subset that one is unable to identify. In the negative case, the outcome of the stabilized CG algorithm is an infeasible primal solution and a weaker bound than z_P (see Proposition 5.1). We show that there is a constructive way of identifying the DIs that have cut off D^*. This can be seen as a *recovery procedure*. In order to accelerate the overall CG approach, we therefore propose to alternate between solving the (over-)stabilized formulation $\tilde{P}$ and the recovery procedure until a feasible primal solution results.

Fifth and finally, we show that the dynamic generation of DIs during the CG process is another option that often helps reducing the computation time. Note that up to now, DOIs and DDOIs have been used in a static fashion. Indeed, when the number of known DOIs or DDOIs is relatively small, e.g., polynomial in the size of the input instance, then $\tilde{P}$ is not significantly larger than P, and the advantage of stability can then outreach the resulting larger RMP re-optimization times. Such a family of DOIs has been used in a branch-and-price approach for the capacitated arc-routing problem (Bode and Irnich, 2012). However, in all problems considered here, the families of DIs, DOIs, and DDOIs are exponential in size. For the problems CS, BP, VC, and BPC we demonstrate that often the dynamic generation, sometimes in combination with an a priori addition of the expectedly strongest DIs provides the best trade-off between stabilization, effort of DI generation, and effort for re-optimizing the RMP.

The remainder of this paper is organized as follows: Section 5.2 presents the generalized proposition on the relations of solutions of models P, D, $\tilde{P}$, and $\tilde{D}$. In Section 5.3, we derive useful properties of the coefficient matrix allowing to identify DIs that are DOIs and/or DDOIs. Along with introducing the new properties we present their application to different problems. This includes the proper introduction of the CG formulations of these problems and the specification of the different new classes of DOIs and DDOIs. Section 5.4 clarifies the dynamic identification of violated DIs and presents the recovery procedure. In Section 5.5, we present aggregated computational results for BP, CS, VC, and BPC. Final conclusions are drawn in Section 5.6.

5.2 Equivalence Conditions for the Original and Extended Models

The central question of this section is the following: Under which conditions are the models P and $\tilde{P}$ and the models D and $\tilde{D}$ equivalent? Conditions for the equivalence depend on the set of DIs $E^\top \pi \leq e$. Equivalence of the respective models means that they provide identical LP-bounds. Moreover, it is required that optimal solutions to the extended models $\tilde{P}$ and $\tilde{D}$ can be transformed into optimal solutions of the corresponding original models P and D, respectively. In case of equivalence, the stabilized model $\tilde{P}$ can now be solved by CG instead of the non-stabilized model P.

The next proposition generalizes the findings of Ben Amor *et al.* (2006) also including some results of Valério de Carvalho (2005). In essence, if the $E^\top \pi \leq e$ are DDOIs then the models $P, D, \tilde{P}$, and $\tilde{D}$ are equivalent, and vice versa.

Proposition 5.1. *The following statements are equivalent:*

(i) $E^\top \pi \leq e$ *are DDOIs.*
(ii) *There exists a* $\pi^* \in D^*$ *which is feasible also for* $\tilde{D}$.
(iii) $z_D = z_{\tilde{D}}$.
(iv) $z_P = z_{\tilde{P}}$.
(v) *For every feasible primal solution* $(\tilde{\lambda}, y)$ *to* $\tilde{P}$ *there exists a primal feasible solution* λ *to* P *with* $c^\top \lambda \leq c^\top \tilde{\lambda} + e^\top y$.
(vi) *There exists a primal optimal solution* $(\tilde{\lambda}^*, y^*)$ *to* $\tilde{P}$ *with* $Ey^* = 0$. *Also,* $\tilde{\lambda}^*$ *is an optimal solution to* P.
(vii) *Every optimal dual solution* π^* *to* $\tilde{D}$ *is optimal for* D.

The proof of Proposition 5.1 and proofs for all other propositions and theorems can be found in Section 5.A of the Appendix.

Ben Amor *et al.* (2006) have shown the implications (i) $\Rightarrow$ (iii), (i) $\Rightarrow$ (iv), and (i) $\Rightarrow$ (vii). Moreover, for DOIs they showed that if there exists a primal optimal solution $(\tilde{\lambda}^*, y^*)$ to $\tilde{P}$ with $Ey^* = 0$, then $\tilde{\lambda}^*$ is optimal for P. Since DOIs are also DDOIs, Proposition 5.1 proves the fact that if $E^\top \pi \leq e$ are DOIs then this implies (i)–(vii). The reverse does not hold in general.

From a practical point of view, the condition (v) is particularly useful for problems for which a constructive procedure is known to transform solutions $(\tilde{\lambda}, y)$ of $\tilde{P}$ so that

they lead to a solution λ of P with less or equal cost. A solution to model P can then be determined by first solving the relaxed model $\tilde{P}$ to optimality, and then converting this solution into a solution to P. This approach has been taken by Valério de Carvalho (2005) when solving CS with CG: He first solves the extended master program $\tilde{P}$, in which several additional y variables corresponding to DIs have been added a priori. As a result, the optimal master program may contain positive y variables. This solution is then transformed into a feasible CS solution with identical cost (our recovery procedure presented in Section 5.4.2 is inspired by this transformation). Thus, Valério de Carvalho (2005) was the first to show that the original CS problem is actually solved to optimality, even if additional y variables are active. In fact, the DIs that he used for CS are DOIs as later shown by Ben Amor *et al.* (2006).

Even if there is no direct transformation from solutions $(\tilde{\lambda}, y)$ of $\tilde{P}$ into solutions λ of P known, the model P can still be solved by solving a stabilized variant of the model $\tilde{P}$. Ben Amor *et al.* (2006) show that for DOIs it suffices to marginally increase the costs e of the y variables to $e + \varepsilon$. As a result, all DOIs are inactive and the corresponding primal variables y must be at 0 (resulting from complementary slackness). Moreover, Ben Amor *et al.* (2006) suggest a two-phase procedure for DDOIs: In the first phase, they determine an optimal solution to $\tilde{P}$. It has a corresponding dual solution π^*. In the second phase, they use a stabilized model P, in which the dual variables π are stabilized with the help of a trust region around π^*. The primal solution λ^* to this stabilized model is then a feasible solution to P, see (Ben Amor *et al.*, 2006, Prop. 5).

5.3 Matrix Properties for Deriving DOIs and DDOIs

The definition of DOIs and DDOIs refers to the set of dual-optimal solutions D^*. Up to now, DOIs and DDOIs have been derived by analyzing structural properties of the problem at hand, i.e., for CS and BP. In this section, we derive some new results by analyzing the structure of the constraint matrix A defining the model P. This allows us to apply the results to new and different problems showing that several classes of DIs are DOIs or DDOIs.

Some additional notation is needed: Recall that I are the row indices and J are the column indices of the coefficient matrix A. We denote the ith unit vector for $i \in I$ by $u_i \in \mathbb{Z}_+^I$. Moreover, $\mathbf{0}$ and $\mathbf{1}$ are vectors with all entries 0 and 1, respectively, of appropriate size. In the following, we consider an integer valued matrix $A \in \mathbb{Z}_+^{I \times J}$, an integer valued, strictly positive right-hand side (rhs) $b \in \mathbb{Z}_{>0}^I$ and unit costs $c = \mathbf{1} \in \mathbb{R}^J$.

5.3.1 Exchange Property

Several results presented in the following use the following exchange property:

Definition 5.1. (Exchange Property) Let $s, t \in \mathbb{Z}_+^I$ be given. A matrix $A \in \mathbb{Z}_+^{I \times J}$ has the (s,t)-*exchange property* if $a_j \in A, a_j \geq s$ implies $a_j - s + t \in A$.

If one or both vectors s and t are unit vectors, e.g., $s = u_h$, we simplify the notation and write (h,t)-exchange property. If one of the vectors is the null vector, e.g., $t = \mathbf{0}$, we write $(s,0)$-exchange property.

The exchange property means that the columns in A are related to each other. Any column that is not too small, i.e., fulfills $a_j \geq s$, can be converted into another column by exchanging entries in some rows against entries in other rows (described by $t - s$). In this case, useful relations between optimal values of primal and dual variables can be derived:

Proposition 5.2. *Let $s = (s_i), t = (t_i) \in \mathbb{Z}^I_+$ and $A \in \mathbb{Z}^{I \times J}_+$ be given. If A has the (s,t)-exchange property and (λ^*, π^*) is a pair of optimal solutions to P and D, then*

$$\text{(Exch-D)} \quad \sum_{i \in I} s_i \pi^*_i \geq \sum_{i \in I} t_i \pi^*_i \qquad \text{or} \qquad \text{(Exch-P)} \quad \lambda^*_k = 0 \quad \text{for all } k \in J \text{ with } a_k \geq s.$$

If s is a unit vector u_h for $h \in I$, the exchange property allows that an entry in row h is replaced by entries in a subset of other rows given by the (strictly) positive entries in t. The following proposition shows that valid DOIs result for the dual variables described by h and the positive components of t.

Proposition 5.3. *Let $h \in I$ and $t \in \mathbb{Z}^I_+$ be given. If a matrix $A \in \mathbb{Z}^{I \times J}_+$ has the (h,t)-exchange property, then any dual-optimal solution $\pi^* \in D^*$ fulfills*

$$\pi^*_h \geq \sum_{i \in I} t_i \pi^*_i.$$

Proposition 5.3 can directly be applied to identify DOIs for problems whose coefficient matrix A has the (h,t)-exchange property. We exemplify this for CS and VP.

Cutting Stock The CS consists of finding cutting patterns for cutting rolls of length L into items $i \in I$ of length $w_i \leq L$ such that the total number of rolls is minimal and given demands b_i of the items $i \in I$ are fulfilled. A feasible cutting pattern cuts a roll into several of the items with $\sum_{i \in I} a_{ij} w_i \leq L$, where a_{ij} denotes the number of items of length w_i that are produced by pattern $j \in J$. An extensive formulation for CS was first presented by Gilmore and Gomory (1961), and several CG-based algorithms have been presented over the years, e.g., by Valerió de Carvalho (1998); Vanderbeck (2000); Ben Amor and Valério de Carvalho (2005). In out notation, model P for CS consists of columns $a_j \in \mathbb{Z}^I_+$ forming A, one for each feasible cutting pattern. The variables λ_j give the number of rolls that are cut according to pattern j (cf. Gilmore and Gomory, 1963). The pricing problem has to identify a cutting pattern with negative reduced cost. This is an *unbounded knapsack problem* (UKP), which can be solved by dynamic programming algorithms as a *longest path problem* in an acyclic digraph (see Kellerer *et al.*, 2004). We provide some more details in Section 5.4.1.

Vector Packing The VP is the d-dimensional ($d \geq 2$) generalization of CS. CG algorithms for VP have been presented by Caprara and Toth (2001) and Alves *et al.*

(2014). In VP, items $i \in I$ have a size $w_{i1}, w_{i2}, \ldots, w_{id}$ in each of the d dimensions (e.g., weight, volume, cost etc.), and *cutting patterns* or *packings* are restricted not only by a single capacity, but by d independent capacities $L_1, L_2, \ldots, L_d$. A straightforward CG model for VP is analog to the one for CS: Columns in the master program correspond to feasible packings, while the subproblem is the *unbounded* version of a *d-dimensional knapsack problem* (DKP) (see Kellerer *et al.*, 2004, Ch. 9).

A direct consequence of Proposition 5.3 is:

Theorem 5.1. *Let $d = 1$ or $d \geq 2$. Moreover, let $w_i, i \in I$ (for $d = 1$) or $w_{ip}, i \in I, p \in \{1, 2, \ldots, d\}$ (for $d \geq 2$) be the d-dimensional weights defining CS and VP, and let $h \in I$, $S \subseteq I \setminus \{h\}$, and $t \in \mathbb{Z}^S_{>0}$ be given. We distinguish between $d = 1$ and $d \geq 2$:*

(i) *If $w_h \geq \sum_{s \in S} t_s w_s$, then the inequality $\pi_h \geq \sum_{s \in S} t_s \pi_s$ is a DOI for CS.*
(ii) *If $w_{hp} \geq \sum_{s \in S} t_s w_{sp}$ for all $p = 1, \ldots, d$, then the inequality $\pi_h \geq \sum_{s \in S} t_s \pi_s$ is a DOI for VP.*

In the following, we refer to DIs of the form $\pi_h \geq \sum_{s \in S} t_s \pi_s$ as *weighted subset inequalities* (WSIs). In the primal model $\tilde{P}$, the corresponding kth column to a WSI is $(E_{ik})_{i \in I} \in \mathbb{Z}^I$ with

$$(E_{ik}) = \begin{cases} t_i & i \in S, \\ -1 & i = h, \\ 0 & otherwise, \end{cases}$$

and a cost of zero, i.e., $e_k = 0$. We refer to such a WSI and its (so-called) WSI column as $(h \leftarrow t, S)$ with $h \in I$, $S \subseteq I \setminus \{h\}$, and $t \in \mathbb{Z}^S_{>0}$ instead of using the column index k. Moreover, for $t = \mathbf{1} \in \mathbb{Z}^S_{>0}$, the DI $\pi_h \geq \sum_{s \in S} \pi_s$ is a *subset inequality* (SI), and we refer to it and its column as $(h \leftarrow S)$.

The WSIs and the associated primal WSI columns have a very intuitive practical interpretation in CS: The quantities t_s for $s \in S$ describe a combination of items including copies whenever $t_s > 1$. Whenever the total length of the combination is not longer than the length w_h of item h, then it is more difficult to include item h in a cutting pattern than all items (and their copies) of the combination. Hence, the marginal cost π_h of producing item h should be not smaller than the overall marginal cost of producing the combination. It also means that in any cutting pattern that includes item h one can safely replace this item h by the combination, i.e., all items $s \in S$ in quantities given by $t_s, s \in S$. The result is a different, but certainly feasible cutting pattern.

When $\tilde{P}$ is solved by CG, the presence of a WSI $(h \leftarrow t, S)$ in the RMP implicitly represents several other cutting pattern columns. For any cutting pattern column a_j including item h, i.e., $a_j \geq u_h$, the sum of column a_j and a WSI column $(h \leftarrow t, S)$ realizes the cutting pattern in which items $s \in S$ are cut an additional t_s times and item h is cut once fewer than given by a_{hj}. Thus, the presence of WSI columns in the RMP prevents having to explicitly generate specific columns because they are already implicitly represented. Moreover, these implicitly represented cutting patterns and other WSIs together represent additional cutting pattern columns.

Note that the WSIs for CS in Theorem 5.1 are generalizations of the SIs introduced by Valério de Carvalho (2005) and proven to be DOIs by Ben Amor *et al.* (2006). Both works exclusively consider DOIs of the form $\pi_h \geq \sum_{s \in S} \pi_s$, hence they neglect the possibility to replace a single item by multiples of one or more items $s \in S$.

5.3.2 Covering Property

It seems to be common knowledge or folklore that in CG (generalized) covering formulations are preferable over partitioning formulations, i.e., inequality over equality constraints. In the light of Proposition 5.3 we can justify this as follows: If the matrix A has the $(h, 0)$-exchange property, it means that matrix entries in the row h can be decreased as long as a column $a_j, j \in J$ has a positive entry a_{hj}. Intuitively, the hth equality of $A\lambda = b$ can then be replaced by a covering constraint. The following remark formalizes the idea from a primal and dual perspective:

Remark 5.1. Let a matrix $A \in \mathbb{Z}_+^{I \times J}$ have the $(h, 0)$-exchange property for every $h \in I$. Then:

(i) For every primal solution λ to the system $A\lambda \geq b, \lambda \geq 0$ there exists another primal solution λ' to P with $\mathbf{1}^\top \lambda' = \mathbf{1}^\top \lambda$.
(ii) Any dual-optimal solution $\pi^* \in D^*$ fulfills $\pi_h^* \geq 0$.

The above result can be used to explain why some (generalized) partitioning problems are equivalent to their covering versions, which is a well-known fact for many problems. A prerequisite is that the coefficient matrix A of the problem has the $(h, 0)$-exchange property for every $h \in I$. Practically speaking, a subset of a feasible structure is again a feasible structure of the same type and with not greater cost (note the unit costs assumption made at the beginning of Section 5.3). The feasibility of subsets is known as the concept of *hereditary* (see, e.g., Pattillo *et al.*, 2013). For CS, e.g., removing an item from a cutting pattern clearly results in another feasible cutting pattern.

Note that for solving these equivalent formulations by CG the covering formulation (with $A\lambda \geq b$) is significantly more stable than the one with equality ($A\lambda = b$). In the latter, the feasible dual space is in $\mathbb{R}^I$, while feasible dual solutions for covering come from the positive quadrant $\mathbb{R}_+^I$. Hence, the dual space is reduced by a factor of 2^m.

Hereditary problems are VP and BP. Additional examples are VC, which is equivalent to partitioning with independent sets and partitioning with cliques in the complement graph, the *edge coloring problem*, and partitioning with matchings. In the context of vehicle routing, most tour minimization problems are hereditary. In all these problems (some are considered and formally introduced later in this article), subsets of packings, independent sets, cliques, and matchings are clearly feasible subsets and, hence, partitioning and covering are equivalent formulations.

For other problems, however, covering and partitioning differ: Covering and partitioning a graph with certain types of subgraphs are different problems if the subgraphs are, e.g., cycles or k-clubs. Here the subgraph formed by a subset of the vertices of a cycle or a k-club is generally not a cycle or k-club (see Pattillo *et al.*, 2013). The same

is true for routing problems with a maximal waiting time constraint, since a subtour of a feasible tour may violate the waiting time constraint.

5.3.3 Row Interchange Properties

We now consider linear programs with only binary coefficients so that $A \in \mathbb{B}^{I \times J}$ and $b = \mathbf{1}$ holds. For these we define a modified version of the exchange property:

Definition 5.2. (Row Replacement Property) Let $h, i \in I$ be given. A binary matrix $A \in \mathbb{B}^{I \times J}$ has the (h,i)-*row replacement property* if $a_j \in A, a_j \geq u_h$ and $a_{ij} = 0$ implies $a_j - u_h + u_i \in A$. In this case, the pair $(h,i) \in I \times I$ is called a *valid replacement* for A.

The analog to Proposition 5.3 for linear problems P whose coefficient matrix A satisfies the row replacement property is:

Proposition 5.4. *Let $E^\top \pi \leq e$ be the set of all inequalities $\pi_i - \pi_h \leq 0$ for valid replacements (h,i) for $A \in \mathbb{B}^{I \times J}$. Then, $E^\top \pi \leq e$ are DDOIs.*

Using Proposition 5.4, we can show that specific classes of DIs for BP, VC, and BPC are DDOIs. We briefly introduce these problems now.

Bin Packing The BP is a restricted CS in which all items $i \in I$ have unit demand $b_i = 1$. Consequently, an item cannot appear more than once in a feasible cutting pattern, called *bin* in BP. Thus, A is a binary matrix ($A \in \mathbb{B}^{I \times J}$) and the CG pricing problem is a *binary knapsack problem* (KP) (see Kellerer *et al.*, 2004). CG algorithms for BP have been suggested by Gilmore and Gomory (1963); Vance *et al.* (1994); Valério de Carvalho (1999). Clearly, for BP, (h,i) is a valid replacement for A if $w_h \geq w_i$.

Vertex Coloring The VC is defined for an undirected graph $G = (I, \mathcal{E})$ with vertex set I and edge set $\mathcal{E}$. VC is the problem of feasibly coloring the vertices with a minimum number of colors such that any two adjacent vertices receive different colors. We denote by $N(i)$ the set of vertices adjacent to vertex $i \in I$. An *independent set* in G is a subset $S \subseteq I$ such that no two vertices of S are adjacent. Clearly, in VC all vertices of the same color form an independent set. CG models for VC were analyzed, e.g., in (Mehrotra and Trick, 1996; Malaguti *et al.*, 2011; Gualandi and Malucelli, 2012; Held *et al.*, 2012), and they are defined as follows: The columns $a_j \in \mathbb{B}^I$ of A are incidence vectors of independent sets, and the task is to properly partition I into independent sets, i.e., the rhs is $b = \mathbf{1}$. The pricing problem in this formulation consist of finding a *maximum-weight independent set* (MWIS).

Bin Packing with Conflicts The BPC is the synthesis of BP and VC: Items $i \in I$ with unit demand $b_i = 1$ and weights w_i have to be packed into a minimum number of bins each with a capacity of L. Moreover a conflict graph $G = (I, \mathcal{E})$ with vertex set I and edge set $\mathcal{E}$ is given, where an edge $\{i,j\}$ indicates that the items i and j are in

conflict. Conflicting items cannot be packed into the same bin. A CG formulation for BPC is analog to the formulations for BP or VC and has been presented by Sadykov and Vanderbeck (2013). Columns can equivalently be seen as *conflict free bins* or *capacity constrained independent sets*.

We refer to a DI of the form $\pi_h \geq \pi_i$ as *pair inequality*. Clearly, the pair inequalities are a special case of the SIs.

Theorem 5.2.

(a) *The pair inequalities* $\{\pi_h \geq \pi_i : w_h \geq w_i\}$ *are DDOIs for BP.*

The pair inequality $\pi_h \geq \pi_i$ *is a DOI for BP, if in addition* $w_h + w_i > L$ *holds.*

The SI $\pi_h \geq \sum_{s \in S} \pi_s$ *is a DOI for BP, if* $w_h \geq \sum_{s \in S} w_s$ *and* $w_h + \min_{s \in S} w_s > L$.

(b) *The pair inequalities* $\{\pi_h \geq \pi_i : N(h) \cup \{h\} \supseteq N(i)\}$ *are DDOIs for VC.*

The pair inequality $\pi_h \geq \pi_i$ *is a DOI for VC, if in addition* $h \in N(i)$ *holds.*

The SI $\pi_h \geq \sum_{s \in S} \pi_s$ *is a DOI for VC, if* $N(h) \cup \{h\} \supseteq N(s)$ *and* $h \in N(s)$ *for all* $s \in S$*, and* S *is an independent set.*

(c) *The pair inequalities* $\{\pi_h \geq \pi_i : w_h \geq w_i, N(h) \cup \{h\} \supseteq N(i)\}$ *are DDOIs for BPC.*

The pair inequality $\pi_h \geq \pi_i$ *is a DOI for BPC, if in addition* $w_h + w_i > L$ *or* $h \in N(i)$ *holds.*

The SI $\pi_h \geq \sum_{s \in S} \pi_s$ *is a DOI for BPC, if* $w_h \geq \sum_{s \in S} w_s$*,* $N(h) \cup \{h\} \supseteq N(s)$ *for all* $s \in S$*,* S *is an independent set, and* $w_h + \min_{s \in S} w_s > L$ *or* $h \in N(s), s \in S$ *holds.*

The above pair inequalities of Theorem 5.2(a) are DOIs for CS and VP, while they are only DDOIs for BP. Note that for BP the *equality constraints* $\pi_h = \pi_i$ for items $h, i \in I$ with $w_h = w_i$ used by Ben Amor *et al.* (2006) are a special case of the DDOIs of Theorem 5.2(a). Equality constraints are analyzed in a more general form in the next Section 5.3.4.

Also note that for BP and CS a linear-sized subset of the DIs of Theorem 5.2 is sufficient to enforce all $\mathcal{O}(m^2)$ pair inequalities: One can assume that all items are sorted by non-decreasing weights so that $w_1 \leq w_2 \leq \cdots \leq w_m$. Then, the $m-1$ *ranking inequalities* $\pi_1 \leq \pi_2 \leq \cdots \leq \pi_m$ imply all pair inequalities (cf. Valério de Carvalho, 2005; Ben Amor *et al.*, 2006, for CS). For VP, VC, and BPC such an ordering of the vertices/items is generally not possible, and no linear system of DIs can impose all pair inequalities.

5.3.4 Constraint Aggregation and Elimination

It is well known for BP that equally-sized items $i_1, \ldots, i_p \in I$ with $w_{i_1} = \cdots = w_{i_p}$ can be aggregated (Vanderbeck, 1999; Ben Amor *et al.*, 2006). It means that the p rows $i_1, \ldots, i_p$ are dropped and a new row, say row $k \in I$, is introduced instead. The result

is a master program, in which the aggregated row has rhs $b_k = p$, and the new entry of a column a_j is the sum $\sum_{\ell=1}^{p} a_{i_\ell,j}$ so that a_{kj} can take the values from $\{0, 1, \ldots, p\}$. Consequently, the pricing problem of an CG approach to BP with aggregation is a *bounded knapsack problem*, see (Kellerer *et al.*, 2004). Note that a similar aggregation is also possible for CS and VP.

In the context of DIs, Ben Amor *et al.* (2006) have characterized this principle of aggregation for BP in more detail. They have shown that the set of *equality constraints* $\pi_h = \pi_i$ for all $h, i \in I$ with $w_h = w_i$ is a set of DDOIs for BP. Furthermore, they proposed two different ways to enforce the equality constraints in a CG algorithm: explicitly adding the corresponding primal columns to the RMP or using constraint aggregation. Their computational results indicated that constraint aggregation is by far superior, which can be attributed to the following facts (see Ben Amor *et al.*, 2006): The size of the RMP decreases significantly in terms of both the number of constraints and the number of feasible packings. The size of the subproblem is also reduced.

With the following proposition we generalize the simple addition of two rows:

Proposition 5.5. *Let $\alpha \in \mathbb{R}$, $h, i \in I$ be two row indices with the primal constraints $a_{h*}\lambda = b_h$ and $a_{i*}\lambda = b_i$ and associated dual variables π_h and π_i, respectively.*

(i) *The following constraints are equivalent:*

$$(a_{h*} + \alpha a_{i*})\tilde{\lambda} = b_h + \alpha b_i \quad \Leftrightarrow \quad \begin{pmatrix} a_{h*} \\ a_{i*} \end{pmatrix} \tilde{\lambda} + \begin{pmatrix} \alpha \\ -1 \end{pmatrix} y = \begin{pmatrix} b_h \\ b_i \end{pmatrix}, \; y \in \mathbb{R}. \quad (5.1)$$

(ii) *Let $\pi^* \in D^*$ be a dual-optimal solution with $\alpha\pi_h^* = \pi_i^*$, which fulfills $E^\top \pi^* \leq e$ for a set of given DDOIs. (This is equivalent to stating that $\alpha\pi_h = \pi_i$ together with $E^\top \pi \leq e$ are DDOIs.)*
Then, P and $\tilde{P}$ are equivalent to an aggregated formulation $\tilde{P}'$ in which the hth and ith equalities are replaced by the lhs of (5.1). The dual solution π'^* defined by

$$\pi_k'^* = \begin{cases} \pi_k^* & \text{for } k \in I \text{ with } k \neq h, i, \\ \pi_h^* & \text{for the new aggregated constraint} \end{cases}$$

is an optimal solution to the dual of $\tilde{P}'$.

Proposition 5.5 gives rise to the following interesting result for instances of CS with no loss.

Remark 5.2. Consider an instance of CS which has a solution without loss. An optimal solution then uses the minimum number of rolls given by $\sum_{i \in I} w_i b_i / L$. Ben Amor *et al.* (2006) have shown that the DIs $\pi_i^* = w_i / L, i \in I$ are a set of DDOIs. Using this information about optimal dual values, the repeated aggregation of rows results in a corresponding LP with a single row, rhs $b' = \sum_{i \in I} b_i w_i / L$, and coefficients $a'_j \leq 1$ for all $j \in J$. Those aggregated columns with entry $a'_j = 1$ correspond to original columns $a_j, j \in J$ that represent a cutting pattern without loss. In the aggregated RMP, exactly one of these columns forms the basic solution. This variable takes the value $\sum_{i \in I} w_i b_i / L$.

The fact that the aggregated model can be solved by finding a cutting pattern without loss seems appealing. However, this result is not useful for computing a primal feasible LP-solution for the original model P. By disaggregation one can find a solution to $\tilde{P}$, where the above DDOIs are generally no WSIs. Thus, the recovery algorithm is not applicable (see Section 5.4.2). The only tool to transform the solution to $\tilde{P}$ with some active DDOIs is running a stabilized CG algorithm as discussed at the end of Section 5.2.

The above example of CS instances without loss is an extreme example in the sense that all constraints can be aggregated into a single constraint. In contrast, if only a partial aggregation but with $\alpha \neq 1$ is possible, one can benefit from Proposition 5.5 and the aggregated formulation because CG can then be applied to the corresponding non-trivial, but smaller instance. The result, the LP bound and the optimal dual values, both for the aggregated formulation and herewith also for the original formulation, may be rather helpful: Faster bounding procedures can be developed and optimal dual values can stabilize the CG algorithm for the original disaggregated model $\tilde{P}$, see again Section 5.2.

We now consider the special case of $\alpha = 0$ in Proposition 5.5. It means that the ith constraint is completely eliminated. The elimination of dominated constraints has been used long since in order to reduce the size of set-covering/set-partitioning instances (e.g., Balinski, 1965; Garfinkel and Nemhauser, 1969; Balas and Padberg, 1976): If for two rows $h, i \in I$ the inequality $a_{ij} \geq a_{hj}$ holds for all $j \in J$, then row i is dominated by row h and can be eliminated. Using this property, it is possible to eliminate specific rows in the CG formulation for VC:

Proposition 5.6. *Consider two vertices $h, i \in I$ in VC. If $N(h) \supseteq N(i)$, row i is dominated by row h and can therefore be eliminated.*

The positive effects of constraint elimination on a CG approach to VC are similar as those of aggregation for BP, CS, and VP. One can solve VC for the graph induced by the non-eliminated vertices, which leads to a row- and column-reduced RMP as well as smaller subproblem instances.

Following Proposition 5.5, the elimination of row i is one possibility to enforce the DI $\pi_i = 0$. Let P' be the primal model resulting from eliminating all dominated rows $I' \subseteq I$ from P. Because $z_{P'} = z_P$, the set of DIs $\pi_i = 0, i \in I'$ is a set of DDOIs (Proposition 5.1). An alternative way of enforcing the DI $\pi_i = 0$ is to explicitly include the corresponding column, i.e., the ith unit vector with cost zero, in the RMP. As for constraint aggregation and equality constraints, constraint elimination is computationally superior to an extended formulation with the DDOIs.

5.4 Separation and Recovery Algorithms

Our approach differs in several aspects from the approaches of Valério de Carvalho (2005) and Ben Amor *et al.* (2006) for stabilizing CG with DOIs or DDOIs. From an application point of view, we cover additional problems (VP, VC, and BPC) and provide general insights on how DOIs and DDOIs may be determined for alternative problems. The focus of this section is, however, on the differing algorithmic parts.

First, because the number of known DOIs and DDOIs is exponential in size, Valério de Carvalho (2005) and Ben Amor *et al.* (2006) suggested using only a linear-sized subset of the SIs with $|S| \leq 2$ when solving CS and BP. The associated primal variables y are here added as additional columns to $\tilde{P}$ before solving the RMP for the first time. We do not limit our approach in this way, but consider exponential classes of DOIs and DDOIs, which makes it necessary to identify violated DIs dynamically in the CG process. Indeed, this is a separation problem, and the first part of this section presents effective separation algorithms for CS, BP, VC, and BPC. The result of separation is the identification of a violated DI for $\tilde{D}$, which is then added as a new column to the primal formulation $\tilde{P}$. In this sense, separation of DIs is also column generation.

Second, we do not restrict our approaches to DIs that have been proven to be DOIs or DDOIs for the problem at hand. Instead, we may perform a kind of over-stabilization by also using classes of DIs that are generally neither DOIs nor DDOIs. As a result, we might end up with an optimal solution to the over-stabilized primal $\tilde{P}$ that cannot be transformed into a feasible solution to P (see Proposition 5.1). In these cases, we apply a recovery procedure to restore primal feasibility. Such a procedure is presented in the second part of this section.

In the following, all classes of DIs under consideration are in fact WSIs so that we present separation algorithms and the recovery algorithm for these. Recall that WSIs and SIs are denoted by $(h \leftarrow t, S)$ and $(h \leftarrow S)$, respectively. Clearly, one should only consider such subclasses of WSIs that are likely to be DDOIs at least for some instances. For pure binary problems such as BP, VC and BPC, we therefore restrict ourselves to SIs $(h \leftarrow S)$.

A WSI $(h \leftarrow t, S)$ or SI $(h \leftarrow S)$ is probably no DDOI if it violates the weight inequality for CS and BP, if the defining set S is no independent set for VC, and both for BPC. Therefore, we require h, t and S to meet the following conditions:

- For CS and VP, the weights inequality $w_h \geq \sum_{s \in S} t_s w_s$ must hold.
- For BP, we use the criterion $w_h \geq \sum_{s \in S} w_s$.
- For VC, h and S need to fulfill $N(h) \cup \{h\} \supseteq \bigcup_{s \in S} N(s)$, and S must constitute an independent set.
- For BPC, we require both of the above conditions of BP and VC.

5.4.1 Dynamic Separation of Violated Dual Inequalities

The careful dynamic generation and addition of violated WSIs prevents the RMP from being stuffed with useless DOIs as it may happen with an a-priori addition of many DIs that one suspects to help stabilizing the CG process. Clearly, mixing both strategies, the a-priori integration of expectedly helpful DIs and the dynamic generation of violated DIs, is a viable strategy that we analyze later in the computational experiments.

For the identification of violated WSIs, a separation procedure is needed. Let $\bar{\pi}_i, i \in I$ be the dual values in a CG iteration after the reoptimization of the RMP. (We need to distinguish between a specific dual solution $\bar{\pi}$ and the dual variables π.) The task of the separation procedure is to identify one or several violated WSIs $(h \leftarrow t, S)$ if any, i.e.,

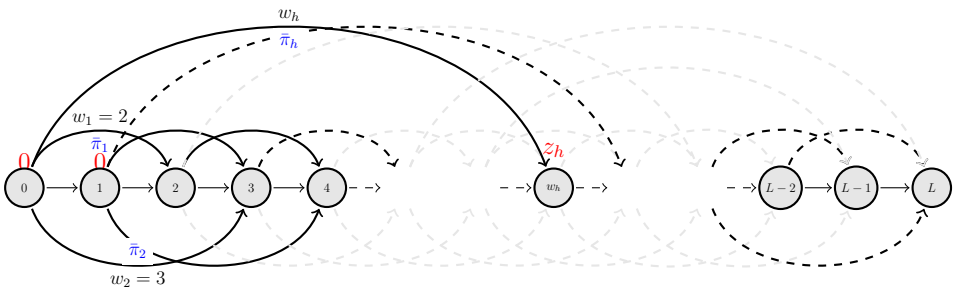


Figure 5.1: Dynamic programming state graph for the unbounded knapsack problem (UKP)

with $\bar{\pi}_h < \sum_{s \in S} t_s \bar{\pi}_s$. In the following, we describe the individual separation algorithms that we later use for CS, BP, VC, and BPC in our computational study.

Cutting Stock For CS, we consider a given item $h \in I$ for which a violated WSI of the form $(h \leftarrow t, S)$ requires the identification of the subset S and the quantities $t_s \in \mathbb{Z}^S_{>0}$. Such a violated WSI exists if and only if $\bar{\pi}_h < z_h$, where $z_h = \max_{i \in I} \bar{\pi}_i t_i$ such that $\sum_{i \in I} w_i t_i \leq w_h$. This is, for each item $h \in I$, an UKP with identical coefficients as in the pricing problem, except for a smaller capacity of w_h instead of L. Note that the WSI $(h \leftarrow t, S)$ with t and $S = \{i \in I : t_i > 0\}$ for the optimal solution to the UKP above is the most violated WSI for this specific item h.

From a worst-case point of view, the best known algorithms for solving the UKP pricing problem are based on dynamic programming (DP), and they require $\mathcal{O}(mL)$ time (see Kellerer *et al.*, 2004). Figure 5.1 shows the state graph, in which states correspond with residual capacities $0, 1, 2, \ldots, L$. For each item $i \in I$, arcs $(p, p + w_i)$ are present for $0 \leq p$ and $p + w_i \leq L$. They all have an associated profit $\bar{\pi}_i$. Additional arcs $(p, p + 1)$ with profit 0 model possible slack in a solution. UKP can now be interpreted as a longest 0-L-path problem in this directed acyclic graph. When solved with DP, the value z_h for $h \in I$ can be found at state w_h. If $\bar{\pi}_h < z_h$, then the associated longest 0-w_i-path defines the subset of selected items $S \subseteq I$ and their quantities $t \in \mathbb{Z}^S_+$. Thus, the exact dynamic separation of violated WSIs is a by-product of solving the pricing problem for CS. The additional effort for identifying a most violated WSI is $\mathcal{O}(m)$ and, therefore, the separation comes almost for free. Note that the approach provides not only a most violated WSI, but several other violated WSIs can also be separated.

For VP, the multi-dimensional extension of CS, the pricing problem is an unbounded DKP (Kellerer *et al.*, 2004, Chapter 9). Finding a most violated WSI requires the solution of m smaller DKPs, which are identical to the pricing problem, but the multi-dimensional capacity $(L_1, \ldots, L_d)$ is set to $(w_{h1}, \ldots, w_{hd})$ for each $h \in I$. The literature reports different exact solution algorithms for DKP (see Ozden, 1988; Fréville, 2004). It seems that none of these approaches allows the direct retrieval of the solution to the WSI separation problem in a straightforward way.

Bin Packing The pricing problem and the SI separation problem for BP are KPs. Again, they again can be solved efficiently in $\mathcal{O}(mL)$ pseudo-polynomial time using dynamic programming (Kellerer *et al.*, 2004, Chapter 5). The state graph however differs from the one of UKP. There is one stage for each item $i \in I = \{1, \dots, m\}$ plus one start stage with the single state 0. At stage $i \in I$, the states $0, 1, 2, \dots, L$ are the possible residual capacities. Stage $i-1$ and stage i for $i \in I$ are connected in the following way: The arcs $(p, p + w_i)$ with profit π_i for $0 \leq p \leq p + \ w_i \leq L$ ($p = 0$ for $i = 1$) model the inclusion of item i in the respective solution, while the arcs (p, p) with profit 0 for $0 \leq p \leq L$ ($p = 0$ for $i = 1$) model the exclusion of item i. Using forward DP, the values at the states w_h at stage m are $z_h = \max_{i \in I} \bar{\pi}_i t_i$ such that $\sum_{i \in I} w_i t_i \leq w_h$ and $t \in \{0, 1\}^I$, i.e., the values of the same KP instance as the pricing problem, but for smaller residual capacities. For each $h \in I$, the corresponding path from 0 to w_h (connecting stages 0 and m) defines a set S of items to include. The respective SI $(h \leftarrow S)$ is violated if and only if $z_h > \bar{\pi}_h$ holds. Summing up, as for CS, if the BP pricing problem is already solved with DP, the additional effort needed to identify violated SIs is $\mathcal{O}(m)$. Thus, separation is a by-product of pricing.

Vertex Coloring For VC, we consider again a fixed vertex $h \in I$. For some $S \subseteq I \setminus \{h\}$, $(h \leftarrow S)$ is a possible SI if S is an independent set and $N(h) \cup \{h\} \supseteq N(s)$ holds for all $s \in S$. Denote by $S_h = \{s \in I : N(h) \cup \{h\} \supseteq N(s)\}$ the set of all vertices that have a subset of the conflicts of vertex h. Then, the SI $(h \leftarrow S)$ is violated if and only if $\bar{\pi}_h < z_h$, where $z_h = \max_{S \subseteq S_h} \sum_{s \in S} \bar{\pi}_s$ such that S is an independent set. This is a MWIS problem (Östergård, 2002). Again, the separation problem is identical to the pricing problem, but defined on the vertex-induced subgraph $G[S_h]$. As such, it is strongly $\mathcal{NP}$-hard and no by-product of solving the pricing problem.

For the separation of violated SIs $(h \leftarrow S)$, we solve a MWIS for each vertex $h \in I$ using the same exact algorithm as for the pricing problem, see Section 5.5. The respective sets S_h for each vertex $h \in I$ are computed once and prior to the CG process. Typically, the sets S_h are very small compared to I in the VC benchmark instances. As a result, separation times are negligible. Analog to CS and BP, the separation procedure provides not only the most violated SI, but also several others.

Bin Packing with Conflicts Since the BPC can be seen as the synthesis of BP and VC, it can be expected that ideas for the separation of DIs can be adapted from these problems. First note that the CG subproblem is a binary *knapsack problem with conflicts* (KPC). For general conflicts, i.e., defined by arbitrary conflict graphs $(I, \mathcal{E})$, the KPC is strongly $\mathcal{NP}$-hard and also practically very hard to solve (Hifi and Michrafy, 2007; Bettinelli *et al.*, 2014). If the conflict graph is of bounded treewidth or is a chordal graph, an efficient DP algorithm with complexity $\mathcal{O}(m^2 L)$ has been proposed by Pferschy and Schauer (2009). The CG literature on BPC focuses on an even more restricted class of conflict graphs, namely interval graphs. Herein, each item $i \in I$ has an associated interval $\mathcal{I}_i := (a_i, b_i)$, and two items $i, j \in I$ are in conflict if and only if $\mathcal{I}_i \cap \mathcal{I}_j \neq \varnothing$. For interval graphs, the KPC subproblem can be solved by dynamic programming in $\mathcal{O}(mL)$

time and space with the algorithm of Sadykov and Vanderbeck (2013). The following computational analysis is likewise restricted to the case of interval conflict graphs.

We briefly sketch the DP algorithm for the KPC proposed by Sadykov and Vanderbeck (2013): The state graph consists of the same $\mathcal{O}(mL)$ states as the one for KP, one for each residual capacity $0, 1, 2, \ldots, L$ and item $i \in I$ plus one start state at stage 0. The crucial difference, however, is that items $i \in I$ need to be sorted according to the rhs b_i of their conflict intervals. Let $pred_i$ denote the a last vertex in $\{0, 1, 2, \ldots, i-1\}$ that is not in conflict with a vertex i (items are sorted in the above order and 0 is an artificial start item not in conflict to any other). Then, the arcs $(p, p + w_i)$ connect stage $pred_i$ to stage i (instead of stages $i-1$ and i as for KP). They represent the selection of the item i and result in the profit $\bar{\pi}_i$. Similar to KP, the arcs (p, p) connect stages $i-1$ and i. They model that item i is not part of the KPC solution with resulting profit of 0. It is easy to see that by construction paths in the state graph correspond to conflict free packings.

As before, for the separation of SIs $(h \leftarrow S)$, we consider the possible items $h \in I$ one by one. The most violated SI $(h \leftarrow S)$ results from the solution of a smaller KPC subproblem, which is defined by the smaller capacity w_h $(\leq L)$ and on the item subset $S_h = \{s \in I : w_h \geq w_s, N(h) \cup \{h\} \supseteq N(s)\}$. This separation problem is *no* by-product of the BPC subproblem in this case. In fact, the DP value for the last item m and the residual capacity w_h results in a solution, which is an independent set S with weight not exceeding w_h. However, the necessary condition $S \subseteq S_h$ does generally not hold. Thus, the DI $(h \leftarrow S)$ does typically not qualify as a possible SI.

Since separation is non-trivial, the following three procedures for identifying violated SIs will be considered:

pairs Pair inequalities can be separated by explicit inspection, which takes $\mathcal{O}(m^2)$ time.

DP Based on the solution of the KPC pricing problem, the following DP-based heuristic can be applied: One inspects all states $w = 1, 2, \ldots, L$ at the final stage m. Each state reached represents a solution, which is an independent set S_w with weight w. For this set S_w, one can find a best item $h \in I$ (if any) with $w_h \geq w$ and $S_h \supseteq S_w$ so that h and S_w induce a SI. Section 5.B in the Appendix explains that the entire procedure can be implemented to run in $\mathcal{O}(mL)$ time.

exact The exact separation of SIs $(h \leftarrow S)$ must solve a smaller KPC for each vertex $h \in I$ with capacity w_h and vertices $s \in S_h$. This requires $\mathcal{O}(m^2 L)$ time, which will turn out to be rather time consuming. Thus, we only solve the DP for an item $h \in I$, for which the LP-bound of the KP with vertices $s \in S_h$ results in a sufficiently high violation compared to the currently most violated SI found. The procedure is still exact, but potentially misses some SIs when separating multiple SIs within certain quality compared to the most violated one.

5.4.2 Over-Stabilization and Recovery of Feasible Primal Solutions

For further stabilizing the CG process, we propose to not only use DIs that are proven to be DOIs or DDOIs. It may then happen that all dual-optimal solutions π^* to D are cut off. From a primal perspective, this means that we have over-stabilized the CG process and that we terminate with a primal solution to $\tilde{P}$ that cannot be transformed into a feasible solution of P with the same cost, see Proposition 5.1. The following example illustrates this behavior.

Example 5.1. Consider an instance of BP with bin capacity $L = 10$ and four items with $w_1 = 5$, $w_2 = 2$, $w_3 = 2$, and $w_4 = 2$. The linear relaxation uses the following four feasible packings $(1,1,1,0)^\top$, $(1,1,0,1)^\top$, $(1,0,1,1)^\top$, and $(0,1,1,1)^\top$ with corresponding dual constraints $\pi_1 + \pi_2 + \pi_3 \leq 1$, $\pi_1 + \pi_2 + \pi_4 \leq 1$, $\pi_1 + \pi_3 + \pi_4 \leq 1$, and $\pi_2 + \pi_3 + \pi_4 \leq 1$, respectively. The unique primal and dual optimal solutions are $\lambda^* = \pi^* = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})^\top$ with $z(P) = z(D) = \frac{4}{3}$.

After adding the SI $\pi_1 \geq \pi_3 + \pi_4$, the optimal solution π^* is cut off, and a new dual-optimal solution is given by $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})^\top$ with $z(\tilde{D}) = 1.25$. Thus, the SI $\pi_1 \geq \pi_3 + \pi_4$ is no DDOI. The corresponding column in $\tilde{P}$ is $(-1,0,1,1)^\top$, and an optimal solution for $\tilde{P}$ is given by $(\tilde{\lambda}^*, y^*) = ((\frac{1}{2}, \frac{1}{2}, \frac{1}{4}, 0)^\top, \frac{1}{4})$.

Transforming $(\tilde{\lambda}^*, y^*)$ into a feasible solution to P means exchanging item 1 by items 3 and 4 in one of the chosen bin columns that include item 1. Doing so, however, results in a new bin column, in which either item 3 or 4 are packed twice, which is infeasible for BP.

Note that for CS, $(\tilde{\lambda}^*, y^*)$ can be transformed into a feasible solution for P because columns are allowed to have non-negative integer coefficients. This also reflects the intuition that WSIs are DOIs for CS and VP, while they are not for BP.

Before we formally state the recovery algorithm, we introduce some additional notation. Recall that any WSI is of the form $(h \leftarrow t, S)$, where $h \in I$ is the row with the -1 entry in the coefficient matrix E and (t, S) are the non-zero entries $t_s > 0$ for $s \in S$. Let K denote the row indices of E. Thus, in the primal model $\tilde{P}$, these indices refer to the variables $y_k, k \in K$ and columns of $E^\top$ with associated WSI $(h \leftarrow t, S)$. For abbreviation, we write $k = k(h \leftarrow t, S)$ in this case. Two indices $j \in J$ and $k \in K$ are said to be *WSI compatible* if $a_j - u_h + t \in A$ for $h \in I$ defined by $k = k(h \leftarrow t, S)$.

The recovery algorithm is formally stated by Algorithm 2. First note that by definition of compatible pairs $(j,k) \in J \times K$ the value of δ is strictly positive in Step 5. Moreover, the assignments in the Step 7 do neither invalidate the primal feasibility $A\lambda^* + E^\top y = b$ nor alter the cost of the solution. This results from the equalities $a_{j'} = a_j - u_h + t$ and $c_j = c_{j'} = 1$. Therefore, the inner loop (Steps 4–10) is successful if an equivalent primal solution with $y^* = 0$ is constructed. Otherwise (Step 8), there must exist an active WSI with row index h for which no compatible $j \in J$ exists. In this case, replacements to row h cannot be recovered. The recovery algorithm has failed in this iteration, and therefore some active WSIs are eliminated from the RMP and their re-introduction is made impossible. As a result, the next RMP solution (Step 2) does not contain any

Algorithm 2: Column generation recovery algorithm

1 **repeat**
2 Compute solution (λ^*, y^*) to $\tilde{P}$ using CG
3 **while** *WSI compatible pairs* $(j,k) \in J \times K$ *with* $\lambda_j^* > 0$ *and* $y_k^* > 0$ *exist* **do**
4 Select a WSI compatible pair $(j,k) \in J \times K$ with $\lambda_j^* > 0$ and $y_k^* > 0$
5 Let $\delta := \min\{\lambda_j^*, y_k^*\}$
6 Let $j' \in J$ such that $a_{j'} = a_j - u_h + t$
7 Let $\lambda_j^* := \lambda_j^* - \delta$, $y_k^* := y_k^* - \delta$, $\lambda_{j'}^* := \lambda_{j'}^* + \delta$
8 **if** $y^* > 0$ **then**
9 Select one (or several) $h \in I$ such that $y_k^* > 0$ for h defined by $k = k(h \leftarrow t, S)$
10 Eliminate all WSIs $(h \leftarrow t', S')$ for $S' \subseteq I, t' \in \mathbb{Z}_{>0}^I$ from the RMP and prohibit their re-generation
11 **until** $y^* = 0$
Result: Solution λ^* to P

WSIs that refer to the row index $h \in I$ selected in Step 9. Thus, after a maximum of m outer loops, the Algorithm 2 must terminate with a solution $(\lambda^*, 0)$ to $\tilde{P}$. Since the recovery procedure is cost preserving, Proposition 5.1(v) is fulfilled so that the modified λ^* is an optimal primal solution to P.

The Steps 4 and 9 leave different strategies for choices of specific pairs (j,k) and h open. For Step 4, the selection of a pair (j,k) leading to a maximal δ may help to minimize the number of necessary iterations of Algorithm 2. For the selection of a row h in Step 9, we observed in our computational test that the number of possible choices is always small so that we always choose all $h \in I$ with $y_k^* > 0$ and $k(h \leftarrow t, S)$.

5.5 Computational Results

In the following, we present computational results for CS, BP, VC, and BPC showing how the stabilization with DIs affects the CG process. Furthermore, we analyze the effects of using different classes of DIs and show that the dynamic generation of DIs is often beneficial. All algorithms described in this paper were implemented in C++ using CPLEX 12.5 as LP solver. The experiments were conducted on two standard PCs, one with an Intel(R) Core(TM) i7-3770 (for BP) and one with an Intel(R) Core(TM) i7-2600 (for CS, VC, and BPC), each running at 3.4 GHz and with 16.0 GB main memory using a single thread only.

5.5.1 Results for Bin Packing

We use the same basic CG algorithm for each of the different CG strategies that we compare. The main features of the CG algorithm are as follows: We use a *best fit decreasing* heuristic (Martello and Toth, 1990) to generate an initial set of columns for warm starting the CG process. The subproblems are solved by the DP algorithm described in Section 5.4.1 and a single best reduced-cost bin column is generated in

each iteration. Pre-tests indicated that adding a single bin column is generally the best performing strategy. If DIs are generated dynamically, multiple SI columns with a violation of at least 25% of the most violated SI are added. In preliminary computational tests, this appeared to be marginally superior compared to values of 50% and 75% and adding a single DI column only.

The first strategy we consider is the standard CG approach without using any DIs referred to by *standard* in the following. In contrast, the strategy denoted by *aggregation* uses constraint aggregation on items with identical weight. This is the approach followed by Ben Amor *et al.* (2006) and can be seen as the state of the art for BP. It is, therefore, used as the baseline strategy for comparisons with all other strategies. Moreover, all following strategies also make use of this aggregation. We also consider using DIs in a static fashion only (*static*), i.e., the addition of a set of expectedly helpful DIs to the RMP as static columns prior to the CG process. Motivated by preliminary computational tests, we use DIs that are similar to what Ben Amor *et al.* (2006) proposed for CS. More precisely, we use the $m-1$ ranking inequalities, which are DDOIs for BP, see Theorem 5.2. Additionally, $\mathcal{O}(m)$ SIs $(h \leftarrow S)$ with $|S| = 2$ are added, namely one for each item $h \in I$. Herein, the set $S = \{i, j\}$ is chosen (if existent) such that the slack in $w_h \geq w_i + w_j$ is minimal (and non-negative). The pure dynamic generation of violated SIs using the separation procedure of Section 5.4.1 is denoted by *dynamic*. Finally, we consider the combination of the latter two, i.e., the use of static DIs and the dynamic generation of additional violated SIs. This last strategy is referred to as *stat+dyn*.

As test instances we used the benchmark sets by Scholl *et al.* (1997) and Sim and Hart (2013) comprising a total of 1,210 and 15,830 instances, respectively. Both benchmark sets are subdivided into several subclasses differing with respect to the number of items, the capacity, and the variance of the item weights resulting in instances with very different characteristics and degrees of complexity. In the following, we present results averaged over all subclasses. However, we include only those instances, for which the computation time of *aggregation* is at least one second. This reduces the number of benchmark instances to 207 and 4,032, respectively. The other instances are solved in too little time with the stronger algorithmic strategies. Hence, the consideration of these instances would, if they were taken into account, produce unreliable values for the computation times.

We further differentiate two subgroups for each of the benchmark sets: Groups 1 comprise the instances with computation times between one and ten seconds for *aggregation*, and it consist of 157 and 3,352 instances for the sets of Scholl *et al.* (1997) and Sim and Hart (2013), respectively. The hardest instances in terms of solution time (> 10 seconds for *aggregation*) are in groups 2 consisting of the remaining 50 and 680 instances.

Table 5.2 summarizes the results for BP, where the table columns have the following meaning:

m	The number of rows in the RMP relative to *aggregation*
Solution time	The time for solving the linear relaxation of the RMP relative to *aggregation*; we present average, maximum, and minimum values over the instances

# Iterations	The number of iterations relative to *aggregation*; we present average, maximum, and minimum values over the instances
SP/RMP	The ratio of the times spent for solving the subproblem and re-optimizing the RMP, averaged over the instances.
# Dynamic DIs	The number of DIs that are generated dynamically during the CG process; we present average, maximum, and minimum values over the instances
Recovery	The total number of instances for which a recovery is needed (# inst.) and the maximum number of iterations of the recovering algorithm for one of the instances (max it.).

Table 5.2 shows that the dynamic generation of DIs results in an average speedup of approximately factor two compared to *aggregation*. For the hard group 2 instances, the average speedup even reaches factor three to four. Thereby, *stat+dyn* is slightly faster than the pure use of dynamic DIs. There are also instances for which *stat+dyn* and *dynamic* perform worse than *aggregation* in terms of computations times. However, computation times never go beyond 139% and 130% of *aggregation* for *dynamic* and *stat+dyn*, respectively. The biggest speedups on the contrary exceed factor 15, while *standard* CG is at least one order on magnitude slower than all strategies applying *aggregation*.

The static use of DIs results only in a small speedup of approximately $2-3\%$ on average. Recall that ranking inequalities were not yet known as DDOIs in the work of Ben Amor *et al.* (2006). Therefore, a static DI-based CG algorithm was not considered for BP at that time.

Regarding the number of iterations, the differences between the strategies using dynamic DIs and *aggregation* are even more pronounced than for the computations times. Also *static* performs considerably better as it requires only about two thirds of the iterations of *aggregation*.

Another interesting result is that a recovery is necessary only for a very small number of instances. Even more, the maximum number of iterations of the recovery algorithm is two (cf. Algorithm 2). This demonstrates that the SIs we add to the RMP (static or dynamic) are indeed DDOIs for most of the instances in the benchmark.

Table 5.2 also indicates that there are instances, for which no dynamic DIs are generated when static DIs are added a priori. These are instances in which the item sizes are very similar so that no three items $h, i, j \in I$ with $w_h \geq w_i + w_j$ exist. It is therefore clear that no SIs $(h \leftarrow S)$ with $|S| \geq 2$ exist in these cases so that no additional speedup can be achieved. Our results include a total of 1,427 and 39 such instances for the sets of Sim and Hart and Scholl *et al.*, respectively. In summary, the overall average speedup for the instances, in which SIs with $|S| \geq 2$ exist, actually exceeds factor two significantly.

In Figure 5.2, we display more generally the influence of the number of possible SIs on the performance of the different CG strategies. The ratio $R_{BP} = \max_{i \in I} w_i / \min_{i \in I} w_i$ serves as an approximate measure of how many SIs potentially exist. Clearly, this ratio reveals no information on the distribution of the items weights. In fact, the number of SIs is generally not computable, but R_{BP} can be determined easily for each BP instance. Note that a ratio $R_{BP} < 2$ reflects the case described above, in which no SI with $|S| \geq 2$

Instances	Algorithm	m	Solution time (avg./max/min)	# Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)	Recovery (# inst./max it.)
Sim and Hart All instances ($n = 4{,}032$)	*standard*	3.1	5.00/31.46/0.84	3.45/20.47/0.96	5.8	–	–/–
	aggregation	1	1.00/1.00/1.00	1.00/1.00/1.00	31.7	–	–/–
	static	1	0.97/1.39/0.67	0.66/1.02/0.47	44.3	–	7/1
	dynamic	1	0.54/1.39/0.07	0.43/1.11/0.08	21.3	1462/9852/26	54/2
	stat+dyn	1	0.52/1.30/0.06	0.36/1.02/0.03	26.9	864/7351/0	54/2
Sim and Hart Group 1 ($n = 3{,}352$)	*standard*	3.3	5.35/31.45/0.87	3.73/20.47/0.96	2.9	–	–/–
	aggregation	1	1.00/1.00/1.00	1.00/1.00/1.00	30.1	–	–/–
	static	1	0.96/1.35/0.67	0.65/1.02/0.47	41.7	–	7/1
	dynamic	1	0.57/1.39/0.09	0.45/1.11/0.10	20.1	1007/7029/26	37/2
	stat+dyn	1	0.55/1.30/0.08	0.39/1.02/0.04	23.2	529/5531/0	39/2
Sim and Hart Group 2 ($n = 680$)	*standard*	2.0	3.30/8.55/0.84	2.06/3.70/1.05	20.5	–	–/–
	aggregation	1	1.00/1.00/1.00	1.00/1.00/1.00	38.9	–	–/–
	static	1	1.03/1.39/0.79	0.72/0.87/0.60	56.0	–	0/0
	dynamic	1	0.36/1.21/0.07	0.30/0.81/0.08	26.7	3074/9852/669	17/2
	stat+dyn	1	0.35/1.20/0.06	0.24/0.81/0.03	43.5	2520/7351/0	15/2
Scholl *et al.* All instances ($n = 207$)	*standard*	3.0	5.22/28.78/0.91	3.48/20.26/0.96	4.9	–	–/–
	aggregation	1	1.00/1.00/1.00	1.00/1.00/1.00	29.6	–	–/–
	static	1	0.98/1.22/0.76	0.66/1.00/0.47	41.3	–	1/1
	dynamic	1	0.41/1.07/0.08	0.35/1.03/0.10	15.9	2252/11567/35	6/2
	stat+dyn	1	0.39/1.12/0.07	0.26/1.00/0.04	21.4	1465/7270/0	2/1
Scholl *et al.* Group 1 ($n = 157$)	*standard*	3.3	5.83/28.77/1.08	3.96/20.26/1.06	2.7	–	–/–
	aggregation	1	1.00/1.00/1.00	1.00/1.00/1.00	32.1	–	–/–
	static	1	0.97/1.19/0.76	0.65/1.00/0.47	42.8	–	1/1
	dynamic	1	0.45/1.07/0.13	0.38/1.03/0.14	17.1	1372/4472/35	4/2
	stat+dyn	1	0.43/1.12/0.09	0.29/1.00/0.07	20.9	871/3443/0	2/1
Scholl *et al.* Group 2 ($n = 50$)	*standard*	1.9	3.29/8.11/0.91	1.99/3.57/0.96	11.8	–	–/–
	aggregation	1	1.00/1.00/1.00	1.00/1.00/1.00	22.4	–	–/–
	static	1	0.99/1.22/0.86	0.68/0.79/0.55	37.1	–	0/0
	dynamic	1	0.29/1.07/0.08	0.24/0.70/0.10	12.4	5015/11567/936	2/1
	stat+dyn	1	0.27/1.08/0.07	0.17/0.70/0.41	22.8	3329/7270/0	0/0

Table 5.2: Computational results for the bin packing problem (BP)

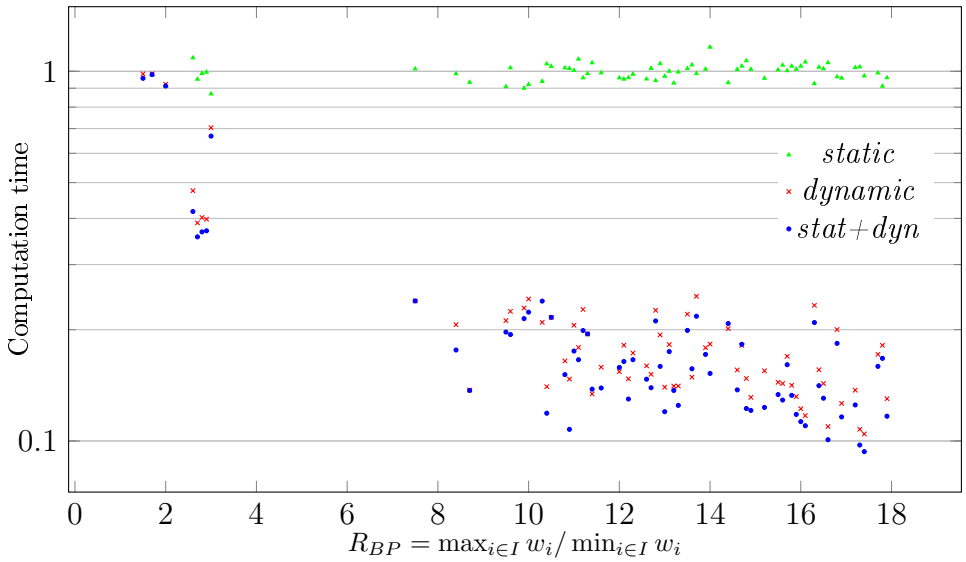


Figure 5.2: Average computations times relative to *aggregation* (ordinate) depending on the ratio $\max_{i \in I} w_i / \min_{i \in I} w_i$ (abscissae) for all considered Sim and Hart instances for the bin packing problem (BP)

exist. The relationship between the ratio R_{BP} and the speedups using DIs is depicted in Figure 5.2: It shows the average computation times of *static*, *dynamic*, and *stat+dyn* relative to *aggregation* for all 4,032 considered Sim and Hart instances depending on R_{BP}. Note that the abscissae groups instances according to R_{BP} (rounded to one decimal) and that the ordinate shows the relative computation times in logarithmic scale. Figure 5.2 clearly demonstrates that the dynamic generation of SIs, i.e., implicitly considering the set of all SIs, works better the bigger the ratio R_{BP}. One can therefore suspect that there is a highly significant positive correlation between the number of SIs and the resulting acceleration of the CG process. For larger values of R_{BP}, average speedups reach factors between five and ten.

5.5.2 Results for Cutting Stock

For CS, the basic CG algorithm and the different stabilization strategies are similar to BP. We only describe the differences. The UKP subproblems are solved with the DP approach of Section 5.4.1. For CS, the basic CG algorithms already makes use of constraint aggregation. Thus, the strategies *standard* and *aggregation* coincide, and results are solely reported as *standard*. Strategy *static* as proposed by Ben Amor *et al.* (2006) is the state of the art for CS. Therefore, *static* is the baseline strategy meaning that results for all other strategies are presented relative to it. Recall that the WSIs are DOIs for CS and no recovery is needed, see Theorem 5.1(i).

Generally, the BP benchmark instances of the previous section can also be interpreted as CS instances, in which items of identical length need to be aggregated. In preliminary experiments, we found that the UKP subproblems of CS are solved significantly faster

than the (binary) KP subproblems of BP for all the benchmark instances. We suspect that this is due to the smaller state space of the DP of the UKP compared to the one of KP. Recall that the state graph of the UKP (depicted in Figure 5.1) has only $L+1$ states, while the state graph of KP has $1+(L+1)m = \mathcal{O}(mL)$ states. We would like to stress that for both UKP and KP we implemented state-of-the-art DP algorithms: In particular, for KP we do not explicitly build the $\mathcal{O}(mL)$ states, but use a list-based implementation as discussed in (Kellerer *et al.*, 2004, Sect. 3.4). In contrast, for UKP we found that a straightforward array-based implementation of the DP approach is faster than the list-based approach. We suspect that on a modern CPU, the smaller state graph of UKP can be accessed much faster (due to caching techniques) so that the solution of the UKP subproblems as they occur in the BP benchmark instances is possible in almost no (measurable) time.

As a result, of the 17,040 benchmark instances from the sets of Scholl *et al.* and Sim and Hart over 99% are solved by *standard* CG in less than 1 second of computation time, most of them in only a few milliseconds, meaning that a large number of UKP can be solved in this short time. Therefore, we considered these CS instances of limited interest and generated new and harder CS instances. These are characterized by huge values for the capacity (in order to complicate the subproblems) and larger numbers of items with distinct lengths. The instances we generated can be grouped into different subclasses of problems. Capacities take values $L \in \{500{,}000, 1{,}500{,}000\}$ and integer item lengths are uniformly drawn from $[2/15\,L, 2/3\,L]$ and $[1/150\,L, 2/3\,L]$ resulting in items for which the ratios w_i/L are analog to those in the benchmark instances used by Ben Amor *et al.* (2006). The number of items with distinct item length takes values $m = \{125, 250, 500\}$. Demands are uniformly distributed in the range $[1, 20]$. For each subclass, 20 instances were generated. The entire benchmark set comprises 240 instances and is available at `http://logistik.bwl.uni-mainz.de/Dateien/CS.zip`.

Table 5.3 presents our results for CS. First and foremost, the overall speedups of using stabilized CG are significantly smaller for CS compared to BP. The *static* use of DIs results in an average speedup of about one third. The additional use of dynamically generated DI saves an extra 17% of the computation time. The pure *dynamic* generation even results in a slightly slower overall algorithm than the *static* version.

The results regarding the number of iterations are more in favor of using stabilized CG. Iterations are on average reduced by a larger factor than the computation times. The results by subclasses are similar to the overall results as can be seen in Section 5.C of the Appendix.

Instances	Algorithm	Solution time (avg./max/min)	# Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)
All instances ($n = 240$)	*standard*	1.50/8.73/0.93	1.77/29.00/1.31	17.6	–
	static	1.00/1.00/1.00	1.00/1.00/1.00	13.7	–
	dynamic	1.03/4.80/0.44	0.86/9.00/0.41	6.5	5590/18100/247
	stat+dyn	0.83/1.53/0.29	0.75/1.00/0.31	9.8	1120/3742/0

Table 5.3: Computational results for the cutting stock problem (CS)

5.5.3 Results for Vertex Coloring

Analog to BP and CS, we use one basic CG algorithm for our experiments and run it with different strategies regarding the stabilization with DIs. To warm start the CG process, a simple greedy coloring is performed. The MWIS pricing problem is solved using the algorithm and its implementation by Held *et al.* (2012). In each iteration, a single best reduced-cost column is added to the RMP. If dynamic generation of DIs is used, violated SIs are separated using the procedure described in Section 5.4.1 and multiple SI columns with a violation of at least 25% of the most violated SI are generated.

Standard denotes the basic CG algorithm without stabilization. It serves as the baseline algorithm for the other strategies. The algorithm using constraint elimination is denoted by *elimination*, and all following strategies also make use of constraint elimination. For an approach that uses DIs in a static fashion, recall that no linear system of DIs can impose all pair inequalities, see Section 5.3.3. Still, we suggest a strategy that uses a linear-sized subset of the pair inequalities similar to the ranking inequalities in CS and BP. For each vertex $h \in I$, we determine a vertex $i = \text{argmax}_{s \in S_h} |N(s)|$. This is the vertex in the candidate set S_h that has the most conflicts, and we add the corresponding SI column to initialize the RMP. The intuition is that this vertex i is the hardest one to cover of all the vertices that are easier to cover than h. Therefore, the repetition of this SI selection for each vertex $h \in I$ should result in a near ranking of all the duals. We refer to this approach as *static*, and no SIs with $|S| > 1$ are used in this case. The strategies *dynamic* and *stat+dyn* are analog to CS and BP.

Note that because constraint elimination is performed in each strategy that uses DIs, the candidate sets S_h for all vertices $h \in I$ comprise only vertices $s \in S_h$ for which $h \in N(s)$ holds. Consequently, by Theorem 5.2(b) all SIs generated by our separation procedure are in fact DOIs and no recovery is needed.

As benchmark problems, we used the instances from the graph coloring benchmark web page `https://sites.google.com/site/graphcoloring/home` of Gualandi and Chiarandini (2014). The entire benchmark comprises 136 instances. We restrict our analysis to those instances in which at least one possible SI exists. This eliminates approximately half of the instances so that 74 VC instances remain. Furthermore, we exclude from our computational analysis those instances, for which the linear relaxation of the RMP cannot be solved within a time limit of one hour using all strategies (15 instances), and those instances, for which the solution time is less than 0.1 seconds for the *standard* strategy (34 instances). This leaves 25 instances for the test set which we further subdivided into two groups: Group 1 comprises all instances for which the ratio $R_{VC} = (\sum_{h \in I} |S_h|)/m^2 > 0.001$, while the remaining instances form group 2. R_{VC} measures the number of possible replacements of items relative to the instance size, thus, giving an approximation on the number of possible SIs relative to the instances size. One can expect that (dynamic) stabilization generally performs better for group 1 than for group 2.

The results of the computational tests are summarized in Table 5.4. The additional column *Poss. repl. %* shows the ratio R_{VC} in percent. A general observation is that with stabilized CG an average speedup of approximately factor two can be achieved

Instances	Algorithm	m	Poss. repl. %	Solution time (avg./max/min)	# Iterations (avg./max/min)	SP / RMP	# Dynamic DIs (avg./max/min)
All instances ($n = 25$)	*standard*	1	0	1.00/1.00/1.00	1.00/1.00/1.00	29.8	–
	elimination	0.80	0	0.88/1.16/0.35	0.93/1.12/0.52	33.0	–
	static	0.80	0.64	0.57/1.17/0.06	0.71/1.01/0.15	13.8	–
	dynamic	0.80	0.64	0.54/0.96/0.03	0.72/1.04/0.08	11.5	349/5314/0
	stat+dyn	0.80	0.64	0.53/0.91/0.06	0.70/0.99/0.08	12.0	121/2015/0
Group 1 ($n = 11$)	*standard*	1	0	1.00/1.00/1.00	1.00/1.00/1.00	59.7	–
	elimination	0.96	0	1.01/1.16/0.90	1.00/1.12/0.88	62.0	–
	static	0.96	1.45	0.35/0.78/0.06	0.60/0.90/0.15	17.5	–
	dynamic	0.96	1.45	0.33/0.71/0.04	0.57/0.92/0.08	13.9	765/5314/26
	stat+dyn	0.96	1.45	0.31/0.70/0.06	0.55/0.90/0.08	15.1	270/2015/0
Group 2 ($n = 14$)	*standard*	1	0	1.00/1.00/1.00	1.00/1.00/1.00	6.3	–
	elimination	0.67	0	0.78/1.04/0.35	0.87/1.03/0.52	10.3	–
	static	0.67	0.01	0.74/1.17/0.27	0.81/1.01/0.43	10.8	–
	dynamic	0.67	0.01	0.71/0.96/0.40	0.81/1.04/0.45	9.6	23/288/0
	stat+dyn	0.67	0.01	0.70/0.91/0.42	0.81/0.99/0.43	9.6	4/52/0

Table 5.4: Computational results for the vertex coloring problem (VC)

compared to *standard*. Thereby, all three strategies using DIs perform comparably well, although the biggest gains can be achieved with *stat+dyn*. For larger values of R_{VC}, stabilization with SIs is even more attractive and the average speedup reaches factor three for group 2. Table 5.4 also reveals that stabilization comes almost for free, since the maximum relative computation time for *static* is only 17% longer than *standard* and the strategies with dynamic separation of SIs never take more computation time than *standard*. On the other hand, the biggest speedups exceed factors 15, 30, and 15 for *static*, *dynamic*, and *stat+dyn*, respectively. Expectedly, *elimination* works better the more vertices can be eliminated. The insights regarding the number of iterations are evident: stabilization by DIs can help to reduce the number of CG iterations and is rarely detrimental.

5.5.4 Results for Bin Packing with Conflicts

Analog to Sections 5.5.1–5.5.3, we run one basic CG algorithm for BPC with different stabilization strategies to analyze their impact on the CG approach. An initial set of columns to warm start the process is obtained by performing a *first fit decreasing* heuristic several times with different orderings of the items. The KPC subproblems, for which the conflicts are defined on interval graphs in all benchmark instances, are solved with the DP algorithm of Sadykov and Vanderbeck (2013) described in Section 5.4.1. Again, a single best reduced-cost bin and multiple SI columns with a minimum violation of 25% of the most violated SI are generated in each iteration.

Standard CG without stabilization serves as the baseline strategy for comparisons with the other strategies. Similar to VC, no linear system of DIs can impose all pair inequalities, see Section 5.3.3. For an approach that uses DIs in a *static* fashion, we therefore follow a strategy using the same basic idea as for VC: For each item $h \in I$ we identify the item $i \in S_h$ (if any) that is the hardest one to cover of all the vertices that are easier to cover than h and add the respective pair inequality to the RMP. For BPC, however, the hardness to cover an item i depends on both its weight w_i and its conflicts $N(i)$. Therefore, as item i we choose one that maximizes the sum of w_i/w_h and $|N(i)|/|N(h)|$. SIs with $|S| > 1$ are not used in this *static* approach. Strategies *dynamic* and *stat.dyn* are analog to the other problems. Both strategies are performed with the different separation procedures for the SIs as described in Section 5.4.1. The exact separation of pair inequalities by inspection is indicated by the suffix *.pairs*, while the suffixes *.DP* and *.exact* refer to the DP-based and exact separation of SIs, respectively.

As test problems, we used the benchmark instances of Fernandes Muritiba *et al.* (2010). The complete set comprises 800 instances with differing characteristics regarding the number of items, the capacity, the item lengths, and the number of conflicts of the items. Similar to BP and VC, we include in our computational analysis only those 169 instances, for which the solution of the RMP took more than one second of computation time in the *standard* CG. We also distinguish two subgroups of instances. Group 1 comprises instances from the classes 1–4 of the benchmark set, while group 2 comprises instances from classes 5–8. The latter are instances for which the item weights are such that no three items i, j, h with $w_h \geq w_i + w_j$ exist, i.e., no SIs with $|S| > 1$ are

possible. Furthermore, we report separate results for the hardest instances in terms of computation time. This includes 49 instances for which the solution time of *standard* CG was more than 10 seconds.

Table 5.5 summarizes the results for BPC. It reveals that a significant improvement can be achieved by using stabilized CG. The biggest gain is already obtained from the *static* use of pair inequalities leading to a speedup of almost factor four. The additional, dynamic separation of violated SIs further decreases computation times resulting in an average speedup of almost factor five over all considered instances. The pure *dynamic* generation of DIs is inferior to the other stabilization strategies. Still, a speedup of approximately factor three compared to *standard* CG can be achieved. For the hard instances, stabilized CG is even more beneficial and leads to an average speedup exceeding factor eight for *stat+dyn.DP*. The results for group 1 are also more in favor for the stabilization strategies than those over all instances. There, average speedups exceed factors between four and five for *static* and *stat+dyn*, respectively. For group 2, speedups reach factors three and four, respectively. Also, Table 5.5 indicates that for group 2 the separation of pair inequalities and the DP-based heuristic dynamically generate exactly the same number of DIs. This can be explained by the fact that no SIs other than pair inequalities exist for these instances.

Regarding the different separation strategies for the dynamic separation of violated DIs, the separation of pair inequalities by inspection and the separation with the DP-based heuristic perform comparably well. While the former is slightly better averaged over all considered instances as well as for subgroups 1 and 2, the additional effort for the separation in the latter pays off for the hard instances resulting in a slightly better performance. Also, it is obvious from Table 5.5 that the exact separation of violated SIs is too time consuming for BPC and does not pay off.

As for the other problems, the results regarding the number of iterations are similar to those for the computation times: Stabilized CG results in a significant reduction of the iterations needed for the optimization of the RMP. Moreover, the dynamic addition of SIs yields an additional, considerable gain compared to using DIs in a static fashion only. Table 5.5 also reveals that the exact separation of violated SIs does not result in a decrease in the number of iterations compared to the other separation strategies. This can be explained by the use of the KP bound (see Section 5.4.1) in the separation and the potential miss of some SIs with a violation of at least 25% of the most violated SI. Indeed, preliminary tests showed that without using the KP bound the exact separation of violated DIs leads to a slight decrease in the number of iterations relative to the other separation strategies. However, computation times increase disproportionately.

Overall, stabilization with DIs works well for our CG approach to BPC. While speedups can reach a factor of 50, the most successful strategies *static*, *stat+dyn.pairs*, and *stat+dyn.DP* are never slower than *standard* CG. In contrast, the worst performance of these strategies still results in a speedup of almost factor two. Finally, Table 5.5 indicates that all SIs that we generated in the test problems were DDOIs, since a recovery was never necessary for any of the instances and strategies.

Instances	Algorithm	Solution time (avg./max/min)	# Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)	Recovery (# inst./max it.)
All instances ($n = 169$)	*standard*	1.00/1.00/1.00	1.00/1.00/1.00	0.2	–	–/–
	static	0.27/0.58/0.03	0.37/0.73/0.06	0.5	–	–/–
	dyn.pairs	0.35/2.71/0.07	0.18/0.59/0.02	0.2	12281/162521/328	–/–
	dyn.DP	0.35/1.33/0.08	0.16/0.59/0.01	0.3	16715/227607/1317	0/0
	dyn.exact	3.11/12.13/0.48	0.20/0.60/0.05	13.5	2968/15235/494	0/0
	stat+dyn.pairs	0.21/0.63/0.02	0.17/0.55/0.02	0.4	3294/23661/35	–/–
	stat+dyn.DP	0.22/0.54/0.02	0.15/0.55/0.01	0.4	7201/60899/342	0/0
	stat+dyn.exact	2.83/11.25/0.18	0.16/0.58/0.03	22.6	1991/15919/272	0/0
Group 1 ($n = 92$)	*standard*	1.00/1.00/1.00	1.00/1.00/1.00	0.1	–	–/–
	static	0.24/0.48/0.15	0.34/0.64/0.20	0.1	–	–/–
	dyn.pairs	0.35/2.71/0.12	0.13/0.58/0.02	0.1	15741/162521/328	–/–
	dyn.DP	0.33/1.33/0.15	0.08/0.58/0.01	0.1	23886/227607/1317	0/0
	dyn.exact	1.18/6.32/0.48	0.13/0.50/0.060	1.7	3701/15235/494	0/0
	stat+dyn.pairs	0.17/0.63/0.07	0.12/0.53/0.02	0.2	3738/23661/35	–/–
	stat+dyn.DP	0.18/0.53/0.07	0.08/0.47/0.01	0.2	10914/60899/342	0/0
	stat+dyn.exact	0.72/5.60/0.18	0.08/0.50/0.03	3.9	2946/15919/281	0/0
Group 2 ($n = 77$)	*standard*	1.00/1.00/1.00	1.00/1.00/1.00	0.5	–	–/–
	static	0.31/0.58/0.03	0.42/0.73/0.06	0.9	–	–/–
	dyn.pairs	0.34/0.73/0.07	0.25/0.59/0.03	0.4	8147/27207/1517	–/–
	dyn.DP	0.37/0.79/0.08	0.25/0.59/0.03	0.5	8147/27207/1517	0/0
	dyn.exact	5.43/12.13/0.87	0.29/0.60/0.05	27.6	2092/4452/663	0/0
	stat+dyn.pairs	0.25/0.56/0.02	0.24/0.55/0.02	0.6	2763/10439/589	–/–
	stat+dyn.DP	0.26/0.54/0.02	0.24/0.55/0.02	0.7	2763/10439/589	0/0
	stat+dyn.exact	5.35/11.25/0.41	0.25/0.58/0.03	45.1	850/2496/272	0/0
Hard instances ($n = 49$)	*standard*	1.00/1.00/1.00	1.00/1.00/1.00	0.1	–	–/–
	static	0.18/0.38/0.03	0.27/0.56/0.06	0.2	–	–/–
	dyn.pairs	0.31/2.71/0.07	0.07/0.42/0.02	0.1	27629/162521/1824	–/–
	dyn.DP	0.25/1.33/0.08	0.05/0.36/0.01	0.1	37699/227607/2852	0/0
	dyn.exact	1.03/2.97/0.48	0.10/0.29/0.05	8.8	5255/15235/1629	0/0
	stat+dyn.pairs	0.13/0.63/0.02	0.06/0.41/0.02	0.2	6322/23661/975	–/–
	stat+dyn.DP	0.12/0.32/0.02	0.04/0.37/0.01	0.2	15838/60899/1497	0/0
	stat+dyn.exact	0.56/2.71/0.18	0.06/0.27/0.03	11.6	3949/15919/416	0/0

Table 5.5: Computational results for the bin packing problem with conflicts (BPC)

5.6 Conclusions

Ben Amor *et al.* (2006) introduced DOIs and DDOIs as a general concept to accelerate the CG process, which was then tested for two well structured problems, CS and BP. As they state, the restriction of the dual space by DIs leads to less possible intermediate values for the dual multipliers so that a dual-optimal solution is computed faster. In this paper, we extend their results in theory, applications, algorithms, and computational results.

On the theoretical side, we generalize the characterization of a system of DDOIs as stated in Proposition 5.1. Moreover, instead of deriving properties of dual-optimal solutions directly from the problem under consideration, we introduce several new properties referring to the coefficient matrix of the underlying primal problem. The new comprehensive class of the so-called weighted subset inequalities (WSI) is shown to be DOIs for any problem whose coefficient matrix fulfills the exchange property. A direct consequence is that WSI also hold for VP, and they generalize the subset inequalities prior known to be DOIs for CS. With the help of the row replacement property, the pair inequalities are proven to form a new class of DDOIs for BP.

On the application side, the same pair inequalities are also valid for VC and BPC. For these two problems, DIs have not been tested before (to the best of our knowledge). Furthermore, DOIs and DDOIs can also be applied to problems with general cost-minimization objective, which is only partly covered in the paper at hand (due to the unit-cost assumption made starting from Section 5.3). In a companion paper (Gschwind and Irnich, 2014), we show that the CG algorithm of Caprara *et al.* (2013) for the temporal knapsack problem benefits from the integration of DOIs. Recently, we discussed and agree with Bianchessi *et al.* (2014) that the split commodities mixed routing problem is an excellent candidate to apply DIs for a vehicle routing problem. We suspect that an observable acceleration of the branch-and-price algorithm can be gained.

We see the most important algorithmic innovation in the *dynamic* generation of DIs. This strategy can either replace or complement the integration of DIs in the form of additional primal columns into the initial RMP. Several exact and heuristic algorithms for the dynamic generation (=separation) of DIs have been presented. The additional effort for the implementation of separation procedures is small, as shown for CS, BP, and VC. In fact, the most violated WSI in CS and BP are an algorithmic by-product, since they result from intermediate solutions that must anyway be computed when the knapsack-type pricing problem is solved by DP (the currently best known approach in this case). Even some tailored separation heuristics like the ones proposed for BPC are relatively simple to implement.

The possible over-stabilization of the CG process with potential DDOIs is another new idea first coined and analyzed in this paper. The idea goes hand in hand with the dynamic generation of DIs because WSIs form an exponential class of DIs and can therefore not entirely be added to the initial RMP. On the downside, the dynamic addition of non-DDOIs, i.e., DIs that may cut off the entire dual-optimal polyhedron, requires the use of a recovery algorithm as the one presented in Section 5.4.2. The

required iterative application of the recovery algorithm may at the end lead to many additional CG iterations. Fortunately, this never happened in our computational studies. It is indeed reverse, only in rare cases more than one recovery was needed, most of the time none was required at all. Overall, over-stabilization turns out to be advantageous on average for BP, VC, and BPC. For BP, we show that finally much larger speedups can result when DIs are added dynamically.

The extensive computational tests on many known and some additional harder benchmark problems allow some clear statements: For all problems analyzed (CS, BP, VC, and BPC), the newly identified and potential DDOIs together reduce the number of CG iterations when added before proving optimality of the RMP, sometimes significantly. Since the LP-reoptimization of the RMP becomes slightly more time consuming with additional DI columns, the overall reduction in computation time is generally smaller than the reduction in CG iterations. However, for all studied problems the average computation times are reduced compared to state-of-the-art CG algorithms. For the BP example, we compare with an implementation already using constraint aggregation as suggested by Ben Amor *et al.* (2006). Here we are able to confirm their substantial and impressive speedups of factors between two and three for many BP instances. With a mix of statically and dynamically added DIs together with over-stabilization we gain another factor of approx. two for the extensive benchmark set of Sim and Hart (2013) and a factor of 2.5 for the widely used benchmark by Scholl *et al.* (1997). Notably, while for this strategy (*stat+dyn*) the maximum slowdown was by factor 1.3, the maximum speedup was more than factor ten on some instances. Thus, the use of DIs seldom and only gently hampers the CG process, but very often it accelerates. For future applications we expect the dynamic addition of DIs to work best for those problems, in which the solution of the pricing almost completely occupies the overall computation time: any reduction in the number of CG iterations should linearly contribute to a reduction in the solution time.

The results presented in this paper all refer to the solution of the linear relaxation of the extensive formulation. Clearly, the same techniques should also be tested when CG is used in branch-and-price. Such a comparison is, however, a non-trivial task: The overall computation times very much depend on the branching scheme applied, which due to degeneracy of the primal solution may make completely different decisions for the alternative DI strategies, leading to actually incomparable search trees. Even more severe is the fact that some branching decisions are incompatible with some DIs, see, e.g., the branch-and-price algorithm of Alves and Valério de Carvalho (2008) for the multiple length CS. Generic strategies to integrate DIs and branching constitute an interesting avenue for future research.

References

Alves, C. and Valério de Carvalho, J. M. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, **35**(4), 1315–1328.

Alves, C., Valério de Carvalho, J., Clautiaux, F., and Rietz, J. (2014). Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, **233**(1), 43–63.

Balas, E. and Padberg, M. W. (1976). Set partitioning: A survey. *SIAM Review*, **18**(4), 710–760.

Balinski, M. (1965). Integer programming: Methods, uses, computation. *Management Science*, **12**(3), 253–313.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Ben Amor, H. and Valério de Carvalho, J. (2005). Cutting stock problems. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Springer, New York, NY.

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.

Bettinelli, A., Cacchiani, V., and Malaguti, E. (2014). Bounds and algorithms for the knapsack problem with conflict graph. Technical Report OR-14-16, DEIS – University of Bologna, Bologna, Italy.

Bianchessi, N., Archetti, C., and Speranza, M. G. (2014). An exact solution approach for the split commodities mixed routing problem. Presented at the VeRoLog Conference, Oslo, Norway.

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

Caprara, A. and Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, **111**(3), 231–262.

Caprara, A., Furini, F., and Malaguti, E. (2013). Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem. *INFORMS Journal on Computing*, **25**(3), 560–571.

Carraghan, R. and Pardalos, P. M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, **9**(6), 375–382.

Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, **8**, 101–111.

Desaulniers, G., Desroisiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constraint vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer, Boston, MA.

Desrosiers, J., Gauthier, J. B., and Lübbecke, M. E. (2014). Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, **236**(2), 453–460.

du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, **194**(1–3), 229–237.

Fernandes Muritiba, A. E., Iori, M., Malaguti, E., and Toth, P. (2010). Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, **22**(3), 401–415.

Fréville, A. (2004). The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, **155**(1), 1–21.

Garfinkel, R. S. and Nemhauser, G. L. (1969). The set-partitioning problem: Set covering with equality constraints. *Operations Research*, **17**(5), 848–856.

Gauthier, J. B., Desrosiers, J., and Lübbecke, M. E. (2014). Tools for primal degenerate linear programs. *EURO Journal on Transportation and Logistics*. (In press.).

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Gilmore, P. and Gomory, R. (1963). A linear programming approach to the cutting stock problem: Part II. *Operations Research*, **11**, 863–888.

Gschwind, T. and Irnich, S. (2014). Solution of temporal knapsack problems by column generation accelerated using dual inequalities. Technical Report LM-2014-05, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany. (In preparation.).

Gualandi, S. and Chiarandini, M. (2014). Graph coloring benchmarks. `https://sites.google.com/site/graphcoloring/home`.

Gualandi, S. and Malucelli, F. (2012). Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, **24**(1), 81–100.

Held, S., Cook, W., and Sewell, E. C. (2012). Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, **4**(4), 363–381.

Hifi, M. and Michrafy, M. (2007). Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Computers & Operations Research*, **34**(9), 2657–2673.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (1993). *Convex Analysis and Minimization Algorithms, Part 2: Advanced Theory and Bundle Methods*, volume 306 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin.

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

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

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

Lee, C. and Park, S. (2011). Chebyshev center based column generation. *Discrete Applied Mathematics*, **159**(18), 2251–2265.

Malaguti, E., Monaci, M., and Toth, P. (2011). An exact approach for the vertex coloring problem. *Discrete Optimization*, **8**(2), 174–190.

Marsten, R., Hogan, W., and Blankenship, J. (1975). The boxstep method for large-scale optimization. *Operations Research*, **23**, 389–405.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. Wiley Intersience Series in Discrete Mathematics and Optimization. Wiley, New York, NY.

Mehrotra, A. and Trick, M. A. (1996). A column generation approach for graph coloring. *INFORMS Journal on Computing*, **8**(4), 344–354.

Östergård, P. R. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, **120**(1-3), 197–207.

Ozden, M. (1988). A solution procedure for general knapsack problems with a few constraints. *Computers & Operations Research*, **15**(2), 145–155.

Pattillo, J., Youssef, N., and Butenko, S. (2013). On clique relaxation models in network analysis. *European Journal of Operational Research*, **226**(1), 9–18.

Pferschy, U. and Schauer, J. (2009). The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, **13**(2), 233–249.

Rousseau, L.-M., Gendreau, M., and Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, **35**(5), 660–668.

Sadykov, R. and Vanderbeck, F. (2013). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, **25**(2), 244–255.

Scholl, A., Klein, R., and Jürgens, C. (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, **24**(7), 627–645.

Sim, K. and Hart, E. (2013). Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1549–1556, New York, NY. ACM.

Valerió de Carvalho, J. M. (1998). Exact solution of cutting stock problems using column generation and branch-and-bound. *International Transactions in Operational Research*, **5**(1), 35–44.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vance, P., Barnhart, C., Johnson, E., and Nemhauser, G. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, **3**, 111–130.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, **48**(1), 111–128.

Vanderbeck, F. (2005). Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Springer, New York, NY.

Appendix

5.A Proofs

Proof of Proposition 5.1:
(i)⇒(ii) and (ii)⇒(iii): Take a dual-optimal solution π^* from $D^* \cap \{\pi : E^\top \pi \leq e\}$, i.e., π^* is feasible for $\tilde{D}$ (showing (ii)). Since $\tilde{D}$ is a restriction of D, π^* is also optimal for $\tilde{D}$ implying $z_D = z_{\tilde{D}}$.
(iii)⇒(iv): Consequence of LP duality.
(iv)⇒(v): Independent of $(\tilde{\lambda}, y)$, take a primal optimal solution λ^* to P. Then, $(\lambda^*, 0)$ is a feasible for $\tilde{P}$ and has identical objective value. Hence, it is an optimal solution to $\tilde{P}$ because of $z_P = z_{\tilde{P}}$. Therefore, $c^\top \lambda^* = z_{\tilde{P}} \leq c^\top \tilde{\lambda} + e^\top y$.
(v)⇒(vi): Let $(\tilde{\lambda}, y)$ be optimal for $\tilde{P}$. Then there exists a λ^* feasible for P with $c^\top \lambda^* \leq c^\top \tilde{\lambda} + e^\top y$. Also, (λ^*, y^*) with $y^* = 0$ is feasible for $\tilde{P}$ and has identical cost. Thus, (λ^*, y^*) is optimal for $\tilde{P}$ and $Ey^* = 0$.
(vi)⇒(vii) and (vii)⇒(i): Since $Ey^* = 0$, the primal solution $\tilde{\lambda}^*$ is feasible for P. Moreover, y^* is an extreme ray for $\tilde{P}$. The optimality of $(\tilde{\lambda}^*, y^*)$ implies that $\tilde{P}$ is bounded so that $e^\top y^* \geq 0$. Then, $z_{\tilde{P}} = c^\top \tilde{\lambda}^* + e^\top y^* \leq z_P \leq c^\top \tilde{\lambda}^*$. As a consequence, $e^\top y^* = 0$ and $z_{\tilde{P}} = z_P = z_D = z_{\tilde{D}}$. Therefore, every optimal dual solution π^* to $\tilde{D}$ is feasible and herewith optimal for D, i.e., $\pi^* \in D^*$ (showing (vii)) and hence $\pi^* \in D^* \cap \{\pi : E^\top \pi \leq e\}$. □

Proof of Proposition 5.2:
Assume that the left part (Exch-D) is not fulfilled, i.e.,

$$\sum_{i \in I} (t_i - s_i)\pi_i^* > 0. \tag{5.2a}$$

We have to show that $\lambda_k^* = 0$ holds for all $k \in J$ with $a_k \geq s$. Consider such a column index $k \in J$: The primal variable λ_k has the associated dual inequality

$$\sum_{i \in I} a_{ik}\pi_i \leq 1. \tag{5.2b}$$

Since A has the (s,t)-exchange property, we have $a_k - s + t \in A$ with the associated dual inequality

$$\sum_{i \in I} (a_{ik} - s_i + t_i)\pi_i = \sum_{i \in I} a_{ik}\pi_i + \sum_{i \in I} (t_i - s_i)\pi_i \leq 1. \tag{5.2c}$$

Since π^* is dual feasible, it follows from (5.2a) and (5.2c)

$$\sum_{i \in I} a_{ik}\pi_i^* \leq 1 - \sum_{i \in I} (t_i - s_i)\pi_i^* < 1$$

such that the dual inequality (5.2b) has strictly positive slack. Complementary slackness

implies $\lambda_k^* = 0$, i.e., (Exch-P), which completes the proof. □

Proof of Proposition 5.3:
This is a direct consequence of Proposition 5.2: The option (Exch-P) is impossible because $\lambda_k^* = 0$ for all $a_k \in A$ with $a_k \geq u_h$, i.e., $a_{hk} \geq 1$ implies $(A\lambda^*)_h = 0 < b_h$. Hence, (Exch-D) must be true implying the above inequality. □

Proof of Theorem 5.1:
For both CS and VP, it is easy to verify that the matrix A has the (h,t)-exchange property for all $h \in I$, $S \subseteq I$, $t \in \mathbb{Z}_{>0}^S$ with $w_h \geq \sum_{s \in S} t_s w_s$ and $w_{hp} \geq \sum_{s \in S} t_s w_{sp}$ for all $p = 1, 2, \ldots, d$, respectively. Then, the result follows directly from Proposition 5.3. □

Proof of Remark 5.1:
(i): We give a constructive proof. If $A\lambda \geq b$ already holds with equality, there is nothing to prove. Otherwise, there exists a row $h \in I$ for which $(A\lambda)_h > b_h$. Choose any $j \in J$ with $\lambda_j > 0$ and $a_{hj} > 0$. Since $a_j \geq u_h$ and A has the $(h,0)$-exchange property, $a_j - u_h \in A$ holds. Thus, let $a_k = a_j - u_h$ be the corresponding column. Then, $\lambda' = \lambda - u_j + u_k$ is another solution with $A\lambda' \geq b$, but with smaller surplus (if any). By repeating this type of exchange procedure, a solution λ' to the system $A\lambda' = b, \lambda' \geq 0$ can be constructed.
(ii): Follows directly from Proposition 5.3. □

Proof of Proposition 5.4:
We base our proof on the equivalence of (v)⇒(i) of Proposition 5.1, i.e., we show (v).

Let $(\tilde{\lambda}, y)$ be a feasible solution to $\tilde{P}$. If $y = 0$ there is nothing to do because $\tilde{\lambda}$ is feasible for P with identical cost and hence optimal also for P.

Otherwise, there is at least one positive component of y corresponding to a valid replacement (h,i). We refer to the respective variable as $y_{(h,i)}$. A replacement cycle $(i_1, i_2, \ldots, i_p, i_1)$ (with the additional definition $i_{p+1} := i_1$) is a cyclic sequence of different row indices such that (i_s, i_{s+1}) is a valid replacement and $y_{(i_s,i_{s+1})} > 0$ for all $s = 1, 2, \ldots, p$. A basic solution to $\tilde{P}$ cannot contain any replacement cycles, since the corresponding columns are linear dependent. Consequently, all valid replacements (h,i) form a *directed acyclic graph* (DAG).

We first show that it is possible to construct, with the help of $(\tilde{\lambda}, y)$, another feasible solution $(\tilde{\lambda}', y')$ to $\tilde{P}$ with identical cost $\mathbf{1}^\top \tilde{\lambda} + e^\top y = \mathbf{1}^\top \tilde{\lambda}' + e^\top y'$ and $\mathbf{1}^\top y' < \mathbf{1}^\top y$. The latter inequality means that we can reduce the infeasibility w.r.t. P. Now, choose a longest path $\mathcal{P} = (h = i_1, i_2, \ldots, i_p = i)$ in the DAG. The maximality of $\mathcal{P}$ implies $(Ey)_h < 0$ and $(Ey)_i > 0$, and, therefore, $(A\tilde{\lambda})_h > (A\tilde{\lambda})_i$. Hence, there exist columns $a_k, k \in J$ with $a_{hk} = 1$ and $a_{ik} = 0$. Let $K \subseteq J$ be the set of these column indices.

All columns $a_k, k \in K$ allow the application of all replacements $(i_1, i_2), \ldots, (i_{p-1}, i_p)$ when these are performed in the following order: Consider a fixed column $a_k, k \in K$. Let $1 \leq q \leq p-1$ be the largest index with $a_{i_q,k} = 1$. Apply the replacements $(i_q, i_{q+1}), (i_{q+1}, i_{q+2}), \ldots, (i_{p-1}, i_p)$. Next, let $1 \leq q' \leq q-1$ be the largest index with

$a_{i_{q'},k} = 1$. Apply the replacements $(i_{q'}, i_{q'+1}), (i_{q'+1}, i_{q'+2}), \ldots, (i_{q-1}, i_q)$. Replace q by q' and iterate until $q' = 1$. Note that we used every replacement of $\mathcal{P}$ exactly once.

Due to the row replacement property of A, each replacement of a column $a_k, k \in K$ by $a_k - u_h + u_i$ results in another column $a_{j(k)} \in A$. Iteratively, we perform replacements and update the solution $(\tilde{\lambda}, y)$. In each iteration, we compute $\varepsilon = \min\{y_{i_1,i_2}, y_{i_2,i_3}, \ldots, y_{i_{p-1},i_p}\}$. The replacement of a_k into $a_{j(k)}$ is performed $\mu_k = \min\{\tilde{\lambda}_k, \varepsilon\}$ times. The new feasible solution is $y_{i_s} := y_{i_s} - \varepsilon$ for $s = 1, \ldots, p$, while the remaining components of y are unchanged. Moreover, $\tilde{\lambda}_k := \tilde{\lambda}_k - \mu_k$ and $\tilde{\lambda}_{j(k)} := \tilde{\lambda}_{j(k)} + \mu_k$. Whenever one of the $y_{i_s,i_{s+1}}$ becomes zero, the procedure stops. The new feasible solution $(\tilde{\lambda}', y')$ is the current (updated) solution $(\tilde{\lambda}, y)$. Note that this new solution has identical cost and $\mathbf{1}^\top y' < \mathbf{1}^\top y$ holds.

Also note that it is always possible to perform replacements until one of the $y_{i_s,i_{s+1}}$ is 0 because $\sum_{k \in K} \tilde{\lambda}_k \geq \min\{y_{i_1,i_2}, y_{i_2,i_3}, \ldots, y_{i_{p-1},i_p}\} = \varepsilon'$ holds: The feasibility of $(\tilde{\lambda}, y)$ implies $(A\tilde{\lambda} + Ey)_h = 1 = (A\tilde{\lambda} + Ey)_i$ which together with the maximality if $\mathcal{P}$ means that $(A\tilde{\lambda})_h \geq 1 + y_{i_1,i_2}$ and $(A\tilde{\lambda})_i = 1 - y_{i_{p-1},i_p}$. Therefore, $\sum_{k \in K} \tilde{\lambda}_k \geq \varepsilon'$.

The iterative updates finally result in $y' = 0$ because the replacement procedure eliminates at least one arc is from the DAG for every chosen path longest $\mathcal{P}$. This concludes the proof. □

Proof of Theorem 5.2:
For all three problems, it is easy to verify that the matrix A has the (h, t)-row replacement property for the respective $h, i \in I$ as specified in the theorem. Then, the results regarding DDOIs follow directly from Proposition 5.4.

For the statements regarding DOIs, note that the matrix A has the (h, i)-exchange property for the respective pairs $h, i \in I$ for all problems and the results follow from Proposition 5.3. □

Proof of Proposition 5.5:
(i) "⇒": Follows by defining $y = a_{i*}\lambda - b_i$ and substituting into the lhs of (5.1).
"⇐": Follows by multiplication of row i with α and the subsequent addition of the two rows.

(ii) P and $\tilde{P}$ are equivalent to a formulation with the additional column corresponding to the equation $\alpha\pi_h = \pi_i$ (Proposition 5.1) which is according to (i) equivalent to the aggregated formulation $\tilde{P}'$. Because P and $\tilde{P}'$ are equivalent, so are D and $\tilde{D}'$. Furthermore, it is easy to verify that $z_D = b^\top \pi^* = b'^\top \pi'^*$ holds. Therefore, π'^* is an optimal solution to $\tilde{D}'$. □

Proof of Remark 5.2:
The aggregation of all rows onto the first row results in a single row $(a'_j)\lambda = b'$ with $a'_j = \sum_{i \in I} a_{ij} w_i / L$, $b' = \sum_{i \in I} b_i w_i / L$, and associated dual value $\pi' = 1$. We show this by induction over the number m of rows that are aggregated onto the first row.

Case $m = 0$: The first row is $(a_{1j})\lambda = b_1$. Using the DDOIs proposed by Ben Amor *et al.* (2006), we can choose the associated dual value as $\pi_1^* = w_1/L$. Multiplication with w_1/L results in $(\frac{a_{1j} w_1}{L})\lambda = \frac{b_1 w_1}{L}$ with associated dual $\pi_1^{*\prime} = 1$.

Case $m{-}1 \to m$: Given is the first row representing the aggregation of rows 1 to $m-1$ of the form $(\frac{\sum_{i=1}^{m-1} a_{ij} w_i}{L})\lambda = \frac{\sum_{i=1}^{m-1} b_i w_i}{L}$ with $\pi_1^{*\prime} = 1$. Also, row m is $(a_{mj})\lambda = b_m$ with $\pi_m^* = \frac{w_m}{L} = \frac{w_m}{L}\pi_1^{*\prime}$. Aggregating both rows according to Proposition 5.5 gives

$$(\frac{\sum_{i=1}^{m-1} a_{ij} w_i}{L} + \frac{w_m}{L} a_{mj})\lambda = (\frac{\sum_{i=1}^{m} a_{ij} w_i}{L})\lambda = \frac{\sum_{i=1}^{m-1} b_i w_i}{L} + \frac{w_m}{L} b_m = \frac{\sum_{i=1}^{m} b_i w_i}{L}.$$

The associated dual value is 1.

Moreover, the aggregated entry a_j represents column $j \in J$ of the original model formulation. Because the instance can be solved without any loss, there is at least one column $j \in J$ representing a cutting pattern with no loss, i.e., $\sum_{i \in I} a_{ij} w_i = L$, in the original model. Therefore, the corresponding aggregated entry is 1. Columns with loss result in aggregated entries $a_j < 1$.

Obviously, an optimal solution to the aggregated model is given by choosing $\sum_{i=1}^{m} b_i w_i / L$ times a column $j \in J$ with $a_j = 1$. □

Proof of Proposition 5.6:
Note first that in set-covering also dominated columns can be eliminated, i.e., a column a_j can be eliminated if there exists another column a_k with $a_j \leq a_k$ (and $c_j \geq c_k$, fulfilled here due to $c_j = c_k = 1$).

Let a_j be the column corresponding to the independent set $S \subseteq I$ with $h \in I$ and $i \notin I$. Because $N(h) \supseteq N(i)$, which implies $i \notin N(h)$ and $h \notin N(i)$, the set $S \cup \{i\}$ is also an independent set and the corresponding column a_j' dominates column a_j. Thus, $a_{hj} > a_{ij}$ cannot hold for an undominated column. As a result, we have $a_{ij} \geq a_{hj}, \forall j \in J$ after eliminating all dominated columns and, hence, row i is dominated by row h and can be eliminated. □

5.B Dynamic Programming-based Separation of Subset Inequalities for the Bin Packing Problem with Conflicts (BPC)

We refer to Section 5.4.1 in which we claimed that the heuristic DP-based separation algorithm for identifying violated SIs can be implemented to run in $\mathcal{O}(mL)$ time. Recall that in the BPC an item $i \in I$ has an associated interval $\mathcal{I}_i := (a_i, b_i)$. Any two items $i, j \in I$ are in conflict (indicated by $\{i, j\} \in \mathcal{E}$) if and only if $\mathcal{I}_i \cap \mathcal{I}_j \neq \varnothing$.

In a preprocessing step, we loop over all items $h \in I$ and compute $S_h := \{i \in I : N(h) \cup \{h\} \supseteq N(i), w_h \geq w_i\}$. Moreover, the smallest interval $\mathcal{I}_{S_h}$ containing $\bigcup_{i \in S_h} (a_i, b_i)$ can be determined by computing the left- and rightmost interval borders of the items in S_h, i.e., $a_{S_h} := \min_{j \in S_h} a_j$ and $b_{S_h} := \max_{j \in S_h} b_j$ and setting $\mathcal{I}_{S_h} = (a_{S_h}, b_{S_h})$. This step requires $\mathcal{O}(m)$ time per item, overall $\mathcal{O}(m^2)$ time.

The actual separation procedure works as follows: Loop over all states $w = 1, 2, \ldots, L$ at the final stage m of the DP. Each state represents a solution, which is an independent set S_w with weight w. The determination of S_w takes $\mathcal{O}(m)$ time. Now determine the smallest interval $\mathcal{I}_{S_w}$ covering $\bigcup_{j \in S_w} \mathcal{I}_j$. This can be done, similar as before, by computing $a_{S_w} := \min_{j \in S_w} a_j$ and $b_{S_w} := \max_{j \in S_w} b_j$ and setting $\mathcal{I}_{S_w} = (a_{S_w}, b_{S_w})$. Also,

this step requires only $\mathcal{O}(m)$ time. An additional loop over all $h \in I$ now allows the direct $\mathcal{O}(1)$ test, whether or not h and S_w together form a possible SI. Indeed, $w_h \geq w$ is trivial and $S_h \supseteq S_w$ is checked via $a_{S_h} \geq a_{S_w}$ and $b_{S_h} \leq b_{S_w}$. In the positive case, the violation of the DI ($h \leftarrow S_w$) is $\sum_{s \in S_w} \bar{\pi}_s - \bar{\pi}_h$. Summing up, the outer loop requires $\mathcal{O}(L)$ time, while all steps inside that loop require $\mathcal{O}(m)$ time, yielding the $\mathcal{O}(mL)$ result.

5.C Detailed Computational Results for CS

Table 5.6 presents results for CS by subclass.

Instances	Algorithm	Solution time (avg./max/min)	# Iterations (avg./max/min)	SP/ RMP	# Dynamic DIs (avg./max/min)
All instances ($n = 240$)	*standard*	1.50/8.73/0.93	1.77/29.00/1.31	17.6	–
	static	1.00/1.00/1.00	1.00/1.00/1.00	13.7	–
	dynamic	1.03/4.80/0.44	0.86/9.00/0.41	6.5	5590/18100/247
	stat+dyn	0.83/1.53/0.29	0.75/1.00/0.31	9.8	1120/3742/0
$L =$ 1.500.000 ($n = 120$)	*standard*	1.49/7.06/0.98	1.70/13.00/1.32	18.6	–
	static	1.00/1.00/1.00	1.00/1.00/1.00	14.4	–
	dynamic	0.97/4.55/0.49	0.85/7.00/0.41	6.6	5567/18100/247
	stat+dyn	0.82/1.11/0.29	0.75/1.00/0.31	10.5	1131/3742/0
$L =$ 500.000 ($n = 120$)	*standard*	1.50/8.73/0.93	1.85/29.00/1.31	9.7	–
	static	1.00/1.00/1.00	1.00/1.00/1.00	8.1	–
	dynamic	1.08/4.80/0.44	0.87/9.00/0.60	3.6	5614/17362/483
	stat+dyn	0.84/1.53/0.29	0.74/1.00/0.35	5.8	1108/3544/0
With small items ($n = 120$)	*standard*	1.33/7.06/0.93	1.54/13.00/1.31	21.8	–
	static	1.00/1.00/1.00	1.00/1.00/1.00	17.3	–
	dynamic	0.99/4.55/0.58	0.86/7.00/0.59	7.9	6711/18100/247
	stat+dyn	0.87/1.45/0.55	0.77/1.00/0.58	12.1	1538/3742/0
Without small items ($n = 120$)	*standard*	1.66/8.73/0.99	2.01/29.00/1.51	9.6	–
	static	1.00/1.00/1.00	1.00/1.00/1.00	7.7	–
	dynamic	1.06/4.80/0.44	0.86/9.00/0.41	3.4	4470/11906/483
	stat+dyn	0.79/1.53/0.29	0.72/1.00/0.31	5.8	701/1731/0

Table 5.6: Detailed computational results for the cutting stock problem (CS)

References

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.

Chapter 6

Summary and Conclusion

The main goal of this thesis was to contribute to the development of new exact solution approaches to different combinatorial optimization problems. In what follows, we chapter-wise summarize the work, restate the main results, and give concluding remarks.

In Chapter 2, we provided insights into the nature of the symmetry inherent in the circular TTP instances. We showed that the largest symmetry group can reduce the search space by a factor approximating $4n$, where n is the number of teams. By selecting representatives from classes of symmetric solutions, it is possible to design algorithms that exactly cut off the symmetric parts of the solution space and, thus, are much faster. Using a branch-and-bound approach we demonstrated that the speedups increase with n. Even more, our empirical results suggest that symmetry reduction can lead to speedups that approximate factor $4n$ for the larger constrained instances. For the unconstrained circular instances, only a smaller symmetry group can be exploited effectively in the branch-and-bound algorithm leading to speedups that seem to approximate factor $2n$. With the accelerated branch-and-bound algorithm we were able to solve two unconstrained instances to proven optimality for the first time.

Chapters 3 and 4 studied routing problems with temporal intra-route synchronization constraints. The special case of maximum ride-time constraints was addressed in Chapter 3. It proposed a new integer CG approach to the DARP which is the prototypical VRP with maximum ride-time constraints. For the first time, all route constraints of the DARP are handled in the CG subproblem. We derived a new dynamic-programming labeling algorithm for its effective solution. The key component of the algorithms is a new, effective dominance rule for comparing partial paths. The labeling procedure also allows feasibility testing of a given route in pseudo-linear time and constitutes the first efficient feasibility check when a partial route is extended in a node-by-node fashion. Furthermore, we conducted an extensive computational study comparing the proposed branch-and-cut-and-price algorithm with the best known approaches from the literature and alternative CG algorithms that handle maximum ride-time constraints either as constraints of the master program or with less effective labeling procedures. The detailed analysis showed that our algorithm significantly outperforms all other exact solution techniques.

In Chapter 4, we introduced the VRPTWTSPD as the prototypical VRP with full temporal intra-route synchronization, i.e., with minimum and maximum ride-time constraints. For the problem with only maximum ride times, Chapter 3 has shown that handling all route constraints in the subproblem is the preferable strategy in an integer CG approach. This is mainly due to the fact that a highly effective solution algorithm

for the subproblem could be derived. The additional presence of minimum ride times, however, significantly complicates the (sub)problem and it is a priori not clear what is the best compromise between the strength of the LP bound and the hardness of the subproblem. Therefore, we developed four branch-and-cut-and-price solution approaches to the VRPTWTSPD based on CG formulations with differing subproblems. Two of these subproblems were considered for the first time in this chapter. One of them is the subproblem that integrates all constraints of the VRPTWTSPD that refer to routes individually. In the other one, maximum ride-time constraints are relaxed. We derived new dominance rules and labeling algorithms for their solution based on our ideas of Chapter 3. Extensive computational experiments demonstrated the applicability of the new labeling algorithms in the sense that they are capable of solving subproblems arising in state-of-the-art benchmark instances in reasonable time. Moreover, the results indicated a clear ranking of the four presented branch-and-cut-and-price algorithms for solving the VRPTWTSPD. The strongest approaches are those that handle either both classes of ride-time constraints or only the maximum ride-time constraints in the subproblem. They performed comparably well and were consistently significantly stronger than the other considered approaches.

In Chapter 5, we extended in theory, applications, algorithms, and computational results the work of Ben Amor *et al.* (2006) who introduced the concept of DOIs and DDOIs to stabilize the CG process. On the theoretical side, we generalized the characterization of a system of DDOIs. Moreover, instead of deriving properties of dual-optimal solutions directly from the problem under consideration, we introduced several new properties referring to the coefficient matrix of the underlying primal problem allowing for the identification of DOIs and DDOIs for all problems whose coefficient matrices fulfill these properties. Therewith, additional classes of DIs were proven to be DOIs for CS and DDOIs for BP. On the application side, we extended the DI-based stabilization to additional problems, namely VP, VC, and BPC. For all problems, we derived DOIs and/or DDOIs using the new matrix properties. On the algorithmic side, we proposed the dynamic generation (=separation) of DIs. This strategy can either replace or complement the integration of DIs into the initial RMP prior to the actual CG process. Several exact and heuristic algorithms for the separation of violated DIs were presented. Furthermore, we suggested the over-stabilization of the CG process with DIs that are potentially but not necessarily DDOIs. Along with that, a recovery procedure that is needed to restore primal feasibility in this case was presented. Computational results indicated that for all analyzed problems (CS, BP, VC, BPC) using the newly identified DDOIs together with dynamic generation and over-stabilization significantly reduces the number of CG iterations and the computation times for solving the LP relaxation of the RMP compared to state-of-the-art CG algorithms. An interesting avenue for future research is the integration and analysis of the DI-based stabilization techniques within a branch-and-price context.

References

Alves, C. and Valério de Carvalho, J. M. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, **35**(4), 1315–1328.

Alves, C., Valério de Carvalho, J., Clautiaux, F., and Rietz, J. (2014). Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, **233**(1), 43–63.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2006). Concorde TSP solver. `http://www.math.uwaterloo.ca/tsp/concorde.html`.

Aschbacher, M. (2000). *Finite Group Theory*. Cambridge University Press.

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, **36**(2), 69–79.

Azi, N., Gendreau, M., and Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, **202**(3), 756–763.

Balas, E. and Padberg, M. W. (1976). Set partitioning: A survey. *SIAM Review*, **18**(4), 710–760.

Baldacci, R. and Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, **120**(2), 347–380.

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

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

Balinski, M. (1965). Integer programming: Methods, uses, computation. *Management Science*, **12**(3), 253–313.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Ben Amor, H. and Valério de Carvalho, J. (2005). Cutting stock problems. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Springer, New York, NY.

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.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, **15**(1), 1–31.

Bettinelli, A., Cacchiani, V., and Malaguti, E. (2014). Bounds and algorithms for the knapsack problem with conflict graph. Technical Report OR-14-16, DEIS – University of Bologna, Bologna, Italy.

Bianchessi, N., Archetti, C., and Speranza, M. G. (2014). An exact solution approach for the split commodities mixed routing problem. Presented at the VeRoLog Conference, Oslo, Norway.

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

Borndörfer, R., Klostermeier, F., Grötschel, M., and Küttner, C. (1997). Telebus Berlin: vehicle scheduling in a dial-a-ride system. Technical report SC 97-23, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany.

Bredström, D. and Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, **191**, 19–29.

Caprara, A. and Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, **111**(3), 231–262.

Caprara, A., Furini, F., and Malaguti, E. (2013). Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem. *INFORMS Journal on Computing*, **25**(3), 560–571.

Carraghan, R. and Pardalos, P. M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, **9**(6), 375–382.

Ceselli, A., Righini, G., and Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, **43**(1), 56–69.

Cherkesly, M., Desaulniers, G., and Laporte, G. (2014). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*. Forthcoming.

Christiansen, M. and Nygreen, B. (1998). Modelling path flows for a combined ship routing and inventory management problem. *Annals of Operations Research*, **82**, 391–413.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, **54**(3), 573–586.

Cordeau, J.-F. and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B*, **37**(6), 579–594.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, **153**(1), 29–46.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, Boston, MA.

Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, **8**, 101–111.

de Werra, D. (1980). Geography, games and graphs. *Discrete Applied Mathematics*, **2**, 327–337.

Desaulniers, G., Desroisiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constraint vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*. Kluwer, Boston, MA.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., and Soumis, F. (2002). VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*. SIAM, Philadelphia, PA.

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.

Desrochers, M. and Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, **26**(3), 191–212.

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. (2005). A primer in column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Springer, New York, NY.

Desrosiers, J., Gauthier, J. B., and Lübbecke, M. E. (2014). Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, **236**(2), 453–460.

Dohn, A., Rasmussen, M. S., and Larsen, J. (2011). The vehicle routing problem with time windows and temporal dependencies. *Networks*, **58**(4), 273–289.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, **46**(3), 297–316.

du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, **194**(1–3), 229–237.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**(1), 7–22.

Easton, K., Nemhauser, G., and Trick, M. (2001). The traveling tournament problem description and benchmarks. In T. Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, volume 2239 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg.

Eveborn, P., Flisberg, P., and Rönnqvist, M. (2006). Laps care – an operational system for staff planning of home care. *European Journal of Operational Research*, **171**(3), 962–976.

Fernandes Muritiba, A. E., Iori, M., Malaguti, E., and Toth, P. (2010). Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, **22**(3), 401–415.

Firat, M. and Woeginger, G. J. (2011). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, **39**(1), 32–35.

Fréville, A. (2004). The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, **155**(1), 1–21.

García, S., Labbé, M., and Marín, A. (2011). Solving large p-median problems with a radius formulation. *INFORMS Journal on Computing*, **23**(4), 546–556.

Garfinkel, R. S. and Nemhauser, G. L. (1969). The set-partitioning problem: Set covering with equality constraints. *Operations Research*, **17**(5), 848–856.

Gauthier, J. B., Desrosiers, J., and Lübbecke, M. E. (2014). Tools for primal degenerate linear programs. *EURO Journal on Transportation and Logistics*. (In press.).

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Gilmore, P. and Gomory, R. (1963). A linear programming approach to the cutting stock problem: Part II. *Operations Research*, **11**, 863–888.

Gschwind, T. and Irnich, S. (2014). Solution of temporal knapsack problems by column generation accelerated using dual inequalities. Technical Report LM-2014-05, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany. (In preparation.).

Gualandi, S. and Chiarandini, M. (2014). Graph coloring benchmarks. `https://sites.google.com/site/graphcoloring/home`.

Gualandi, S. and Malucelli, F. (2012). Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, **24**(1), 81–100.

Haugland, D. and Ho, S. C. (2010). Feasibility testing for dial-a-ride problems. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and B. Chen, editors, *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg.

Held, S., Cook, W., and Sewell, E. C. (2012). Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, **4**(4), 363–381.

Hifi, M. and Michrafy, M. (2007). Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Computers & Operations Research*, **34**(9), 2657–2673.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (1993). *Convex Analysis and Minimization Algorithms, Part 2: Advanced Theory and Bundle Methods*, volume 306 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin.

Houck, D. J., Picard, J. C., Queyranne, M., and Vemuganti, R. R. (1980). The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch*, **17**, 93–109.

Hunsaker, B. and Savelsbergh, M. (2002). Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters*, **30**(3), 169–173.

Ioachim, I., Gélinas, S., Desrosiers, J., and Soumis, F. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, **31**, 193–204.

Ioachim, I., Desrosiers, J., Soumis, F., and Bélanger, N. (1999). Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, **119**(1), 75–90.

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

Irnich, S. (2010). A new branch-and-price algorithm for the traveling tournament problem. *European Journal of Operational Research*, **204**(2), 218–228.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. 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.

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.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

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.

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

Lee, C. and Park, S. (2011). Chebyshev center based column generation. *Discrete Applied Mathematics*, **159**(18), 2251–2265.

Madsen, O., Ravn, H., and Rygaard, J. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of Operations Research*, **60**, 193–208.

Malaguti, E., Monaci, M., and Toth, P. (2011). An exact approach for the vertex coloring problem. *Discrete Optimization*, **8**(2), 174–190.

Marsten, R., Hogan, W., and Blankenship, J. (1975). The boxstep method for large-scale optimization. *Operations Research*, **23**, 389–405.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. Wiley Intersience Series in Discrete Mathematics and Optimization. Wiley, New York, NY.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0–1 knapsack problem. *Management Science*, **45**(3), 414–424.

Mehrotra, A. and Trick, M. A. (1996). A column generation approach for graph coloring. *INFORMS Journal on Computing*, **8**(4), 344–354.

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*. SIAM, Philadelphia, PA.

Östergård, P. R. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, **120**(1-3), 197–207.

Ozden, M. (1988). A solution procedure for general knapsack problems with a few constraints. *Computers & Operations Research*, **15**(2), 145–155.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (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., Doerner, K. F., and Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, **37**(6), 1129–1138.

Pattillo, J., Youssef, N., and Butenko, S. (2013). On clique relaxation models in network analysis. *European Journal of Operational Research*, **226**(1), 9–18.

Pferschy, U. and Schauer, J. (2009). The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, **13**(2), 233–249.

Plum, C. E., Pisinger, D., Salazar-González, J.-J., and Sigurd, M. M. (2014). Single liner shipping service design. *Computers & Operations Research*, **45**, 1–6.

Ropke, S. and Cordeau, J.-F. (2005). Branch and cut and price for the pickup and delivery problem with time windows. Chapter 9 of S. Ropke's cumulative Ph.D. dissertation, Heuristic and exact algorithms for vehicle routing problems, University of Copenhagen, Copenhagen, Denmark.

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.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, **49**(4), 258–272.

Rousseau, L.-M., Gendreau, M., and Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, **35**(5), 660–668.

Russell, R. A. and Morrel, R. B. (1986). Routing special-education school buses. *Interfaces*, **16**(5), 56–64.

Sadykov, R. and Vanderbeck, F. (2013). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, **25**(2), 244–255.

Savelsbergh, M. W. P. (1992). The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, **4**(2), 146–154.

Scholl, A., Klein, R., and Jürgens, C. (1997). Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, **24**(7), 627–645.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin/Heidelberg.

Sim, K. and Hart, E. (2013). Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1549–1556, New York, NY. ACM.

Tang, J., Kong, Y., Lau, H., and Ip, A. W. (2010). A note on "efficient feasibility testing for dial-a-ride problems". *Operations Research Letters*, **38**(5), 405–407.

Toth, P. and Vigo, D. (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, **31**(1), 60–71.

Toth, P. and Vigo, D., editors (2002). *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Uthus, D. C., Riddle, P. J., and Guesgen, H. W. (2009). DFS* and the traveling tournament problem. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. In *6th International Conference, CPAIOR 2009*, Pittsburgh, PA.

Valerió de Carvalho, J. M. (1998). Exact solution of cutting stock problems using column generation and branch-and-bound. *International Transactions in Operational Research*, **5**(1), 35–44.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vance, P., Barnhart, C., Johnson, E., and Nemhauser, G. (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications*, **3**, 111–130.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, **48**(1), 111–128.

Vanderbeck, F. (2005). Implementing mixed integer column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*. Springer, New York, NY.