Effective Handling of Dynamic Time Windows and Synchronization with Precedences for Exact Vehicle Routing

Timo Gschwind[a], Stefan Irnich[a]

[a] *Chair of Logistics Management, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

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 restricted. 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 for 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.

Key words: Dynamic time windows, dial-a-ride problem, labeling algorithm, ride-time constraints, time synchronization, branch-and-price

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 restricted. Thus, the difference is bounded by the 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

Email addresses: gschwind@uni-mainz.de (Timo Gschwind), irnich@uni-mainz.de (Stefan Irnich)

pickup and delivery operations, e.g., the origin can be the person's home and the destination a school, hospital, or training workshop. This synchronization is an intra-route synchronization opposed to inter-route synchronization conditions that couple different routes (Drexl, 2012). Another application is 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).

The contribution of this paper is threefold: First, we derive a new branch-and-cut-and-price solution approach for 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. The crucial point here is the development of an effective column-generation pricing procedure that allows fast shortest-path computations. The most important building block is a strong dominance rule to be used in the shortest-path labeling procedure. 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 (=extended) formulation provides compared to an original formulation, from which the former is derived by Dantzig-Wolfe decomposition (Lübbecke and Desrosiers, 2005). Solutions provided by the pricing problem are convex-combined in the column-generation master program. Thus, the master program produces a stronger linear relaxation than the original formulation whenever these convex combinations form a proper subset of the corresponding original domain. Accordingly, it is known that *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 additional constraints (ESPPRC) that does not have 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 that paths visit nodes or arcs more than once. These relaxations are known as shortest-path problems with resource constraints (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 to compute any subproblem solutions, since 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. Examples are 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.*, 2011).

Similarly, when there are groups of complicated constraints in the subproblem that are 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 for this. The presence of two potentially conflicting temporal constraints (ordinary and dynamic time windows) significantly complicates the subproblem. Even more, traditional labeling algorithms are not able to check both ordinary and dynamic time windows. 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 should generally provide stronger lower bounds to the cost of a harder to solve pricing

problem. To be able to simultaneously deal with time windows and maximum ride times in the subproblem, we derive two new dominance criteria that are valid in the presence of both temporal constraints. This enables effective labeling procedures for solving the shortest-path subproblem of our column-generation approach for the DARP.

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) and Haugland and Ho (2010) a faster $\mathcal{O}(n \log n)$ algorithm. Recently, Firat and Woeginger (2011) provided 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 algorithmic step in both local search-based heuristics and exact approaches that are based on shortest-path subproblems (discussed also later on). A by-product of the dynamic-programming labeling approach proposed here is a labeling-based feasibility check. Herein, several attributes have to be computed repeatedly for each node of a partial route. More precisely, when extending a partial route by one node, $\mathcal{O}(1+|O|)$ additional attributes must be determined, where O is the set of picked requests that are not yet delivered. As the number of open requests is always bounded by the vehicle capacity and typically a small number 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. It is a priori not clear if the additional effort of solving a subproblem that handles all routing constraints of the DARP 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. Furthermore, for the formulation with ride-time constraints in the master program different separation strategies are investigated. For the new branch-and-cut-and-price approach that handles all routing constraints in the column-generation subproblem, two different labeling strategies for the shortest-path computations of the subproblem are compared. Finally, we include the two strongest approaches from the DARP literature (Ropke *et al.*, 2007; Ropke and Cordeau, 2005) in our study. The detailed analysis will show 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 2 provides a literature review on the DARP. In Section 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 4. The basic components of the branch-and-cut-and-price solution algorithms are briefly discussed in Section 5 followed by computational experiments in Section 6. The paper ends with final conclusions and an outlook given in Section 7.

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 VRP 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. 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 where 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. 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. 0 denotes the origin and $2n+1$ 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 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 all or least the obviously infeasible arcs are excluded. 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 will 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):

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{k \in K} x^k(\delta^+(i)) = 1 \qquad \forall i \in P \tag{2}$$

$$x^k(\delta^+(i)) - x^k(\delta^+(n+i)) = 0 \qquad \forall i \in P, k \in K \tag{3}$$

$$x^k(\delta^+(i)) - x^k(\delta^-(i)) = \begin{cases} 1 & i = 0 \\ -1 & i = 2n+1 \\ 0 & i \in P \cup D \end{cases} \qquad \forall i \in N, k \in K \tag{4}$$

$$T_j^k \geq (T_i^k + s_i + t_{ij}) x_{ij}^k \qquad \forall (i,j) \in A, k \in K \tag{5}$$

$$a_i \leq T_i^k \leq b_i \qquad \forall i \in N, k \in K \tag{6}$$

$$Q_j^k \geq (Q_i^k + d_i) x_{ij}^k \qquad \forall (i,j) \in A, k \in K \tag{7}$$

$$\max\{0, d_i\} \leq Q_i^k \leq \min\{C, C + d_i\} \qquad \forall i \in N, k \in K \tag{8}$$

$$L_i^k = T_{n+i}^k - (T_i^k + s_i) \qquad \forall i \in P, k \in K \tag{9}$$

$$t_{i,i+n} \leq L_i^k \leq L_i \qquad \forall i \in P, k \in K \tag{10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K \tag{11}$$

The objective is to minimize the total routing costs (1). Constraints (2) guarantee that each requests is served exactly once, and (3) impose pairing. The flow conservation constraints (4) require that each route starts at the depot 0, is continued, and finally ends in the depot $2n+1$. Constraints (5) and (7) ensure consistency of the time and load variables with the routing variables. Time variables are bounded the by time windows (6), and load variables by the vehicle capacity (8). The ride times are defined by equations (9) and are bounded by the maximum ride times (10). Due to the non-negativity of the ride times constraints, constraints (10) also impose the precedences. This three-index formulation (1)–(11) is non-linear because of constraints (5) and (7). A linearization, however, is straightforward (see, e.g., Cordeau (2006) or Ropke *et al.* (2007)).

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 (1)–(8) and (11), i.e., without constraints (9)–(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 (9)–(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

$$x^k(I) \leq |I| - 1 \qquad \forall I \in \mathcal{I}, k \in K, \tag{12}$$

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 for 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)–(11) and also the IPEC (12) are all non-coupling constraints as they refer to each vehicle individually. A straightforward Dantzig-Wolfe decomposition has covering constraints (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-fesible routes is denoted by Ω^{DARP}.

An alternative Dantzig-Wolfe decomposition leaves the covering constraints (2) and the IPEC (12) in the master program. Since IPEC replace the ride-time constraints (9)–(10) in this case, the subproblem comprises only the PDPTW constraints (3)–(8). The variables λ_r of this extensive formulation model PDDTW-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 will 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)

$$\min \sum_{r \in \Omega^{PDPTW}} c_r \lambda_r \tag{13}$$

$$\text{s.t.} \sum_{r \in \Omega^{PDPTW}} a_{ir} \lambda_r = 1 \quad \forall i \in P \tag{14}$$

$$\sum_{r \in \Omega^{PDPTW}} \lambda_r = |K| \tag{15}$$

$$\sum_{r \in \Omega^{PDPTW}} b_{Ir} \lambda_r \leq |I| - 1 \quad \forall I \in \mathcal{I} \tag{16}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \Omega^{PDPTW} \tag{17}$$

(IMP)

$$\min \sum_{r \in \Omega^{DARP}} c_r \lambda_r \tag{13}$$

$$\text{s.t.} \sum_{r \in \Omega^{DARP}} a_{ir} \lambda_r = 1 \quad \forall i \in P \tag{14}$$

$$\sum_{r \in \Omega^{DARP}} \lambda_r = |K| \tag{15}$$

$$\lambda_r \in \{0,1\} \quad \forall r \in \Omega^{DARP} \tag{17}$$

The objective (13) is the minimization of the total costs, the pendant to request covering constraints (2) is (14), and constraints (15) are generalized convexity constraints (resulting from aggregation by vehicles, see Desaulniers *et al.*, 1998). The reformulation of the IPEC (12) in the route variables is (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 will also add the suffix -I to identify its components), while the approach presented in this paper will rely 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 (13)–(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. Our contribution is the development of an effective solution procedure for the MP subproblem. We will 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.

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 (14), (15), and (16), respectively. To identify feasible routes with negative reduced cost, one first has to compute the reduced costs of the routing variables x_{ij} as

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - \pi_i - \sum\limits_{I \in \mathcal{I}} b_{Ir} \rho_I & \text{if } i \in P \\ c_{ij} - \sum\limits_{I \in \mathcal{I}} b_{Ir} \rho_I & \text{otherwise} \end{cases} \qquad \tilde{c}_{ij} = \begin{cases} c_{ij} - \pi_i & \text{if } i \in P \\ c_{ij} & \text{otherwise} \end{cases}, \tag{18}$$

and then the following subproblem has to be solved:

$$\begin{array}{ll} \overset{\text{(PP-I)}}{=\text{(SPPPDPTW)}} & \min \sum\limits_{(i,j) \in A} \tilde{c}_{ij} x_{ij} - \mu \\ & \text{s.t. (3)–(8) and (11)} \end{array} \qquad \begin{array}{ll} \overset{\text{(PP)}}{=\text{(SPPDARP)}} & \min \sum\limits_{(i,j) \in A} \tilde{c}_{ij} x_{ij} - \mu \\ & \text{s.t. (3)–(11)} \end{array} \tag{19}$$

where in the constraints (3)–(11) the index k for the specific vehicle is dropped.

Subproblem Solution by Dynamic-Programming Labeling Algorithms. Pricing problems of most VRP variants are elementary shortest-path problems with resource constraints (ESPPRC) (Desaulniers *et al.*, 1998). ESPPRC and their relaxations are typically solved using a dynamic-programming labeling algorithm (Irnich and Desaulniers, 2005). 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.

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 excluded from further consideration.

Note that the second condition requires that the partial path $\mathcal{P}(\ell_1)$ must be feasible. We call a label ℓ *feasible*, if the represented partial path $\mathcal{P}(\ell)$ is 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, the second condition is often proved indirectly using conditions that compare 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 resource extension functions (REFs) for the validity of dominance rules was stressed and exemplified in (Desaulniers *et al.*, 1998; Irnich, 2008).

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 REFs.

A final remark about IMP-I subproblem for the generation of PDPTW routes seems appropriate here. The advantage of replacing (9)–(10) by IPEC, as suggested by Ropke and Cordeau, is that the reduced costs (18) in PP-I solely depend on the chosen arcs, but not on a route's schedule. 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).

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 each feasible completion to visit the delivery nodes of those open requests. Conversely, however, partial paths with open delivery nodes are not allowed in the labeling process and thus considered infeasible.

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*. It may be fulfilled by definition (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 delivery triangle inequality. Ropke and Cordeau (2009) show how to transform the reduced-cost matrix by addition of constants into one that satisfies the delivery triangle inequality. From now on, we assume that the delivery triangle inequality holds. In particular, the validity of the following and all other dominance rules in this paper is based on this property.

The standard dominance criterion for SPPPDPTW (Dumas *et al.*, 1991; Ropke and Cordeau, 2005) 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, and the set of open requests O_ℓ. Dumas *et al.* (1991) and Ropke and Cordeau (2005) have shown:

Proposition 1. (Dom_{PDPTW}) *A feasible label ℓ_1 dominates a label ℓ_2 if*

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

The dominance rule of Proposition 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 that resolve 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 1 results from the generalization of the second condition of the standard dominance. Condition 2. can be replaced by

Labels	nodes in $\mathcal{P}(\ell_1)$, $\mathcal{P}(\ell_1')$	0	j	i	$j+n$	η	$i+n$
ℓ_1 for path $(0,...,\eta)$	time window $[a_i, b_i]$	$[0,100]$	$[0,15]$	$[20,35]$	$[15,30]$	$[50,65]$	$[60,75]$
ℓ_1' for path $(0,..,i+n)$	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 1: Label ℓ_1 dominates label ℓ_2 in the SPPPDPTW sense, but is inferior regarding the ride-time constraint of request i

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 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 delivery triangle inequality holds. $\mathcal{P}_1$ respects the capacity constraint, since $\mathcal{P}_2$ satisfies it and $O_{\ell_1} \subseteq O_{\ell_2}$ implies that the load of ℓ_1 is not larger than of ℓ_2. Finally, time-window constraints are also respected by $\mathcal{P}_1$ when they are respected by $\mathcal{P}_2$, as $t_{\ell_1} \leq t_{\ell_2}$ and visiting the additional deliveries in Q compared to Q' can never decrease the time, since travel times satisfy the triangle inequality. From a higher perspective, the PDPTW dominance rules result from the modeling with non-decreasing REFs for pairing and precedence, capacity, and time-window constraints as shown in (Irnich and Desaulniers, 2005).

Ropke and Cordeau (2005) also discussed three additional aspects of Dom_{PDPTW}. First, an extension to the elementary case is straightforward. 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, Dom_{PDPTW} is only valid as long as the delivery triangle inequality holds for the reduced costs $\tilde{c}_{ij}$. If no assumptions on the reduced costs can be made, dominance is only possible between labels with identical sets $O_{\ell_1} = O_{\ell_2}$ of open requests (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 (1)–(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 delivery triangle inequality.

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 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 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.*

4.1. Weak Dominance for SPPDARP

One may ask why we present a weak dominance relation if a stronger dominance is also available (Section 4.3). The reason is that, on the one hand, the weak dominance provides some useful insights about basic information needed for applying any dominance rule. These insights will be used for the development of the strong dominance rules. On the other hand, there is a trade-off between the strength of the dominance rules 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 6 will therefore compare branch-and-price algorithms based on both weak and strong dominance rules.

Example 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. On the other hand, it is easy to see that 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 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 2. (Dom_{weak}) *A feasible label ℓ_1 dominates a label ℓ_2, if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t_{\ell_1} \leq t_{\ell_2}, \text{ and}$$
$$|O_{\ell_1}| = \begin{cases} 0, & \text{if } \eta \in D \\ 1, & \text{if } \eta \in P \end{cases}. \tag{20}$$

Note that condition (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 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 2. *Consider label ℓ_2' from Example 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 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 successing 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. To check the DARP-feasibility of a given path, the procedures proposed by Tang *et al.* (2010); Haugland and Ho (2010); Firat and Woeginger (2011) can be applied.

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 1. For convenience, we also include a resource for the load 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

$$x \notin O_\ell \quad \text{if} \quad x \in P \tag{21}$$
$$x - n \in O_\ell \quad \text{if} \quad x \in D \tag{22}$$
$$O_\ell = \varnothing \quad \text{if} \quad x = 2n + 1. \tag{23}$$

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. A feasible 0-$(2n+1)$-path in the SPPDARP has to respect pairing and precedence, capacity, time-window, and ride-time constraints. 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

$$\tau_i \in [a_{h_i}, b_{h_i}] \quad \forall i = 1, \ldots, q \tag{24}$$
$$\tau_i + s_{h_i} + t_{h_i, h_{i+1}} \leq \tau_{i+1} \quad \forall i = 1, \ldots, q-1 \tag{25}$$
$$\tau_i + s_{h_i} + L_{h_i} \geq \tau_j \quad \text{if } h_i + n = h_j. \tag{26}$$

The time schedule $T_\mathcal{P}$ is then said to be feasible. Note again, that elementarity of $\mathcal{P}$ is assumed. Otherwise, inequalities (26) have to be satisfied for every pair of corresponding pickup and delivery nodes.

For a *partial* path, feasibility in the SPPDARP sense means respecting time-window, capacity, and precedence constraints. Furthermore, ride-time constraints have to be met for each request that has been picked up and delivered on the partial path. Then, there exists at least one feasible time schedule for each feasible partial path. 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 $\eta \in P$ and $|O_{\ell'}| = 1$ follows immediately from the feasibility of its parent label ℓ together with $t_\ell + t_{\eta_\ell, x} \leq b_x$.

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. Moreover, in many cases it is not necessary to consider in the feasibility test the entire partial path that is given by ℓ' from origin depot 0 up to node $\eta_{\ell'}$. Instead, 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, such as the $\mathcal{O}(n \log n)$ test by Haugland and Ho (2010) or the linear-time test by Firat and Woeginger (2011). Besides, the algorithm of Tang *et al.* (2010) can uses some of the data already computed during the labeling process, while the approach of 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.

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

$$\eta_{\ell'} = x \quad \tilde{c}_{\ell'} = \tilde{c}_\ell + \tilde{c}_{\eta_\ell, x} \quad t_{\ell'} = \max\{a_x, t_\ell + t_{\eta_\ell, x}\} \quad l_{\ell'} = l_\ell + d_x \tag{27}$$
$$O_{\ell'} = \begin{cases} O_\ell \cup \{x\} & \text{if } x \in P \\ O_\ell \backslash \{x - n\} & \text{if } x \in D \end{cases} \tag{28}$$

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 as 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.

4.3. Strong Dominance for SPPDARP

Proposition 2 has shown that when restricting the set of dominating labels according to (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 $Q' = Q \backslash \{n + (O_{\ell_2} \backslash O_{\ell_1})\}$. 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 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 will show (in Proposition 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^η at the current node η, for all open request $i \in O$ the *latest possible delivery time* $ld^i(t)$ as a function of the start of service t at η. In other words, $ld^i(t)$ 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 are violated, and the start of service at η 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* with $\tau_\eta \leq t$. Then, $ld^{h_i}_\ell(t), t \geq t^\eta_\ell, h_i \in O_\ell$ can be defined as

$$ld^{h_i}_\ell(t) = \min \left\{ b_{h_i+n}, \max_{T_{\mathcal{P}(\ell)} = (\tau_i) \in \mathscr{T}_{\mathcal{P}(\ell)}(t)} \{\tau_i\} + s_{h_i} + L_{h_i} \right\}. \tag{29}$$

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 1. *Let $\mathcal{P} = (h_1, ..., h_q)$ be a feasible partial path, and let $\tau_i^*(t) = \max_{T_\mathcal{P} = (\tau_i) \in \mathscr{T}_\mathcal{P}(t)} \{\tau_i\}$ 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 A of the Appendix.

Integrating the information $ld^i_\ell(t)$ 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. *(Dom^*_{strong}) A feasible label ℓ_1 dominates a label ℓ_2, if*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \leq \tilde{c}_{\ell_2}, \quad t^\eta_{\ell_1} \leq t^\eta_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \text{ and} \tag{30}$$

$$ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t) \quad \forall i \in O_{\ell_1}, t \in [t^\eta_{\ell_2}, b_{\eta_{\ell_2}}]. \tag{31}$$

Equations (31) 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 for ℓ_2, it is feasible to deliver request i for ℓ_1 whenever it is feasible for ℓ_2. Note that the functions $ld^i_\ell(t)$ 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 (31) 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 $i \in O_{\ell_1}, t \in [t^\eta_{\ell_2}, b_{\eta_{\ell_2}}]$.

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 requires the comparison of several functions in (31), the dominance relation is practically not yet applicable in a labeling algorithm. We therefore characterize the shape of the functions $ld^i(t)$ next enabling a version of Dom^*_{strong} that is easy to handle.

Proposition 4. *Let ℓ be a feasible label at node η_ℓ with earliest start of service t^η_ℓ and open requests O_ℓ. The functions $ld^i_\ell(t), t \geq t^\eta_\ell$ are of the form $\min\{k^i_1 + t, k^i_2\}$ for all $i \in O_\ell$, where k^i_1 and k^i_2 are constants.*

Using Proposition 4, Dom^*_{strong} can be simplified to a dominance rule where instead of having to keep track of and compare the entire functions $ld^i_\ell(t)$ it is sufficient to store and check the values of $ld^i_\ell(t)$ only at two distinct points of time. We have chosen the earliest possible start of service t^η_ℓ and time B^i_ℓ where the latter is the time when $ld^i_\ell(t)$ becomes constant. Then a simplified version of Dom^*_{strong} is as follows (see Figure 1):

Proposition 5. *(Dom_{strong}) A label ℓ_1 dominates a label ℓ_2, if*

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

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 will only describe additional aspects. According to Proposition 5, the two additional resources $ld^i_\ell(t^\eta_\ell)$ 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.

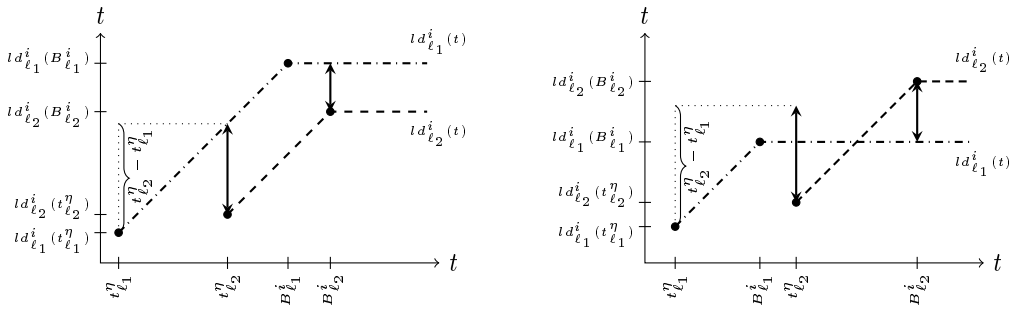

Figure 1: Dominance condition for ld^i fullfilled (left hand side) and not fullfilled (right hand side)

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 $t^\eta_\ell + s_{\eta_\ell} + t_{(\eta_\ell,x)} \leq ld^i_\ell(B^i_\ell)$ 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 6. *Let the label ℓ' result from the extension of a label ℓ along the arc (η_ℓ, x). Then, the resources $ld^i_\ell(t^\eta_\ell)$, $ld^i_\ell(B^i_\ell)$, and B^i_ℓ are either initialized or updated according to the following REFs:*

$$B^i_{\ell'} = \begin{cases} \min\{\tilde{b}_x, b_{x+n} - s_x - L_i\} & \text{if} \quad i = x \\ \min\{B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x}, \tilde{b}_x\} & \text{otherwise} \end{cases} \tag{32}$$

$$ld^i_{\ell'}(t^x_{\ell'}) = \begin{cases} \min\{t^x_{\ell'} + s_x + L_i, b_{x+n}\} & \text{if} \quad i = x \\ ld^i_\ell(t^{\eta_\ell}_\ell) + (\min\{t^x_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x}, B^i_\ell\} - t^{\eta_\ell}_\ell) & \text{otherwise} \end{cases} \tag{33}$$

$$ld^i_{\ell'}(B^i_{\ell'}) = \begin{cases} B^i_{\ell'} + s_x + L_i & \text{if} \quad i = x \\ ld^i_\ell(B^i_\ell) - ((B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x}) - B^i_{\ell'}) & \text{otherwise,} \end{cases} \tag{34}$$

where

$$\tilde{b}_x = \begin{cases} b_x & \text{if } x \in P \\ \min\{b_x, ld^{x-n}_\ell(B^{x-n}_\ell)\} & \text{if } x \in D \end{cases}. \tag{35}$$

Example 3. *A small example to illustrate the REFs and the underlying intuition for B^i, $ld^i(t^\eta)$ and $ld^i(B^i)$ is given in Table 2. 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 2. 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 $B^i_{\ell_i}, ld^i_{\ell_i}(t^i_{\ell_i})$ 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., $ld^i_{\ell_i}(t^i_{\ell_i}) = 5 + 20 = 25$ 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 the label ℓ_j with $t^j_{\ell_j} = t^i_{\ell_i} + 5 = 10$. Regarding the latest delivery times for request i, starting the service at j at time $t^j_{\ell_j}$ means that the predecessor node i has to be serviced at the earliest possible time $t^i_{\ell_i}$, and consequently $ld^i_{\ell_i}(t^i_{\ell_i}) = ld^j_{\ell_j}(t^j_{\ell_j}) = 25$. 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 $B^i_{\ell_j} = b_j = 18$.

	ℓ_0	ℓ_i	ℓ_j	ℓ_k
η	0	i	j	k
$[a_\eta, b_\eta]$	$[0, 100]$	$[5, 15]$	$[10, 18]$	$[17, 25]$
t^η	0	5	10	17
B^i	-	15	18	23
$ld^i(t^\eta)$	-	25	25	27
$ld^i(B^i)$	-	35	33	33

Table 2: REFs for B^i, $ld^i(t^\eta)$ and $ld^i(B^i)$ along the path $\mathcal{P} = (0, i, j, k)$ assuming a maximum ride time of $L_i = 20$

Considering label ℓ_k, the earliest start of service is $t^k_{\ell_k} = 17$ 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 of node i. In other words, when serving k at time $t^k_{\ell_k} = 17$ the service of i should not be started at $t^i_{\ell_i}$ but instead be delayed until $t^i_{\ell_i} + 2 = 7$. This also implies that j is serviced at time $t^j_{\ell_j} + 2 = 12$. The corresponding latest delivery times of i for the earliest feasible service at k are then given by $ld^i_{\ell_k}(t^k_{\ell_k}) = ld^i_{\ell_j}(t^i_{\ell_i} + 2) = ld^i_{\ell_j}(t^i_{\ell_i}) + 2 = 27$. 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 an 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 $ld^i_{\ell_k}(B^i_{\ell_k}) = ld^i_{\ell_j}(B^i_{\ell_j}) = 33$.

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.

4.5. A Pseudo-Linear Feasibility Test for 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 (27)–(28) and (32)–(34) 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, there exists to the best of our knowledge only one algorithm that runs in linear time (Firat and Woeginger, 2011). Feasibility tests with a worst-case runtime of $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^2)$ were provided by Haugland and Ho (2010) and Tang *et al.* (2010), respectively. The eight-step route evaluation procedure of Cordeau and Laporte (2003) also serves as an feasibility check and runs in $\mathcal{O}(n^2)$. Even more, in the SPPRC context, when a given feasible partial route is extended node by node, neither of these algorithms is suited to derive a simple and efficient check. Our labeling algorithm, however, allows to decide on the feasibility of such an extension in $\mathcal{O}(M)$.

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 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

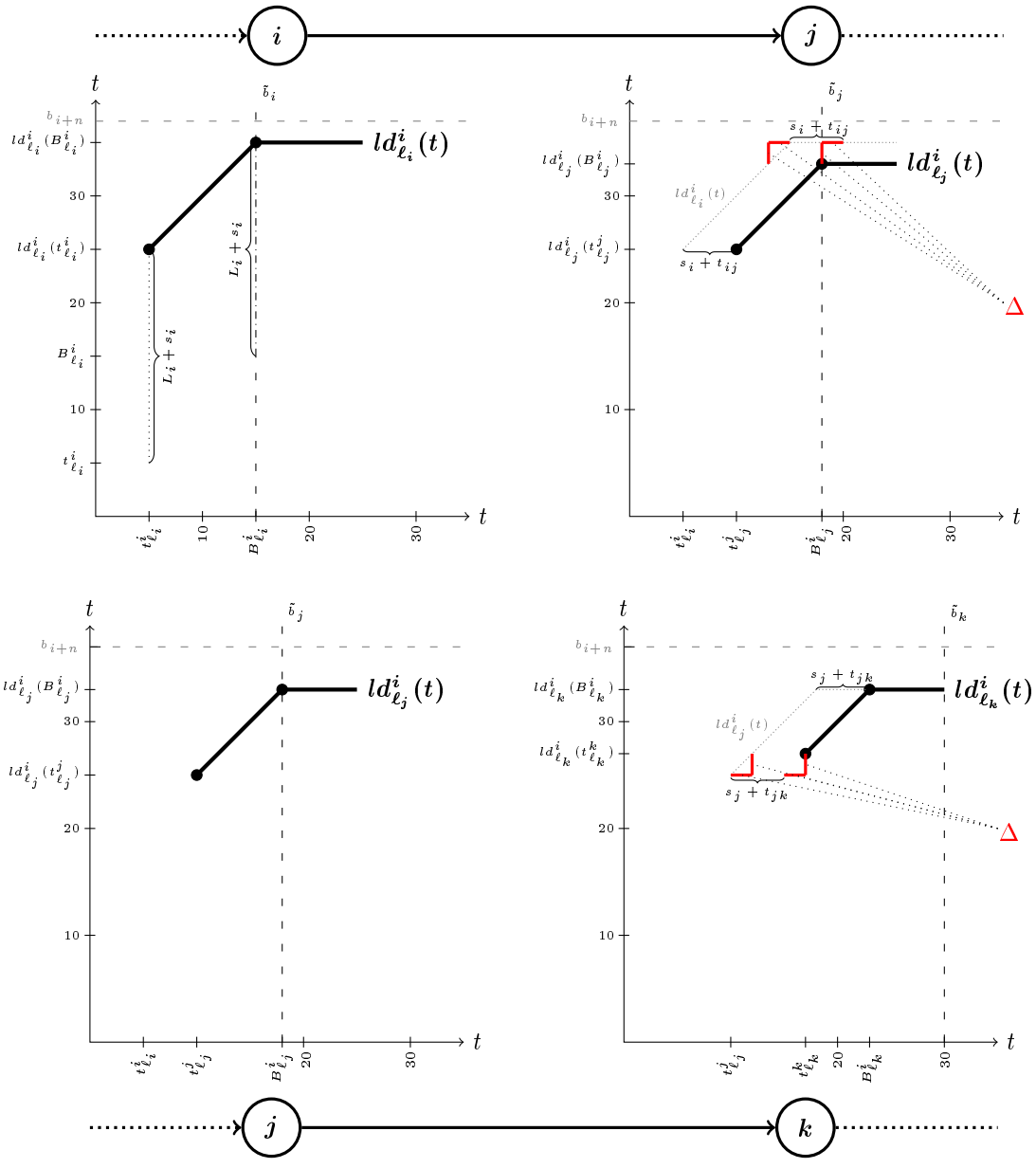

Figure 2: Extension of labels: Latest delivery times $ld^i(t)$ for an open request i for the extension along arc (i, j) (upper part) and along arc (j, k) (lower part)

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 maximum service times τ_j for ℓ' at the nodes $h_1, ..., h_q$, we consider inequalities (24) and (25), and inequalities (26) 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 $\eta_{\ell'} = h_q$. For any feasible completion Q of ℓ', the resulting path $(\mathcal{P}(\ell'), Q)$ must among others satisfy $\tau_i + s_{h_i} + L_{h_i} \geq \tau_j$ for all $h_i \in O_{\ell'}, h_i + n = h_j$. Consequently, each feasible partial schedule must respect $\tau_q + s_{\eta_{\ell'}} + t_{\eta_{\ell'}, h_i+n} \leq \tau_i + s_{h_i} + L_{h_i}$ for all $h_i \in O_{\ell'}$ in order to be feasibly completed. These constraints bound τ_q. Using this bound in equation (35), 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 which imposes even stronger bounds on τ_q. The computational effort to find the strongest bound on τ_q, however, seems to be prohibitively large in general. We therefore 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 $B^i_{\ell'}$, $ld^i_{\ell'}(t^\eta_{\ell'})$, and $ld^i_{\ell'}(B^i_{\ell'})$ an upper bound $\hat{b}_{\eta_{\ell'}}$ on the latest feasible start of service at the current node x is computed. In the REFs (32)–(34), the value $\tilde{b}_x$ is then replaced by $\hat{b}_x$.

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 $t \in [t^\eta_{\ell_2}, \hat{b}_{\eta_{\ell_2}}]$ than for all $t \in [t^\eta_{\ell_2}, b_{\eta_\ell}]$. 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.

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 will be compared in Section 6. Mainly, we will 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 will also compare the two new labeling algorithms of Sections 4.2 and 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 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

$$x_{ij} = \sum_{k \in K} x_{ij}^k = \sum_{r \in \Omega} a_{ij,r} \lambda_r \qquad \forall (i,j) \in A \tag{36}$$

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 B of the Appendix gives further details on these inequalities and their separation. For the computational studies, we will 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 $_{sepRT}$.

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 (36) 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 $x(\delta^+(0)) \geq \lceil \bar{x}(\delta^+(0)) \rceil$. Second, we branch on the outflow of a node set $S = \{h,i\}$ 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.

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 B of the Appendix, eqs. (37) and (38)) 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 B of the Appendix). Following the notation of Section 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.

Algorithm	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$^{cut}_{strg}$
acronym										
Formulation	2-index	IMP-I	IMP-I	IMP-I	IMP-I	IMP-I	IMP	IMP	IMP	IMP
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: Overview of DARP solution approaches used for comparison

The approaches based on IMP where the subproblem is an SPPDARP solved using the proposed labeling algorithm with Dom_{weak} and Dom_{strong} are denote by IMP$_{weak}$ and IMP$_{strg}$, respectively. As before, a superscript ^{cut} indicates that cuts are used, giving rise to IMP-I and IMP-I^{cut}. All algorithms are summarized in Table 3.

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 (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. 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.

6.1. Linear Relaxation Results

We start the analysis with results on the linear relaxation of the master program (13)–(17). Tables 4 and 5 show details for the lower bound values lb and summarize the computation times. The notation in the tables has the following meaning:

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

More detailed tables with individual computation times per instance can be found in Section C 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.

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

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

Lower Bounds. The computed lower bound values lb for linear relaxations shows 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 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 corresponding 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 a 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-I_{sepRT} and $\text{MP-I}^{cut}_{sepRT}$, i.e., approaches with full consideration of ride-time feasibility, with MP-I and MP-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 $\text{MP-I}^{cut}_{sepRT}$. 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-I_{sepRT} and $\text{MP-I}^{cut}_{sepRT}$. Interestingly, for the type `b` instances, the opposite is true for the solution times of $\text{MP-I}^{cut}_{sepRT}$ and MP-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.

6.2. Integer Solution Results

Results for the computation times of optimal integer solutions are given in Tables 6 and 7. The entry *1h* indicates that the respective algorithm was unable to solve the instances within the time limit of one hour. There was no time limit for B&C and B&P-RC was terminated after two hours (*2h*).

Our approach IMP^{cut}_{strg} with the SPPDARP subproblem and cuts was able 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) 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-I^{cut} fails on three more instances, the approaches without any cuts (except those, necessary to handle ride-time constrains) IMP-I and IMP-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.

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

Table 6: Computation times for optimal integer solutions of type `a` instances in seconds.

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 simplest example of a 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

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

Table 7: Computation times for optimal integer solution of type b instances in seconds.

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: 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 effective dominance rules for comparing partial paths represented by labels. As obvious when looking back on Section 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 presented in this paper. First, more general intra-route synchronization with dynamic time windows of the form $[K_i, L_i]$ with $K_i > 0$ 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 and dynamic time windows $[K_i, L_i]$ 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 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; 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 inner-route and inter-route synchronization constraints can be modeled and solved. This relates to exact as well as heuristic approaches.

Acknowledgment

The second author's research is partially funded by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/5-1.

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.

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.

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: Methodological*, **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*, pages 327–357. Springer US, Boston and MA.

Desaulniers, G., Desroisiers, J., Ioachim, I., Solomon, 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*, pages 57–93. Kluwer, Norwell and MA.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M., and Soumis, F. (2002). VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 9, pages 225–242. SIAM, Philadelphia.

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.

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*, pages 170–179. Springer, Berlin and 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. (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*, pages 33–65. Springer, New York.

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

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

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

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.

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

A. Proofs

Proof of Lemma 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 (24)–(26) and $\tau_q \leq t$. By definition, for each $i = 1, ..., q$, there exists a feasible schedule $T^i_{\mathcal{P}}(t) = (\tau^i_1(t), ..., \tau^*_i(t), ..., \tau^i_q(t))$ with $\tau^i_q(t) \leq t$ and $\tau^i_j(t) \leq \tau^*_j(t)$ 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 $\tau^*_i(t) + s_{h_i} + t_{h_i,h_{i+1}} \leq \tau^i_{i+1}(t)$ and $\tau^i_{i+1}(t) \leq \tau^*_{i+1}(t)$ hold for $i = 1, ..., q-1$, $T^*_{\mathcal{P}}(T)$ satisfies inequalities (25). Finally, using $\tau^j_i(t) + s_{h_i} + L_{h_i} \geq \tau^*_j(t)$ and $\tau^*_i(t) \geq \tau^j_i(t)$ for each pair i, j with $h_i + n = h_j$, $T^*_{\mathcal{P}}(T)$ also satisfies inequalities (26) and thus $T^*_{\mathcal{P}}(T) \in \mathscr{T}_{\mathcal{P}}(t)$. □

Proof of Proposition 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 1 that $\mathcal{P}_1$ satisfies pairing, precedence, and capacity constraints and has smaller costs 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 τ^{ℓ_2} at the current node η_{ℓ_2}. Denote by $T^*_{\mathcal{P}(\ell_1)} = (\tau^*_1(\tau^{\ell_2}), ..., \tau^*_q(\tau^{\ell_2})) \in \mathscr{T}_{\mathcal{P}(\ell_1)}(\tau^{\ell_2})$ with $h_q = \eta_{\ell_1}$ 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 $\tau_{\eta_{\ell_1}} \leq \tau^{\ell_2}$. Lemma 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, $\tau^*_{\eta_{\ell_1}} \leq \tau^{\ell_2}$, and $\tau_i \leq ld^{h_i-n}_{\ell_2}(\tau^{\ell_2}) \leq ld^{h_i-n}_{\ell_1}(\tau^{\ell_2})$ 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 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 $\tau^*_q(t), t \geq t^\eta_\ell$ at the last node h_q is of the form $\tau^*_q(t) = \min\{t, k\}$,

where k is a constant. By definition, $\tau_q^*(t) = \max_{T_{\mathcal{P}(\ell)} \in \mathscr{T}_{\mathcal{P}(\ell)}(t)} \{\tau_q\}$ 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 sub-path of $\mathcal{P}$ starting at h_i. We can now show by induction over the number m that $ld_\ell^{h_i}(t) = \min\{k_1^{h_i} + t, k_2^{h_i}\}, t \geq t_\ell^\eta$ 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 $T_{\mathcal{P}(\ell')}^{*,\ell'}(t) = (\tau_1^{*,\ell'}(t), ..., \tau_q^{*,\ell'}(t), \tau_x^{*,\ell'}(t)) \in \mathscr{T}_{\mathcal{P}(\ell')}(t)$ 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 maximation problems for $T_{\mathcal{P}(\ell')}^{*,\ell'}(t)$ and $T'_{\mathcal{P}(\ell' \setminus x)}(t')$ we have that $T_{\mathcal{P}(\ell')}^{*,\ell'}(t) = (T'_{\mathcal{P}(\ell')\setminus\{x\}}(t'), \tau_x^{*,\ell'}(t))$ 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 6: Consider first the case $i = x$. We have shown in the proof of Proposition 4 that the latest possible delivery times for i in this case are $ld_{\ell'}^i(t) = \min\{\min\{t, b_x\} + s_x + L_i, b_{i+n}\}$. The point of time when $ld_{\ell'}^i(t)$ becomes constant is then clearly given by $B_{\ell'}^i = \min\{b_x, b_{i+n} - s_x - L_i\}$. The formulas for $ld_{\ell'}^i(t_{\ell'}^x)$ and $ld_{\ell'}^i(B_{\ell'}^i)$ are obvious in this case.

For $i \neq x$, it is again known from the proof of Proposition 4 that $T_{\mathcal{P}(\ell')}^{*,\ell'}(t) = (T_{\mathcal{P}(\ell)}^{*,\ell}(t'), \tau_x^{*,\ell'}(t))$ and thus $ld_{\ell'}^i(t) = ld_\ell^i(t')$ holds, with $t' = \tau_x^{*,\ell'}(t) - s_{\eta_\ell} - t_{\eta_\ell,x}$ and $\tau_x^{*,\ell'}(t) = \min\{t, \tilde{b}_x\}$. If $t \geq \tilde{b}_x$ then $\tau_x^{*,\ell'}(t)$ is constant and hence $ld_{\ell'}^i(t)$ is constant for all such t. For all $t < \tilde{b}_x$, $\tau_x^{*,\ell'}(t)$ increases linear in t and so does $ld_\ell^i(t')$ as long as $B_\ell^i \geq t'$. If $B_\ell^i < t'$, then $ld_\ell^i(t')$ is constant and so is $ld_{\ell'}^i(t)$. As a result, the time when $ld_{\ell'}^i(t)$ becomes constant is $B_{\ell'}^i = \min\{B_\ell^i + s_{\eta_\ell} + t_{\eta_\ell,x}, \tilde{b}_x\}$.

For $ld_{\ell'}^i(t_{\ell'}^x)$ and $ld_{\ell'}^i(B_{\ell'}^i)$ note first that $\tau_x^{*,\ell'}(t_{\ell'}^x) = t_{\ell'}^x$ and $\tau_x^{*,\ell'}(B_{\ell'}^i) = B_{\ell'}^i$. Then, we have $ld_{\ell'}^i(t_{\ell'}^x) = ld_\ell^i(t_{\ell'}^x - s_{\eta_\ell} - t_{\eta_\ell,x}) = ld_\ell^i(t_{\ell'}^x - s_{\eta_\ell} - t_{\eta_\ell,x} + t_\ell^{\eta_\ell} - t_\ell^{\eta_\ell})$ and $ld_{\ell'}^i(B_{\ell'}^i) = ld_\ell^i(B_{\ell'}^i - s_{\eta_\ell} - t_{\eta_\ell,x}) = ld_\ell^i(B_{\ell'}^i - s_{\eta_\ell} - t_{\eta_\ell,x} + B_\ell^i - B_\ell^i)$. Using the property that $ld_\ell^i(t)$ increases linear for all $t \in \left[t_\ell^{\eta_\ell}, B_\ell^i\right]$ and is constant for all $t > B_\ell^i$ we have that $ld_{\ell'}^i(t_{\ell'}^x) = ld_\ell^i(t_\ell^{\eta_\ell}) + \min\{(t_{\ell'}^x - s_{\eta_\ell} - t_{\eta_\ell,x}) - t_\ell^{\eta_\ell}, B_\ell^i - t_\ell^{\eta_\ell}\}$ and $ld_{\ell'}^i(B_{\ell'}^i) = ld_\ell^i(B_\ell^i) - ((B_\ell^i + s_{\eta_\ell} + t_{\eta_\ell,x}) - B_{\ell'}^i)$ □

B. 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.

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 tourna-

ment constraint is

$$x([I]) \leq |I| - 1 \quad (= q - 2), \tag{37}$$

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:

$$x(I) \leq |I| - 2 \quad (= q - 3) \tag{38}$$

For the separation of both types of infeasible path inequalities, we use a straightforward 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 $S \in P \cup D$ 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 differentiate 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 (in)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. For this purpose, we use an adaptation of our labeling algorithm for SPPDARP with Dom_{strong}. The enumeration of all paths that possibly need to be checked is not promising to attain efficient separation procedures. Thus, when comparing the approaches that can handle ride-times in a labeling algorithm to one that cannot, it seems fair not to include in the latter case the ride-time constraints in that part of the separation procedures that account for pairing and precedence constraints. In Section 6, we will present results for two different versions of the separation procedures when using SPPPDPTW as subproblem. The first one integrates the ride-times in the all separation procedures and is marked with the subscript $_{sepRT}$ in the computational results. The other one considers ride-time constraints only in feasibility checks that do *not* incorporate the predecessor $\pi(S)$ and successor nodes $\sigma(S)$.

Care must also be taken when including the predecessor and successor information to identify infeasible paths in the case of the lifted IPEC (38). 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 (38), 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 (38).

C. Detailed Computational Results

The Tables 8 and 9 show the detailed computation times for the root node lower bounds. All times are given in seconds.

Instance	MP-I	MP-I^{cut}	MP-I$_{sepRT}$	MP-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 8: Computation times for root lower bounds and type `a` instances in seconds.

Instance	MP-I	MP-I^{cut}	MP-I$_{sepRT}$	MP-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 9: Computation times for root lower bounds and type `b` in seconds.