Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

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

Timo Gschwind, Stefan Irnich

To cite this article:
Timo Gschwind, Stefan Irnich (2015) Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem. Transportation Science 49(2):335-354. https://doi.org/10.1287/trsc.2014.0531

TRANSPORTATION SCIENCE
Vol. 49, No. 2, May 2015, pp. 335–354
ISSN 0041-1655 (print) | ISSN 1526-5447 (online)

http://dx.doi.org/10.1287/trsc.2014.0531

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

Timo Gschwind, Stefan Irnich
Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, D-55128 Mainz, Germany
{gschwind@uni-mainz.de, irnich@uni-mainz.de}

A dynamic time window relates to two operations that must be executed within a given time meaning that the difference between the points in time when the two operations are performed is bounded from above. The most prevalent context of dynamic time windows is when precedence is 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 precedence is the dial-a-ride problem (DARP), where user-specified transportation requests from origin to destination points must 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 (to our knowledge), both ordinary and dynamic time windows are handled in the column-generation subproblem. For the 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.

Keywords: transportation; vehicle routing; dynamic time windows; dial-a-ride problem; labeling algorithm; ride-time constraints; time synchronization; branch-and-price
History: Received: April 2013; revision received: November 2013; accepted: January 2014. Published online in *Articles in Advance* September 18, 2014.

1. Introduction

In vehicle routing and scheduling, an ordinary or static time window restricts the point in time when 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 needed to fulfill a request consisting of several working steps. In contrast, a *dynamic time window* relates to two operations: Both operations must be executed within a given time meaning that the difference between the points in time when the two operations are performed is bounded from above. Thus, the difference is bounded by the dynamic time window. The most prevalent context of dynamic time windows is when precedence is 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 precedence is the dial-a-ride problem (DARP), where user-specified transportation requests from origin to destination points must 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, Ravn, and Rygaard 1995; Toth and Vigo 1997; Borndörfer et al. 1997). To guarantee a certain level of service, the duration a passenger is on board the vehicle is either penalized or restricted by a so-called ride-time constraint. We consider the latter case, which is, obviously, an example for a dynamic time window. It refers to pickup and delivery operations, e.g., the origin can be the person's home and the destination a school, hospital or training workshop.

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

Intra-route synchronization constraints have additional applications in home care where patients must be monitored and attended to by nurses in given intervals (Eveborn, Flisberg, and Rönnqvist 2006). In the service industries, a service technician might need to come back to a location to finish a job that was begun earlier.

For example, perhaps a workpiece must dry, harden, rest, etc., before the next working step. Moreover, security guards must inspect facilities repeatedly with a guaranteed time between two inspections (Bredström and Rönnqvist 2008). Note that in these applications lower and upper bounds on the time difference can be present. In this paper we focus exclusively on the case of upper bounds given by dynamic time windows, e.g., maximum ride-time constraints in the DARP.

The contribution of this paper is, first and foremost, the presentation of an effective solution to VRPs with intra-route synchronization constraints. Specifically, we establish the following three results: First, we derive a new branch-and-cut-and-price solution for the DARP, the prototypical VRP with ordinary and dynamic time windows. For the first time (to our knowledge), both time-window and ride-time constraints are handled in the column-generation subproblem. The development of an effective column-generation pricing procedure that allows fast shortest-path computations based on new strong dominance rules is decisive. Second, checking for a given route whether there is a feasible time schedule with service times that simultaneously obey the ordinary and the dynamic time window constraints is intricate. We provide a labeling procedure that 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.

Next we elaborate on the three contributions in more detail. First, there are numerous examples of VRP variants for which highly successful exact solution approaches are based on integer column generation (e.g., Desrochers, Desrosiers, and Solomon 1992; Jepsen et al. 2008; Baldacci and Mingozzi 2009). The success of integer column generation can be attributed to the stronger lower bounds (we assume a minimization problem) that a column-generation (= extensive) formulation provides compared to an original formulation, from which the former is derived by Dantzig-Wolfe decomposition (Lübbecke and Desrosiers 2005). *No* lower bound improvement can be gained if the subproblem has the *integrality property* (Lübbecke and Desrosiers 2005). In many routing and scheduling applications, however, the subproblem is an elementary shortest-path problem with resource constraints (ESPPRC) and 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 allow paths visiting nodes more than once (resulting in an SPPRC). Both ESPPRC and SPPRC are typically solved using dynamic-programming labeling algorithms (Irnich and Desaulniers 2005). In early column-generation algorithms, SPPRC was the only viable approach because pseudo-polynomial algorithms are known in this case (Desrochers and Soumis 1988). Later on, exploitation of the trade-off between the hardness of the pricing problem and the quality of the lower bound led to several other ESPPRC relaxations such as SPPRC with 2-cycle and k-cycle free paths (Houck et al. 1980; Irnich and Villeneuve 2006), partial elementary paths (Desaulniers, Lessard, and Hadjar 2008), and ng-paths (Baldacci, Mingozzi, and Roberti 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. In the presence of two potentially conflicting temporal constraints (ordinary and dynamic time windows), traditional labeling algorithms are unable to check both. Thus, instead of including all tour constraints of the DARP in the subproblem, Ropke and Cordeau (2005) chose to relax the ride-time constraints. The resulting subproblem is well studied and can be solved effectively (Dumas, Desrosiers, and Soumis 1991; Ropke and Cordeau 2009). In their algorithm the ride-time constraints are handled with infeasible-path elimination inequalities in the master program. Our approach, in contrast, is to integrate both time-window and ride-time constraints into the column-generation subproblem. This generally provides stronger lower bounds at the cost of a harder to solve pricing problem. We derive two new dominance criteria leading to effective labeling procedures for this heretofore unsolvable subproblem.

Second, concerning feasibility testing of a given DARP route, Tang et al. (2010) presented an $\mathcal{O}(n^2)$ algorithm (where n is the length of the route), Haugland and Ho (2010) presented a faster $\mathcal{O}(n \log n)$ algorithm, and 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, none 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 based on (E)SPPRC subproblems. A byproduct of the approach proposed here is a $\mathcal{O}(nM)$ labeling-based feasibility check where M denotes the maximum number of open requests along a feasible route. As M is always bounded by the vehicle capacity and is typically small in real-world applications, the proposed labeling procedure is a step towards simple and efficient feasibility checking.

Third, we provide an extensive computational study comparing several exact solution approaches for the DARP including the two strongest approaches from the

literature (Ropke, Cordeau, and Laporte 2007; Ropke and Cordeau 2005). It is a priori unclear whether the additional effort of solving a subproblem that handles all DARP routing constraints pays off in the branch-and-bound tree. Thus, we implemented different branch-and-price algorithms based on two basic formulations that handle ride-time constraints in the subproblem or in the master program. In algorithmic variants, different classes of valid inequalities are added to the master program to strengthen the respective formulations, and different separation strategies are investigated. Moreover, for the new branch-and-cut-and-price approach that handles all routing constraints in the subproblem, two different labeling strategies for the shortest-path computations are compared. The detailed analysis shows that the new branch-and-cut-and-price algorithm outperforms all exact solution approaches from the DARP literature.

The rest of the paper is structured as follows: §2 provides a literature review on the DARP. In §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 §4. The basic components of the branch-and-cut-and-price solution algorithms are briefly discussed in §5 followed by computational experiments in §6. Conclusions and a future outlook are given in §7.

2. The Dial-a-Ride Problem

The DARP is a pickup-and-delivery VRP for passenger transportation where user specified transportation requests from origin to destination points must be served by a fleet of vehicles 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, to our knowledge, no common problem definition exists. For a comprehensive overview of DARP variants and solution approaches we refer to recent surveys (Cordeau and Laporte 2007; Parragh, Doerner, and Hartl 2008; Cordeau, Laporte, and Ropke 2008).

The DARP variant we consider below 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 the vehicle. In other words, the ride-time constraints impose a dynamic time window $[0, L_i]$ between pickup and delivery of the request i. As a generalization of many other VRPs the DARP is $\mathcal{NP}$-hard.

Heuristic approaches to 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, Doerner, and Hartl (2010) suggesting a variable neighborhood search. These and similar heuristic approaches are indispensable for solving huge instances of the DARP as they occur in practice: They often include more than 3,000 requests per day and heterogeneous fleets with small buses and dedicated taxis. For this reason, ride-time constraints are relaxed in practice and partially enforced by appropriately adjusting regular time windows.

For exact solution approaches, Cordeau (2006) developed a branch-and-cut algorithm based on a three-index formulation. Maximum route duration constraints are considered in the model, but they coincide with the time windows of the depots in the benchmark instances. Thus, they are not explicitly present there. Known inequalities from the TSP, VRP(TW), and PDPTW were adapted to the DARP and used along with new inequalities valid for the DARP. The algorithm solved 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, Cordeau, and Laporte (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, Cordeau, and Laporte (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. Node 0 denotes the origin and node $2n+1$ the destination depot, $P=\{1,\dots,n\}$ the pickup nodes, and $D=\{n+1,\dots,2n\}$ 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 must be picked up together. The same passengers are dropped at the delivery node $i+n$, indicated by the negative demand $d_{i+n}=-d_i$. Furthermore, a nonnegative service time s_j (with $s_0=s_{2n+1}=0$) and a time window $[a_j,b_j]$ in which the service must be started are given for each node $j\in N$. This means that arriving before a_j is allowed, in which case the vehicle must 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 cannot be started after b_j. The possibility to delaying the start of service at some nodes is crucial for the feasibility of routes in the presence of maximum ride times (see §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 nonnegativity holds for both. We assume that from the arc set A at least the obviously infeasible arcs are excluded (see §5). Thus, the arc set A is a subset of

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

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 must be visited on the same route, and i must 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 must 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.

Unlike Cordeau and Laporte (2003), we do not impose an explicit bound on the maximum duration of a route. 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 (MIP), 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 be 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 $\delta^-(i)=\{(h,i)\in A\colon h\in N\}$ and $\delta^+(i)=\{(i,j)\in A\colon j\in N\}$, respectively. We use a condensed notation, where for the vector $x^k\in\{0,1\}^A$ and any subset $B\subseteq A$, the term $x^k(B)$ means $\sum_{b\in B}x_b^k$.

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

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

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

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

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

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

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

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

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

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

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

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

The objectives are to minimize the total routing costs (1). Constraints (2), guarantee that each request 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 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 by the time windows (6), and load variables by the vehicle capacity (8). The ride times are defined by Equation (9) and are bounded by the maximum ride times (10). Because of the nonnegativity of the ride times constraints, constraints (10) also impose the precedence. This three-index formulation (1)–(11) is nonlinear because of constraints (5)

and (7). A linearization, however, is straightforward (see, e.g., Cordeau 2006 or Ropke, Cordeau, and Laporte 2007), but useless if the subproblem is solved with dynamic programming.

Ropke, Cordeau, and Laporte (2007) present two tighter problem formulations. Using appropriate formulated precedence inequalities, they can ensure pairing and precedence without the use of an additional index for the vehicle and thus can formulate the DARP with a two-index model. These MIPs 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 intensively researched (Dumas, Desrosiers, and Soumis 1991; Desaulniers et al. 2002; Ropke and Cordeau 2009); we can rely on this research because it is a relaxation of the DARP.

Another possibility for modeling the ride-time constraints is to replace (9)–(10) with *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

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

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 to solve the traveling salesman problem (TSP) with time windows (Ascheuer, Fischetti, and Grötschel 2000), but offer a universal approach for handling (complex) constraints in routing models that consist solely of routing variables. On the downside, because IPEC generally form an exponential family of valid inequalities, they cannot be included en masse when solving the MIP. Violated IPEC must 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 the IPEC (12) are all noncoupling constraints as they refer to each vehicle individually. A straightforward Dantzig-Wolfe decomposition has partitioning 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 partitioning constraints (2) and the IPEC (12) in the master program. Because IPEC replaces 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 PDPTW-feasible routes where the set of all such routes is Ω^{PDPTW}.

In the master programs, the cost of a route $r \in \Omega^{\text{DARP}}$ or $r \in \Omega^{\text{PDPTW}}$ is c_r, respectively. For each request $i \in P$ and each route r, let $a_{ir} \in \mathbb{Z}_+$ be the number of times route r performs request i. In an elementary route, a_{ir} is binary. However, we also allow relaxations where nonelementary 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 here presented side-by-side:

(IMP-I)	(IMP)	
$\min \sum_{r\in\Omega^{\text{PDPTW}}} c_r\lambda_r$	$\min \sum_{r\in\Omega^{\text{DARP}}} c_r\lambda_r$	(13)
s.t. $\sum_{r\in\Omega^{\text{PDPTW}}} a_{ir}\lambda_r = 1, \quad \forall i \in P,$	s.t. $\sum_{r\in\Omega^{\text{DARP}}} a_{ir}\lambda_r = 1, \quad \forall i \in P,$	(14)
$\sum_{r\in\Omega^{\text{PDPTW}}} \lambda_r = \lvert K\rvert,$	$\sum_{r\in\Omega^{\text{DARP}}} \lambda_r = \lvert K\rvert,$	(15)
$\sum_{r\in\Omega^{\text{PDPTW}}} b_{Ir}\lambda_r \le \lvert I\rvert - 1, \quad \forall I \in \mathcal{I},$		(16)
$\lambda_r \in \{0,1\}, \quad \forall r \in \Omega^{\text{PDPTW}}.$	$\lambda_r \in \{0,1\}, \quad \forall r \in \Omega^{\text{DARP}}.$	(17)

The objective (13) is the minimization of the total costs. The pendant to request partitioning 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 also add the suffix -I to identify its components). The approach presented in this paper relies on the right-hand side formulation. As the number of feasible routes in the masters is too large to solve directly, we 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. Regrettably, this advantage of a stronger model makes the subproblem harder to solve. Next we develop an effective solution for the MP subproblem. We also show empirically that, with the proposed subproblem algorithm, the trade-off (between bounds and effort to gain the bounds) clearly favors 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 $\rho=(\rho_I)_{I\in\mathcal{J}}$ be the values of the dual variables to the RMP constraints (14), (15), and (16), respectively. Let $b_{I,ij}$ be the number of times that arc $(i,j)\in A$ is present in the path $I\in\mathcal{J}$. To identify feasible routes with negative reduced costs, we must first compute the reduced costs of the routing variables x_{ij} as follows:

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

Then the following subproblem must be solved:

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

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

A final remark about the IMP-I subproblem for the generation of PDPTW routes seems appropriate here. The advantage of replacing (9)–(10) with IPEC, as suggested by Ropke and Cordeau, is that the reduced costs (18) in PP-I depend solely on the chosen arcs, but not on a route's schedule, i.e., values T_i^k in (3)–(11). If schedules were involved, Desaulniers et al. (1998) have explained that highly complex variants of SPPRC would result, where linear node costs and profits must be included (Ioachim et al. 1998).

Subproblem Solution by Dynamic-Programming Labeling Algorithms. The basic principle of a labeling algorithm is the following: Starting at a distinct source node, partial paths are iteratively extended along arcs of the underlying graph until the sink node is reached and the path is complete. The partial paths are represented by labels in which attributes such as the accumulated cost or time consumption along the partial path are stored. To 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. A more detailed discussion of general dominance principles is provided in Desaulniers et al. (1998), Irnich and Desaulniers (2005) and in the introductory paragraph of the proofs in Online Appendix A (available as supplemental material at http://dx.doi.org/10.1287/trsc.2014.0531).

Next 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 no longer valid. 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 nondecreasing resource extension functions (REFs).

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

In the SPPPDPTW, a feasible path must satisfy time-window, capacity, pairing, and precedence constraints. For a nonelementary 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 nonelementary paths.

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

A general prerequisite for the following dominance rule is that the reduced costs $\tilde{c}$ fulfill $\tilde{c}_{ij}\le\tilde{c}_{ik}+\tilde{c}_{kj}$ for all arcs $(i,j),(i,k),(k,j)\in A$ with $k\in D$. Ropke and Cordeau (2009) call this property the *delivery triangle inequality* (DTI). It may be fulfilled by definition (18) and permits a more efficient solution of the PDPTW pricing problem (Dumas, Desrosiers, and Soumis 1991). However, adding additional cuts to the RMP formulated in the x_{ij} variables may lead to reduced costs that do not satisfy the DTI. Ropke and Cordeau (2009) show how to transform the reduced-cost matrix into an equivalent that satisfies the DTI. Hereafter, we assume that the DTI holds. In particular, the validity of the following

and all other dominance rules in this paper is based on this assumption.

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

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

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

Ropke and Cordeau (2009) discussed three additional aspects of Dom_{PDPTW}. First, for an extension to the elementary case one just needs to keep track of the set of serviced requests U for each label and require $U_{\ell_1}\subseteq U_{\ell_2}$ in the dominance rules. Second, if the DTI does not hold dominance is only possible between labels with identical sets $O_{\ell_1}=O_{\ell_2}$ (and $U_{\ell_1}=U_{\ell_2}$ in the elementary case). Third, when column generation is embedded into branch-and-bound, the use of Dom_{PDPTW} as dominance criteria 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 subpath. In this case, it is not possible to meet the DTI.

PDPTW and Ride-Time Constraints. We now show that ride-time constraints are not compatible with the standard SPPPDPTW dominance rule. The main difficulty is to address 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. The following example illustrates this.

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, ℓ_1 seems to dominate ℓ_2. However, this is not true for the SPPDARP. If the maximum ride time of requests i is $35\le L_i<40$, ℓ_2 can be feasibly extended to delivery node $i+n$, while ℓ_1 cannot because 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 (§4.3). On one hand, the weak dominance provides some useful insights about basic information needed for applying any dominance rule, later used for the development of the strong dominance rules. On the other hand, there is a trade-off between the strength of the rule and the effort needed to implement and perform the comparison of labels: The weak dominance rule is easy to understand, simple to implement, and computationally cheap. Section 6 compares 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. This is not an issue if there are no *active* ride-time constraints at a label. Consequently, whenever a label ℓ is feasible and represents an empty vehicle at the current node $\eta_\ell\in D$, i.e., $O_\ell=\varnothing$, it can dominate other labels according to the rule Dom_{PDPTW}.

When extending a label with $O_{\ell'}=\varnothing$ to a customer node, this needs to be a pickup node $\eta_\ell\in\mathcal{P}$. The only active ride-time constraint for the resulting label ℓ is that of the request picked up at the current node η_ℓ. Thus, there are no ride times connecting the preceding part of the path with η_ℓ, which allows for delay of the start of service at η_ℓ for ℓ to at least the same time as 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*

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

Table 1 Label ℓ_1 Dominates Label ℓ_2 in the SPPPDPTW Sense, But Is Inferior Regarding the Ride-Time Constraint of Request i

Labels	Nodes in $\mathcal{P}(\ell_1)$, $\mathcal{P}(\ell_1')$	0	j	i	$j+n$	η	$i+n$
ℓ_1 for path $(0,\ldots,\eta)$	Time window $[a_i,b_i]$	[0,100]	[0,15]	[20,35]	[15,30]	[50,65]	[60,75]
ℓ_1' for path $(0,\ldots,i+n)$	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 $(0,\ldots,\eta)$	Time window $[a_i,b_i]$	[0,100]	[20,35]	[20,35]	[45,60]	[50,65]	[60,75]
ℓ_2' for path $(0,\ldots,i+n)$	Nodes in $\mathcal{P}(\ell_2)$, $\mathcal{P}(\ell_2')$	0	h	i	$h+n$	η	$i+n$

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 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 insufficient to decide on the label's feasibility.

Example 2. Consider label ℓ'_2 from Example 1. When arriving at node $i+n$, it is unclear whether the ride-time constraint of request i is satisfied (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 $L_i = 30$, ℓ'_2 appears to be infeasible as the actual ride time for i is $65-30=35$. Yet, it is possible to voluntarily delay the start of service at node i for 5 units. This reduces the actual ride time of request i from 35 to 30 (without affecting the service times of the succeeding nodes), which means that ℓ'_2 does represent a feasible path.

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

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

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

$$\tau_i \in [a_{h_i}, b_{h_i}], \quad \forall i = 1, \dots, q, \tag{21}$$

$$\tau_i + s_{h_i} + t_{h_i, h_{i+1}}, \le \tau_{i+1}, \quad \forall i = 1, \dots, q-1, \tag{22}$$

$$\tau_i + s_{h_i} + L_{h_i} \ge \tau_j \quad \text{if } h_i + n = h_j. \tag{23}$$

The time schedule $T_{\mathcal{P}}$ is then said to be feasible. Note again that elementarity of $\mathcal{P}$ is assumed. Otherwise, inequalities (23) must be satisfied for every pair of corresponding pickup and delivery nodes.

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

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

$$\begin{aligned} \eta_{\ell'} &= x, \quad \tilde{c}_{\ell'} = \tilde{c}_\ell + \tilde{c}_{\eta_\ell, x}, \\ t_{\ell'} &= \max\{a_x, t_\ell + s_{\eta_\ell} + t_{\eta_\ell, x}\}, \quad l_{\ell'} = l_\ell + d_x, \end{aligned} \tag{24}$$

$$O_{\ell'} = \begin{cases} O_\ell \cup \{x\}, & \text{if } x \in P, \\ O_\ell \setminus \{x-n\}, & \text{if } x \in D. \end{cases} \tag{25}$$

The pairing constraints enable the elimination of labels whenever there exists no feasible extension to the destination depot $2n+1$. Rules for label elimination were formulated in Dumas, Desrosiers, and Soumis (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, ℓ can be eliminated. When travel times satisfy the triangle inequality, this comes down to solving a TSP with time windows over the nodes $\{i+n \in D\colon i \in O_\ell\} \cup \{\eta_\ell, 2n+1\}$, which is $\mathcal{NP}$-hard. Thus, Dumas, Desrosiers, and Soumis (1991) consider only subsets of O_ℓ with not more than two requests. We follow the same approach, but also 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, e.g., the order in which they have been picked up, can be used. 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 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) the ride-time constraints can be ignored in the dominance relation between two labels. In general, however, the ride times of open requests must 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 must be known. In particular, whenever the open requests $i \in O_{\ell_1}$ can be ride-time feasibly delivered in a completion Q of ℓ_2 it must also be possible to feasibly deliver them for ℓ_1 using the completion Q' that results from Q by leaving out those deliveries that belong to nonopen requests of $\mathcal{P}(\ell_1)$ (i.e., open in $\mathcal{P}(\ell_2)$, but not open in $\mathcal{P}(\ell_1)$). As the ride-time constraints only bound the maximum time on board, delivering early is never a problem, but delivering late can be. Thus, the latest possible delivery times, still respecting the maximum ride times of the open requests, are of importance. As seen in Example 2, the possibility of delaying the start of service at pickup nodes has to be incorporated. Consequently, we must always 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 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 to maximize the latest delivery times of open requests. However, we need to ensure that it is always, i.e., for any feasible time the current node η_ℓ can be serviced, possible that a dominating label ℓ_1 can deliver its open requests if a dominated label ℓ_2 can do also. We show (in Proposition 3) 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 criteria, we therefore store, in addition to the earliest start of service t_ℓ at the current node η_ℓ, for all open request $i \in O_\ell$ the *latest possible delivery time* $ld^i_\ell(t)$ as a function of the start of service t at η_ℓ. In other words, $ld^i_\ell(t)$ is the latest delivery time for request i, i.e., the latest ride-time feasible start of service at the delivery node $i+n$, where the start of service at its pickup node i has been delayed as much as possible. No constraints that are already known for the partial path up to η_ℓ are violated, and the start of service at η_ℓ is not delayed beyond t.

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

$$ld^{h_i}_\ell(t) = \min\left\{ b_{h_i+n}, \max_{T_{\mathcal{P}(\ell)}=(\tau_1,\dots,\tau_q)\in\mathcal{T}_{\mathcal{P}(\ell)}(t)} \{\tau_i\} + s_{h_i} + L_{h_i} \right\}. \quad (26)$$

To maximize τ_i, generally, the starts of service τ_j at several nodes h_j of the path $\mathcal{P}$ must be changed compared to the respective earliest starts of service. However, this does not lead to any conflicts when the latest delivery times must 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 $\mathcal{P} = (h_1, \dots, h_q)$ be a feasible partial path, and let $\tau^*_i(t) = \max_{T_{\mathcal{P}}=(\tau_1,\dots,\tau_q)\in\mathcal{T}_{\mathcal{P}}(t)}\{\tau_i\}$ be the maximum value for the start of service at node h_i with $t \le \tau_q$. Then, $T^*_{\mathcal{P}}(t) = (\tau^*_1(t), \dots, \tau^*_q(t)) \in \mathcal{T}_{\mathcal{P}}(t)$.*

Proofs of this and all other lemmas and propositions are provided in Online Appendix A.

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*

$$\eta_{\ell_1} = \eta_{\ell_2}, \quad \tilde{c}_{\ell_1} \le \tilde{c}_{\ell_2}, \quad t_{\ell_1} \le t_{\ell_2}, \quad O_{\ell_1} \subseteq O_{\ell_2}, \quad \text{and} \quad (27)$$

$$ld^i_{\ell_1}(t) \ge ld^i_{\ell_2}(t), \quad \forall i \in O_{\ell_1}, t \in [t_{\ell_2}, b_{\eta_{\ell_2}}]. \quad (28)$$

Equation (28) ensures that for all open requests $i \in O_{\ell_1}$ of the dominating label ℓ_1 and for each time t that may represent a feasible start of service at η_{ℓ_2} for ℓ_2, it is feasible to deliver request i for ℓ_1 whenever it is feasible for ℓ_2. Note that the functions $ld^i_\ell(t)$ 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 Equation (28) is that it is not necessary for the start of service of the dominating label ℓ_1 to actually reach all feasible times t for the start of service of ℓ_2, as long as $ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t)$ holds for all $i \in O_{\ell_1}, t \in [t_{\ell_2}, b_{\eta_{\ell_2}}]$.

Note that it is straightforward to extend Dom^*_{strong} to a valid dominance rule for the DARP subproblem 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 on the maximum route duration.

Because the dominance rule of Proposition 3 requires the comparison of several functions in (28), the dominance relationship is practically not yet applicable in a labeling algorithm. We therefore characterize the shape of the functions $ld^i_\ell(t)$ 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_ℓ and open requests O_ℓ. The functions $ld^i_\ell(t), t \geq t_\ell$ 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.*

The visualization of the result of Proposition 4 together with notation is given in Figure 1. Using Proposition 4, Dom^*_{strong} can be simplified to a dominance rule where instead of having to compute 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_ℓ and time B^i_ℓ where the latter is the time when $ld^i_\ell(t)$ becomes constant. This choice also enables an elegant way to compute the respective values of $ld^i_\ell(t)$ using REFs (see §4.4). Then a simplified version of Dom^*_{strong} is as follows (see Figure 2):

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

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

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}. Next we only describe additional aspects. According to Proposition 5, the two additional resources $ld^i_\ell(t_\ell)$ and $ld^i_\ell(B^i_\ell)$ for each open request $i \in O_\ell$ must be stored within each label ℓ. Furthermore, the computation of

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

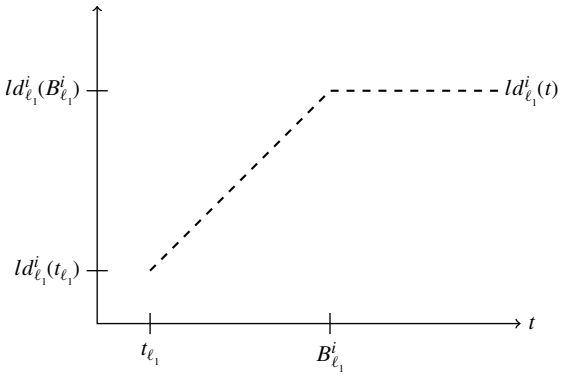

Figure 1 **Resources of a Label ℓ Used Within the Labeling Algorithms**

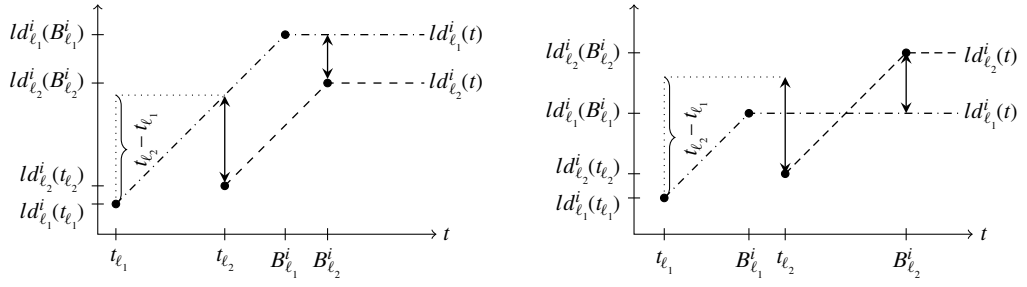

Figure 2 **Dominance Condition for ld^i Fulfilled (Left Panel) and Not Fulfilled (Right Panel)**

$ld^i_\ell(B^i_\ell)$ requires carrying along B^i_ℓ for each $i \in O_\ell$ as an additional resource.

With $ld^i_\ell(B^i_\ell)$, precise information on the actual ride times of the open requests $i \in O_\ell$ and thus on the feasibility of a label ℓ is available. Specifically, the extension of ℓ along an arc (η_ℓ, x) can only be feasible if $t_\ell + s_{\eta_\ell} + t_{\eta_\ell,x} \le ld^i_\ell(B^i_\ell)$ 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 $ld^i_\ell(t_\ell)$, $ld^i_\ell(B^i_\ell)$, and B^i_ℓ for all open request $i \in O_\ell$ are either initialized or updated according to the following REFs*:

$$B^i_{\ell'} = \begin{cases} \min\{\tilde{b}_x, b_{x+n} - s_x - L_x\}, & \text{if } i = x, \\ \max\{t_{\ell'}, \min\{B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x}, \tilde{b}_x\}\}, & \text{otherwise}, \end{cases} \tag{29}$$

$$ld^i_{\ell'}(t_{\ell'}) = \begin{cases} \min\{t_{\ell'} + s_x + L_x, b_{x+n}\}, & \text{if } i = x, \\ ld^i_\ell(t_\ell) + (\min\{t_{\ell'} - s_{\eta_\ell} - t_{\eta_\ell,x}, B^i_\ell\} - t_\ell), & \text{otherwise}, \end{cases} \tag{30}$$

$$ld^i_{\ell'}(B^i_{\ell'}) = \begin{cases} B^i_{\ell'} + s_x + L_x, & \text{if } i = x, \\ ld^i_\ell(B^i_\ell) - \max\{0, B^i_\ell + s_{\eta_\ell} + t_{\eta_\ell,x} - \tilde{b}_x\}, & \text{otherwise}, \end{cases} \tag{31}$$

where

$$\tilde{b}_x = \begin{cases} b_x, & \text{if } x \in P, \\ \min\{b_x, ld^{x-n}_\ell(B^{x-n}_\ell)\}, & \text{if } x \in D. \end{cases} \tag{32}$$

A detailed example of applying the labeling algorithm with the strong dominance rule is provided in Online Appendix B.

Label elimination is performed in the same way as 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 determine 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, the number of resources at each node and the complexity of the updates (24)–(25) and (29)–(31) is $\mathcal{O}(M)$. Hence, the feasibility test requires $\mathcal{O}(nM)$ time and space, where n is the length of the route. Assuming integer values for node demands d and vehicle capacity C (which seems natural for passenger transportation), then, e.g., $M \le C$ holds. The travel and service times combined with the time windows allow similar estimations. Thus, feasibility testing with the labeling procedure is pseudo-linear.

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

The only point where the labeling algorithm with Dom_{strong} of §4.4 uses 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 for improving the dominance Dom_{strong}. More precisely, let ℓ' result from the extension of a label ℓ to node x so that $\mathcal{P}(\ell') = (h_1, \ldots, h_q = x)$. When computing the values $B^{h_i}_{\ell'}$ and $ld^i_{\ell'}(B^{h_i}_{\ell'})$, we consider inequalities (21) and (22), and inequalities (23) for requests already picked up and delivered. No information on the ride-time constraints of the open requests is included. Instead of ignoring these maximum ride times, they can be used to impose additional constraints on the service time τ_q of the last node $x = h_q$. Consequently, each feasible partial schedule must respect $\tau_q + s_x + t_{x,h_i+n} \le ld^i_\ell(B^{h_i}_\ell)$ for all $h_i \in O_{\ell'}$ to be feasibly completed. These constraints bound τ_q. Using this bound in Equation (32), a tighter upper bound $\hat{b}_x$ instead of $\tilde{b}_x$ can be computed. As a result, the values $B^i_{\ell'}$ and $ld^i_{\ell'}(B^i_{\ell'})$ may also decrease.

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

To summarize, the labeling algorithm is then altered as follows: Prior to the propagation of the resources $B^i_{\ell'}$, $ld^i_{\ell'}(t_{\ell'})$, and $ld^i_{\ell'}(B^i_{\ell'})$ an upper bound $\hat{b}_x$ on the

latest feasible start of service at the current node x is computed. In the REFs (29)–(31), the value $\tilde{b}_x$ is then replaced by $\hat{b}_x$. Algorithm 1 outlines the label-extension and label-elimination step of the refined labeling algorithm with strong dominance.

Algorithm 1 (Label-Extension and Label-Elimination Step of the Refined Labeling Algorithm with Strong Dominance)

input: Label ℓ, arc (η_ℓ, x)
output: New label ℓ' resulting from the extension of ℓ along arc (η_ℓ, x), if feasible
if *Extension of* ℓ *along* (η_ℓ, x) *is feasible* **then**
  Create label ℓ'
  Compute $\eta_{\ell'}$, $\tilde{c}_{\ell'}$, $t_{\ell'}$, $l_{\ell'}$, and $O_{\ell'}$ according to (24)–(25)
  Determine refined bound $\hat{b}_x$ on the latest start of service at x
  for *all* $i \in O_{\ell'}$ **do**
    Compute $B^i_{\ell'}$, $ld^i_{\ell'}(t_{\ell'})$, and $ld^i_{\ell'}(B^i_{\ell'})$ according to (29)–(31)
  Perform label elimination using information $ld^i_{\ell'}(B^i_{\ell'})$ of open requests $i \in O_{\ell'}$.

Bounding the start of service at the current node has two positive effects on the dominance rule. First, the dominance relation is less restrictive because it is a relaxed condition to require $ld^i_{\ell_1}(t) \geq ld^i_{\ell_2}(t)$ for all $t \in [t_{\ell_2}, \hat{b}_{\eta_{\ell_2}}]$ rather than for all $t \in [t_{\ell_2}, \tilde{b}_{\eta_{\ell_2}}]$. Second, the possible bounding effect on B^i_ℓ and $ld^i_\ell(B^i_\ell)$ gets stronger when 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 compared in §6. Mainly, we analyze the difference between using the two formulations IMP-I and IMP. For a fair comparison, we do not only compare with the results of Ropke and Cordeau (2005) directly, but we also implement versions for solving IMP-I, i.e., where the pricing problem is an SPPPDPTW and ride-time constraints are enforced on the master problem level. The use of formulation IMP requires that a SPPDARP subproblem is solved, and we also compare the two new labeling algorithms of §§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, Desrosiers, and Solomon 1992; Dumas, Desrosiers, and Soumis 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 must be invoked such as those described in §4. Our implementation comes with two straightforward pricing heuristics. The first is solving the pricing problem on a reduced network only. The second 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}. Speed ups are small, however, when using Dom_{strong} for SPPDARP or when solving the SPPPDPTW.

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

$$x_{ij} = \sum_{k \in K} x^k_{ij} = \sum_{r \in \Omega} a_{ij,r} \lambda_r, \quad \forall (i,j) \in A, \tag{33}$$

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

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

Moreover, some separation procedures rely on checking ride-time constraints. Those that use an SPPDARP-based separation procedure, i.e., ride-time constraints are comprehensively taken into account, are indicated with an index sepRT. Details on the different separation procedures are provided in Online Appendix C.

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 (33) we branch on the number $\bar{x}(\delta^+(0))$ of vehicles if fractional, and create the two branches $x(\delta^+(0)) \le \lfloor \bar{x}(\delta^+(0)) \rfloor$ and $x(\delta^+(0)) \ge \lceil \bar{x}(\delta^+(0)) \rceil$. This rule was proposed by Desrochers, Desrosiers, and Solomon (1992). Second, we branch on the outflow of a node set S as proposed by Naddef and Rinaldi (2002) where we restrict the choice of S to sets of size two. A set S with outflow $\bar{x}(\delta^+(S))$ closest to 1.5 is chosen and the two branches $x(\delta^+(S)) \le 1$ and $x(\delta^+(S)) \ge 2$ are created. For each branch of either branching decision, a linear constraint is added to the master program. No structural changes must 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, Cordeau, and Laporte (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 eight 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, Cordeau, and Laporte (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 Online Appendix C, Equations (EC.1) and (EC.2)) to ensure the ride-time constraints, but no additional cuts. Because the ride-time constraints are not handled in the pricing problem, the resulting subproblem is a SPPPDPTW that 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$^{\text{cut}}$ refers to the version where 2-path cuts, rounded capacity inequalities, and fork inequalities are separated (see Online Appendix C). Following the notation of §5, IMP-I$_{\text{sepRT}}$ and IMP-I$^{\text{cut}}_{\text{sepRT}}$ integrate the ride-time constraints in all separation procedures, while IMP-I and IMP-I$^{\text{cut}}$ do not.

The approaches based on IMP where the subproblem is an SPPDARP solved using the proposed labeling algorithm with Dom_{weak} and Dom_{strong} are denoted by IMP$_{\text{weak}}$ and IMP$_{\text{strg}}$, respectively. As before, a superscript cut indicates that cuts are used, giving rise to IMP and IMP$^{\text{cut}}$. All algorithms are summarized in Table 2.

In preceding experiments, we found only small improvements in the root lower bounds when the elementary variants of the pricing problems replace the nonelementary. Because of the hardness of the elementary subproblem the labeling algorithms are slower. Overall, this results in slightly longer computation times for all of the branch-and-price algorithms. These findings coincide with those of Ropke and Cordeau (2005). Consequently, we decided not to report any computational results for the elementary case.

All of our algorithms were coded in C++ using the callable library of CPLEX 12.2 to re-optimize the RMP. Arc costs and travel times are computed with double precision. For stability, we replaced partitioning constraints (14) with covering constraints (du Merle et al. 1999). The computations were carried out on a standard PC with an Intel Core 2 Duo E8400 at 3 GHz with 4 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 3 and 4 show details for the lower bound values *lb* and summarize the computation times. The meaning of the notations in the tables is as follows:

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 are provided in Online Appendix D. Note that for B&C and B&P-RC the computation times were reported for different computers so that a direct comparison is critical. According to http://www.spec.org/cpu2006/results our computer is approximately twice as fast as an AMD Opteron 254 which is in turn about

Table 2 Overview of DARP Solution Approaches Used for Comparison

Algorithm	Acronym	B&C (Ropke, Cordeau, and Laporte 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$^{cut}_{strg}$
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 and sequences	sepRT	?	?			•	•	•	•	•	•

Table 3 Lower Bound Values *lb* for Type a Instances, ∗ If Lower Bound = Opt

Instance	Opt	B&C (Ropke, Cordeau, and Laporte 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	1,033.4	1,025.2	1,028.6	1,022.9	1,029.2	1,024.4	1,032.0	1,029.8		∗	
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	1,229.7	1,213.4	1,224.5	1,205.3	1,212.8	1,209.1	1,225.1	1,228.8		∗	
# 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

Note. Note that MP$_{weak}$ and MP$_{strg}$ as well as MP$^{cut}_{weak}$ and MP$^{cut}_{strg}$ have identical lower bounds.

15% faster (http://www.spec.org/cpu2000/#Results) than the AMD Opteron 250 used by Ropke, Cordeau, and Laporte (2007) and Ropke and Cordeau (2005).

Lower Bounds. The computed lower bound values *lb* for linear relaxations show that integrating ride-time constraints into the subproblem yields significantly stronger lower bounds compared to handling ride times in the master program. This holds for 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 is mainly because 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

Table 4 Lower Bound Values *lb* for Type b Instances, ∗ If Lower Bound = Opt

Instance	Opt	B&C (Ropke, Cordeau, and Laporte 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	1,203.4	1,193.2	1,202.0	1,201.9	1,202.0	1,201.9	1,202.0	1,201.3		1,202.0	
b8-64	839.9	834.7	836.9	836.4	837.4	836.4	838.0	836.6		838.1	
b8-80	1,036.3	1,032.6	1,036.2	1,035.7	1,035.7	1,035.7	1,036.2	1,035.9		1,036.2	
b8-96	1,185.6	1,165.1	1,181.5	1,181.4	1,181.9	1,181.4	1,182.2	1,181.3		1,183.8	
# 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

Note. Note that MP_{weak} and MP_{strg} as well as MP^{cut}_{weak} and MP^{cut}_{strg} have identical lower bounds.

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 11 of the type a instances in the root node. For the type b instances, these numbers decrease to 10 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. Because 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. In addition, the necessity to repeatedly solve pricing problems and separation problems to achieve an optimal ride-time feasible solution imposes that many more pricing and separation problems must be solved. This complicates and slows down the re-optimization of the master program. As a result, 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 still seem acceptable.

Next we compare computation times for MP-I_{sepRT} and 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, computation times also become longer, particularly for MP-I^{cut}_{sepRT}. There are a number of reasons for this. 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 MP-I^{cut}_{sepRT}. Interestingly, for the type b instances, the opposite is true for the solution times of 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, fewer pricing problems must be solved.

Extended Instances. Standard instances from the literature are characterized by narrow time windows and small capacities. Consequently, solutions comprise relatively short routes with a small number of requests that are open at the same time so that the new approach either solves the instances in the root node or with a small integrality gap. Next we analyze the behavior of the two most powerful approaches, MP-I^{cut}_{sepRT} and MP^{cut}_{strg}, in instances with fewer tight constraints.

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

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

6.2. Integer Solution Results

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

Our approach IMP^{cut}_{strg} with the SPPDARP subproblem and cuts solved all 42 instances from the benchmark set. Without using the cuts, IMP_{strg} fails on instance b8-96. The only other approach to solving all of 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 constraints) IMP-I and IMP-I_{sepRT} fail on four additional instances.

For the instance a8-96, we computed an optimal integer solution with value 1,229.66 that differs from the value 1,232.61 reported in Ropke, Cordeau, and Laporte (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 10 instances with IMP^{cut}_{weak} and 12 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, 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. Clearly, the weak dominance rule is too weak to solve larger instances.

Table 5 Linear Relaxation Results for Extended Instances

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

Table 6 Computation Times for Optimal Integer Solutions of Type a Instances in Seconds

Instance	B&C (Ropke, Cordeau, and Laporte 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	1 h	1 h	1 h	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	1,721.6	1 h	1 h	1 h	686.0	1 h	1 h	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	1,436.5	1 h	2,686.7	1 h	146.9	1 h	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	2 h	1 h	1 h	1 h	1 h	1 h	395.7	67.4	43.0
a8-96	4,233.0	2 h	1 h	1 h	1 h	1 h	1 h	178.4	20.5	11.1
# opt	21	19	16	17	16	19	17	20	21	21
Avg. time	312.4	884.7	994.2	912.1	898.7	410.3	716.2	204.9	14.6	5.1

Table 7 Computation Times for Optimal Integer Solution of Type b Instances in Seconds

Instance	B&C (Ropke, Cordeau, and Laporte 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	1 h	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	1 h	1 h	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	1,024.2	402.7	468.8	230.8	600.3	212.9	1 h	1 h	136.1	31.4
b7-56	807.6	66.2	32.1	10.8	26.7	9.3	1 h	1 h	30.4	50.3
b7-70	127.2	65.1	42.9	12.7	158.6	9.4	1 h	1 h	163.7	13.0
b7-84	396.6	237.6	30.4	45.4	75.4	36.8	1 h	1 h	42.0	71.7
b8-64	150.0	105.1	92.1	43.4	31.3	63.7	1 h	1 h	36.7	23.1
b8-80	183.0	73.1	17.7	20.8	18.7	9.6	1 h	1 h	14.4	10.3
b8-96	7,205.4	3,403.5	1 h	1 h	1 h	3,031.6	1 h	1 h	1 h	898.8
# opt	21	21	20	20	20	21	13	13	20	21
Avg. time	484.2	220.5	211.6	224.4	256.1	172.5	1,403.9	1,543.2	193.2	53.9

Computation Times. We now discuss computation times for integer solutions 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 combined with an effective labeling algorithm make solving these instances seemingly easy. For example,

Table 8 Integer Solution Results for Extended Instances

	Orig.		4/3		5/3		6/3	
	MP-I^{cut}_{sepRT}	MP^{cut}_{strg}	MP-I^{cut}_{sepRT}	MP^{cut}_{strg}	MP-I^{cut}_{sepRT}	MP^{cut}_{strg}	MP-I^{cut}_{sepRT}	MP^{cut}_{strg}
				Type a instances				
# opt	19	21	15	21	13	20	9	21
Avg. time	410.3	5.1	1,121.1	28.1	1,549.2	220.1	2,064.4	187.2
				Type b instances				
# opt	21	21	16	21	11	17	9	15
Avg. time	172.5	53.9	917.8	237.9	1,860.7	699.7	2,206.8	1,197.3

with IMP^{cut}_{strg} all but three instances can be solved within 10 seconds (exceptions are a6-72: 24.4 s, a8-80: 43.0 s, a8-96: 11.1 s). 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 faster by at least a factor of 10 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 in instances that are already solved in the root node.

To summarize, 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, integration of the ride-time information in all separation procedures leads to overall faster computations. This is particularly true for type a instances. Note that the best strategy for all approaches is to separate violated inequalities only at the root node.

Extended Instances. We use the same three sets of extended benchmark instances as for the linear relaxation results in §6.1. Online Appendix E provides detailed integer computational results for these instances that we have summarized in Table 8. Again, the more the constraints are relaxed, the harder the instances. The new algorithm MP^{cut}_{strg} can solve all instances except for one type a, while MP-I^{cut}_{sepRT} fails in 28 instances. For type b instances, the former fails in 10 and the latter in 27 instances. It seems that the new approach MP^{cut}_{strg} is more advantageous for the type a instances, for which ride-time constraints are more binding.

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 personnel (technicians, security guards, etc.), and in home care. In a recent survey, Drexl (2012) outlined a growing interest in vehicle routing with synchronization constraints. The work presented here can be seen as the central building block for handling dynamic time windows of the form $[0, L_i]$ to synchronize two operations i and $i+n$.

The DARP, in the variant where the service level is controlled by means of ride-time constraints, is the prototypical VRP with dynamic time windows. We proposed a new column-generation based solution approach where for the first time (to our knowledge) both time-window and ride-time constraints are handled in the subproblem. The detailed computational study has shown that this approach outperforms all other exact solution techniques based either 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 to 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 core is an effective dominance rule for comparing partial paths represented by labels. As is clear when looking back on §4, the dominance rules were nontrivial to derive, but their application comes 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, Cordeau, and Laporte (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

constraints of the form $K_i \le T_{i+n} - T_i \le L_i$ with an additional parameter $K_i > 0$ should be considered. This 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- or three-index compact formulation. However, based on attempts to generalize our results, we suspect that the simultaneous handling of regular time windows and both minimum and maximum ride times in a dynamic programming labeling algorithm is not straightforward, but highly intricate. A possible approach to such a problem is the handling of *minimum* ride times using IPEC in the column-generation master program. The subproblem is then identical to the DARP subproblem for which the actual implementation can be used.

Even more challenging types of VRP result from a mix of intra-route and inter-route synchronization constraints. It is clear that inter-route synchronization leads to additional constraints in the column-generation master program (Desaulniers et al. 1998; 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 handle effectively, and it seems completely unclear how VRP combining intra-route and inter-route synchronization constraints can be modeled and solved. This relates to exact as well as heuristic approaches.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/trsc.2014.0531.

Acknowledgments

The second author's research is partially funded by the Deutsche Forschungsgemeinschaft (DFG) [Grant IR 122/5-1].

References

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

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

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

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

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

Bredström D, Rönnqvist M (2008) Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *Eur. J. Oper. Res.* 191(1):19–29.

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

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

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

Cordeau J-F, Laporte G (2007) The dial-a-ride problem: Models and algorithms. *Ann. Oper. Res.* 153(1):29–46.

Cordeau J-F, Laporte G, Ropke S (2008) Recent models and algorithms for one-to-one pickup and delivery problems. Golden B, Raghavan S, Wasil E, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, Boston), 327–357.

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

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

Desaulniers G, Desroisiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D (1998) A unified framework for deterministic time constraint vehicle routing and crew scheduling problems. Crainic TG, Laporte G, eds. *Fleet Management and Logistics* (Kluwer, Norwell, MA), 57–93.

Desrochers M, 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, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.* 40(2):342–354.

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

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

Dumas Y, Desrosiers J, Soumis F (1991) The pickup and delivery problem with time windows. *Eur. J. Oper. Res.* 54(1):7–22.

Eveborn P, Flisberg P, Rönnqvist M (2006) Laps care—An operational system for staff planning of home care. *Eur. J. Oper. Res.* 171(3):962–976.

Firat M, Woeginger GJ (2011) Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Oper. Res. Lett.* 39(1):32–35.

Haugland D, Ho SC (2010) Feasibility testing for dial-a-ride problems. Chen B, ed. *Algorithmic Aspects in Information and Management*, Lecture Notes Comput. Sci., Vol. 6124 (Springer, Berlin Heidelberg), 170–179.

Houck DJ, Picard JC, Queyranne M, Vemuganti RR (1980) The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *OPSEARCH* 17:93–109.

Ioachim I, Desrosiers J, Soumis F, Bélanger N (1999) Fleet assignment and routing with schedule synchronization constraints. *Eur. J. Oper. Res.* 119(1):75–90.

Ioachim I, Gélinas S, Desrosiers J, Soumis F (1998) A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks* 31(3):193–204.

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, New York), 33–65.

Irnich S, Villeneuve D (2006) The shortest-path problem with resource constraints and k-cycle elimination for $k \ge 3$. *INFORMS J. Comput.* 18(3):391–406.

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

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res.* 53(6):1007–1023.

Madsen O, Ravn H, Rygaard J (1995) A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Ann. Oper. Res.* 60(1):193–208.

Naddef D, Rinaldi G (2002) Branch-and-cut algorithms for the capacitated VRP. Toth P, Vigo D, eds. *The Vehicle Routing Problem* (SIAM, Philadelphia), 53–84.

Parragh SN, Doerner KF, Hartl RF (2008) A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *J. für Betriebswirtschaft* 58(2):81–117.

Parragh SN, Doerner KF, Hartl RF (2010) Variable neighborhood search for the dial-a-ride problem. *Comps. Oper. Res.* 37(6): 1129–1138.

Ropke S, Cordeau J-F (2005) Branch and cut and price for the pickup and delivery problem with time windows. Unpublished doctoral dissertation, Heuristic and exact algorithms for vehicle routing problems, University of Copenhagen, Copenhagen.

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

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

Russell RA, Morrel RB (1986) Routing special-education school buses. *Interfaces* 16(5):56–64.

Savelsbergh MWP (1992) The vehicle routing problem with time windows: Minimizing route duration. *INFORMS J. Comput.* 4(2):146–154.

Tang J, Kong Y, Lau H, Ip AWH (2010) A note on "efficient feasibility testing for dial-a-ride problems." *Oper. Res. Lett.* 38(5): 405–407.

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