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

Timo Gschwind[*,a], Stefan Irnich[a], Ann-Kathrin Rothenbächer[a], Christian Tilk[a]

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

Abstract

For the exact solution of many types of vehicle-routing problems, column-generation based algorithms have become predominant. The column-generation subproblems are then variants of the shortest-path problem with resource constraints which can be solved well with dynamic-programming labeling algorithms. For vehicle-routing problems with a pickup-and-delivery structure, the strongest known dominance between two labels requires the delivery triangle inequality (DTI) for reduced costs to hold. When the direction of labeling is altered from forward labeling to backward labeling, the DTI requirement becomes the pickup triangle inequality (PTI). DTI and PTI cannot be guaranteed at the same time. The consequence seemed to be that bidirectional labeling, one of the most successful acceleration techniques developed over the last years, is not applicable with a strong dominance in both directions for problems with a pickup-and-delivery structure. Surely, relying on a weak dominance in one direction is feasible but makes the bidirectional approach less powerful. In this paper, we show that bidirectional labeling with the strongest dominance rules in forward as well as backward direction is possible and computationally beneficial. A full-fledged branch-cut-and-price algorithm is tested on the pickup-and-delivery problem with time windows (PDPTW).

Key words: vehicle routing, pickup-and-delivery, shortest-path problem with resource constraints, bidirectional labeling, column generation

1. Introduction

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

For the exact solution of many types of VRPs, column-generation based algorithms have become predominant (Desaulniers *et al.*, 2005). The column-generation subproblems are then variants of the *elementary shortest-path problem with resource constraints* (ESPPRC) which can be solved well with dynamic-programming labeling algorithms (Irnich and Desaulniers, 2005). For ESPPRCs with a pickup-and-delivery structure, the strongest known dominance between two labels requires the *delivery triangle inequality* (DTI) for reduced costs to hold (Ropke and Cordeau, 2009). When the direction of labeling is altered from forward labeling to backward labeling, the DTI requirement becomes the *pickup triangle inequality* (PTI). DTI and

[*]Corresponding author.

Email addresses: gschwind@uni-mainz.de (Timo Gschwind), irnich@uni-mainz.de (Stefan Irnich), anrothen@uni-mainz.de (Ann-Kathrin Rothenbächer), tilk@uni-mainz.de (Christian Tilk)

PTI cannot be guaranteed at the same time. The consequence seemed to be that bidirectional labeling (Righini and Salani, 2006), one of the most successful acceleration techniques developed over the last years, is not applicable with a strong dominance in both directions for problems with a pickup-and-delivery structure. Clearly, relying on a weak dominance in one direction is feasible but makes the bidirectional approach less powerful.

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

The remainder of this paper is structured as follows: Section 2 formally defines the PDPTW and sketches the column-generation algorithm by discussing its extended set-covering master program and ESPPRC subproblem. Moreover, the role of the DTI/PTI for weak and strong dominance in ESPPRC labeling algorithms is clarified. The section also derives the main theoretical result of the paper. Computational results are provided in Section 3. Final conclusions are drawn in Section 4.

2. Branch-Cut-and-Price for PDPTW

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

2.1. Master Program

The branch-cut-and-price for the PDPTW uses an extended set-covering formulation that we describe now. Let Ω be the set of all feasible PDPTW routes. A route $r \in \Omega$ is defined as an elementary 0-$(2n+1)$-path. It is feasible if it fulfills time-window, capacity, and pairing and precedence constraints (see Dumas *et al.*, 1991, for details). The cost c_r of a route $r \in \Omega$ is defined as the sum of the travel costs of the traversed arcs. The goal of the PDPTW is to find a set of feasible routes serving each request exactly once minimizing the total routing costs. The set-covering formulation uses binary variables λ_r indicating whether route $r \in \Omega$

is used or not.

$$\min \quad \sum_{r\in\Omega} c_r\lambda_r \tag{1a}$$

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

$$\sum_{r\in\Omega} \lambda_r \leq K \tag{1c}$$

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

The objective (1a) minimizes the total travel costs. Due to the validity of the triangle inequalities for travel times and routing costs, the covering constraints (1b) ensure that each request is served exactly once. Herein, the binary coefficients a_{ir} are equal to 1 if and only if request $(i,i+n)\in R$ is served by route r. The number of routes is limited by (1c), and the variable domains are given in (1d).

For solving the linear relaxation of formulation (1), a column-generation algorithm (Desaulniers *et al.*, 2005) is employed. Starting with a subset $\bar{\Omega}\subset\Omega$ of the feasible routes, the linear relaxation of formulation (1) defined over $\bar{\Omega}$ is denoted as the *restricted master program* (RMP). The column-generation algorithm alternates between the re-optimization of the RMP and the solution of the column-generation pricing problem that adds negative reduced-cost variables (=columns) to the RMP, if one exists. The linear relaxation can be strengthened by adding valid inequalities (see Section 2.3), and branching is required to finally ensure integer solutions (see Section 2.4). The most time-consuming part of this branch-cut-and-price algorithm is the solution of the pricing subproblem, described in the following.

2.2. Subproblem

Let π_i for $(i,i+n)\in R$ be the dual prices of the covering constraints (1b) and μ the dual price of constraint (1c). The pricing problem must compute at least one feasible route $r\in\Omega$ with negative reduced cost $\bar{c}_r := c_r - \mu - \sum_{(i,i+n)\in R} a_{ir}\pi_i$ (or guarantee that no such route exists). This problem is an ESPPRC which can be solved by means of a dynamic-programming labeling algorithm (Irnich and Desaulniers, 2005). Herein, for defining reduced costs of arcs, note that the dual price π_i for fulfilling request $(i,i+n)\in R$ can be associated to the pickup vertex i, the delivery vertex $i+n$, or be arbitrarily split among them. Accordingly, for any given and fixed $\alpha\in\mathbb{R}$, define

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

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

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

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

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

Righini and Salani (2006) and several subsequent works have shown that bounded bidirectional labeling algorithms are usually superior to their monodirectional counterparts. A requirement for bidirectional labeling is that REFs can be inverted, as discussed by Irnich (2008). The respective backward labeling component starts with an initial label at the destination depot and propagates labels in backward direction,

i.e., against the orientation of the arcs, toward the origin depot. Each backward label stores the resource consumption of a partial path $(i, \dots, 2n+1)$ starting at some vertex $i \in V$ and ending at the destination depot $2n+1$. A major design decision for bidirectional labeling is the definition of a so-called *half-way point*. Its purpose is to alleviate combinatorial explosion, since both forward and backward labels are extended only up to the half-way point. When labeling terminates, suitable forward and backward labels are merged to obtain complete feasible 0-$(2n+1)$-paths. For VRPs with time windows, traditional (static) approaches often choose the middle $(a_0 + b_{2n+1})/2$ of the planning horizon as the half-way point. The recent work of Tilk *et al.* (2017) has shown that dynamically adjusting the half-way point by carefully estimating the remaining workload in the forward and backward labeling can improve the overall performance.

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

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

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

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

$$\bar{c}'_f = \bar{c}_f + \bar{c}_{ij}(\alpha) \tag{2a}$$

$$t'_f = \max\{t_f + t_{ij}, a_j\} \tag{2b}$$

$$l'_f = l_f + q_j \tag{2c}$$

$$S'_f = \begin{cases} S_f \cup \{(j-n, j)\} & \text{if } j \in D \\ S_f & \textit{otherwise} \end{cases} \tag{2d}$$

$$O'_f = \begin{cases} O_f \cup \{(j, j+n)\} & \text{if } j \in P \\ O_f \setminus \{(j-n, j)\} & \text{if } j \in D \\ O_f & \textit{otherwise} \end{cases} \tag{2e}$$

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

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

Rule 1. *(Weak Dominance Forward) A forward label $L_f = (i, \bar{c}_f, t_f, l_f, S_f, O_f)$ dominates another forward label $L'_f = (i, \bar{c}'_f, t'_f, l'_f, S'_f, O'_f)$ with identical last vertex i if $\bar{c}_f \leq \bar{c}'_f$, $t_f \leq t'_f$, $S_f \subseteq S'_f$, and $O_f = O'_f$ hold.*

Using $\alpha = 1$ in the forward labeling, the dual prices of covering requests are completely assigned to the pickup vertices as already suggested by Dumas *et al.* (1991). Formally, $\alpha = 1$ gives $\bar{\pi}_i(\alpha) = \pi_i$ for all pickup vertices $i \in P$ and $\bar{\pi}_{i+n}(\alpha) = 0$ for all delivery vertices $i + n \in D$. With this definition, the forward reduced-cost matrix $(\bar{c}^f_{ij}) := (\bar{c}_{ij}(1))$ satisfies $\bar{c}^f_{ij} \leq \bar{c}^f_{ik} + \bar{c}^f_{kj}$ for all $(i, j) \in A$ and $k \in D$. Ropke and Cordeau (2009) call this property the DTI. Roughly speaking, the DTI ensures that visiting an additional delivery

vertex is never beneficial. This property enables the use of the following stronger dominance rule (Dumas *et al.*, 1991):

Rule 2. *(Strong Dominance Forward) Let the forward reduced-cost matrix fulfill the DTI. A forward label* $L_f = (i, \bar{c}_f, t_f, l_f, S_f, O_f)$ *dominates another forward label* $L'_f = (i, \bar{c}'_f, t'_f, l'_f, S'_f, O'_f)$ *with identical last vertex* i *if* $\bar{c}_f \leq \bar{c}'_f$, $t_f \leq t'_f$, $S_f \subseteq S'_f$, *and* $O_f \subseteq O'_f$ *hold.*

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

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

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

$$\bar{c}'_b = \bar{c}_b + \bar{c}_{ij}(\alpha) \tag{3a}$$

$$t'_b = \min\{t_b - t_{ij}, b_i\} \tag{3b}$$

$$l'_b = l_b + q_i \tag{3c}$$

$$S'_b = \begin{cases} S_b \cup \{(j, j+n)\} & \text{if } j \in P \\ S_b & \textit{otherwise} \end{cases} \tag{3d}$$

$$O'_b = \begin{cases} O_b \cup \{(j-n, j)\} & \text{if } j \in D \\ O_b \setminus \{(j, j+n)\} & \text{if } j \in P \\ O_b & \textit{otherwise} \end{cases} \tag{3e}$$

As in the forward case, also backward REFs are non-decreasing in the resources $\bar{c}_f$ and S_f as well as non-increasing in the resource t_f so that the following dominance rule is valid:

Rule 3. *(Weak Dominance Backward) A backward label* $L_b = (j, \bar{c}_b, t_b, l_b, S_b, O_b)$ *dominates another backward label* $L'_b = (j, \bar{c}'_b, t'_b, l'_b, S'_b, O'_b)$ *with identical last vertex* j *if* $\bar{c}_b \leq \bar{c}'_b$, $t_b \geq t'_b$, $S_b \subseteq S'_b$, *and* $O_b = O'_b$ *hold.*

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

Rule 4. *(Strong Dominance Backward) Let the backward reduced-cost matrix fulfill the PTI. A backward label* $L_b = (j, \bar{c}_b, t_b, l_b, S_b, O_b)$ *dominates another backward label* $L'_b = (j, \bar{c}'_b, t'_b, l'_b, S'_b, O'_b)$ *with identical last vertex* j *if* $\bar{c}_b \leq \bar{c}'_b$, $t_b \geq t'_b$, $S_b \subseteq S'_b$, *and* $O_b \subseteq O'_b$ *hold.*

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

The bidirectional labeling limits the effort in both directions, since not all forward (backward) labels are propagated to the destination (origin) depot. Instead, labels are propagated only up to a so-called *half-way*

point, thus limiting the overall number of created labels. In the PDPTW, the half-way point h is best defined on the time resource so that forward labels L_f are propagated only if $t_f \leq h$ and backward labels L_b only if $t_b > h$. Compatible forward and backward labels must then be merged to obtain complete 0-$(2n{+}1)$-paths.

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

$$S_f \cap S_b = \varnothing \tag{4a}$$

$$t_f \leq t_b \tag{4b}$$

$$\begin{cases} O_b = O_f \setminus \{(i, i+n)\} & \text{if } i \in P \\ O_f = O_b \setminus \{(i-n, i)\} & \text{if } i \in D \\ O_b = O_f & \text{otherwise} \end{cases} \tag{4c}$$

where (4a) ensures that each request is fulfilled at most once, (4b) guarantees that the path is feasible with respect to the time resource, and (4c) requires that the concatenation completes all open requests.

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

Proposition 1. *The reduced cost of a route r generated by a feasible merge of a forward label L_f and a backward label L_b can be obtained as*

$$\bar{c}_r = \bar{c}_f + \bar{c}_b + \sum_{\substack{(k,k+n)\in \\ \{O_b \cap O_f\}}} \pi_k + \sum_{\substack{(k,k+n)\in \\ \{O_b \cup O_f\}\setminus\{O_b \cap O_f\}}} \frac{\pi_k}{2}. \tag{5}$$

Proof. Obviously, for all requests $(k, k+n) \in R$ that are completely fulfilled (picked up and delivered) either in L_f or L_b, the dual price π_k is correctly incorporated. Let i be the vertex at which the merge is performed. For all requests $(k, k+n) \in R$ with $k \neq i$ and $k+n \neq i$ that are open in both directions, i.e., $(k, k{+}n) \in O_f \cap O_b$, the dual price π_k was subtracted twice (once in $\bar{c}_f$ and once in $\bar{c}_b$). This is corrected by adding it once to $\bar{c}_f + \bar{c}_b$. Finally, if the merge vertex i is a pick-up or a delivery vertex, the corresponding request is open either in L_f or L_b and its dual price π_i (π_{i-n}) for $i \in P$ ($i \in D$) was subtracted exactly 1.5 times, which is corrected by adding $\pi_i/2$ in (5). □

The following proposition shows that our bidirectional labeling algorithm is exact, although we perform forward and backward labeling on different cost matrices.

Proposition 2. *A bidirectional labeling algorithm that uses the strong dominance (Rule 2 and Rule 4) in both directions and the described merge procedure with* (4) *and* (5) *finds an optimal solution to the pricing subproblem of the PDPTW.*

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

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

partial path ending at the merge vertex i, the label $L_f = (i, \bar{c}_f, t_f, l_f, S_f, O_f)$ is the label of F generated in the forward part of the bidirectional labeling algorithm, and B is the backward partial path starting at the merge vertex i.

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

If $O'_b = O_b$, then L_f and L'_b fulfill the merge conditions (4), since L_f and L_b fulfill it due to the feasibility of $P = (F, B)$. Moreover, the reduced cost of the path $P' = (F, B')$ resulting from the merge of L_f and L'_b has reduced cost not greater than $P = (F, B)$, because the only difference in (5) is that $\bar{c}_b$ is replaced by $\bar{c}'_b \leq \bar{c}_b$. Thus, $P' = (F, B')$ must be another feasible 0-$(2n+1)$-path with minimum reduced cost. It is found in the bidirectional labeling algorithm.

Assume now that $O'_b \subsetneq O_b$ holds. One can then remove all pickup vertices of requests $(k, k+n) \in O'_b \setminus O_b$ from the forward partial path F. Let F' be the resulting partial path. The path $P^* = (F', B')$ is feasible and it fulfills a smaller number of requests than P because $S'_b \subseteq S_b$. Note that the latter implication relies on elementarity of feasible paths. We now show that the reduced cost of P^* is not greater than the reduced cost of P leading to a contradiction to the request-minimality of P: Consider $P^* = (F', B')$ as a backward path. Due to the PTI, the backward reduced costs of F' and F fulfill $\bar{c}_b(F') \leq \bar{c}_b(F)$. Moreover, $\bar{c}'_b \leq \bar{c}_b$ due to the dominance between L'_b and L_b. Hence, the reduced cost $\bar{c}_b(F') + \bar{c}'_b$ of P^* is not greater than the reduced cost $\bar{c}_b(F) + \bar{c}_b$ of P. □

2.3. Cutting Planes

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

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

The implication of both families of cuts are additional dual prices on arcs (i, j) that have to be included in the reduced costs $\bar{c}^f_{ij}$ and $\bar{c}^b_{ij}$ in the pricing subproblem. As a result, the forward (backward) reduced cost matrix does generally not fulfill the DTI (PTI). In order to use the strong dominance rule in forward (backward) labeling, however, the DTI (PTI) is indispensable. Ropke and Cordeau (2009) have shown how to transform an arbitrary cost matrix into one that that fulfills the DTI while at the same time keeping the reduced costs of all feasible routes unchanged. They compute for each request $(j, j+n) \in R$ the maximum violation $\theta^f_j = \max_{i,k \in V} \{\bar{c}^f_{ik} - (\bar{c}^f_{i,j+n} + \bar{c}^f_{j+n,k})\}^+$ of the DTI. Then, $\theta^f_j/2$ is subtracted from the reduced cost $\bar{c}^f_{ij}$ and $\bar{c}^f_{ji}$ of all in- and outgoing arcs of the corresponding pickup vertex $j \in P$. For the delivery vertex $j+n \in D$, the same amount $\theta^f_j/2$ is added to the reduced costs $\bar{c}^f_{i,j+n}$ and $\bar{c}^f_{j+n,i}$ of all in- and outgoing arcs. Since each feasible route either visits both the pickup and the delivery vertex of a request or none of the two, its reduced cost remains unchanged with this transformation. Clearly, an analog transformation can be performed to ensure the PTI for the backward reduced cost matrix $\bar{c}^b_{ij}, i, j \in V$ using the terms $\theta^b_j = \max_{i,k \in V} \{\bar{c}^b_{ik} - (\bar{c}^b_{ij} + \bar{c}^b_{jk})\}^+$ for all $(j, j+n) \in R$.

The merge procedure has to take into account the modified cost matrices in forward and backward labeling. When merging suitable forward and backward labels L_f and L_b the reduced cost of the resulting

route r can be computed as:

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

The correctness follows straightforwardly by analog considerations as in the proof of Proposition 1. Note also that the proof of Proposition 2 is not affected by the altered reduced-cost formula.

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

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

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

The computation of the reduced cost of a merged path in the bidirectional labeling algorithm also has to account for the new resources and dual prices associated with each cut $SR(U_k)$. Let L_f and L_b be a pair of forward and backward labels fulfilling the merge conditions (4). If both L_f and L_b have visited an odd number of requests in U_k, i.e., $sr_f(k) = sr_b(k) = 1$, then the dual price σ_k of the subset-row cut $SR(U_k)$ has to be subtracted once more in the reduced cost of the merged path. This is similar to the merge procedure in labeling algorithms for many other ESPPRC variants. Additional care has to be taken in the proposed bidirectional labeling for the PDPTW because visited requests $(i, i+n) \in U_k$ are counted at the pickups in forward and at the deliveries in backward direction. Thus, if some requests $(i, i+n) \in U_k$ are open in the forward label L_f or in the backward label L_b, they have been counted twice (once at the pickup vertex in L_f and once at the delivery vertex in L_b) and, therefore, the dual price σ_k might have been subtracted too often. To repair this defect, we compute the number of requests that have been counted twice as $cnt_k := |U_k \cap (O_f \cup O_b)|$. Furthermore, let $\Theta_k := 1 - |sr_f(k) - sr_b(k)|$ be the indicator telling whether the sum of counted visits in both directions is even. If this sum is even, we have to add the dual price for every odd number of double-counts. Vice versa, if the sum of counted visits is odd, we have to add the dual price for every even number of double-counts. Consequently, we have to add $\lfloor \frac{cnt_k + \Theta_k}{2} \rfloor$ times σ_k to the reduced cost of the merged path to adjust for possible double-inclusions. Again, these changes in the computation of the reduced cost of the merge of labels L_f and L_b do not influence the validity of the arguments of the proof of Proposition 2.

2.4. Branching

Note first that branching on arcs is not possible in PDPs if one wants to preserve the DTI or PTI (see Ropke and Cordeau, 2009, for details). We therefore use the following hierarchical branching scheme: First, we try to branch on the number of vehicles. If the number of vehicles is integer, we branch on the outflow of a set of vertices as proposed for the CVRP by Naddef and Rinaldi (2002) and applied to the PDPTW by Ropke and Cordeau (2009). Both branching decisions can be implemented by including a single inequality of the form $x(\delta^+(S)) \leq rhs$ or $x(\delta^+(S)) \geq rhs$ (see Section 2.3) in the master problem.

3. Computational Results

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

Benchmark Instances. The computational studies use two groups of instances from the literature, the 40 instances of Ropke and Cordeau (2009) called `RC`, and 30 instances of Li and Lim (2001) denoted by `LL`. The `RC` instances comprise 30 to 75 requests and have high vehicle fixed costs such that the minimization of the number of vehicles is the primary objective. Moreover, they are characterized by small vehicle capacities, narrow time windows, and time windows of corresponding pickup and delivery locations that are very close to each other. Consequently, these instances rarely allow multiple open requests at the same time. To be precise, on average only 22% of the generated labels during a forward labeling have more than one open request. As a result, most routes visit the pickup and delivery vertices of each request in direct sequence. Thus, the `RC` instances resemble more VRPTW instances without capacity restrictions rather than being true PDPTW instances. As a consequence, there is no considerable difference between weak and strong dominance in the labeling algorithm on such instances. We therefore created an additional set of "real" PDPTW instances based on the `RC` instances by increasing the right hand side of all time windows by 25 units and the vehicle capacity by 10 units. These instances are denoted `RC+`.

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

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

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

Overall, we compare six different *labeling strategies* for the solution of the ESPPRC pricing problems: Pure (monodirectional) forward and backward labeling (`Fw-S` and `Bw-S`), bidirectional labeling with a static half-way point using strong dominance in the forward part and weak or strong dominance in the backward

part (`Bi-St-SW` or `Bi-St-SS`), and the two analog strategies `Bi-Dy-SW` and `Bi-Dy-SS` for bidirectional labeling with a dynamic half-way point as described in (Tilk *et al.*, 2017). The different strategies are summarized in Table 1.

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

Table 1: Labeling strategies

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

For each PDPTW instance and labeling strategy, we compute the average ratio of the pricing time compared to the baseline strategy `Fw-S` as the geometric mean over all pricing iterations. In order to reduce inaccuracies in time measurement, we take into account only those pricing iterations that consumed more than 0.1 seconds of computation time for all six labeling strategies. Whenever the linear relaxation is not completely solved within the time limit, the average is taken over the iterations solved by all strategies up to this point.

Table 2 summarizes the results for this first type of experiment. The first column indicates whether only the pricing iterations for the linear relaxation of (1) (*LP*) or those of the entire branch-and-bound tree (*B&B*) are considered. The second column (*Group*) gives the instance group and the third column (*#inst*) gives the number of instances for which there was at least one pricing iteration requiring at least the minimal time of 0.1 seconds. The remaining columns present the geometric means over the instances of the pricing time ratios of the different labeling strategies compared to forward labeling.

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

Table 2 also shows that bidirectional labeling with a static half-way point and weak dominance in the backward part performs rather disappointing: `Bi-St-SW` is clearly inferior to strategy `Fw-S`. These findings are especially true for the original instance sets (recall that they are much harder to solve in backward than in forward direction, see also column `Bw-S` of Table 2) and are in line with the findings of Ropke and Cordeau (2009) for their prototype implementation of strategy `Bi-St-SW`.

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

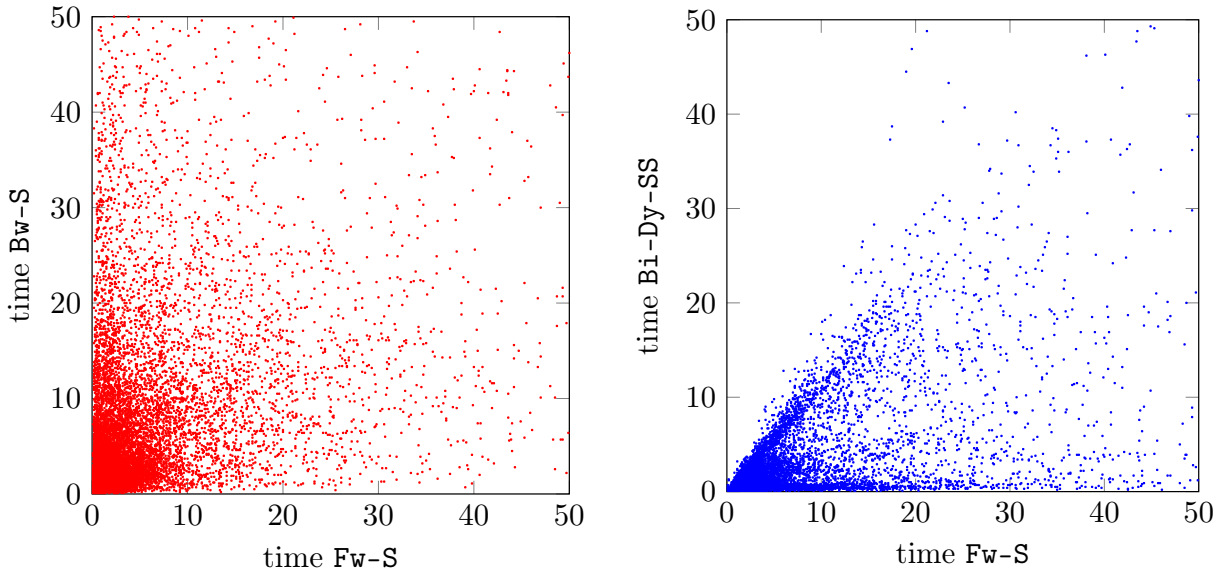
Table 2: Comparison of different labeling strategies on same ESPPRC instances (identical dual prices)

Figure 1 displays two scatter plots of the solution times of individual pricing iterations of Bw-S and Bi-Dy-SS compared to Fw-S. Figure 1a (on the left-hand side) shows that solution times of the two pure monodirectional strategies Bw-S to Fw-S behave almost randomly. Indeed, it seems impossible to predict whether forward or backward labeling requires more computational effort for a given ESPPRC instance. Figure 1b (on the right-hand side) shows the same plot for the new and best-performing labeling strategy Bi-Dy-SS in comparison to Fw-S. The majority of the data points lay under the diagonal showing that in almost all cases Bi-Dy-SS is performing at least as well as Fw-S. The many data points close to the abscissa stand for those cases in which Bi-Dy-SS is strongly superior to Fw-S.

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

The results of the second type of experiment can be found in Table 3. It reports for each instance group (*Group*) and each of the six labeling strategies the following values: the number of instances solved to proven optimality within the time limit (*#opt*) and the average solution *time* in seconds. The average is taken only over those instances that were solved to optimality with at least one of the six branch-cut-and-price algorithms (counting the time limit for labeling strategies that did not finish). Overall, Table 3 confirms the result of the first type of test: The branch-cut-and-price algorithm with labeling strategy Bi-Dy-SS produces the best results concerning both number of solved instances and total solution times. Moreover, all algorithms using bidirectional labeling outperform those using monodirectional labeling.

These observations are also confirmed by the *performance profiles* of the six branch-cut-and-price algorithms depicted in Figure 2. According to Dolan and Moré (2002), given a set $\mathcal{A} = \{A_1, A_2, \ldots, A_p\}$ of algorithms, the performance profile $\rho_A(\tau)$ of an algorithm $A \in \mathcal{A}$ is the fraction of instances that algorithm A



(a) `Bw-S` against `Fw-S`

(b) `Bi-Dy-SS` against `Fw-S`

Figure 1: Comparison of individual pricing times (in seconds) of three labeling strategies

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

Table 3: Comparison of different labeling strategies on separate runs: number of instances solved to proven optimality (*#opt*) and average run times (in seconds)

can solve within a factor τ of the fastest algorithm of $\mathcal{A}$. Any unsolved instances are taken into account with infinite run time. In particular, $\rho_A(1)$ is the percentage of instances on which A wins. For the same algorithm A, $100 - \rho_A(\infty)$ is the percentage of unsolved instances.

Overall, the performance profiles in Figure 2 confirm that the branch-cut-and-price with labeling strategy `Bi-Dy-SS` clearly dominates all other algorithms on the 220 PDPTW test instances. Indeed, it is at most 1.5 times slower than the fastest algorithm for more than 90% of the instances solved by at least one algorithm. Moreover, the branch-cut-and-price algorithms with bidirectional labeling and strong dominance in both directions are consistently and significantly superior to their counterparts with weak dominance in the backward labeling.

Comparison with Baldacci et al. *(2011).* A dedicated exact approach for the PDPTW based on a set-partitioning formulation, bounding procedures, cut-and-column generation procedures, and route enumeration was suggested by Baldacci *et al.* (2011). The integer programming solver CPLEX is finally used to obtain integer solutions. The authors use two different algorithmic setups for the `RC` and `LL` instances exploiting their specific characteristics. Although this approach is based on a different column-generation

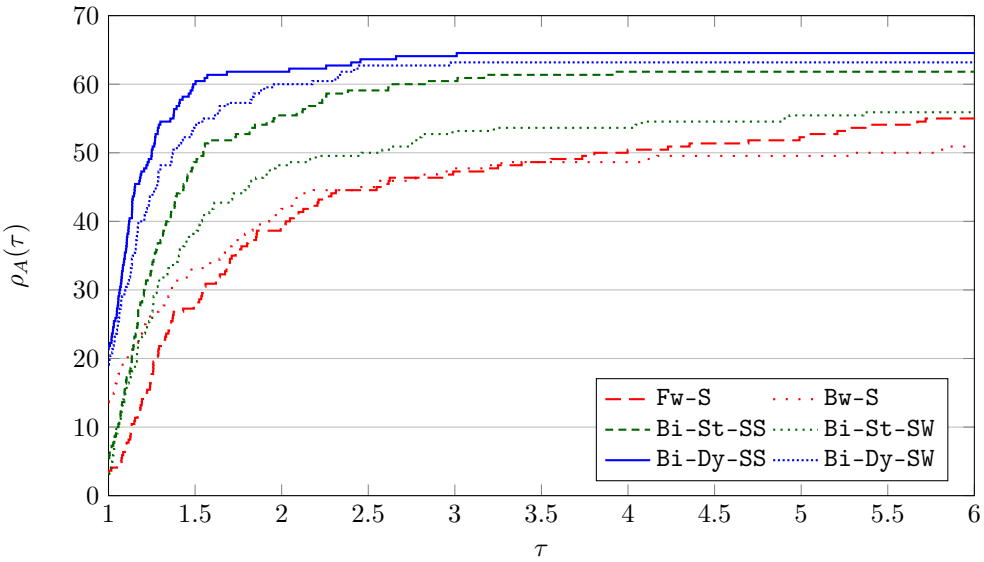


Figure 2: Performance profiles of branch-cut-and-price algorithms using the six different labeling strategies

principle, the overall results are comparable: For the `RC` benchmark, they solved 32 of the 40 instances to proven optimality within one hour (`Bi-Dy-SS` produces 33 optima, see Table 3). Seven of the `LL` benchmark instances remain completely untractable for both approaches, since the algorithm of Baldacci *et al.* runs out of memory and ours fails to solve the linear relaxation (root node). Within one hour of computation time, the approach of Baldacci *et al.* cannot close the integrality gap for one other instance while ours fails on another five instances. As the enumeration approach is very much tailored to the PDPTW, it seems hard to generalize it to other vehicle-routing problems with pickup-and-delivery structure.

4. Conclusions

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

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

A promising avenue for future research is the extension of the bidirectional labeling to the *ng*-path relaxations of the ESPPRC (Baldacci *et al.*, 2012). Such an extension is, however, not straightforward. When using an *ng*-path relaxation in the PDPTW with LIFO loading, Cherkesly *et al.* (2016) have shown that non-elementary paths can be incorrectly dominated. Note that similar anomalies result from the widely-used preprocessing and arc-elimination techniques in non-elementary pricing for PDPs. As a consequence, the lower bounds computed with these relaxations are not unique. However, bounds are valid as exploited in the selective pricing paradigm of Desaulniers *et al.* (2016). The incorrect dominance is particularly critical in the bidirectional labeling because the existence of compatible forward and backward pairs of labels can no longer be ensured with arguments similar to those used in the proof of Proposition 2.

References

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

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

Battarra, M., Cordeau, J.-F., and Iori, M. (2014). Pickup-and-delivery problems for goods transportation. In Toth and Vigo (2014), chapter 6, pages 161–191.

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

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

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

Desaulniers, G., Contardo, C., and Pecin, D. (2016). Selective pricing in branch-and-price algorithms for vehicle routing. Presentation at the Column Generation Conference 2016. `https://www.gerad.ca/colloques/ColumnGeneration2016/PDF/Desaulniers.pdf`.

Doerner, K. F. and Salazar-González, J.-J. (2014). Pickup-and-delivery problems for people transportation. In Toth and Vigo (2014), chapter 7, pages 193–212.

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

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

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

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers *et al.* (2005), chapter 2, pages 33–65.

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

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

Li, H. and Lim, A. (2001). A metaheuristic for the pickup and delivery problem with time windows. In D. Moldovan, editor, *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01)*, pages 160–167. IEEE Press.

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

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

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

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

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530 – 539.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

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