Submitted to *Transportation Science*
manuscript (Please, provide the mansucript number!)

Branch-and-Price for the Pickup and Delivery Problem with Time Windows and Scheduled Lines

Veaceslav Ghilas
Eindhoven University of Technology, School of Industrial Engineering,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands, v.ghilas@tue.nl

Jean-François Cordeau
HEC Montréal and CIRRELT, 3000, chemin de la Côte-Sainte-Catherine,
Montréal, H3T 2A7, Canada, jean-francois.cordeau@hec.ca

Emrah Demir
Panalpina Centre for Manufacturing and Logistics Research, Cardiff Business School, Cardiff University,
Cardiff CF10 3EU, UK, demire@cardiff.ac.uk

Tom Van Woensel
Eindhoven University of Technology, School of Industrial Engineering,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands, t.v.woensel@tue.nl

The Pickup and Delivery Problem with Time Windows and Scheduled Lines (PDPTW-SL) consists of routing and scheduling a set of vehicles, by integrating them with scheduled public transportation lines, to serve a set of freight requests within their time windows. This paper presents an exact solution approach based on a branch-and-price algorithm. A path-based set partitioning formulation is used as the master problem, and a variant of the elementary shortest path problem with resource constraints is solved as the pricing problem. In addition, the proposed algorithm can also be used to solve the PDPTW with transfers (PDPTW-T) as a special case. Results of extensive computational experiments confirm the efficiency of the algorithm: it is able to solve small- and medium-size instances to optimality within reasonable execution time. More specifically, our algorithm solves the PDPTW-SL with up to 50 requests and the PDPTW-T with up to 40 requests on the considered instances.

Key words: pickup and delivery problem; freight transportation; column generation; scheduled lines
History:

1. Introduction

A successful integration of freight and public transportation creates a seamless movement for both people and goods. This integration achieves socially desirable and economically viable transport options in urban areas (Ghilas et al. 2016b) as it reduces congestion and air pollution (Demir et al. 2014, 2015). There have been several practical attempts to investigate the potential benefits of such integrated systems. MULI was a demonstration project in Germany between 1996 and 1999

(Trentini and Malhene 2010). Special-design buses were used to transport both passengers and small-size parcels. The aim of the project was to achieve the environmental and economical benefits of the integrated transport system, but one of the most important obstacles for implementation proved to be passenger acceptance. City Cargo Amsterdam was set up as a pilot experiment in 2007 (Cargo Tram 2012). Two cargo trams were used to transport packages into the city center of Amsterdam. In 2009, the project was abandoned due to the lack of public funds. Later, Masson et al. (2015) investigated an integrated freight and public transport distribution system in the French city of La Rochelle. The results showed that an integrated transportation system can lead to improved vehicle utilization and reduced operational costs. In our view, the aforementioned projects did not succeed due to a number of practical challenges, such as inappropriate decision support, low passenger acceptance rate, deficient information sharing between involved parties and insufficient funding for upgrading existing transportation systems.

This paper focuses on opportunities to make use of available public transportation as a part of freight journey from a routing and scheduling (i.e., decision support) point of view. It is assumed that each considered public transport vehicle, which operates according to predetermined routes and schedules, has a finite carrying capacity for freight requests apart from available spaces destined for passengers. Therefore, transferring freight requests to available scheduled line (SL) services may benefit the whole transportation system.

With the possibility of using SL services, there can be two delivery options for serving the transport requests. These include direct and indirect (via SL) deliveries. The first one implies that the origin and destination points of a request are visited by using only one pickup and delivery (PD) vehicle. The second type of delivery implies that a request is picked up by a PD vehicle and transported to a transfer node. From there, the request continues its journey on scheduled lines. Afterwards, the request is picked up again by another PD vehicle to be delivered to its destination point. In this paper, we denote this specific transportation problem as the *Pickup and Delivery Problem with Time Windows and Scheduled Lines* (PDPTW-SL). It is important to note that the PDPTW-SL contains the *Pickup and Delivery Problem with Time Windows and Transfers* (PDPTW-T) as a special case, where transfers are allowed between PD vehicles at predefined nodes. A schematic overview of the associated network is provided in Figure 1. This figure presents two requests that have their pickup and delivery points close to two different transfer nodes. It would thus make sense to use the scheduled line service that connects these two transfer nodes instead of using a PD vehicle as in classical pickup and delivery systems. Hence, travel time savings for PD vehicles and operational cost reductions can be expected with the proposed integrated system.

The existing literature on the PDPTW mainly focuses on heuristic algorithms (see e.g., Nanry and Barnes (2000), Røpke and Pisinger (2006)). However, there are also a few studies that have

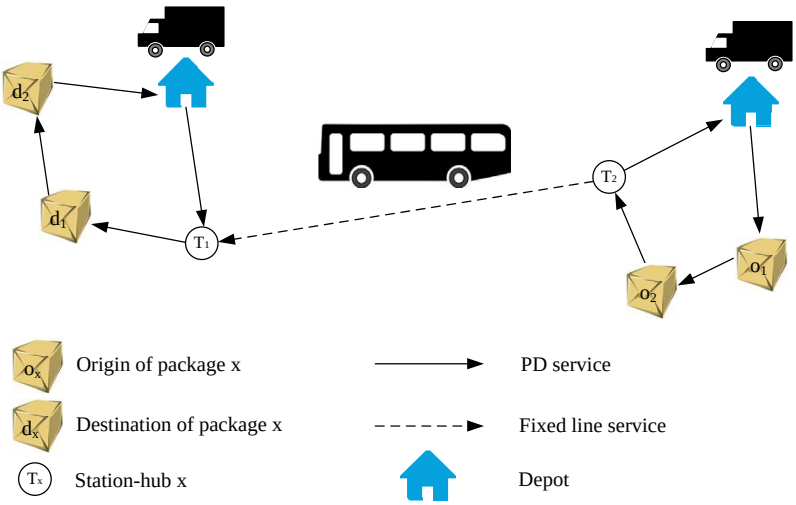

Figure 1 **An Illustration of the Integrated Transportation Network**

introduced exact solution algorithms (see, e.g., Dumas et al. (1991), Savelsbergh and Sol (1995), Røpke and Cordeau (2009), Baldacci et al. (2011a)). To solve the PDPTW-SL, this paper proposes a branch-and-price (B&P) algorithm. We formulate the problem as a path-based mixed-integer program (MIP). Since it is prohibitively time-consuming to generate all possible routes and consequently solve the formulation, the proposed algorithm uses column generation to construct promising routes. The column generator solves the *Elementary Shortest Path Problem with Resource and Precedence Constraints* (ESPPRPC), which is the natural pricing problem for the PDPTW-SL, in order to generate promising routes. *Precedence Constraints* mean that a request must be picked up from either its pickup, or a transfer node, and subsequently delivered to its delivery location, or a transfer node.

The scientific contributions of this paper are threefold: *(i)* we formulate the PDPTW-SL as a path-based MIP, *(ii)* we develop a B&P algorithm to solve small to medium-size PDPTW-SL instances, and *(iii)* we show how the algorithm can be applied to the PDPTW-T.

The remainder of the paper is structured as follows. Section 2 provides a brief review of related work. Section 3 introduces the MIP formulation for the PDPTW-SL, which is followed by the proposed solution methodology in Section 4. Section 5 reports the results obtained from extensive computational experiments. Conclusions are given in Section 6.

2. Literature Review

In this section, we first review existing studies on pickup and delivery problems (PDPs) with transfers, which are the most closely related to the PDPTW-SL. Then, we present recent studies on column generation algorithms applied to PDPs.

2.1. Pickup and Delivery Problem with Transfers

One of the first studies to investigate the dial-a-ride problem (DARP) with scheduled line services was done by Liaw et al. (1996). The authors proposed a heuristic algorithm and solved instances with up to 120 requests. The results show that significant operating cost savings can be achieved due to transfer opportunities. Later, Aldaihani and Dessouky (2003) considered an integrated DARP (IDARP) with public transportation services and proposed a two-stage heuristic algorithm, i.e., construction and improvement stages. The authors concluded that shifting some requests to public transportation services reduces the total traveled distance by PD-vehicles and the overall trip time of requests. Häll et al. (2009) introduced an arc-based MIP formulation to solve the IDARP without considering the corresponding schedules of the public transportation. The authors managed to solve instances with up to four requests. In the context of integrated passenger and freight transportation systems, Trentini et al. (2012) investigated a two-echelon vehicle routing problem with transshipment (VRPT) to public transportation. The authors developed an adaptive large neighborhood search (ALNS) heuristic to solve VRPT instances with up to 50 customers.

Cortes et al. (2010) proposed an exact branch-and-cut (B&C) algorithm for the PDPTW-T, which makes use of combinatorial Benders cuts. The proposed cuts provided significant CPU time savings, but the algorithm was limited to solving instances with up to six requests. Moreover, Masson et al. (2012, 2014) proposed an ALNS algorithm to solve both the PDP-T and DARP-T. The authors developed a constant-time feasibility check mechanism for the inter-dependent vehicle routes. In both cases, real-life instances with up to 193 requests were solved, yielding operational cost savings of up to 9%. In addition, Rais et al. (2013) introduced a new MIP formulation for the PDP-T, where transfers are allowed at any request node. The authors used a general-purpose MIP solver to tackle small-size instances with up to seven requests and observed 7% cost savings due to transfer opportunities.

The problem variant studied here was first introduced by Ghilas et al. (2016b), who proposed an arc-based MIP formulation for the PDPTW-SL. The model was tightened using several families of valid inequalities. The authors solved instances with up to 11 requests using a general-purpose MIP solver. Later, Ghilas et al. (2016a) introduced an ALNS algorithm to tackle the PDPTW-SL and reported results on instances with up to 100 requests. Finally, the authors concluded that significant operating cost savings can be achieved (i.e., up to 20%) with the use of scheduled lines. Ghilas et al. (2016c) extended the ALNS and embedded it into a sample average approximation framework to solve the PDPTW-SL with stochastic demands. The authors concluded that making use of SLs is still a promising alternative in uncertain environments, if the uncertainty is taken into account in the planning process.

2.2. Column Generation Algorithms

For a detailed overview of column generation algorithms, the interested reader is referred to Lübbecke and Desrosiers (2005). The first branch-and-price algorithm for the PDPTW was proposed by Dumas et al. (1991). In their formulation, a set-partitioning master problem considers columns which represent feasible routes. The resulting pricing problem is a shortest path problem with resource and precedence constraints. Later, Savelsbergh and Sol (1995) proposed an improved B&P algorithm by using heuristics in the pricing problem, by reducing the master problem size (e.g., removing unused columns), and finally by applying more effective branching rules. Xu et al. (2003) and Sigurd et al. (2004) applied column generation to address variants of the PDPTW arising in long-haul transportation and in the transportation of live animals, respectively. Mues and Pickl (2005) proposed a column generation method for the PDP-T, but the implementation of this idea was not provided and no results were reported. In another study, Røpke and Cordeau (2009) proposed a branch-and-cut-and-price algorithm to solve the PDPTW and successfully tackled large instances with up to 500 requests. The authors investigated two pricing problems, namely the elementary shortest path problem with resource and precedence constraints (ESPPRPC) and the non-elementary variant of the problem (SPPRPC), respectively. They concluded that solving the SPPRPC as pricing problem for the PDPTW does not yield significant benefits. This is mainly due to the fact that SPPRPC is strictly NP-hard. In order to speed up the algorithm used to solve the elementary shortest path problem with resource constraints (ESPPRC), Righini and Salani (2006, 2008) proposed various techniques, including bi-directional search and decremental state space relaxation methods. Later, Baldacci et al. (2011a) proposed another exact approach to solve the PDPTW. The authors describe a bounding procedure that combines heuristics with a cut-and-column generation procedure. Relying on the calculations of high quality lower and upper bounds, the algorithm can sometimes enumerate all columns with a negative reduced cost that may belong to an optimal solution. If this proves impossible, a branch-and-cut-and-price method is used. This algorithm has outperformed previous ones and was able to solve many previously unsolved instances.

Even though there were several attempts to develop exact solution algorithms for the PDPTW with transfers, solving medium to large-size instances remains challenging. To the best of our knowledge, this paper is the first attempt to apply column generation to the PDPTW-SL and PDPTW-T.

3. Description of the PDPTW-SL

In this section, we first introduce the notations and assumptions used in the rest of the paper. We then present the set partitioning formulation of the PDPTW-SL.

3.1. Definitions and Assumptions

Let $\mathcal{P}$ and $\mathcal{D}$ denote the sets of pickup and delivery nodes, respectively. We consider a set of n requests, where each request is associated with a pickup node $r \in \mathcal{P}$ and a delivery node $r+n \in \mathcal{D}$. We refer to a specific request r by its pickup node, e.g., $r \in \mathcal{P}$. Each request r is associated with two desired time windows: one for the origin ($[l_r,\ u_r]$), and one for the destination ($[l_{r+n},\ u_{r+n}]$). In addition, each request r has a demand d_r. Let $\mathcal{V}$ denote the set of vehicles. Each vehicle $v \in \mathcal{V}$ has a carrying capacity Q_v and an assigned depot $o(v) \in \mathcal{O}$ with a time window ($[l_{o(v)},\ u_{o(v)}]$).

The set of all physical transfer nodes is given by $\mathcal{S}$ while the set $\mathcal{E}$ contains all physical scheduled lines. Line $(i,\ j) \in \mathcal{E}$ is represented by a directed arc between the start and the end of the line and it has a set $\mathcal{K}^{ij}$ of indices w for the departure times p_{ij}^w from terminal i. We note that each SL may have a different frequency, thus the size of the sets $\mathcal{K}^{ij}$ may differ. Furthermore, it is assumed that SL vehicles are designed to carry a limited number of requests, thus implying a carrying capacity Q_{ij} for every $(i,j) \in \mathcal{E}$.

In order to model the waiting times of the PD vehicles and to allow multiple visits at transfer nodes, each physical SL $(i,\ j) \in \mathcal{E}$ is replicated n times as in Häll et al. (2009). The set of all replicated SLs is denoted by $\mathcal{F}$. In addition, in order to reduce the number of decision variables, each request r is assigned one replication of each SL $(i,\ j) \in \mathcal{E}$. The set of replicated SLs associated with request r is given by $\mathcal{F}^r$. Note that the replication process implies that each physical transfer node in $\mathcal{S}$ is replicated n times. The set of all replicated transfer nodes is denoted by $\mathcal{T}$.

To simplify the modeling, we also introduce the following additional notation. Let $\psi(t)$, $\forall t \in \mathcal{T}$, be the physical transfer node represented by t (i.e., $\psi(t) \in \mathcal{S}$). Set $\mathcal{T}^r$ represents the replicated transfer nodes associated with request r. In addition, $\mathcal{E}_t^-$ gives the set of physical scheduled lines that start at transfer node t and $\mathcal{E}_t^+$ represents the set of physical scheduled lines that end at transfer node t.

Figure 2 illustrates the one-SL replication for two requests. For the considered example, $\mathcal{E} \equiv \{(1,\ 2),\ (2,\ 1)\}$ (Figure 2.a) and $\mathcal{F} \equiv \{(1a,\ 2a),\ (2a,\ 1a),\ (1b,\ 2b),\ (2b,\ 1b)\}$ (Figure 2.b). Furthermore, $\mathcal{S} \equiv \{1,\ 2\}$ and $\mathcal{T} \equiv \{1a,\ 2a,\ 1b,\ 2b\}$, $\mathcal{T}^a \equiv \{1a,\ 2a\}$, $\mathcal{F}^a \equiv \{(1a,\ 2a),\ (2a,\ 1a)\}$ and $\psi(1a) = \psi(1b) = 1$.

We also make the following practical assumptions: (i) a storage space for packages that need to be shipped on a SL is available at each physical transfer node (see, e.g., DHL-Packstation (2015)), and these automatic lockers may be used to temporarily store the packages from a transfer node; (ii) as multiple freight carriers may be using SL services, each of these carriers is assigned a part of the storage space and SL-vehicle capacity (e.g., contract-based agreement); (iii) the cost η_{ij} per unit shipped on the SL $(i,\ j)$ includes transportation, handling (transshipment) and storage costs.

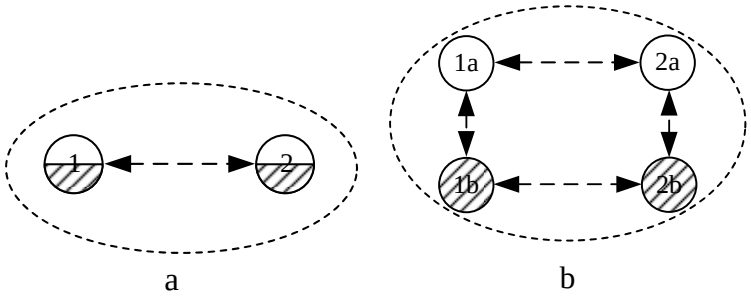

Figure 2 **Physical and Virtual Scheduled Lines (Ghilas et al. 2016b)**

To formulate the PDPTW-SL as a set-partitioning problem, we let Ω denote the set of all feasible routes (paths) satisfying precedence, elementarity, time windows and PD-vehicle capacity constraints. Moreover, we let Ω^v be the set of feasible paths that can be executed by PD vehicle $v \in \mathcal{V}$. A path is feasible if it satisfies the precedence and capacity constraints (see Section 4.1.1 for more details on the feasibility of PD vehicle routes). The routing cost of each path $p \in \Omega$ is given by c_p. In addition, constant z_p^i takes value 1 if node $i \in \mathcal{P} \cup \mathcal{D}$ is visited in path p, 0 otherwise. Constant a_p^{ij} takes value 1 if path p contains arc (i, j), 0 otherwise. Finally, m_p^{it} is a parameter taking value 1 if request node $i \in \mathcal{P} \cup \mathcal{D}$ and replicated transfer node $t \in \mathcal{T}$ are visited in path p by respecting precedence constraints, 0 otherwise.

The decision variables used to represent the routing and scheduling of the PD vehicles, along with the timing of the requests, are given as follows. Binary variable x_p takes value 1 if path $p \in \Omega$ is in the solution, 0 otherwise. If request r departs on the SL (i, j) at scheduled departure time p_{ij}^w, binary variable q_{ij}^{rw} takes value 1, otherwise 0. Continuous variables β_i, γ_i and α_v indicate the departure time of a vehicle from node i, the departure time of request r from transfer node i, and the time at which vehicle v returns to its depot, respectively.

Finally, the PDPTW-SL can be defined on a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes (i.e, $\mathcal{N} \equiv \mathcal{O} \cup \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$) and $\mathcal{A} \equiv \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$ is the set of arcs defined as follows:

$\mathcal{A}_1 = ((\mathcal{P} \cup \mathcal{D})) \times (\mathcal{P} \cup \mathcal{D})) \setminus \{(r+n, r): r \in \mathcal{P}\}$
$\mathcal{A}_2 = \{(i, j): i \in \mathcal{O}, j \in \mathcal{P}\} \cup \{(i, j): i \in \mathcal{D}, j \in \mathcal{O}\} \cup (\mathcal{O} \times \mathcal{T})$
$\mathcal{A}_3 = ((\mathcal{P} \cup \mathcal{D})) \times \mathcal{T}) \setminus (\{(j, r): r \in \mathcal{P}, j \in \mathcal{T}^r\} \cup \{(r+n, j): r \in \mathcal{P}, j \in \mathcal{T}^r\})$
$\mathcal{A}_4 = \{(i, j): i, j \in \mathcal{T}, (\psi(i), \psi(j)) \notin \mathcal{E}\}$.

This definition of the set of arcs considers all possible connections except those that would be infeasible (e.g., direct paths from a depot to a delivery node). Figure 3 presents an example with one depot, two requests and one SL along with all corresponding arcs. Sets $\mathcal{A}_1$ and $\mathcal{A}_2$ are given in Figure 3.a whereas sets $\mathcal{A}_3$ and $\mathcal{A}_4$ are given in Figure 3.b. Finally, each arc $(i, j) \in \mathcal{A}$ has a deterministic travel time t_{ij} and service time at node $i \in \mathcal{N}$ is given by s_i.

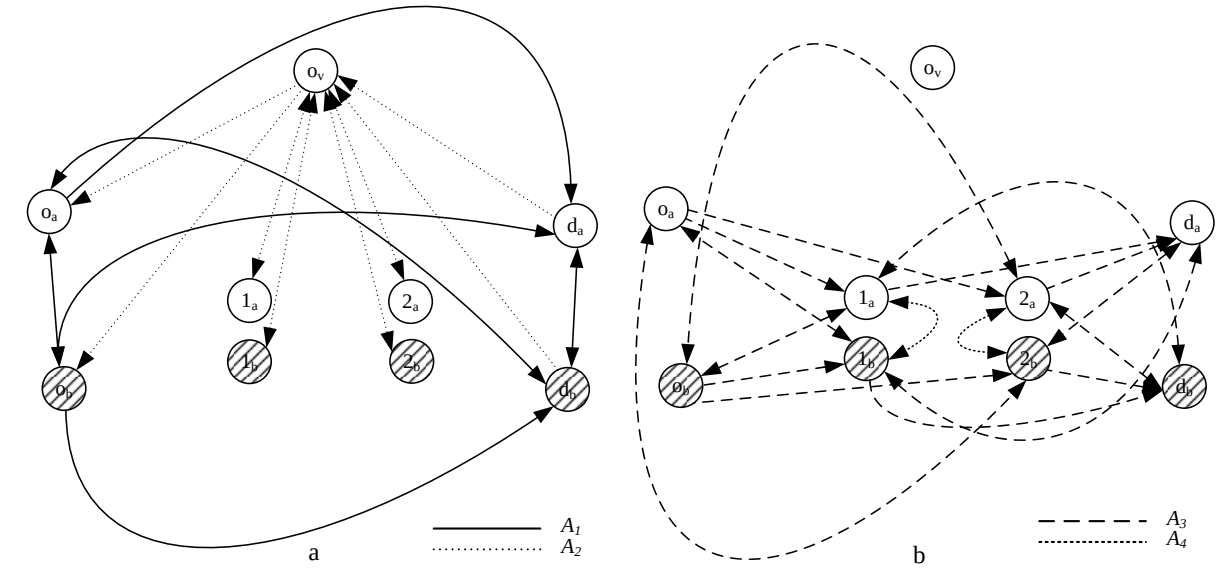

Figure 3 An Example Network with Two Requests and One SL

3.2. A Set-Partitioning Formulation

The PDPTW-SL can be formulated as the following set-partitioning problem:

$$\text{Minimize} \quad \sum_{p\in\Omega} c_p x_p + \sum_{r\in\mathcal{P}} \sum_{(i,j)\in\mathcal{E}} \sum_{w\in\mathcal{K}^{ij}} \eta_{ij} d_r q_{ij}^{rw} \tag{1}$$

subject to

$$\sum_{p\in\Omega^v} x_p \le 1 \qquad \forall v \in \mathcal{V} \tag{2}$$

$$\sum_{p\in\Omega} z_p^i x_p = 1 \qquad \forall i \in \mathcal{P}\cup\mathcal{D} \tag{3}$$

$$\sum_{p\in\Omega} m_p^{rt} x_p \le \sum_{p\in\Omega} \sum_{t_1\in\mathcal{T}^r\setminus\{t\}} m_p^{r+n,t_1} x_p \qquad \forall r\in\mathcal{P}, t\in\mathcal{T}^r \tag{4}$$

$$\sum_{p\in\Omega} m_p^{rt} x_p + \sum_{(i,j)\in\mathcal{E}^+_{\psi(t)}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw} = \sum_{p\in\Omega} m_p^{r+n,t} x_p + \sum_{(i,j)\in\mathcal{E}^-_{\psi(t)}} \sum_{w\in\mathcal{K}^{ij}} q_{ij}^{rw} \qquad \forall r\in\mathcal{P}, t\in\mathcal{T}^r \tag{5}$$

$$\sum_{r\in\mathcal{P}} d_r q_{ij}^{rw} \le Q_{ij} \qquad \forall (i,j)\in\mathcal{E}, w\in\mathcal{K}^{ij} \tag{6}$$

$$\sum_{w\in\mathcal{K}^{\psi(i),\psi(j)}} q_{\psi(i),\psi(j)}^{rw} = 1 \implies \gamma_j^r \ge \gamma_i^r + t_{ij} + s_j \qquad \forall r\in\mathcal{P}, (i,j)\in\mathcal{F}^r \tag{7}$$

$$\sum_{p\in\Omega} a_p^{ij} x_p = 1 \implies \beta_j \ge \beta_i + t_{ij} + s_j \qquad \forall i\in\mathcal{N}, j\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \tag{8}$$

$$\sum_{p\in\Omega^v} a_p^{i,o(v)} x_p = 1 \implies \alpha_v \ge \beta_i + t_{i,o(v)} + s_{o(v)} \qquad \forall i\in\mathcal{P}\cup\mathcal{D}\cup\mathcal{T}, v\in\mathcal{V} \tag{9}$$

$$\beta_{r+n} \ge \beta_r + t_{r,r+n} + s_{r+n} \qquad \forall r\in\mathcal{P} \tag{10}$$

$$l_i \leq \beta_i - s_i \leq u_i \qquad \forall i \in \mathcal{P} \cup \mathcal{D} \tag{11}$$

$$l_{o(v)} \leq \alpha_v \leq u_{o(v)} \qquad \forall v \in \mathcal{V} \tag{12}$$

$$q^{rw}_{\psi(i),\psi(j)} = 1 \implies \gamma^r_i = p^w_{\psi(i),\psi(j)} \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{F}^r, w \in \mathcal{K}^{\psi(i),\psi(j)} \tag{13}$$

$$\sum_{(i,j) \in \mathcal{E}^+_{\psi(t)}} \sum_{w \in \mathcal{K}^{ij}} q^{rw}_{ij} = 1 \implies \gamma^r_t = \beta_t \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{14}$$

$$\beta_t \leq \gamma^r_t \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{15}$$

$$x_p \in \{0,1\} \qquad \forall p \in \Omega \tag{16}$$

$$q^{rw}_{ij} \in \{0,1\} \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{E}, w \in \mathcal{K}^{ij} \tag{17}$$

$$\beta_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{18}$$

$$\gamma^r_i \geq 0 \qquad \forall r \in \mathcal{P}, i \in \mathcal{T} \tag{19}$$

$$\alpha_v \geq 0 \qquad \forall v \in \mathcal{V}. \tag{20}$$

The objective (1) is the sum of two cost functions. The first function is related to routing and the second one considers SL-related costs. Constraints (2) assure that each PD vehicle is used at most once. Constraints (3) ensure that every request node (pickup or delivery) is visited exactly once. Constraints (4) enforce the visit of transfer nodes if the pickup and delivery nodes of a request are visited in different paths (vehicle routes). Constraints (5) are the flow balance constraints for each transfer node. In other words, a request can travel from a transfer node either on a SL or on a PD vehicle. The capacity of the SLs is considered in constraints (6) and these consider each specific scheduled departure time. Constraints (7) assure timing at transfer nodes if there is a request flow on the corresponding SL. The scheduling of the PD vehicles is considered in constraints (8) and (9). Constraints (10) assure that the pickup node is visited earlier than the associated delivery node. Constraints (11) and (12) force time windows to be respected. The timetabling of the available SLs is considered in constraints (13). In particular, these constraints link the request flows on the SLs with the corresponding timing variables. Constraints (14) assure that a request departs from a destination transfer node at the same time as a PD vehicle. Constraints (15) make sure a request departs from a transfer node no earlier than the arrival of a PD vehicle at that node. Note that constraints (7)–(9) and (13)–(14) are written as implications and standard linearization techniques can be used to express them as one or two linear inequalities.

We note that physical transfer nodes ($\mathcal{S}$) as well as replicated transfer nodes ($\mathcal{T}$) that represent them are needed in the presented model in order to assure capacity constraints (6) on the SLs. The capacity constraints apply to each physical SL and to each scheduled departure time. Therefore, decision variables q^{rw}_{ij} consider both the request flow on the physical SLs and the scheduled departure times. In this case, the SL flow is performed between physical transfer nodes, whereas

replicated transfer nodes are used to consider timing of the requests on SLs and PD vehicles, considering waiting times and pickup/delivery operations at the same physical transfer node.

4. Branch-and-Price Algorithm

In this section, we introduce the B&P algorithm used to solve the PDPTW-SL. We first describe the column generation method, which includes the pricing problem definition, the labeling algorithm used to solve it and some acceleration techniques. Then, we present the branching strategy to explore the enumeration tree, and we discuss potential improvements to the algorithm.

4.1. Column Generation

Column generation is used to solve the linear programming (LP) relaxation in each node of the branch-and-bound tree. A lower bound (LB) on the optimal value of the PDPTW-SL can be obtained by solving the LP relaxation of (1)–(20), which is obtained by replacing the integrality constraints (16) and (17) with the following:

$$0 \leq x_p \leq 1 \qquad \forall p \in \Omega \tag{21}$$

$$0 \leq q_{ij}^{rw} \leq 1 \qquad \forall r \in \mathcal{P}, (i,j) \in \mathcal{E}, w \in \mathcal{K}^{ij}. \tag{22}$$

Since the size of Ω is usually very large, it is almost impossible to solve or even to explicitly represent model (1)–(20). Instead, a restricted master problem (RMP) is obtained by considering a subset of paths $\overline{\Omega} \subseteq \Omega$ in each iteration. Because $\overline{\Omega}$ might not yield a feasible solution, artificial variables with a large cost are added to the RMP to ensure that primal and dual solutions can be obtained. Once the RMP is solved, its dual solution is used to define the objective function of the pricing subproblems. The subproblem for each PD vehicle aims to generate negative reduced cost variables (columns) x_p with respect to this dual solution. If such variables are found, they are added to the RMP, which is then solved again to start a new iteration. Otherwise, when no subproblem can find any such column, the column generation process stops and the obtained solution to the current RMP is optimal for the master problem.

4.1.1. Pricing Problem Formulation. The pricing problem of the PDPTW-SL is a variant of the *Elementary Shortest Path Problem with Resource and Precedence Constraints* (ESPPRPC). The main difference in the ESPPRPC for the PDPTW-SL and for the classical PDPTW is that in the former problem, each request can be served in two ways: direct (i.e., by one vehicle from its origin to its destination) or indirect (i.e., using a scheduled line as part of the journey, thus the vehicle needs to visit a transfer node). In particular, in each of the generated paths, the requests are served in one of the following three ways: *(i)* classical way – first visit the pickup node $r \in \mathcal{P}$, and then deliver the request to node $r+n \in \mathcal{D}$; *(ii)* first visit the pickup node $r \in \mathcal{P}$, and consequently deliver the request to a replicated transfer node e.g., $t_1 \in \mathcal{T}^r$. At this point, the labeling algorithm

does not assure that this request is re-collected by another route. The synchronization takes place in the master problem in constraints (4). Consider a request r that uses a scheduled line. Hence, constraints (4) make sure that if a route, which first visits pickup node $r \in \mathcal{P}$ and then a transfer node $t_1 \in \mathcal{T}^r$ (related to r), is chosen, then a different route (performed by another vehicle), in which another replicated transfer node $t_2 \in \mathcal{T}^r \backslash \{t_1\}$ (related to r) is visited before its corresponding delivery node $r+n \in \mathcal{D}$, must be selected. Finally, in *(iii)* a request r is picked up from a transfer node e.g., $t_2 \in \mathcal{T}^r$, and is subsequently delivered to node $r+n \in \mathcal{D}$. Again, the pricing problem does not assure the synchornization between the routes, but only the precedence and capacity constraints. Since each physical transfer node is replicated for each request, it is possible that all requests can be transferred at or picked up from a transfer point by visiting the corresponding replicated transfer nodes sequentially. Figure 4 illustrates the possible delivery options in the pricing problem, where the flow from t_1 to t_2 is assured in the master problem only.

Note that the triangle inequality in terms of reduced cost does not hold for the pricing problem, meaning that $\overline{c}_{r,r+n}$ may be larger than $\overline{c}_{r,t_1} + \overline{c}_{t_2,r+n} + \eta_{t_1,t_2}$, where $\overline{c}_{ij}$ is the reduced cost of arc $(i,\, j)$, $\forall r \in \mathcal{P}$, $t_1,\, t_2 \in \mathcal{T}^r$.

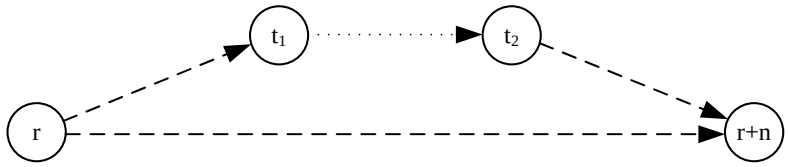

Figure 4 An Illustration of Possible Shipment Options

Since the vehicles may be heterogeneous and multiple depots are available, the ESPPRPC can be solved for each PD vehicle v. We note that the set of nodes considered in the pricing problem is $\mathcal{N} \equiv \mathcal{P} \cup \mathcal{D} \cup \mathcal{T} \cup \{o(v)\} \cup \{o(v^{'})\}$, where $o(v^{'})$ is a replicated depot of vehicle v. The aim of the ESPPRPC is to generate a least reduced cost elementary shortest path from node $o(v)$ to $o(v^{'})$ by considering the available resources, such as time windows and capacity.

Let x_{ij} be a binary variable which takes value 1 if arc $(i,\, j)$ is in the path, and 0 otherwise. In addition, let q_i be a continuous decision variable, which indicates the total load of vehicle v after visiting node $i \in \mathcal{N}$. The pricing problem can now be formulated for each PD vehicle $v \in \mathcal{V}$ as the following MIP:

$$\text{Minimize} \quad \sum_{(i,j)\in\mathcal{A}} \overline{c}_{ij} x_{ij} \tag{23}$$

subject to

$$\sum_{i\in\mathcal{N}} x_{o(v),i} = \sum_{i\in\mathcal{N}} x_{i,o(v')} = 1 \tag{24}$$

$$\sum_{j\in\mathcal{N}} x_{ij} = \sum_{j\in\mathcal{N}} x_{ji} \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \tag{25}$$

$$\sum_{i\in\mathcal{N}} x_{ir} \leq \sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}^r} x_{it} + \sum_{i\in\mathcal{N}} x_{i,r+n} \qquad \forall r \in \mathcal{P} \tag{26}$$

$$\sum_{i\in\mathcal{N}} x_{i,r+n} \leq \sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}^r} x_{it} + \sum_{i\in\mathcal{N}} x_{ir} \qquad \forall r \in \mathcal{P} \tag{27}$$

$$x_{ij} = 1 \implies \beta_j \geq \beta_i + t_{ij} + s_j \qquad \forall i,j \in \mathcal{N} \tag{28}$$

$$\sum_{i\in\mathcal{N}\setminus\mathcal{T}^r} x_{ir} = 1 \text{ and } \sum_{i\in\mathcal{N}} x_{it} = 1 \implies \beta_t \geq \beta_r + t_{rt} + s_t \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{29}$$

$$\sum_{i\in\mathcal{N}\setminus\{r\}} x_{i,r+n} = 1 \text{ and } \sum_{i\in\mathcal{N}\setminus\{r\}} x_{it} = 1 \implies \beta_{r+n} \geq \beta_t + t_{t,r+n} + s_{r+n} \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r \tag{30}$$

$$\beta_{r+n} \geq \beta_r + t_{r,r+n} + s_{r+n} \qquad \forall r \in \mathcal{P} \tag{31}$$

$$l_i \leq \beta_i - s_i \leq u_i \qquad \forall i \in \mathcal{N} \tag{32}$$

$$\sum_{i\in\mathcal{N}\setminus\mathcal{T}^r} x_{ir} = 1 \text{ and } x_{jt} = 1 \implies q_t^v = q_j^v - d_r \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r, j \in \mathcal{N} \tag{33}$$

$$\sum_{i\in\mathcal{N}\setminus\{r\}} x_{i,r+n} = 1 \text{ and } x_{jt} = 1 \implies q_t^v = q_j^v + d_r \qquad \forall r \in \mathcal{P}, t \in \mathcal{T}^r, j \in \mathcal{N} \tag{34}$$

$$x_{ir} = 1 \implies q_r^v = q_i^v + d_r \qquad \forall r \in \mathcal{P}, i \in \mathcal{N} \tag{35}$$

$$x_{i,r+n} = 1 \implies q_{r+n}^v = q_i^v - d_r \qquad \forall r \in \mathcal{P}, i \in \mathcal{N} \tag{36}$$

$$q_i^v \leq Q_v \qquad \forall i \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{T} \tag{37}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in \mathcal{N} \tag{38}$$

$$\beta_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{39}$$

$$q_i \geq 0 \qquad \forall i \in \mathcal{N}. \tag{40}$$

The objective function (23) minimizes the reduced cost of a path. Constraints (24) assure that there is exactly one outgoing arc from $o(v)$ and one incoming arc to $o(v')$. The flow balance for each node is given in constraints (25). Precedence constraints are imposed by (26) and (27). These constraints make sure that each request can be picked up either *(i)* from its pickup node, and consequently delivered to its delivery node or to one of the related transfer nodes, or *(ii)* from one of the related transfer nodes and transferred to its delivery node. Time windows are enforced in constraints (28)–(32). Capacity constraints are imposed by (33)–(37). To help clarity, some constraints are written as implications and can be linearized by using big-M techniques. In addition, expressions (23) and (41) can be made equivalent by adding additional decision variables in the model and linearizing the constraints written as implications.

Let A_v, B_i, C_t^r, D_t^r, E_{ij} and F_i^v be the dual variables associated with constraints (2)–(5), (8) and (9), respectively. Moreover, let $r(j) \in \mathcal{P}$ be the request related to node $j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$. In addition, M_{ij} are large coefficients used in the linearized form of constraints (8) and (9). The reduced cost of a path p can be calculated as follows:

$$\begin{aligned}\overline{c}_p^v = c_p - A_v - \sum_{i \in N} \sum_{j \in \mathcal{P} \cup \mathcal{D}} a_p^{ij} \left(B_j + M_{ij} E_{ij}\right) \\ - \sum_{i \in N} \sum_{j \in \mathcal{T}} a_p^{ij} \left(M_{ij} E_{ij} + m_p^{r(j),j} C_j^{r_j} - \sum_{k \in \mathcal{T}^{r(j)}: j \neq k} m_p^{r(j)+n,k} C_k^{r(j)} + m_p^{r(j),j} D_j^{r(j)} - m_p^{r(j)+n,j} D_j^{r(j)} \right) \\ - \sum_{i \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}} a_p^{i,o(v)} \left(M_{i,o(v)} F_i^v\right), \quad \forall v \in \mathcal{V}, p \in \Omega^v, \end{aligned} \tag{41}$$

where constants a_{ij}^p and $m_p^{i,t}$ were defined in Section 3.1. Recall that constant $a_{ij}^p \ \forall \ i,j \in \mathcal{N}$ has value 1 if arc (i,j) is traversed in path p, 0 otherwise. In addition, $m_p^{i,t} \ \forall \ i \in \mathcal{P}$, $t \in \mathcal{T}$ is a constant with value 1 if pickup node i is visited before visiting transfer node t in path p, 0 otherwise. Constant $m_p^{i+n,t} \ \forall \ i \in \mathcal{P}$, $t \in \mathcal{T}$ has value 1 if the delivery node $i+n$ is visited after visiting transfer node t in path p, 0 otherwise.

Resource constrained shortest path problems used in column generation approaches are usually solved using dynamic programming methods called label-setting (or labeling) algorithms. In the following sections we explain how the ESPPRPC can be solved by using the proposed labeling algorithm.

4.1.2. Labeling Algorithm. The objective of the ESPPRPC is to find a cheapest path from a source node s to a sink node t in a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. Every arc in this graph is associated with a cost and the cost of a path is the sum of the costs of all arcs traversed by the path. The path must satisfy the available resources, such as cost, capacity and time, used between the source node and sink node. An overview of resource constrained shortest path problems and of appropriate solution methodologies was given by Irnich and Desaulniers (2005).

Labeling algorithms build partial paths in graph $\mathcal{G}$. Each such path starts at s and ends at any node $i \in \mathcal{N}$. Existing partial paths are extended along the arcs leaving the end node of the partial path. In general, labeling algorithms generate all possible feasible paths in the graph. To speed up the labeling algorithm, *dominated* paths can be removed during the process. A path p dominates another path p' if both end at the same node $i \in \mathcal{N}$, resource consumption (including the cost) in p is smaller than or equal to what it is in p', and all feasible extensions of p' by a partial path to the sink node are also feasible for p. Hence, *sufficient* conditions can be used to determine dominance criteria. The conditions used in this paper are discussed in the corresponding

sections below. Partial paths are represented by *labels* that contain information about resource consumption at the end node of the partial path and dominance is verified in terms of the labels.

In this section, we introduce a new labeling algorithm for the ESPPRPC that takes advantage of restricted dominance conditions, since the triangle inequality does not hold for the cost. We assume that the source and sink nodes are $o(v)$ and $o(v^{'})$, $\forall v \in \mathcal{V}$, respectively. To speed up the labeling algorithm, a bi-directional search is performed where labels are extended both forward from $o(v)$ to its successors and backward from $o(v^{'})$ to its predecessors. Both forward and backward labels are not allowed to cross a certain threshold (e.g., the middle of the planning horizon). At the end, forward and backward labels are merged to construct entire feasible paths. It has been observed that a bi-directional search may in practice lead to substantially shorter running times for the related ESPPRC as discussed by Righini and Salani (2006, 2008) and by Dabia et al. (2016).

The Forward Labeling Algorithm. In the forward labeling algorithm, labels are extended from the source (i.e., $o(v)$) to its successor nodes. For each forward label (L_f), we store the following data:

- η last node of the partial path;
- t departure time from the last node;
- q total load after visiting the last node;
- c accumulated cost (initialized $c = 0$);
- $\mathcal{O}$ set of requests that have been started, but not completed (i.e., the request has been picked up either from the pickup node or from a transfer node, but not delivered to its delivery or transfer node);
- $\mathcal{C}$ set of completed requests (i.e., $\mathcal{C} \cap \mathcal{O} \equiv \emptyset$);
- $\mathcal{O}^T$ set of started requests from a transfer node, such that $\mathcal{O}^T \subseteq \mathcal{O}$.

We also store the reference of the parent label in each label. The resources are t, q, c, $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$. Note that the resources are initialized as follows: $t = 0$, $q = 0$, $c = 0$, and finally, $\mathcal{O} \equiv \mathcal{O}^T \equiv \mathcal{C} \equiv \emptyset$. The notation t_{L_f} is used to refer to the departure time in forward label L_f and similar notation is used for the rest of the resources (e.g., η_{L_f}, q_{L_f}, etc.).

When extending a forward label L_f along an arc (η_{L_f}, j), the extension is feasible if:

$$t_{L_f} + t_{\eta_{L_f},j} \leq u_j, \tag{42}$$

$$q_{L_f} + d_j \leq Q_v, \tag{43}$$

where $t_{\eta_{L_f},j}$ is the traveling time between nodes η_{L_f} and j, and u_j is the upper bound of node j's time window. Moreover, d_j is the demand of node j, and Q_v is the capacity of PD vehicle v.

The capacity constraints are satisfied by expression (43). We note that time window constraints are satisfied in the master problem due to synchronization constraints between multiple routes. In the subproblem, time windows are used to eliminate infeasible label extensions and to reduce the solution space. Moreover, L_f and j must satisfy one of the following five conditions, which ensure that the extension is feasible with the sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$:

$$j \in \mathcal{P} \wedge j \notin \mathcal{O}_{L_f} \wedge j \notin \mathcal{C}_{L_f} \tag{44}$$

$$j \in \mathcal{D} \wedge j - n \in \mathcal{O}_{L_f} \tag{45}$$

$$j \in \mathcal{T} \wedge r(j) \notin \mathcal{O}_{L_f} \wedge r(j) \notin \mathcal{C}_{L_f} \tag{46}$$

$$j \in \mathcal{T} \wedge r(j) \in \mathcal{O}_{L_f} \setminus \mathcal{O}^T_{L_f} \wedge r(j) \notin \mathcal{C}_{L_f} \tag{47}$$

$$j = o(v^{'}) \wedge \mathcal{O}_{L_f} \in \emptyset. \tag{48}$$

Expression (44) states that visiting a pickup node $j \in \mathcal{P}$ is feasible if request j is not started (has not been picked up earlier in the partial path from either a transfer node $t \in \mathcal{T}^j$ or from the pickup node $j \in \mathcal{P}$), and if this request is not completed (has not been served) in this partial path. Expression (45) implies that the extension to a delivery node j is feasible if the corresponding request is started (has been picked up, either from its origin node $j-n \in \mathcal{P}$, or from a replicated transfer node $t \in \mathcal{T}^{j-n}$). In addition, expression (46) implies that the extension to a replicated transfer node t is feasible if the request related to t is not started (has not been picked up in the partial path) and the request is not completed. In this case, the request related to the transfer node t will be picked up from t. Furthermore, expression (47) states that the extension to a transfer node t is feasible if the request related to t is started from its pickup node, and not from a transfer node, and if it is not completed. In this case, the request will be transferred to t. Finally, expression (48) assures that no partial path ends before all started requests are served.

The above conditions assure that a path is feasible in terms of precedence, capacity, and time-window constraints. We note that timing is synchronized in the master problem. However to reduce the complexity of the pricing problem, we assure timing (time window constraints) for each individual path generated, irrespective of other paths.

When label η_{L_f} and node j satisfy conditions (42) and (43), along with one of the following conditions, the corresponding updates on sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$ are performed as follows:

- expression (44): $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \cup \{j\}$;
- expression (45): $\mathcal{C}_{L_f} \leftarrow \mathcal{C}_{L_f} \cup \{j-n\}$, $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \setminus \{j-n\}$, and $\mathcal{O}^T_{L_f} \leftarrow \mathcal{O}^T_{L_f} \setminus \{j-n\}$;
- expression (46): $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \cup \{r(j)\}$ and $\mathcal{O}^T_{L_f} \leftarrow \mathcal{O}^T_{L_f} \cup \{r(j)\}$;
- expression (47): $\mathcal{C}_{L_f} \leftarrow \mathcal{C}_{L_f} \cup \{r(j)\}$ and $\mathcal{O}_{L_f} \leftarrow \mathcal{O}_{L_f} \setminus \{r(j)\}$;
- expression (48).

The following dominance criteria are valid for the PDPTW-SL.

Proposition 1 (Dominance 1) *A label L^2_f is dominated by label L^1_f if:*

1. $\eta_{L^1_f} = \eta_{L^2_f}$,
2. $t_{L^1_f} \leq t_{L^2_f}$,
3. $c_{L^1_f} \leq c_{L^2_f}$,

4. $\mathcal{O}_{L_f^1} \equiv \mathcal{O}_{L_f^2}$,
5. $\mathcal{C}_{L_f^1} \subseteq \mathcal{C}_{L_f^2}$.

The aforementioned conditions make sure that any feasible extension of L_f^2 is also feasible for L_f^1 and resource consumption in any extension of L_f^2 is larger than or equal to the corresponding consumption in L_f^1. The dominance criteria proposed here are inspired from those of Dumas et al. (1991) and Røpke and Cordeau (2009). We note that the triangle inequality in terms of reduced costs does not hold for the pricing problem of the PDPTW-SL, since there is the possibility of serving requests in two ways. In other words, a transfer node can either be an origin node for a request r (i.e., $r \in \mathcal{O}_{L_f}$ and $\{r\} \notin \mathcal{C}_{L_f}$) or can be a destination node for r (i.e., $\{r\} \in \mathcal{C}_{L_f}$). Therefore, the dominance criteria (1)–(5) are restrictive.

Proof of Proposition 1: Let L_f^1 and L_f^2 be two labels that satisfy the five conditions in Proposition 1. In addition, let $R_{L_f^1}$ and $R_{L_f^2}$ be the sets of all feasible extensions of labels L_f^1 and L_f^2, respectively. We show that *(i)* any feasible extension L of L_f^2 is also a feasible extension of L_f^1 (i.e., $R_{L_f^2} \subseteq R_{L_f^1}$) and *(ii)* for any feasible extension L of L_f^2 we have that $c_{L_f^1 \oplus L} \leq c_{L_f^2 \oplus L}$.

(i) Because of an elementarity assumption and conditions (2), (4) and (5) of the dominance test, the relation between the sets of all feasible extensions of L_f^1 and L_f^2 implies that $R_{L_f^2} \subseteq R_{L_f^1}$.

(ii) Because of condition (3) of the dominance test, the best extension of L_f^1 will never be worse than the best extension of L_f^2. Hence, L_f^1 dominates label L_f^2. Finally, we note that it is not necessary to consider the load q in the dominance test because of condition (4). □

The Backward Labeling Algorithm. The backward labeling algorithm is an inverted analogue of the forward labeling. The main changes needed to adapt the forward labeling algorithm are as follows:

- Labels are extended from the sink node to its predecessors (e.g., arcs are reversed);
- In the considered paths, delivery(destination) transfer nodes are visited before corresponding pickup(origin) transfer nodes.

The label extension function and dominance criteria of the backward labeling algorithm can be adapted similarly to their forward analogue. More details on the backward labeling algorithm can be found in the Appendix.

Merging Forward and Backward Labels. When all forward and backward labels are generated, they are merged to construct feasible paths with negative reduced cost. A forward label L_f and a backward label L_b can be merged if the following conditions are satisfied:

$$\mathcal{O}_{L_f} \equiv \mathcal{O}_{L_b} \tag{49}$$

$$\mathcal{O}^T_{L_f} \cap \mathcal{O}^T_{L_b} \equiv \emptyset \tag{50}$$

$$\mathcal{C}_{L_f} \cap \mathcal{C}_{L_b} \equiv \emptyset \tag{51}$$

$$t_{L_f} + t_{\eta_{L_f},\eta_{L_b}} + s_{\eta_{L_b}} \leq t_{L_b}. \tag{52}$$

Condition (49) assures that the started (but not completed) requests in the forward label are the same as the finished (but not started) requests in the backward label. Expression (50) makes sure that a request cannot be picked up at a transfer node and delivered to another transfer node within the same route. Condition (51) makes sure that no request is served twice (once in each of the forward and backward labels) in the considered route. Finally, condition (52) assures that merging the two labels leads to feasible scheduling.

Since we use the dominance test, not all feasible paths may be generated. However, a path that minimizes the reduced cost will necessarily be created. Labels are discarded only if another label exists that can be extended in the same ways as the discarded label. Hence, the extension of the dominating label will always lead to a path with lower or equal reduced cost compared to the same extension of the discarded label.

4.1.3. Acceleration techniques. To speed up the column generation, in addition to the bi-directional search in the labeling algorithm, we apply the following ideas.

Pricing problem heuristics. The performance of the B&P algorithm can be accelerated by using heuristics to solve the pricing problem. Heuristics search for easy-to-find negative-reduced cost paths and add them to the master problem. When heuristics fail to find any such path, the exact algorithm is used. Ideally, for every node in the branching tree, the exact algorithm should be called only once to prove optimality, i.e., that no more paths with negative reduced cost exist. In our branch-and-price algorithm, we employ two heuristic algorithms.

First, *H1* is a truncated labeling algorithm as in Dabia et al. (2016), in which only a limited number of labels (with the best cost) for each node are stored and considered for possible extension. Here, we store only 50 labels at each node.

Second, *H2* uses the proposed labeling algorithm with stronger dominance criteria. More specifically, for both the forward and backward labeling algorithms, we use the same dominance criteria as before except condition (5).

Pre-processing and label elimination. In this section, we provide some important problem-specific pre-processing techniques that improve the performance of the algorithm. These are valid under the assumption that triangle inequality holds in terms of travel time, namely, $t_{r,r+n} \leq t_{r,k} + t_{k,r+n}$.

Let $r \in \mathcal{P}$ and $t \in \mathcal{T}^r$ be a destination transfer node of r, then E_t^d is the earliest arrival time at t on a scheduled line as a destination transfer node. It is computed as follows:

$$E_t^d = \underset{k \in \mathcal{O}, (i,j) \in \mathcal{F}^t}{\arg\min} \{\lceil (\arg\max\{t_{k,r} + s_r; l_r + s_r\} + t_{r,i} + s_i) \rceil + t_{i,j} + s_j\}, \tag{53}$$

where $\mathcal{F}^t$ is the set of replicated scheduled lines that have the destination node t.

In addition, let L_t^o be the latest feasible departure time of request r on a scheduled line from transfer node t, $\forall r \in \mathcal{P}$, $t \in \mathcal{T}^r$. It is computed as follows:

$$L_t^o = \underset{(i,j)\in\mathcal{F}_t}{\arg\min}\{\lfloor(u_{r+n} + s_{r+n} - t_{j,r+n} - s_j - t_{i,j})\rfloor\}, \tag{54}$$

where $\mathcal{F}_t$ is the replicated scheduled line set that contains SLs which have the origin in node t.

Note that E_t^d and L_t^o are both used in the labeling algorithm to restrict the solution space. In other words, if e.g., arrival time at a *destination* transfer node is smaller than E_t^d, then the departure time is set to E_t^d. Similarly, if the arrival time at an *origin* transfer node is larger than L_t^o, than this extension is considered as infeasible.

Moreover, for each request r and each replicated transfer node $t \in \mathcal{T}^r$, the time window of node t is updated as follows:

$$l_t = \underset{k\in\mathcal{O}}{\arg\min}\{(\arg\max\{t_{k,r} + s_r; l_r + s_r\} + t_{r,t})\}, \tag{55}$$

$$u_t = \underset{k\in\mathcal{O}}{\arg\max}\{(\arg\min\{u_k - t_{r+n,k} - s_{r+n}; u_{r+n} + s_{r+n}\} - t_{t,r+n} - s_t)\}. \tag{56}$$

We note that if $L_t^o < l_t$ and $E_t^d > u_t$ or $u_t < l_t$, there is no feasible solution by visiting node t. Therefore, any label extension to node t can be eliminated. In addition, any arc related to t is also eliminated from the graph.

Similarly to Dumas et al. (1991), we investigate all requests $r \in \mathcal{O}_{L_f}$. If there is at least one request r that cannot be delivered from η_{L_f} to its delivery node $r+n$ by satisfying its time window, then the forward label is eliminated. Similarly, if there is at least one request $r \in \mathcal{O}_{L_b}$ that cannot be picked up before η_{L_b} within its time window, then the backward label is removed.

In addition, for each replicated transfer node, label extension is reduced by breaking the symmetry for replicated transfer nodes. For example, consider a label that represents a replicated transfer node t. Let the other replicated transfer node t_1 represent the same physical transfer node as t (i.e., $\psi(t) = \psi(t_1)$) and let t_1 have the same functionality as t (origin or destination transfer node). Then, the extension from t to t_1 is valid only if $t_1 > t$. This is done to avoid symmetric sequences of nodes visited within the paths (e.g., [..., t, t_1, ...] and [..., t_1, t, ...], $\forall t$, $t_1 \in \mathcal{T}$, such that $\psi(t) = \psi(t_1)$ and t, t_1 have the same functionality, i.e., origin or destination transfer nodes).

Infeasible arcs can be eliminated from the graph by considering time windows. A simple analysis leads to the following observations:

- arcs $(o(v), i+n)$, $(i, o(v))$ and $(i+n, i)$ are infeasible for $i \in \mathcal{P}$, $v \in \mathcal{V}$;
- arc (i, j) with $i, j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T}$ is infeasible if $l_i + s_i + t_{ij} > u_j$.

More relaxed versions of the elimination rules proposed by Dumas et al. (1991) for PDPTW are also valid for the PDPTW-SL as follows:

- arcs (i, j) and $(j, i+n)$ are infeasible if $i \in \mathcal{P}$, $j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{T} \setminus \{i+n\}$ and path $\{i, j, i+n\}$ is infeasible.

4.2. Two-paths cuts

In this section we present the valid inequalities used to strengthen the LP relaxation of the master problem. We adapt the two-paths cuts for the PDPTW-SL, as these helped improve the lower bounds the most for the classical PDPTW (Røpke and Cordeau 2009). In particular, let set $\mathcal{H} \subseteq \mathcal{P} \cup \mathcal{D}$, be such that it cannot be served by one vehicle. The following inequality is then valid for the PDPTW-SL:

$$\sum_{i \in \mathcal{H}} \sum_{j \in \mathcal{N} \setminus \mathcal{H}} \sum_{p \in \Omega} a_{ij}^{p} x_p \geq 2. \tag{57}$$

To determine whether set $\mathcal{H}$ can be served by one vehicle, we solve a variant of the Travelling Salesman Problem (TSP) with time windows, capacity, and pickup and delivery. In this problem variant, the pickups and deliveries can also be performed at transfer nodes, and the tour must satisfy the capacity, time windows and precedence constraints by entering and leaving set $\mathcal{H}$ exactly once. Note that in the case of heterogeneous vehicles, the TSP must be solved for all vehicle types, considering depots and capacities.

Let $\delta^{+}(\mathcal{H}) \subseteq \mathcal{P} \cup \mathcal{T}$ be the corresponding nodes that must be visited before nodes in $\mathcal{H}$. In this tour, a request can be picked up from its pickup node or one of the corresponding transfer nodes. In addition, let $\delta^{-}(\mathcal{H}) \subseteq \mathcal{D} \cup \mathcal{T}$ be the corresponding nodes that need to be visited after visiting set $\mathcal{H}$. In other words, a request can be delivered to its delivery node or one of the corresponding transfer nodes. If a feasible path cannot be found, then set $\mathcal{H}$ defines a valid inequality (57). We separate the two-path cuts in a similar manner as Røpke and Cordeau (2009), using an adapted labeling algorithm.

4.3. Branching

The branch and bound tree is explored using the best bound strategy. The algorithm first branches on arcs (i, j). It imposes two branches: $\sum_{p \in \Omega} a_p^{ij} x_p \geq 1$ and $\sum_{p \in \Omega} a_p^{ij} x_p \leq 0$. Strong branching is used, i.e., the impact of branching on several candidates is investigated every time a branching decision has to be made. For each candidate, we estimate the lower bound in the two child nodes by solving the associated LP relaxation using a quick pricing heuristic, namely *H1*. The branch that maximizes the lower bound in the weakest of the two child nodes is chosen. We consider 10 branching candidates at every node. When all arcs have integer values (i.e., $\sum_{p \in \Omega} a_p^{ij} x_p = 1$ or $\sum_{p \in \Omega} a_p^{ij} x_p = 0$, $\forall\ (i, j) \in \mathcal{A}$), the following might happen: $\exists\ p \in \Omega \mid 0 < x_p < 1$ (i.e., some x_p has a fractional value), as the same route can be traversed by multiple vehicles (e.g., path p can be assigned to vehicles v_1 and v_2, respectively, both with value 0.5). In this case, the algorithm looks for an arc $(i, j) \in \mathcal{A}$ traversed by a vehicle v that has a fractional value and imposes two branches: $\sum_{p \in \Omega^v} a_p^{ij} x_p \geq 1$ and $\sum_{p \in \Omega^v} a_p^{ij} x_p \leq 0$. Finally, if $x_p \in \{0, 1\}$, $\forall\ p \in \Omega$ (i.e., all x_p variables

have integer values), then the MIP formulation (1)–(20) is solved for the considered routes (i.e., for x_p which have value 1) only. Let $\bar{\Omega}_x$ be the restricted set of routes, which may not necessarily be feasible in terms of time windows (by considering time synchronization between routes) and SL-capacity constraints. If the considered MIP model finds a feasible solution, then the solution is accepted. Otherwise, the following valid inequality is added to the formulation to disregard this solution:

$$\sum_{p \in \bar{\Omega}_x} x_p \leq |\bar{\Omega}_x| - 1. \tag{58}$$

The inequality is similar to a combinatorial Benders cut, where a solution that is not feasible is discarded from the search tree. Thus, at most $|\bar{\Omega}_x| - 1$ of the considered paths can be a part of the solution. In the path-based formulations, adding such inequalities to the model will make the considered paths regenerate in the following column generation iteration. Therefore, expression (58) should be reformulated in terms of arcs, consequently pricing out the considered arcs within the paths in the subproblem. Hence, the expression (58) can be reformulated to

$$\sum_{p \in \bar{\Omega}_x} \nu_p x_p \leq \sum_{p \in \bar{\Omega}_x} \nu_p - \mu, \tag{59}$$

where ν_p represents the number of arcs in path p and μ is the number of arcs within the path $p \in \bar{\Omega}_x$ that visits the fewest nodes.

Proposition 2 *Inequality (59) is equivalent to (58) in terms of integer feasible solutions.*

Proof of Proposition 2: Let x be an infeasible routing solution in terms of time windows or SL capacity. These routes are infeasible due to the inter-route synchronization constraints, time or load. In addition, let $\bar{\Omega}_x$ be the set of the infeasible combination of the considered routes (paths), where each PD vehicle is used at most once. Furthermore, let ν_p be the number of arcs within path $p \in \bar{\Omega}_x$ and $\mu = \arg\min_{p \in \bar{\Omega}_x} \{\nu_p\}$. Expression (58) assures that at least one path $p \in \bar{\Omega}_x$ will take value zero. This means that at least μ arcs from the considered paths $p \in \bar{\Omega}_x$ will take value zero, since the path that visits the fewest nodes involves μ arcs. Thus, (59) is equivalent to (58) in terms of integer feasible solutions. □

The dual information of inequalities (59) is considered in the pricing problem for each considered arc in $\bar{\Omega}_x$. Generally, this procedure is performed to avoid branching on q variables, which is time-consuming and leads to a slow convergence of the algorithm.

4.4. Potential Improvements

We note that two stabilization techniques have been tested in our implementation. These include the smoothing method proposed by Pessoa et al. (2013) and the box method introduced by Marsten

et al. (1975). The use of these techniques did not provide significant benefits to the overall performance of the algorithm, therefore no stabilization is used in the computational experiments presented in the following section.

Recently, Desaulniers et al. (2008) and Baldacci et al. (2011b) suggested to relax the elementarity conditions of the pricing problem to speed up the labeling algorithm for some type of VRPs. The elementarity constraints are then imposed in the master problem. Allowing cycles in the pricing problem may lead to an easier problem to solve, for which pseudo-polynomial algorithms are known (see for example Desrochers et al. (1992)). We have implemented and tested the ng-path relaxation of Baldacci et al. (2011b) but it did not show promising results. According to Røpke and Pisinger (2007), the shortest path problem with pairing and precedence constraints is still an NP-hard problem.

5. Computational Experiments

We have implemented the branch-and-price algorithm in Java. Since the subproblems defined on each PD vehicle are independent from each other, they are solved in parallel using four threads. All experiments were conducted on a 2.6 GHz Intel Core i5 processor with 4 GB of RAM. The LP relaxations of the restricted master problems were solved with CPLEX 12.3. We note that a limit of three hours has been imposed on the solution time. For the experiments we used generated datasets, which can be found on `www.smartlogisticslab.nl`. Instances with six to 12 requests, along with those that contain 25 and 50 requests have been used in Ghilas et al. (2016b,a). Additional instances (i.e., with 15, 20, 40 and 60 requests) were generated in the same way as in Ghilas et al. (2016b).

Three sets of instances, namely R, C, and RC, are used in this section. Each instance considers one to three available scheduled lines in a triangular topology and a departure frequency of 30 time units. The instances are classified with respect to the geographical locations of the nodes. For example, C involves clustered nodes around transfer nodes, R considers uniform-randomly distributed request nodes, and finally RC involve randomly clustered nodes. More specifically, C and RC have the nodes positioned within at most 30, respectively 80, time units to one of the three available transfer nodes. In addition, up to 60 requests (i.e., 60 pick-up and 60 delivery nodes) over 200×200 units on a Euclidean space are considered. Instances follow a naming convention of $G\_n\_sl\_v$, where G is the geographic distribution of the customers, n is the number of requests that need to be served, sl is the number of SLs, and v is the number of available vehicles. In all cases, two depots with eight heterogeneous PD vehicles are considered.

The considered planning horizon is 600 time units. The widths of the time windows are uniform-randomly generated between 30 and 90 time units. Service times are considered to be up to three time units. Each demand is assigned between one and three units. The capacity of each PD vehicle

is randomly chosen between six and 20 units. The carrying capacity on the considered SLs is assumed to be 15 demand units.

We assume the driving cost of the PD vehicles to be 0.5 per time unit, unless indicated. The cost of SL service per demand unit is equal to 1.

5.1. The Impact of Two-path Cuts

In this section, we test the performance of the algorithm with and without two-paths cuts. Table 1 presents the results obtained. The columns are self-explanatory.

Table 1 The Impact of Two-Paths Cuts.

	No Two-paths cuts				With Two-paths				
Instance	Root	$\underline{z}$	$\overline{z}$	Time (sec)	Root	Cuts	$\underline{z}$	$\overline{z}$	Time (sec)
C6_2_4	369.36	369.36	369.36	0	369.36	0	369.36	369.36	0
C7_2_4	390.31	390.31	390.31	0	390.31	0	390.31	390.31	0
C8_2_4	416.93	446.35	446.35	21	416.93	0	446.35	446.35	21
C9_2_4	441.20	471.16	471.16	117	441.20	3	471.16	471.16	146
C10_2_4	471.40	510.01	510.01	67	471.40	12	510.01	510.01	55
C11_2_4	498.14	522.85	522.85	9	498.14	10	522.85	522.85	27
C12_2_4	541.60	541.60	541.60	4	541.60	16	541.60	541.60	4
C15_2_6	649.93	649.93	649.93	13	649.93	6	649.93	649.93	21
C20_2_8	745.38	751.23	751.23	83	745.38	26	751.23	751.23	69
C25_2_8	874.93	879.94	879.94	2,114	878.90	55	879.94	879.94	1,215
C40_2_8	1,199.31	1,199.31	–	10,800	1,202.13	118	1,202.13	–	10,800
RC6_2_4	572.76	572.76	572.76	0	572.76	0	572.76	572.76	0
RC7_2_4	575.95	575.95	575.95	0	575.95	0	575.95	575.95	0
RC8_2_4	585.32	585.32	585.32	1	585.32	0	585.32	585.32	1
RC9_2_4	593.70	593.70	593.70	1	593.70	0	593.70	593.70	1
RC10_2_4	599.95	599.95	599.95	1	599.95	0	599.95	599.95	1
RC11_2_4	624.48	624.48	624.48	1	624.48	0	624.48	624.48	1
RC12_2_6	662.03	662.03	662.03	1	662.03	0	662.03	662.03	1
RC15_2_6	1,068.16	1,068.16	1068.16	1	1,068.16	0	1,068.16	1,068.16	1
RC20_2_8	1,200.36	1,200.36	1200.36	6	1,200.36	6	1,200.36	1,200.36	3
RC25_2_8	1,528.02	1,530.43	1530.43	28	1,528.02	18	1,530.43	1,530.43	33
RC40_2_12	1,955.79	1,969.45	1969.45	738	1,965.08	65	1,969.45	1,969.45	682
RC50_2_14	2,199.79	2,200.88	2200.88	1,235	2,200.88	60	2,200.88	2,200.88	921
RC60_2_16	2,611.51	2,636.18	–	10,800	2,625.76	102	2,636.22	–	10,800
R6_2_4	416.16	416.16	416.16	0	416.16	0	416.16	416.16	0
R7_2_4	458.28	473.06	473.06	1	459.33	3	473.06	473.06	1
R8_2_4	547.45	558.17	558.17	0	547.45	0	558.17	558.17	2
R9_2_4	607.48	632.42	632.42	2	607.48	8	632.42	632.42	1
R10_2_4	619.83	636.05	636.05	1	619.83	6	636.05	636.05	3
R11_2_4	746.03	748.29	748.29	1	746.03	5	748.29	748.29	4
R12_2_6	897.60	913.68	913.68	2	897.60	9	913.68	913.68	7
R15_2_6	1,058.97	1,083.50	1083.50	15	1,062.97	2	1,083.50	1,083.50	13
R20_2_8	1,533.21	1,542.93	1542.93	12	1,533.21	25	1,542.93	1,542.93	15
R25_2_8	1,613.81	1,636.12	1636.12	50	1,631.29	50	1,636.12	1,636.12	24
R40_2_12	1,948.56	1,969.77	1969.77	895	1,967.15	42	1,969.77	1,969.77	589
R50_2_14	2,592.15	2,595.14	2595.14	580	2,592.15	56	2,595.14	2,595.14	543
R60_2_16	2,854.99	2,858.69	–	10,800	2,855.24	52	2,871.13	–	10,800
Average	1,007.32	1,016.64		177	1,009.29		1,017.05		130

According to these results, two-paths cuts helped improve the overall performance of the algorithm. In particular, the root node lower bounds improved by up to 1.07% (i.e., R_25_2_8). For instances that could not be solved by the algorithm in either settings valid inequalities helped get better best bounds. Finally, in terms of CPU time, two-path cuts helped achieve better performance. We note that two-path cuts are used in the rest of the computational experiments described below.

5.2. Bi-directional vs. Mono-directional Labeling Algorithm

In this section, we study the benefits of using a bi-directional search over the mono-directional search. The detailed results can be found in Table 10 of the Appendix. The bi-directional labeling

algorithm performs better than the mono-directional version: the lower bounds obtained within the imposed time limit by the bi-directional labeling algorithm are better over all instances. Unlike the mono-directional algorithm, the bi-directional search found lower bounds for all instances within time limit. This is mainly due to the fact that the number of labels that need to be processed in the bi-directional labeling algorithm is considerably smaller compared with the mono-directional version. Lastly, bi-directional search proved to be approximately twice faster than mono-directional search, based on instances that were optimally solved by both algorithms.

5.3. The Impact of Acceleration Techniques

In this section, we investigate the effect of the preprocessing, label elimination techniques and the proposed valid inequality on the overall performance of the algorithm. More specifically, we define three algorithmic setups, as follows: *S1* – without preprocessing and label elimination, but with the cut proposed in Section 4.1.3, *S2* – with the preprocessing and label elimination, but without the valid inequality, and finally *All* – with all acceleration techniques described in this paper. Table 2 presents the obtained results within a three-hour time limit. Columns "$\underline{z}$" and "$\overline{z}$" indicate the best lower and upper bounds found within a three-hour time limit. "Nodes" indicate the number of branch-and-bound nodes explored during the search and, finally, "Time" indicates the computational times in seconds.

Table 2 The Impact of Acceleration Techniques.

Instance	S1				S2				All			
	$\underline{z}$	$\overline{z}$	Nodes	Time (sec)	$\underline{z}$	$\overline{z}$	Nodes	Time (sec)	$\underline{z}$	$\overline{z}$	Nodes	Time (sec)
C6_6_4	282.51	–	3,105	10,800	310.85	310.85	7,767	5,183	310.85	310.85	567	2,437
C7_6_4	263.92	–	1,804	10,800	313.37	–	7,542	10,800	313.41	313.41	951	4,890
C8_6_4	279.70	–	721	10,800	336.33	–	5,239	10,800	337.12	–	2,975	10,800
C9_6_4	300.02	–	15	10,800	355.56	–	2,025	10,800	356.36	–	3,531	10,800
C10_6_4	–	–	–	10,800	364.82	–	1,313	10,800	364.82	–	1,317	10,800
C11_6_4	–	–	–	10,800	380.08	–	935	10,800	380.08	–	947	10,800
RC6_6_4	520.11	520.11	1	2	520.11	520.11	1	1	520.11	520.11	1	0
RC7_6_4	482.55	–	2,804	10,800	539.08	539.08	3	1	539.08	539.08	1	0
RC8_6_4	482.94	–	1,432	10,800	548.45	548.45	3	1	548.45	548.45	1	1
RC9_6_4	488.64	–	226	10,800	552.71	552.71	3	3	552.71	552.71	1	1
RC10_6_4	558.96	558.96	3	1,181	558.96	558.96	3	2	558.96	558.96	1	1
RC11_6_4	585.72	–	1	10,800	585.72	585.72	7	11	585.72	585.72	1	2
R6_6_4	416.16	416.16	123	125	416.16	416.16	1	0	416.16	416.16	1	0
R7_6_4	452.74	–	2,123	10,800	470.15	470.15	21	12	470.15	470.15	12	7
R8_6_4	465.92	–	621	10,800	490.77	490.77	3	1	490.77	490.77	1	3
R9_6_4	510.59	–	62	10,800	579.38	579.38	7	5	579.38	579.38	1	1
R10_6_4	511.93	–	14	10,800	584.89	584.89	7	4	584.89	584.89	1	2
R11_6_4	–	–	–	10,800	687.36	687.36	3	5	687.36	687.36	1	2

The preprocessing and label elimination significantly improve the overall performance of the algorithm. The former tightens the formulation (i.e., better root lower bounds by on average 9.56%), and thus leads to fewer nodes to be processed in the search tree. In addition, it reduces the solution space of the problem, hence decreases the computational effort needed. The latter (i.e., label elimination) has a positive impact on the CPU time to process each B&B node. As can be seen in Table 2, *S1* could solve only three out of 18 instances, compared to *S2* and *All* setups,

with 13 and 14, respectively. Finally, the cut proposed in Section 4.1.3 helped close the gap for one instance (i.e., C7_6_4). In addition, the cut decreased the number of explored nodes in the B&B tree and improved the best lower bound found within time limit.

Table 3 presents the root node values for the formulation given in Ghilas et al. (2016b) and the solution approach presented in this paper (both exact approaches). Column "Root" reports the root node lower bound and "$Time_{root}$" presents the time needed to obtain this lower bound. In addition, column "GAP" presents the relative optimality gap between the obtained root lower bound and the optimal solution, which is given in column "$\overline{z}$". The best lower bound found within the imposed three-hour time limit is given in "$\underline{z}$". Finally, "# of columns" indicates the number of columns that were generated to compute the valid lower bound using B&P.

Table 3 Root Node Values.

	Ghilas et al. (2016b)						B&P						
Instance	Root	$Time_{root}$	GAP%	$\underline{z}$	$\overline{z}$	Time	Root	# of columns	$Time_{root}$	GAP %	$\underline{z}$	$\overline{z}$	Time
C6_6_4	238.08	1	23.41	267.91	310.85	10,800	303.34	160	1	2.42	310.85	310.85	2,437
C7_6_4	213.69	1	31.82	275.41	480.21	10,800	308.50	198	1	1.57	313.41	313.41	4,890
C8_6_4	231.40	3	–	290.15	670.24	10,800	333.05	194	1	–	337.12	–	10,800
C9_6_4	253.72	9	–	305.71	–	10,800	353.54	396	4	–	356.36	–	10,800
C10_6_4	262.07	15	–	307.33	–	10,800	362.80	667	12	–	364.82	–	10,800
C11_6_4	272.42	20	–	316.95	–	10,800	378.93	892	16	–	380.08	–	10,800
RC6_6_4	417.61	0	19.71	520.11	520.11	8	520.11	70	0	0.00	520.11	520.11	0
RC7_6_4	414.58	0	23.09	539.08	539.08	18	539.08	88	0	0.00	539.08	539.08	0
RC8_6_4	420.09	0	23.40	548.45	548.45	33	548.45	124	1	0.00	548.45	548.45	1
RC9_6_4	434.28	1	21.43	552.71	552.71	52	552.71	140	1	0.00	552.71	552.71	1
RC10_6_4	449.03	1	19.67	558.95	558.95	109	558.96	152	1	0.00	558.96	558.96	1
RC11_6_4	472.00	3	19.42	585.72	585.72	786	585.72	222	2	0.00	585.72	585.72	2
R6_6_4	363.40	0	12.68	416.16	416.16	3	416.16	116	0	0.00	416.16	416.16	0
R7_6_4	388.52	0	17.36	470.15	470.15	13	453.62	120	0	3.52	470.15	470.15	7
R8_6_4	427.69	0	12.85	490.77	490.77	63	490.77	180	3	0.00	490.77	490.77	3
R9_6_4	466.08	1	19.55	579.38	579.38	616	579.38	238	1	0.00	579.38	579.38	1
R10_6_4	463.54	1	20.75	584.89	584.89	4,998	584.89	282	2	0.00	584.89	584.89	2
R11_6_4	571.62	2	16.84	632.97	–	10,800	687.36	286	2	0.00	687.36	687.36	2
Average			20.06							0.51			

It can easily be noticed that the proposed set-partitioning formulation solved by B&P significantly outperforms the arc-based formulation proposed in Ghilas et al. (2016b) in terms of root lower bounds, hence in terms of relative optimality gap. The lower bounds obtained within a three-hour time limit are significantly stronger than the ones reported by Ghilas et al. (2016b). Finally, more instances with three SLs could be solved using B&P, within shorter computing times.

5.4. Lower Bound Values

Table 4 provides the results of the root nodes for the PDPTW-SL with homogeneous and heterogeneous routing costs (see Appendix for explanations). Column "Root" provides the lower bound at the root node and "# of columns" indicate the number of columns generated to solve the LP relaxation of the problem. "$Time_{root}$" shows the CPU time needed to solve the root node, and "GAP" presents the relative optimality gap of the lower bound from the known optimal solution. Finally, "$\overline{z}$" indicates the value of the known optimal solution.

Table 4 Lower Bounds.

Instance	Homogeneous					Heterogeneous				
	Root	# of columns	$Time_{root}$	GAP%	$\overline{z}$	Root	# of columns	$Time_{root}$	GAP%	$\overline{z}$
C6_2_4	369.36	92	0	0.00	369.36	385.49	83	0	0.00	385.49
C7_2_4	390.31	92	0	0.00	390.31	410.76	89	0	0.00	410.76
C8_2_4	416.93	162	1	6.59	446.35	437.67	142	0	6.54	468.30
C9_2_4	441.20	212	1	6.36	471.16	464.39	214	1	6.30	495.64
C10_2_4	471.40	312	1	7.57	510.01	494.39	204	1	7.59	535.01
C11_2_4	498.14	262	2	4.73	522.85	523.83	261	2	4.51	548.55
C12_2_4	541.60	416	4	0.00	541.60	568.30	343	4	0.00	568.30
C15_2_6	649.93	920	9	0.00	649.93	961.25	592	6	1.96	980.46
C20_2_8	745.38	1,232	8	0.78	751.23	886.49	888	13	0.40	890.02
C25_2_8	878.90	4,485	261	0.12	879.94	1,330.48	1,863	235	0.00	1,330.48
C40_2_8	1,202.13	6,524	9,660	–	–	1,441.80	5,379	10,217	–	–
RC6_2_4	572.76	54	0	0.00	572.76	593.52	54	0	0.00	593.52
RC7_2_4	575.95	72	0	0.00	575.95	596.89	74	0	0.00	596.89
RC8_2_4	585.32	88	0	0.00	585.32	606.78	85	0	0.00	606.78
RC9_2_4	593.70	96	1	0.00	593.70	615.50	91	1	0.00	615.50
RC10_2_4	599.95	100	1	0.00	599.95	621.75	94	1	0.00	621.75
RC11_2_4	624.48	104	1	0.00	624.48	646.28	97	1	0.00	646.28
RC12_2_6	662.03	194	1	0.00	662.03	693.75	178	1	0.00	693.75
RC15_2_6	1,068.16	178	1	0.00	1,068.16	1,586.84	179	1	0.00	1,586.84
RC20_2_8	1,200.36	532	3	0.00	1,200.36	1,639.68	562	7	0.00	1,639.68
RC25_2_8	1,528.02	1,140	9	0.16	1,530.43	2,395.89	697	6	0.33	2,403.87
RC40_2_12	1,965.08	2,659	175	0.22	1,969.45	3,033.48	2,381	185	3.57	3,145.84
RC50_2_14	2,200.88	4,782	921	0.00	2,200.88	3,432.43	2,897	1,652	0.29	3,442.28
RC60_2_16	2,625.76	7,044	1,341	–	–	3,977.42	3,267	545	–	–
R6_2_4	416.16	64	0	0.00	416.16	416.16	87	0	0.00	416.16
R7_2_4	459.33	82	0	2.90	473.06	458.28	119	0	3.12	473.06
R8_2_4	547.45	118	0	1.92	558.17	547.45	146	0	1.92	558.17
R9_2_4	607.48	136	0	3.94	632.42	607.48	226	0	3.94	632.42
R10_2_4	619.83	150	1	2.55	636.05	619.83	188	0	2.55	636.05
R11_2_4	746.03	150	1	0.30	748.29	746.03	173	1	0.30	748.29
R12_2_6	897.60	300	1	1.76	913.68	929.70	316	0	1.78	946.53
R15_2_6	1,062.97	300	2	1.89	1,083.50	1,345.53	407	1	2.29	1,377.04
R20_2_8	1,533.21	683	5	0.63	1,542.93	1,817.17	567	3	1.21	1,839.34
R25_2_8	1,631.29	908	10	0.30	1,636.12	1,992.37	647	2	0.97	2,011.86
R40_2_12	1,967.15	3,733	151	0.13	1,969.77	2,511.39	2,206	137	0.26	2,518.05
R50_2_14	2,592.15	4,276	358	0.12	2,595.14	3,168.85	2,131	445	0.42	3,182.19
R60_2_16	2,855.24	4,672	2,034	–	–	3,465.54	3,767	3,465	–	–
Average		855		1.26			567		1.48	

Regarding the lower bounds obtained, on average they are 1.51% away from the optimal solutions. For instances with homogeneous routing costs, the lower bounds are slightly tighter than for the instances with heterogeneous costs. Also, according to the CPU times, instances with heterogeneous costs were more time-consuming to solve. These facts indicate that having heterogeneous costs adds extra complexity to the problem. On average, solving the root node took the same amount of time in both cases.

5.5. PDPTW-SL vs. PDPTW

This section discusses the computational results obtained when solving the PDPTW-SL (with one SL) and PDPTW instances using the proposed algorithm. A similar comparison was made by Ghilas et al. (2016a,b) but the results were optimal only for small instances (up to 11 requests) and heuristic for larger instances. In this section, we confirm that the previously found conclusions still hold for the optimal solutions of larger instances. The detailed results are shown in Tables 11 and 12 of the Appendix.

According to the results, PDPTW-SL solutions proved to be cheaper compared to the PDPTW solutions. However, the cost savings are dependent on the relative positioning of the request nodes and SLs, time windows and available capacities. More specifically, C instances provided the most savings, and R instances the least. Overall, the results support the findings of Ghilas et al. (2016a,b):

the use of SLs benefits the pickup and delivery network. In order to maximize the benefits of such systems, tactical decisions, such as configuration of the SLs, storage areas at transfer nodes and re-design of the SL vehicles, must be properly considered.

5.6. PDPTW-SL with Multiple Scheduled Lines

This section illustrates the effect of having multiple available SLs on the operational costs of the system. Table 13 of the Appendix indicates the results obtained when solving the instances with one, two (i.e., two connected lines) and three (i.e., triangular network) available SLs within a three-hour time limit. The routing costs are assumed to be homogeneous. As expected, the complexity drastically increases with the number of available SLs as the increase in graph size makes the pricing problem more difficult to solve. As a side effect, more SLs lead to significantly slower convergence during the search. The results support the observations made by Ghilas et al. (2016a,b) for similar settings: more SLs lead to more cost savings compared to PDPTW solutions.

5.7. Comparison to Other Algorithms

In this section, we present a computational comparison of the proposed B&P algorithm to the methodologies used by Ghilas et al. (2016a,b): the arc-based MIP solved by CPLEX with the valid inequalities given in Ghilas et al. (2016b) along with the preprocessing introduced in Section 4.1.3 of this paper, and the proposed ALNS (the best out of ten runs). Table 5 indicates the results obtained by the different methodologies. The GAP column indicates the relative optimality gap between the best lower bound found and the optimal solution. The other columns are self-explanatory.

The results indicate that the proposed B&P performs significantly better than CPLEX. The lower bounds obtained by CPLEX within the time limit proved to be relatively weak, i.e., on average 7.58% from the optimal solution. The proposed B&P algorithm solved 34 out of 37 instances, and CPLEX could solve only 22 of them. In addition, B&P solved significantly faster the instances that were solved by CPLEX as well.

The ALNS performs relatively well, compared to the proposed B&P. More specifically, the ALNS found the optimal solutions for 27 out of 37 instances. The optimality gap did not exceed 2.3% for the 10 instances for which the ALNS found suboptimal solutions. As expected, the heuristic is significantly faster than B&P.

5.8. Sensitivity Analyses

In this section, we present some computational results to shed some light on how the algorithm's performance changes with some instance parameters, such as the width of the time windows, number of PD vehicles, and PD vehicle capacity.

Table 6 presents the results considering modified time windows. Instances $C15\_2\_6$, $C20\_2\_8$, $RC15\_2\_6$, $RC25\_2\_8$, $R15\_2\_6$, $R20\_2\_8$ are tested. All these instances could be solved optimally

Table 5 Comparison between Algorithms.

Instance	B&P			CPLEX				ALNS		
	$\underline{z}$	$\overline{z}$	Time (sec)	$\underline{z}$	$\overline{z}$	GAP%	Time (sec)	$\overline{z}$	Time (sec)	Gap (%)
C6_2_4	369.36	369.36	0	369.36	369.36	0.00	68	369.36	1	0.00
C7_2_4	390.31	390.31	0	390.31	390.31	0.00	160	390.31	1	0.00
C8_2_4	446.35	446.35	3	446.35	446.35	0.00	365	446.35	2	0.00
C9_2_4	471.16	471.16	178	471.16	471.16	0.00	6,351	472.68	2	0.32
C10_2_4	510.01	510.01	73	501.25	536.50	1.72	10,800	510.01	2	0.00
C11_2_4	522.85	522.85	37	453.44	528.08	13.27	10,800	522.84	2	0.00
C12_2_4	541.60	541.60	4	470.04	–	13.21	10,800	541.60	2	0.00
C15_2_6	649.93	649.93	32	546.29	–	19.95	10,800	659.30	3	1.42
C20_2_8	751.23	751.23	79	544.95	–	27.46	10,800	751.23	4	0.00
C25_2_8	879.94	879.94	2,627	–	–	–	10,800	879.94	8	0.00
C40_2_8	1,202.13	–	10,800	–	–	–	10,800	1,226.77	8	–
RC6_2_4	572.76	572.76	0	572.76	572.76	0.00	3	572.75	1	0.00
RC7_2_4	575.95	575.95	0	575.95	575.95	0.00	3	575.95	1	0.00
RC8_2_4	585.32	585.32	0	585.32	585.32	0.00	5	585.32	1	0.00
RC9_2_4	593.70	593.70	1	593.70	593.70	0.00	5	593.69	2	0.00
RC10_2_4	599.95	599.95	1	599.95	599.95	0.00	8	599.94	2	0.00
RC11_2_4	624.48	624.48	1	624.48	624.48	0.00	8	624.47	2	0.00
RC12_2_6	662.03	662.03	1	662.03	662.03	0.00	11	662.03	2	0.00
RC15_2_6	1,068.16	1,068.16	1	1,068.16	1,068.16	0.00	65	1,068.16	1	0.00
RC20_2_8	1,200.36	1,200.36	3	1,062.37	–	11.50	10,800	1,203.98	2	0.30
RC25_2_8	1,530.43	1,530.43	40	1,085.66	–	29.06	10,800	1,530.43	2	0.00
RC40_2_12	1,969.45	1,969.45	1,063	–	–	–	10,800	1,988.36	6	0.95
RC50_2_14	2,200.88	2,200.88	1,199	–	–	–	10,800	2,214.09	10	0.60
RC60_2_16	2,636.22	–	10,800	–	–	–	10,800	2,646.34	13	–
R6_2_4	416.16	416.16	0	416.16	416.16	0.00	4	416.16	1	0.00
R7_2_4	473.06	473.06	1	473.06	473.06	0.00	4	473.05	1	0.00
R8_2_4	558.17	558.17	2	558.17	558.17	0.00	4	558.17	1	0.00
R9_2_4	632.42	632.42	6	632.42	632.42	0.00	39	632.41	1	0.00
R10_2_4	636.05	636.05	3	636.05	636.05	0.00	24	636.05	2	0.00
R11_2_4	748.29	748.29	3	748.29	748.29	0.00	39	748.29	2	0.00
R12_2_6	913.68	913.68	6	913.68	913.68	0.00	182	934.73	2	2.30
R15_2_6	1,083.50	1,083.50	14	1,083.50	1,083.50	0.00	317	1,083.50	1	0.00
R20_2_8	1,542.93	1,542.93	19	1,280.74	–	16.99	10,800	1,542.93	1	0.00
R25_2_8	1,636.12	1,636.12	59	1,326.77	–	18.91	10,800	1,638.42	1	0.14
R40_2_12	1,969.77	1,969.77	606	–	–	–	10,800	1,969.77	5	0.00
R50_2_14	2,595.14	2,595.14	1,634	–	–	–	10,800	2,595.14	7	0.00
R60_2_16	2,871,13	–	10,800	–	–	–	10,800	2,890.34	12	–
Average			226				4,673		2	0.18

within a three-hour time limit and all had at least one feasible solution in all considered settings. In particular, each request node is assigned a time window $[l_i^1, u_i^1]$, such that $(u_i^1 - l_i^1) = (u_i - l_i)\rho$ and $|u_i^1 - u_i| = |l_i^1 - l_i|$, where $\rho \in \{0.5, 1, 1.5, 2\}$. Most columns are self-explanatory. Note that column "# via SLs" indicates the number of demand units that were shipped using available SLs.

The time window width plays an important role in the algorithm's performance. Generally, the wider the time windows are, the more difficult it is to solve the root node (i.e., "$Time_{root}$" column). More specifically, the pricing problem becomes more difficult as more labels need to be processed, thus increasing the time needed to solve each B&B node. However, no relationship between the root node optimality gap and the width of the time windows can be observed. In most cases, tighter time windows lead to stronger lower bounds. In some exceptional cases with $\rho = 2.0$ (i.e., R15_2_6) the problem is solved in the root node, thus leading to a shorter overall CPU time than in cases with smaller values for ρ. Finally, as expected, wider time windows lead to lower operating costs compared to cases with tight time windows.

Table 7 shows the impact of the PD vehicle capacities on the performance of the algorithm. Let μ be a factor by which the PD vehicle capacities are modified i.e., $\mu \times Q_v \ \forall \ v \in \mathcal{V}$. We investigate four settings: $\mu \in \{0.3, 0.7, 1.0, 1.3, 1.7\}$. To assure feasibility in all settings, time windows are assumed to be twice wider than in the original case (i.e., $\rho = 2.0$). Only instances $C15\_2\_6$, $C20\_2\_8$,

Table 6 Impact of Time-window width.

Instance	Root	# of columns	$Time_{root}$ (sec)	Cuts	$\underline{z}$	$\overline{z}$	Nodes	Time (sec)	# via SLs	# of vehicles	Gap%
					ρ=0.5						
C15_2_6	735.03	572	3	2	736.08	736.08	9	11	8	5	0.14
C20_2_8	848.52	958	3	21	848.52	848.52	1	3	9	7	0.00
RC15_2_6	1,078.91	140	1	0	1,078.91	1,078.91	1	1	0	6	0.00
RC25_2_8	1,620.10	629	4	32	1,642.97	1,642.97	9	18	4	8	1.39
R15_2_6	1,130.87	237	1	0	1,130.87	1,130.87	1	1	0	6	0.00
R20_2_8	1,573.60	540	1	8	1,573.60	1,573.60	1	1	0	8	0.00
					ρ=1.0						
C15_2_6	649.93	920	9	6	649.93	649.93	3	21	8	4	0.00
C20_2_8	745.38	1,232	8	26	751.23	751.23	5	69	2	5	0.78
RC15_2_6	1,068.16	178	1	0	1,068.16	1,068.16	1	1	0	6	0.00
RC25_2_8	1,528.02	1,140	9	18	1,530.43	1,530.43	3	33	6	8	0.16
R15_2_6	1,062.97	300	2	2	1,083.50	1,083.50	5	13	0	6	1.89
R20_2_8	1,533.21	683	5	25	1,542.93	1,542.93	3	15	0	8	0.63
					ρ=1.5						
C15_2_6	639.81	933	9	19	649.93	649.93	91	315	8	4	1.56
C20_2_8	728.60	1,262	11	30	747.46	747.46	41	302	4	5	2.52
RC15_2_6	1,068.16	181	1	0	1,068.16	1,068.16	1	1	0	6	0.00
RC25_2_8	1,473.40	1,242	13	8	1,473.40	1,473.40	1	13	4	7	0.00
R15_2_6	1,044.94	351	2	4	1,058.99	1,058.99	11	14	2	5	1.33
R20_2_8	1,519.42	698	5	25	1,526.97	1,526.97	5	16	1	8	0.49
					ρ=2.0						
C15_2_6	628.73	1,274	23	24	649.12	649.12	49	318	8	4	3.14
C20_2_8	712.89	1,310	30	54	713.74	713.74	3	73	4	5	0.12
RC15_2_6	1,068.16	187	1	0	1,068.16	1,068.16	1	1	0	6	0.00
RC25_2_8	1,415.60	1,429	22	15	1,415.60	1,415.60	23	78	4	7	0.00
R15_2_6	993.80	359	2	8	993.80	993.80	1	2	0	5	0.00
R20_2_8	1,511.98	698	5	6	1,526.97	1,526.97	7	17	1	8	0.98

RC20_2_8, and R12_2_6 have at least one feasible solution in all considered setups. Therefore, only their corresponding results are presented in Table 7.

Surprisingly, smaller capacities do not necessarily lead to easier instances. For more constrained instances the root nodes are solved faster, by generating fewer columns. However, the resulting optimality gaps may not necessary be tighter than in less constrained instances. This may lead to longer overall CPU times. Finally, tighter vehicle capacities lead to higher operating costs in optimal solutions, compared to instances with larger capacities.

Table 8 presents the effect of the number of available vehicles on the algorithm's performance. In particular, it indicates the results obtained by solving the following instances: C15_2_6, C20_2_8, RC20_2_8, R12_2_6. Not all instances were feasible in all the considered setups. To make more instances feasible, we consider twice wider time windows (i.e., $\rho = 2.0$) than in original cases (i.e., $\rho = 1$). The order in which the vehicles are removed is by the corresponding ID (largest ID first).

Intuitively the fewer vehicles are available, the tighter the problem tends to be, and hence easier to solve. However, it is not always the case, as removing certain (heterogeneous) vehicles may lead to different problem structures that may result in more difficult problems. For example, C20_2_4

Table 7 Impact of PD Vehicles Capacities.

Instance setting	Root	# of columns	$Time_{root}$ (sec)	Cuts	$\underline{z}$	$\overline{z}$	Nodes	Time (sec)	# via SLs	# of vehicles	Gap%
$\rho = 2$, $\mu = 0.3$											
C15_2_6	718.83	654	3	11	742.04	742.04	85	115	11	5	3.13
C20_2_8	866.08	1,029	6	20	872.75	872.75	29	85	9	6	0.76
RC20_2_8	1,551.85	665	2	7	1,563.37	1,563.37	631	1,244	8	8	0.74
R12_2_6	866.23	282	1	0	866.23	866.23	1	1	0	4	0.00
$\rho = 2.0$, $\mu = 0.7$											
C15_2_6	628.73	1,031	13	30	649.12	649.12	117	501	8	4	3.14
C20_2_8	713.12	1,121	16	30	713.74	713.74	3	37	4	5	0.09
RC20_2_8	1,142.33	684	2	0	1,142.33	1,142.33	1	2	4	6	0.00
R12_2_6	866.23	282	1	0	866.23	866.23	1	1	0	4	0.00
$\rho = 2.0$, $\mu = 1.0$											
C15_2_6	628.73	1,274	23	24	649.12	649.12	49	318	8	4	3.14
C20_2_8	712.89	1,310	30	54	713.74	713.74	3	73	4	5	0.12
RC20_2_8	1,142.33	747	4	0	1,142.33	1,142.33	1	4	4	6	0.00
R12_2_6	866.23	282	1	0	866.23	866.23	1	1	0	4	0.00
$\rho = 2.0$, $\mu = 1.3$											
C15_2_6	628.73	1,369	38	33	649.12	649.12	37	305	8	4	3.14
C20_2_8	712.89	1,722	47	50	713.74	713.74	3	96	4	5	0.12
RC20_2_8	1,142.33	1,102	4	0	1,142.33	1,142.33	1	4	4	6	0.00
R12_2_6	866.23	282	1	0	866.23	866.23	1	1	0	4	0.00
$\rho = 2.0$, $\mu = 1.7$											
C15_2_6	628.73	1,373	39	33	649.12	649.12	95	610	8	4	3.14
C20_2_8	712.89	1,798	48	53	713.74	713.74	3	120	4	5	0.12
RC20_2_8	1,142.33	1,120	4	0	1,142.33	1,142.33	1	4	4	6	0.00
R12_2_6	866.23	282	1	0	866.23	866.23	1	1	0	4	0.00

Table 8 Impact of Number of Available PD Vehicles.

Instance	Root	# of columns	$Time_{root}$ (sec)	Cuts	$\underline{z}$	$\overline{z}$	Nodes	Time (sec)	# via SLs	# of vehicles	Gap%
Original case											
C15_2_6	628.73	1,274	23	24	649.12	649.12	49	318	8	4	3.14
C20_2_8	712.89	1,310	30	54	713.74	713.74	3	73	4	5	0.12
RC20_2_8	1,142.33	747	4	0	1,142.33	1,142.33	1	4	4	6	0.00
R12_2_6	866.23	282	1	0	866.23	866.23	1	1	0	4	0.00
Without one vehicle											
C15_2_5	628.73	869	11	17	649.12	649.12	83	484	8	4	3.14
C20_2_7	712.89	1,066	16	13	713.74	713.74	3	75	4	5	0.12
RC20_2_7	1,142.33	725	4	0	1,142.33	1,142.33	1	4	4	6	0.00
R12_2_5	871.58	235	1	0	871.58	871.58	1	1	0	4	0.00
Without two vehicles											
C15_2_4	631.23	960	48	27	649.12	649.12	59	433	8	4	2.76
C20_2_6	712.99	1,313	28	27	713.74	713.74	3	57	4	5	0.11
RC20_2_6	1,142.33	679	4	0	1,142.33	1,142.33	1	4	4	6	0.00
R12_2_4	871.58	182	1	0	871.58	871.58	1	1	0	4	0.00
Without three vehicles											
C15_2_3	651.89	787	18	31	652.59	652.59	25	186	5	3	0.11
C20_2_5	712.99	996	25	55	713.74	713.74	7	134	4	5	0.11
Without four vehicles											
C20_2_4	724.89	944	35	57	748.73	748.73	25	472	2	4	3.18

can be generated by removing four vehicles from the original instance ($C20\_2\_8$) according to the given vehicle order in the dataset. In the instances with fewer vehicles, the root lower bound and the optimal solution values are at least as high as in the corresponding instances with more vehicles. Note that fewer vehicles do not always lead to smaller computing times.

5.9. PDPTW with Transfers

As a by-product, we can simplify the proposed set partitioning master problem and solve the PDPTW-T. In this context, transfers among PD vehicles can be made only at pre-defined locations. For the formal description of the problem the reader is referred to Cortes et al. (2010). From the modeling perspective, each transfer node can be considered as a SL with zero traveling time (i.e., the same physical location) and with no imposed schedule and capacity constraints. In other words, PDPTW-T is a special case of PDPTW-SL, where multiple individual SLs of line network topology can exist. However, certain service times at these nodes exist (i.e., loading/unloading operations).

Unfortunately, the instances previously used in the literature are not available. In addition, the authors of the related articles used random elements in their instance generators and thus, we could not re-create the same instances in order to compare the performances of the algorithms. Therefore, we have generated a new set of instances with up to 50 requests, two depots and two transfer nodes (available at `www.smartlogisticslab.nl`). The request nodes were randomly generated on a 200×200 Euclidean space. Transfer nodes were positioned in the center of the considered area. Time windows, service times and PD-vehicle capacities were randomly generated. Instances follow a naming convention of *pdptwt_n_m_v*, where n is the number of requests, m is the number of transfer nodes, and v is the number of vehicles.

In order to remove the SL scheduling part from the model, variables q_{ij}^{rw} are simplified to q_{ij}^{r}, binary variables taking value 1 if request r is transferred at node i, and 0 otherwise $\forall r \in \mathcal{P}$, ($i$, j) $\in \mathcal{E}$. The objective function of the PDPTW-T represents only the first term of expression (1), i.e., minimize routing costs. The constraints of the restricted master problem remain the same as in formulation (2)–(20), without constraints (6) and (13). The results are reported in Table 9.

The results indicate that PDPTW-T instances with up to 40 requests can be solved to optimality. The lower bounds proved to be relatively strong (i.e., 0.37% for both problems). More specifically, as expected the root node lower bounds for the PDPTW-T (i.e., 0.64%) are slightly worse than the ones for the PDPTW (i.e., 0.37%). The reason is that the PDPTW-T is more complicated than the PDPTW, due to the transfer opportunity. Overall, similarly to the PDPTW-SL, transfer opportunities lead to cost savings, compared to the PDPTW solutions. More specifically, average savings of 5% can be achieved with one or two available transfer nodes. In addition, a slight decrease in the number of vehicles used is observed.

Table 9 Comparison between PDPTW-T and PDPTW.

Instance	PDPTW-T							PDPTW				
	Root	$\underline{z}$	$\overline{z}$	# via SLs	# of vehicles	Time (sec)	Savings (%)	Root	$\underline{z}$	$\overline{z}$	# of vehicles	Time (sec)
20_2_8	1,237.84	1,256.04	1,256.04	10	7	362	2.84	1,292.80	1,292.80	1,292.80	7	3
20_2_10	1,326.29	1,326.29	1,326.29	7	7	22	11.77	1,503.16	1,503.16	1,503.16	8	5
20_2_10.1	1,477.37	1,477.37	1,477.37	1	9	17	2.27	1,511.71	1,511.71	1,511.71	9	3
25_2_10	1,623.77	1,623.77	1,623.77	6	9	398	7.93	1,763.54	1,763.54	1,763.54	9	4
25_2_8	1,440.71	1,463.86	1,463.86	3	8	659	6.05	1,532.16	1,558.08	1,558.08	8	48
25_2_10.1	1,373.89	1,409.90	1,409.90	14	8	6,127	5.31	1,476.11	1,489.00	1,489.00	8	34
30_1_12	1,829.70	1,829.70	1,829.70	1	9	77	2.63	1,864.49	1,879.07	1,879.07	9	15
30_2_10	1,830.57	1,831.37	1,831.37	8	9	381	4.84	1,924.51	1,924.51	1,924.51	9	9
30_1_12.1	1,838.15	1,838.15	1,838.15	10	10	77	4.06	1,915.98	1,915.98	1,915.98	10	6
35_1_10	1,812.72	1,838.17	1,838.17	7	9	8,146	5.57	1,946.68	1,946.68	1,946.68	10	9
35_1_12	2,084.32	2,084.32	2,084.32	6	10	138	3.44	2,158.53	2,158.53	2,158.53	11	4
35_1_12.1	1,979.09	1,986.74	1,986.74	3	9	3,104	2.21	2,015.85	2,031.65	2,031.65	10	44
40_1_12	2,118.97	2,120.48	2,120.48	12	11	1,367	6.30	2,263.00	2,263.00	2,263.00	11	11
40_1_12.1	2,139.45	2,146.15	2,146.15	4	11	291	5.54	2,271.95	2,271.95	2,271.95	12	7
40_1_12.2	2,124.26	2,135.27	–	–	–	10,800	–	2,195.56	2,195.56	2,195.56	11	9
50_1_14	2,428.42	2,445.66	–	–	–	10,800	–	2,501.05	2,501.05	2,501.05	12	33
50_1_16	2,415.18	2,439.23	–	–	–	10,800	–	2,469.07	2,469.07	2,469.07	12	18
50_1_14.1	2,554.16	2,559.78	–	–	–	10,800	–	2,588.02	2,590.23	2,590.23	13	155
Average	1,722.35	1,730.88	1,730.88	6.57	9.00	1,512	5.05	1,817.18	1,822.12	1,822.12	9.36	14

6. Conclusions

We have proposed a B&P algorithm to solve the PDPTW-SL. To fully evaluate the effectiveness of the algorithm, we have used a library of PDPTW-SL instances. We have shown that small to medium-size instances, with up to 50 freight requests, can be optimally solved by the proposed algorithm. Moreover, average savings of 8% can be achieved by using available SLs to perform the deliveries, compared to the corresponding PDPTW solutions. Additionally, a special case of the problem, the PDPTW-T, is solved using the proposed algorithm. Instances with up to 40 requests provided average operational savings of 5% compared to the PDPTW solutions.

In addition to the computational complexity of the problem discussed in this paper, there are practical challenges as well that need to be overcome. First, transfer points require appropriate infrastructure for storing securely the packages until their departure (e.g., DHL-Packstation (2015) lockers). Additionally, handling the transfer operations (i.e., loading/unloading packages) can be a significant cost element of using SLs, which needs to be carefully accounted for. In the PDPTW-SL it is assumed that all resources needed to integrate freight and public transportation are available. However, many of them may have setup costs. Hence, before implementing such a system, strategic/tactical level decisions need to be investigated. In particular, the trade-offs between the number of SLs, the number of re-designed SL vehicles, SL capacity, and SL frequency have to be considered.

Acknowledgments

The authors gratefully acknowledge funds provided by Dinalog, the Dutch Institute for Advanced Logistics, under the grant titled “Cargo Hitching”, number 2011 4 086R and by the Canadian Natural Sciences and Engineering Research Council under grant 04959-2014. We also thank the Associate Editor and the two anonymous reviewers for their valuable comments which have helped improve the paper.

References

Aldaihani, M, M Dessouky. 2003. Hybrid scheduling methods for paratransit operations. *Computers & Industrial Engineering* **45**(1) 75–96.

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

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

Cargo Tram. 2012. Official webpage (accessed on March 24, 2014). Available at: www.eltis.org/index.php?id=13&study_id=1547.

Cortes, E-C, M Matamala, C Contardo. 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research* **200**(3) 711–724.

Dabia, S, E Demir, T Van Woensel. 2016. An exact approach for a variant of the pollution-routing problem. *Transportation Science* **51**(2) 607–628.

Demir, E, T Bektaş, G Laporte. 2014. A review of recent research on green road freight transportation. *European Journal of Operational Research* **237**(3) 775–793.

Demir, E, Y Huang, S Scholts, T Van Woensel. 2015. A selected review on the negative externalities of the freight transportation: Modeling and pricing. *Transportation Research Part E: Logistics and Transportation Review* **77** 95–114.

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

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

DHL-Packstation. 2015. DHL official webpage (accessed on November 29, 2015). Available at: www.dhl.de/en/paket/pakete-empfangen/packstation.html.

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

Ghilas, V, E Demir, T Van Woensel. 2016a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research* **72** 12–30.

Ghilas, V, E Demir, T Van Woensel. 2016b. The pickup and delivery problem with time windows and scheduled lines. *INFOR: Information Systems and Operational Research* **54**(2) 147–167.

Ghilas, V, E Demir, T Van Woensel. 2016c. A scenario-based planning for the pickup and delivery problem with time windows, scheduled lines and stochastic demands. *Transportation Research part B: Methodological* **91** 34–51.

Häll, CH, H Andersson, JT Lundgren, P Varbrand. 2009. The integrated dial-a-ride problem. *Public Transportation* **1**(1) 39–54.

Irnich, S, G Desaulniers. 2005. Shortest path problems with resource constraints. G Desaulniers, J Desrosiers, M M. Solomon, eds., *Column Generation*. Springer US, 33–65.

Liaw, C-F, C White, J Bander. 1996. A decision support system for the bimodal dial-a-ride problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* **26**(5) 552–565.

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

Marsten, R, W Hogan, J Blankenship. 1975. The boxstep method for large-scale optimization. *Operations Research* **23**(3) 389–405.

Masson, R, F Lehuede, O Peton. 2012. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science* **47**(3) 1–12.

Masson, R, F Lehuede, O Peton. 2014. The dial-a-ride problem with transfers. *Computers & Operations Research* **41** 12–23.

Masson, R, A Trentini, F Lehuede, N Malhene, O Peton, H Tlahig. 2015. Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics* 1–29.

Mues, C, S Pickl. 2005. Transshipment and time windows in vehicle routing. *8th International Symposium Parallel Architectures, Algorithms and Networks, Piscataway, NJ. Proceedings*. 113–119.

Nanry, W, W Barnes. 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological* **34**(2) 107–121.

Pessoa, A Alves, R Sadykov, E Uchoa, F Vanderbeck. 2013. In-out separation and column generation stabilization by dual price smoothing. *Experimental Algorithms, 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings*, *Lecture Notes in Computer Science*, vol. 7933. Springer, 354–365.

Rais, A, F Alvelos, M S Carvalho. 2013. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research* **235**(3) 530–539.

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

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

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

Røpke, S, D Pisinger. 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* **40**(4) 455–472.

Røpke, S, D Pisinger. 2007. Complexity of shortest path problems arising in column generation for vehicle routing problems with pairing and precedence constraints. Tech. rep., DTU: Technical University of Denmark.

Savelsbergh, M, M Sol. 1995. The general pickup and delivery problem. *Transportation Science* **29**(1) 17–29.

Sigurd, M, D Pisinger, M Sig. 2004. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science* **38**(2) 197–209.

Trentini, A, N Malhene. 2010. Toward a shared urban transport system ensuring passengers & goods cohabitation. *Trimestrale del Laboratorio Territorio Mobilita e Ambiente - TeMALab* **4** 37–44.

Trentini, A, R Masson, F Lehuede, N Malhene, O Peton, H Tlahig. 2012. A shared "passenger & goods" city logistics system. *4th International Conference on Information Systems, Logistics and Supply Chain, Quebec, Canada*.

Xu, H, Z-L Chen, S Rajagopal, S Arunapuram. 2003. Solving a practical pickup and delivery problem. *Transportation Science* **37**(3) 347–364.

Appendix.

Details on the Backward Labeling Algorithm

In the backward labeling algorithm, labels are extended from the sink (i.e., $o(v')$) to its predecessors. It is similar to the forward labeling algorithm. To a backward label L_b, we associate the following data:

- η last node of the path;
- t departure time from the node;
- q total load after visiting the last node;
- c accumulated cost;
- $\mathcal{O}$ set of requests that have been finished, but not started (i.e., the request has been delivered either to the delivery node or to a transfer node, but not picked up from its pickup or transfer node);
- $\mathcal{C}$ set of completed requests;
- $\mathcal{O}^T$ set of finished requests at a transfer node, such that $\mathcal{O}^T \subseteq \mathcal{O}$.

Similarly to forward labeling, the resources are t, q, c, $\mathcal{O}$, $\mathcal{O}^T$, and $\mathcal{C}$. These are initialized as follows: $t = u_{o(v)}$, $q = Q_v$, $c = 0$, and $\mathcal{O} \equiv \mathcal{O}^T \equiv \mathcal{C} \equiv \emptyset$.

When extending a backward label L_b along an arc (j, η_{L_b}), the extension is feasible only if:

$$t_{L_b} - s_{\eta_{L_b}} - t_{j,\eta_{L_b}} \geq l_j + s_j \tag{60}$$

$$q_{L_b} + d_j \geq 0. \tag{61}$$

In this case, we ensure the capacity constraints. Similarly to forward labeling, time windows are used to eliminate infeasible extensions. Moreover, L_b and j must also satisfy one of the following five conditions, which ensure that the extension is compatible with the sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$:

$$j \in \mathcal{D} \wedge j - n \notin \mathcal{O}_{L_b} \wedge j - n \notin \mathcal{C}_{L_b} \tag{62}$$

$$j \in \mathcal{P} \wedge j \in \mathcal{O}_{L_b} \tag{63}$$

$$j \in \mathcal{T} \wedge r(j) \notin \mathcal{O}_{L_b} \wedge r(j) \notin \mathcal{C}_{L_b} \tag{64}$$

$$j \in \mathcal{T} \wedge r(j) \in \mathcal{O}_{L_b} \setminus \mathcal{O}^T_{L_b} \wedge r(j) \notin \mathcal{C}_{L_b} \tag{65}$$

$$j = o(v) \wedge \mathcal{O}_{L_b} \in \emptyset. \tag{66}$$

Condition (62) states that visiting a delivery node $j \in \mathcal{D}$ is feasible if the corresponding request $j-n$ has not been delivered at the delivery node, nor at a transfer node earlier in the partial path, and this specific request is not completed. Expression (63) implies that extending the label to a pickup node $j \in \mathcal{P}$ is feasible if j has already been delivered to its delivery node, or to a transfer node in the partial path. Condition (64) states that visiting a transfer node $j \in \mathcal{T}$ is feasible if the corresponding request $r(j) \in \mathcal{P}$ has not been delivered to its delivery node, or to a transfer node in the partial path, and if this request has not been completed. Expression (65) states that extending to a transfer node $j \in \mathcal{T}$ is feasible if the corresponding request $r(j) \in \mathcal{P}$ has been delivered to its delivery node, and not to a transfer node, and if this request has not been completed in the partial path. Finally, condition (66) implies that a partial path cannot be ended at the origin depot (i.e., $o(v)$), if at least one request is delivered, but not picked up (from the origin, or from a transfer node) in the partial path.

When label η_{L_b} and node j satisfy the conditions (60) and (61), along with one of the following conditions, the corresponding updates on sets $\mathcal{O}$, $\mathcal{O}^T$ and $\mathcal{C}$ are performed as follows:

- expression (62): $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \cup \{j-n\}$;
- expression (63): $\mathcal{C}_{L_b} \leftarrow \mathcal{C}_{L_b} \cup \{j\}$, $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \setminus \{j\}$, and $\mathcal{O}^T_{L_b} \leftarrow \mathcal{O}^T_{L_b} \setminus \{j\}$;
- expression (64): $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \cup \{r(j)\}$ and $\mathcal{O}^T_{L_b} \leftarrow \mathcal{O}^T_{L_b} \cup \{r(j)\}$;
- expression (65): $\mathcal{C}_{L_b} \leftarrow \mathcal{C}_{L_b} \cup \{r(j)\}$ and $\mathcal{O}_{L_b} \leftarrow \mathcal{O}_{L_b} \setminus \{r(j)\}$;
- expression (66).

The dominance condition used in the backward label setting algorithm is the following: a label L^1_b dominates a label L^2_b if conditions (1) and (3)–(5) are satisfied, along with the following condition:

$$t_{L^1_b} \geq t_{L^2_b}. \tag{67}$$

The proof is similar to that given for the forward dominance test.

Table 10 reports detailed results concerning the performance of the bi-directional and the mono-directional labeling algorithms. Columns "$\underline{z}$" and "$\overline{z}$" provide the best lower and upper bounds found within the imposed time limit, and "Time" provides the time needed to solve an instance.

Table 10 Comparison between Bi-directional and Mono-directional Algorithms.

Instance	Mono-directional			Bi-directional		
	$\underline{z}$	$\overline{z}$	Time (sec)	$\underline{z}$	$\overline{z}$	Time (sec)
C6_2_4	369.36	369.36	0	369.36	369.36	0
C7_2_4	390.31	390.31	0	390.31	390.31	0
C8_2_4	446.35	446.35	3	446.35	446.35	21
C9_2_4	471.16	471.16	178	471.16	471.16	146
C10_2_4	510.01	510.01	73	510.01	510.01	55
C11_2_4	522.85	522.85	37	522.85	522.85	27
C12_2_4	541.60	541.60	4	541.60	541.60	4
C15_2_6	649.93	649.93	32	649.93	649.93	21
C20_2_8	751.23	751.23	79	751.23	751.23	69
C25_2_8	879.94	879.94	2,627	879.94	879.94	1,215
C40_2_8	–	–	10,800	1,202.13	–	10,800
RC6_2_4	572.76	572.76	0	572.76	572.76	0
RC7_2_4	575.95	575.95	0	575.95	575.95	0
RC8_2_4	585.32	585.32	0	585.32	585.32	1
RC9_2_4	593.70	593.70	1	593.70	593.70	1
RC10_2_4	599.95	599.95	1	599.95	599.95	1
RC11_2_4	624.48	624.48	1	624.48	624.48	1
RC12_2_6	662.03	662.03	1	662.03	662.03	1
RC15_2_6	1,068.16	1,068.16	1	1,068.16	1,068.16	1
RC20_2_8	1,200.36	1,200.36	3	1,200.36	1,200.36	3
RC25_2_8	1,530.43	1,530.43	40	1,530.43	1,530.43	33
RC40_2_12	1,969.45	1,969.45	1,063	1,969.45	1,969.45	682
RC50_2_14	2,200.88	2,200.88	1,199	2,200.88	2,200.88	921
RC60_2_16	2,636.18	–	10,800	2636.22	–	10,800
R6_2_4	416.16	416.16	0	416.16	416.16	0
R7_2_4	473.06	473.06	1	473.06	473.06	1
R8_2_4	558.17	558.17	2	558.17	558.17	2
R9_2_4	632.42	632.42	6	632.42	632.42	1
R10_2_4	636.05	636.05	3	636.05	636.05	3
R11_2_4	748.29	748.29	3	748.29	748.29	4
R12_2_6	913.68	913.68	6	913.68	913.68	7
R15_2_6	1,083.50	1,083.50	14	1,083.50	1,083.50	13
R20_2_8	1,542.93	1,542.93	19	1,542.93	1,542.93	15
R25_2_8	1,636.12	1,636.12	59	1,636.12	1,636.12	24
R40_2_12	1,969.77	1,969.77	606	1,969.77	1,969.77	589
R50_2_14	2,595.14	2,595.14	1,634	2,595.14	2,595.14	543
R60_2_16	2,868.86	–	10,800	2,871.13	–	10,800
Average			226.35			129.56

In Table 11, under columns "$\underline{z}$" and "$\overline{z}$" we indicate the best lower and upper bounds found within the time limit. The column titled "# via SLs" shows the number of demand units shipped on scheduled lines. In addition, "# of vehicles" indicates the number of vehicles used in the solutions. Finally, "Time" and "Savings" show the time needed to obtain a solution and cost savings from using SLs as part of freight journey, respectively.

Table 11 Comparison between PDPTW-SL and PDPTW with Homogeneous Routing Costs.

Instance	PDPTW-SL						PDPTW			
	$\underline{z}$	$\overline{z}$	# via SLs	# of vehicles	Time (sec)	Savings (%)	$\underline{z}$	$\overline{z}$	# of vehicles	Time (sec)
C6_2_4	369.36	369.36	5	3	0	20.00	461.70	461.70	3	0
C7_2_4	390.31	390.31	6	3	0	15.97	464.49	464.49	3	0
C8_2_4	446.35	446.35	5	4	21	7.68	483.50	483.50	3	0
C9_2_4	471.16	471.16	7	4	146	5.43	498.19	498.19	3	0
C10_2_4	510.01	510.01	4	4	55	1.02	515.25	515.25	3	0
C11_2_4	522.85	522.85	4	4	27	0.35	524.69	524.69	3	0
C12_2_4	541.60	541.60	5	4	4	9.51	598.55	598.55	3	0
C15_2_6	649.93	649.93	8	4	21	10.25	724.12	724.12	4	1
C20_2_8	751.23	751.23	2	5	69	5.45	794.53	794.53	5	2
C25_2_8	879.94	879.94	11	5	1,215	18.27	1,076.71	1,076.71	6	20
C40_2_8	1,202.13	–	–	–	10,800	–	1,428.59	1,428.59	8	1004
RC6_2_4	572.76	572.76	1	3	0	13.09	658.99	658.99	4	0
RC7_2_4	575.95	575.95	1	3	0	13.02	662.18	662.18	4	0
RC8_2_4	585.32	585.32	1	3	1	11.74	663.16	663.16	4	0
RC9_2_4	593.70	593.70	3	3	1	15.58	703.30	703.30	4	0
RC10_2_4	599.95	599.95	3	3	1	15.45	709.55	709.55	4	0
RC11_2_4	624.48	624.48	3	3	1	15.90	742.52	742.52	4	0
RC12_2_6	662.03	662.03	3	4	1	16.66	794.42	794.42	5	0
RC15_2_6	1,068.16	1,068.16	0	6	1	0.00	1,068.16	1,068.16	6	0
RC20_2_8	1,200.36	1,200.36	4	7	3	16.48	1,437.13	1,437.13	7	4
RC25_2_8	1,530.43	1,530.43	6	8	33	2.65	1,572.04	1,572.04	7	12
RC40_2_12	1,969.45	1,969.45	2	10	682	3.16	2,033.66	2,033.66	10	73
RC50_2_14	2,200.88	2,200.88	5	10	921	3.08	2,270.83	2,270.83	10	48
RC60_2_16	2,636.22	–	–	–	10,800	–	2,682.66	2,682.66	12	1518
R6_2_4	416.16	416.16	0	3	0	0.00	416.16	416.16	3	0
R7_2_4	473.06	473.06	0	3	1	0.00	473.06	473.06	3	1
R8_2_4	558.17	558.17	0	3	2	0.00	558.17	558.17	3	0
R9_2_4	632.42	632.42	3	4	1	3.17	653.12	653.12	3	3
R10_2_4	636.05	636.05	3	4	3	3.42	658.60	658.60	4	5
R11_2_4	748.29	748.29	3	4	4	0.54	752.38	752.38	4	0
R12_2_6	913.68	913.68	3	5	7	2.41	936.23	936.23	5	3
R15_2_6	1,083.50	1,083.50	0	6	13	0.00	1,083.50	1,083.50	6	1
R20_2_8	1,542.93	1,542.93	0	8	15	0.00	1,542.93	1,542.93	8	1
R25_2_8	1,636.12	1,636.12	3	8	24	4.73	1,717.39	1,717.39	8	24
R40_2_12	1,969.77	1,969.77	0	9	589	0.00	1,969.77	1,969.77	9	62
R50_2_14	2,595.14	2,595.14	2	12	543	0.98	2,620.89	2,620.89	13	120
R60_2_16	2,871.13	–	–	–	10,800	–	2,959.71	2,959.71	14	1158
Average		909.46	3.12	5.06	129.56	6.94		965.88	5.12	11.18

Table 12 illustrates the results obtained after solving the proposed instances with heterogeneous routing costs. The minimum-capacity vehicle is assumed to cost 0.5 per operating time unit. Larger vehicles are assigned a cost that linearly increases with the carrying capacity. The columns are self-explanatory, similar to the previously presented table.

Table 12 Comparison between PDPTW-SL and PDPTW with Heterogeneous Routing Costs.

Instance	PDPTW-SL						PDPTW			
	$\underline{z}$	$\overline{z}$	# via SLs	# of vehicles	Time (sec)	Savings (%)	$\underline{z}$	$\overline{z}$	# of vehicles	Time (sec)
C6_2_4	385.49	385.49	5	3	0	21.07	488.38	488.38	3	0
C7_2_4	410.76	410.76	6	3	0	16.82	493.81	493.81	3	0
C8_2_4	468.30	468.30	5	4	4	8.75	513.19	513.19	3	0
C9_2_4	495.64	495.64	7	4	186	6.25	528.70	528.70	3	0
C10_2_4	535.01	535.01	4	4	100	2.14	546.71	546.71	3	0
C11_2_4	548.55	548.55	4	4	16	1.46	556.67	556.67	3	0
C12_2_4	568.30	568.30	5	4	4	9.46	627.67	627.67	3	1
C15_2_6	980.46	980.46	8	5	67	18.81	1,207.67	1,207.67	4	9
C20_2_8	890.02	890.02	6	4	91	15.49	1,053.12	1,053.12	5	91
C25_2_8	1,330.48	1,330.48	13	6	5,452	20.64	1,676.53	1,676.53	6	20
C40_2_8	1,441.80	–	–	–	10,800	–	1,800.52	1,800.52	8	3,417
RC6_2_4	593.52	593.52	1	3	0	13.66	687.39	687.39	4	0
RC7_2_4	596.89	596.89	1	3	0	13.59	690.76	690.76	4	0
RC8_2_4	606.78	606.78	1	3	0	12.28	691.73	691.73	4	0
RC9_2_4	615.50	615.50	3	3	1	15.90	731.87	731.87	4	0
RC10_2_4	621.75	621.75	3	3	1	15.81	738.47	738.47	4	1
RC11_2_4	646.28	646.28	3	3	1	16.42	773.28	773.28	4	0
RC12_2_6	693.75	693.75	3	4	1	15.72	823.19	823.19	5	0
RC15_2_6	1,586.84	1,586.84	0	6	1	1.44	1,610.10	1,610.10	6	1
RC20_2_8	1,639.68	1,639.68	4	7	7	18.89	2,021.58	2,021.58	7	6
RC25_2_8	2,403.87	2,403.87	6	8	25	4.16	2,508.24	2,508.24	7	13
RC40_2_12	3,145.84	3,145.84	1	10	10,768	3.19	3,249.36	3,249.36	10	59
RC50_2_14	3,442.28	3,442.28	5	10	5,048	6.03	3,663.11	3,663.11	10	43
RC60_2_16	4,024.96	–	–	–	10,800	–	4,077.43	4,077.43	13	1,103
R6_2_4	416.16	416.16	0	3	0	0.00	416.16	416.16	3	0
R7_2_4	473.06	473.06	0	3	1	0.00	473.06	473.06	3	1
R8_2_4	558.17	558.17	0	3	2	0.00	558.17	558.17	3	0
R9_2_4	632.42	632.42	3	4	4	3.17	653.12	653.12	3	1
R10_2_4	636.05	636.05	3	4	5	3.42	658.60	658.60	4	1
R11_2_4	748.29	748.29	3	4	3	0.54	752.38	752.38	4	0
R12_2_6	946.53	946.53	3	5	7	2.46	970.42	970.42	5	5
R15_2_6	1,377.04	1,377.04	0	6	18	0.00	1,377.04	1,377.04	6	3
R20_2_8	1,839.34	1,839.34	1	8	132	0.05	1,840.22	1,840.22	7	1
R25_2_8	2,011.86	2,011.86	1	8	27	5.80	2,135.65	2,135.65	8	35
R40_2_12	2,518.05	2,518.05	0	9	329	1.80	2,564.27	2,564.27	9	108
R50_2_14	3,182.19	3,182.19	2	12	2,684	1.40	3,227.26	3,227.26	13	466
R60_2_16	3,479.13	–	–	–	10,800	–	3,628.98	3,628.98	14	663
Average		1,133.68	3.24	5.09	734.85	8.14		1,220.82	5.09	25.44

Table 13 indicates the results obtained when solving the instances with up to three SLs. The average is computed over all instances that we optimally solved in all SL configurations. As expected, multiple SLs lead to more savings compared to corresponding PDPTW solutions. In particular, 2-SLs and 3-SLs cases led to cost savings of 4.58%, and 8.39% respectively. In is important to note that the cost savings are mainly obtained from using available capacities, thus minimizing the total traveling time of the PD vehicles. In addition, the average number of vehicles used did not significantly change compared to the classical system.

Table 13 The Effect of Multiple Available SLs.

Instance	One SL (sl=2)				Two SLs (sl=4)				Three SLs (sl=6)			
	$\bar{z}$	# of vehicles	# via SLs	Time (sec)	$\bar{z}$	# of vehicles	# via SLs	Time (sec)	$\bar{z}$	# of vehicles	# via SLs	Time (sec)
C6_2_4	369.36	3	5	0	369.36	3	5	1	310.85	3	8	2,437
C7_2_4	390.31	3	6	0	381.21	3	6	104	313.41	3	9	4,890
C8_2_4	446.35	4	5	21	415.84	3	5	229	–	–	–	10,800
C9_2_4	471.16	4	7	146	459.98	4	7	3,731	–	–	–	10,800
RC6_2_4	572.76	3	1	0	572.76	3	1	0	520.11	4	4	0
RC7_2_4	575.95	3	1	0	575.95	3	1	0	539.08	4	4	0
RC8_2_4	585.32	3	1	1	585.32	3	1	1	548.45	4	4	1
RC9_2_4	593.70	3	3	1	593.70	3	3	1	552.71	4	6	1
RC10_2_4	599.95	3	3	1	599.95	3	3	1	558.96	4	6	1
RC11_2_4	624.48	3	3	1	624.48	3	3	2	585.72	4	7	2
RC12_2_6	662.03	4	3	1	662.03	4	3	2	619.59	4	7	2
RC15_2_6	1,068.16	6	0	1	1,059.38	6	1	3	1,059.38	6	1	3
RC20_2_8	1,200.36	7	4	3	1,127.02	8	8	1,128	1,127.02	8	8	1,389
RC25_2_8	1,530.43	8	6	33	1,388.18	7	8	34	1,371.47	7	14	1,641
RC40_2_12	1,969.45	10	2	682	–	–	–	10,800	–	–	–	10,800
R6_2_4	416.16	3	0	0	416.16	3	0	0	416.16	3	0	0
R7_2_4	473.06	3	0	1	473.06	3	0	2	470.15	3	1	7
R8_2_4	558.17	3	0	2	535.27	4	1	3	490.77	3	2	3
R9_2_4	632.42	4	3	1	579.38	4	3	1	579.38	4	3	1
R10_2_4	636.05	4	3	3	584.89	4	3	1	584.89	4	3	2
R11_2_4	748.29	4	3	4	687.36	4	3	1	687.36	4	3	2
R12_2_6	913.68	5	3	7	858.33	5	4	889	822.64	5	5	43
R15_2_6	1,083.50	6	0	13	1,023.99	6	3	28	921.56	6	6	884
R20_2_8	1,542.93	8	0	15	1,425.42	8	3	105	1,378.70	7	8	5,738
R25_2_8	1,636.12	8	3	24	1,495.46	8	9	20	1,493.46	8	7	78
R40_2_12	1,969.77	9	0	589	1,922.00	9	4	346	–	–	–	10,800