Les Cahiers du GERAD ISSN: 0711–2440

A unified branch-price-and-cut algorithm for multi-compartment pickup and delivery problems

M. Aerts-Veenstra, M. Cherkesly, T. Gschwind

G–2023–26

July 2023

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2023
– Bibliothèque et Archives Canada, 2023

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2023
– Library and Archives Canada, 2023

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A unified branch-price-and-cut algorithm for multi-compartment pickup and delivery problems

Marjolein Aerts-Veenstra [a]

Marilène Cherkesly [b, c]

Timo Gschwind [d]

[a] *Department of Operations, Faculty of Economics and Business, University of Groningen, Groningen, The Netherlands*

[b] *Département d'analytique, opérations et technologies de l'information, Université du Québec à Montréal, Montréal (Qc), Canada, H2X 1L7*

[c] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[d] *Chair of Logistics, Faculty of Business Studies and Economics, RPTU Kaiserslautern-Landau, 67663 Kaiserslautern, Germany*

m.aerts-veenstra@rug.nl
cherkesly.marilene@uqam.ca
gschwind@rptu.de

July 2023
Les Cahiers du GERAD
G–2023–26

Abstract : In this paper, we study the pickup and delivery problem with time windows and multiple compartments (PDPTWMC). The PDPTWMC generalizes the pickup and delivery problem with time windows to vehicles with multiple compartments. In particular, we consider three compartment-related attributes: 1) compartment capacity flexibility which allows the capacities of the compartments to be fixed or flexible, 2) item-to-compartment flexibility that specifies which items are compatible with which compartments, and 3) item-to-item compatibility which considers that incompatible items cannot be simultaneously in the same compartment. To solve the PDPTWMC, we propose an exact branch-price-and-cut algorithm in which the pricing problem is solved by means of a unified bidirectional labeling algorithm. The labeling algorithm can tackle all possible combinations of the studied compartment-related attributes of the PDPTWMC. Furthermore, we implement several acceleration techniques that allow, amongst others, to reduce the symmetry in the label extensions with empty compartments, the symmetry in the dominance between compartments with similar attributes, and the complexity of the algorithm with fixed compartment capacity. Finally, we introduce benchmark instances for the PDPTWMC and conduct an extensive computational campaign to test the limits of our algorithm and to derive relevant managerial insights in order to highlight the applicability of considering the studied compartment-related attributes.

Keywords : Vehicle routing with pickup and delivery, multiple compartments, flexible capacity, item incompatibility, column generation, branch-price-and-cut

Acknowledgements: This work was supported by the Canadian Natural Sciences and Engineering Research Council under grant 2017-06106. This support is gratefully acknowledged.

1 Introduction

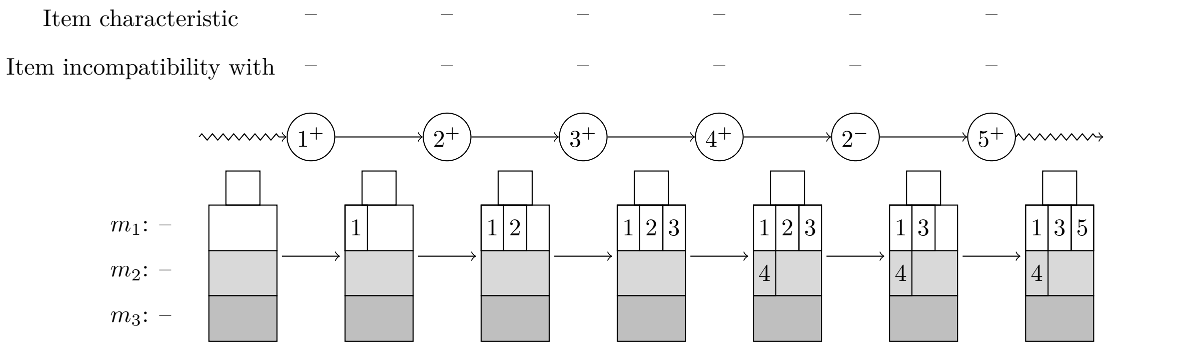
In this paper we introduce, model and solve the pickup and delivery problem with time windows and multiple compartments (PDPTWMC). The PDPTWMC belongs to the family of pickup and delivery problems (PDPs). More specifically, it generalizes the well-studied pickup and delivery problem with time windows (PDPTW) to vehicles with compartmented loading spaces, i.e., separated independent areas referred to as compartments. For each customer request, an item has to be transported by one vehicle from a given pickup location to its corresponding delivery location. Each item has known characteristics (e.g., frozen or ambient good) which can be compatible (or incompatible) with other items or vehicle compartments. All pickup and delivery locations have a time window during which the service must start. A set of homogeneous vehicles with a fixed capacity is available to complete the customer requests, and each vehicle has a maximum number of compartments with given characteristics (capacity and item-compatibility). We study variants of the PDPTWMC according to three attributes that arise in the context of multi-compartment routing problems, i.e., compartment capacity flexibility, item-to-compartment flexibility, and item-to-item compatibility.

Compartment capacity flexibility allows the capacities (i.e., the sizes) of the compartments to be fixed or flexible. This can arise, for example, in food distribution where vehicles can have a fresh and a frozen compartment separated by a wall that may be adjustable before starting the route. With *flexible* compartment capacity, a minimum and a maximum capacity are defined for each compartment, and the total assigned capacity over all compartments must never exceed the vehicle's capacity. The capacity remains the same throughout the route. Note that the minimum and maximum compartment capacity can be set to zero and the total vehicle capacity, respectively. This allows to use only a subset of the compartments. With *fixed* compartment capacity, each compartment has a given capacity (i.e., its minimum and maximum capacities are the same).

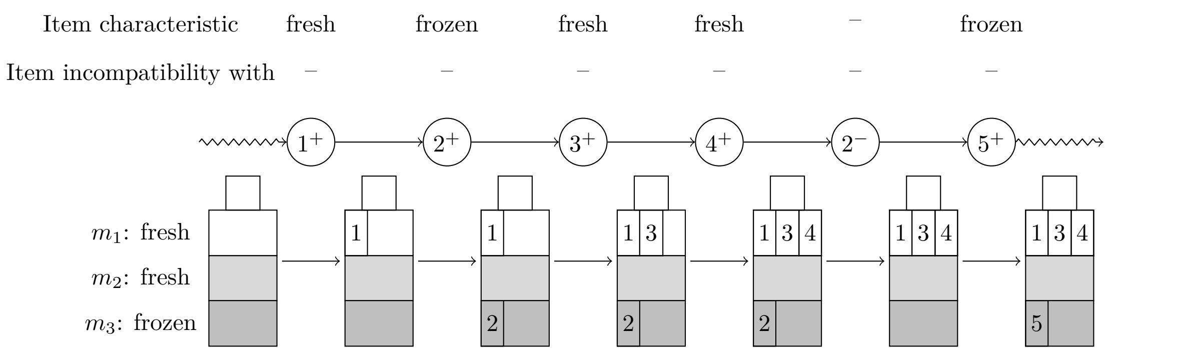
Item-to-compartment flexibility implies that each compartment has known characteristics that are either compatible or incompatible with specific items. This arises for example in waste collection where vehicles have specific compartments dedicated to general waste, organic waste, and mixed (or non-mixed) recycling (e.g., glass, paper, plastic). Item-to-compartment flexibility allows for *full flexibility*, *partial flexibility*, and *no flexibility*. With *full flexibility*, all compartments have the same characteristics and each item is compatible with each compartment. With *no flexibility*, every compartment has specific characteristics such that each item is compatible with exactly one compartment, e.g., frozen goods can only be loaded in the frozen compartment and fresh goods can only be loaded in the fresh compartment. With *partial flexibility*, compartments have specific characteristics that are compatible with only some of the items, i.e., a subset of items is compatible with a subset of compartments (e.g., ambient goods can be loaded in the fresh or ambient compartment). Note that our definition of *partial flexibility* excludes the cases of *full flexibility* and *no flexibility*.

For *item-to-item compatibility*, items can be incompatible with other items, and incompatible items cannot be simultaneously in the same compartment. This arises, for example, in chemical distribution where interactions between the products could be harmful. Item-to-item compatibility can be *fully compatible*, *partially compatible*, or *fully incompatible*. *Full compatibility* implies that all items are compatible, *partial compatibility* implies that there are both compatible and incompatible items, *full incompatibility* implies that each item is incompatible with all other items.

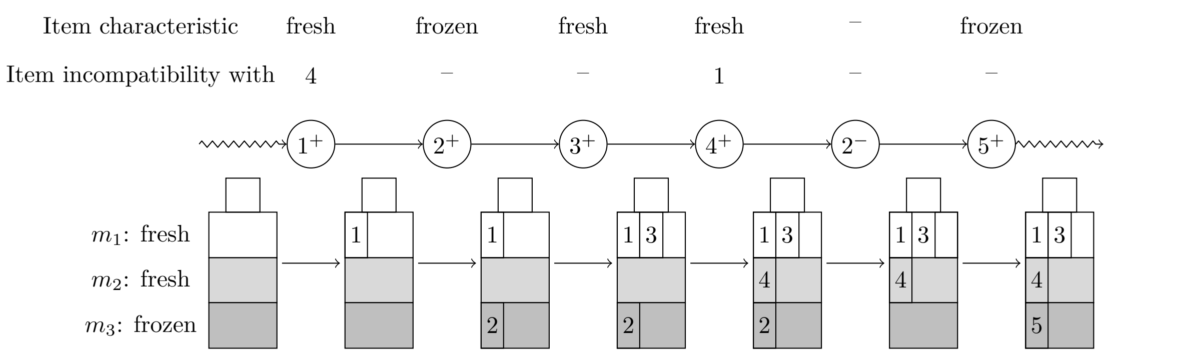
Figures 1a–1c illustrate a path, i.e., a partial route, with a feasible vehicle configuration according to given realizations of the compartment-related attributes. In each figure, all customers have unit demand, the vehicle has a total capacity of nine and consists of three compartments with a fixed capacity equal to one third of the vehicle's total capacity. Each customer i is associated with its pickup vertex i^+ and its delivery vertex i^-. In Figure 1a, all items and compartments have the same characteristics (*full* item-to-compartment flexibility and *full* item-to-item compatibility). In the vehicle configuration, items can be loaded in each compartment as long as the compartment capacity is not exceeded, i.e., at most three items can be in each compartment simultaneously. In Figure 1b, there are



(a) PDPTWMC with fixed compartment capacity, *full* item-to-compartment flexibility, and *full* item-to-item compatibility



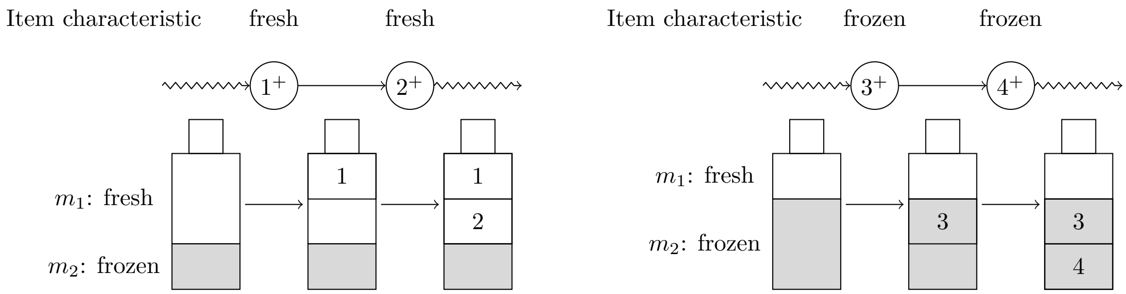
(b) PDPTWMC with fixed compartment capacity, *partial* item-to-compartment flexibility, and *full* item-to-item compatibility



(c) PDPTWMC with fixed compartment capacity, *partial* item-to-compartment flexibility, and *partial* item-to-item compatibility

Figure 1: Example of a path with different vehicle configurations according to the PDPTWMC attributes

item-to-compartment flexibility restrictions in addition to the capacity restrictions of the compartment already present in Figure 1a. Items are either frozen or fresh and compartments m_1 and m_2 are only compatible with fresh items, whereas compartment m_3 is only compatible with frozen items, implying a *partial* item-to-compartment flexibility. There are no incompatibilities between items, i.e., *full* item-to-item compatibility. Then, at most three items can be in each compartment simultaneously and items 1, 3, and 4, can only be loaded in compartments m_1 or m_2, whereas items 2 and 5 can only be loaded in compartment m_3. In Figure 1c, item-to-item compatibility restrictions are added to the compartment capacity restrictions and the item-to-compartment flexibility restrictions of the example in Figure 1b. More precisely, items 1 and 4 are assumed to be incompatible leading to *partial* item-to-item compatibility. In this case, at most three items can be in each compartment simultaneously and items 1, 3, and 4, can only be loaded in compartments m_1 or m_2, whereas items 2 and 5 can only



(a) Fresh compartment requires at least capacity six **(b) Frozen compartment requires at least capacity six**

Figure 2: Example of two paths requiring different capacities for the compartments

be loaded in compartment m_3. Moreover, the incompatibility between items 1 and 4 implies that they cannot be in the same compartment simultaneously so that they need to be loaded in two different compartments, namely compartments m_1 and m_2.

The example in Figure 2 highlights the benefits of compartment capacity flexibility. All customers have a demand of three. The vehicle has a total capacity of nine and it consists of two compartments: one for fresh items and one for frozen items. There are no item-to-item compatibility restrictions. Figure 2 shows two different paths, each of which can be feasibly realized by a vehicle with fixed compartment capacities, e.g., with capacities of six (fresh compartment) and three (frozen compartment) for the route in Figure 2a and capacities of three (fresh compartment) and six (frozen compartment) for the route in Figure 2b. However, there exists no feasible solution including both paths using a single type of vehicle with fixed compartment capacity. In contrast, with flexible compartment capacity and minimum and maximum compartment capacities of three and six, respectively, both paths can be part of a feasible solution.

It is easy to see that the PDPTW is a special case of the PDPTWMC with the following realizations of the compartment-related attributes: *full* item-to-compartment flexibility, *full* item-to-item compatibility, at least one compartment with a maximum capacity equal to the vehicle's capacity, and a minimum capacity of zero for all other compartments. Then, any feasible solution can be modified *a posteriori* to a feasible PDPTW solution, and no modifications are needed for those routes, for which all items are loaded in the same compartment.

To the best of our knowledge, this is the first time that the PDPTWMC is studied. It is also the first time that a unified algorithm is proposed to tackle three important compartment-related attributes. We are not aware of any related work in the literature that studies different variants of each combination of these attributes. The problem is closely related to vehicle routing problems with multiple compartments (VRPMC) and to the pickup and delivery problem with multiple stacks (PDPMS). For VRPMCs, Ostermeier et al. (2021) highlight three compartment-related attributes that have been considered in the literature: the flexibility of compartment sizes, the assignment of product type to compartments and the shareability of compartments. It is easy to see that for PDPs the first two attributes are similar to compartment capacity flexibility and item-to-compartment flexibility. For the third attribute, item-to-item compatibility seems to be the best generalization of the shareability of compartments to a PDP setting. In VRPMCs, compartments can either have one item (no shareability) or more than one item (shareability), but in the PDP setting, compartments can be used to transport more than one item (i.e., items can be unloaded in the route and other items can be loaded later in the route). In the PDPMS (see e.g., Cherkesly et al. 2016), stacks are independent and operated using specific loading rules, such as last-in-first-out loading (LIFO). Stacks can be seen as a variant of compartments as compartments do not have to be operated using such loading rules which allows for more flexibility. Additional attributes, like the above-mentioned compartment-related attributes, have not been considered in the multiple stacks literature.

Ostermeier et al. (2021) point out that most algorithms that have been developed to solve MCVRPs are heuristics with a few exact algorithms based on branch-and-cut (e.g., Henke, Speranza, and Wäscher 2019), column generation (e.g., Cornillier et al. 2008), and branch-and-price (B&P, see Section 2). Column-generation based approaches like B&P constitute the leading exact methodology for many routing problems (Costa, Contardo, and Desaulniers 2019) including the PDPTW and related variants (e.g., Baldacci, Bartolini, and Mingozzi 2011, Gschwind et al. 2018, Ropke and Cordeau 2009). In this paper, we develop a unified exact branch-price-and-cut (BPC) algorithm to solve the PDPTWMC. We also adapt the bidirectional labeling proposed by Gschwind et al. (2018) which is an important recent development for B&P algorithms for PDPs.

The contributions of this paper are five-fold. First, we introduce a new family of routing problems which considers multi-compartment vehicles in a PDP setting. Second, we adapt the three compartment-related attributes studied in VRPMCs to the PDPTWMC (i.e., compartment capacity flexibility, item-to-compartment flexibility, and item-to-item compatibility) and study multiple different combinations of these attributes. Third, we model the PDPTWMC with a set-partitioning formulation and propose a unified BPC algorithm that can handle all three considered compartment-related attributes. Fourth, we derive a bidirectional labeling algorithm to solve the pricing problems and propose ways to reduce the symmetry and the complexity of the algorithm according to the problem's characteristics. Finally, we conduct extensive computational experiments to understand the impact of considering (or not) some compartment-related attributes on the performance of the algorithm and to derive related managerial insights.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 defines the PDPTWMC by presenting a set-partitioning formulation. The BPC algorithm including the bidirectional labeling algorithm to solve the pricing problem is detailed in Section 4. Computational results are presented in Section 5, and final conclusions are drawn in Section 6.

2 Literature review

The PDPTWMC belongs to the family of vehicle routing problems (VRPs, see Mor and Speranza 2020, for a recent survey). It generalizes one-to-one PDPs (Battarra, Cordeau, and Iori 2014, Berbeglia et al. 2007), and more specifically the PDPTW, to routing problems with multiple compartments (Ostermeier et al. 2021). In this section, we first review characteristics of compartments that are studied in multi-compartment vehicle routing problems (MCVRPs). We then focus on B&P algorithms developed for MCVRPs. Finally, we review two related variants of PDPs, i.e., the pickup and delivery problems with multiple stacks (PDPTWMS) and the pickup and delivery problem with incompatibilities (PDPTW-IC), as well as state-of-the-art algorithms developed for these problems.

MCVRPs extend routing problems to consider vehicles with multiple compartments and arise in many applications, such as, food and fuel distribution, agricultural and maritime transportation, and waste collection (see Coelho and Laporte 2015, Ostermeier et al. 2021, for reviews). Three compartment-related attributes (i.e., flexiblity of compartment sizes, assignment of product types to compartments, and shareability of compartments) and two order fulfillment-related attributes (i.e., total number of visits per customer and mode of demand fulfillment) are studied in the literature (Ostermeier et al. 2021). As pointed out by Ostermeier et al. (2021), most papers study only one or two compartment-related attributes. Most works consider compartments with a fixed capacity (e.g., Coelho and Laporte 2015, Zbib and Laporte 2020), but a few authors consider flexible compartment capacities (e.g., Henke, Speranza, and Wäscher 2015, Heßler 2021, Hübner and Ostermeier 2019). In the MCVRP, each customer demands a specified set of items to be delivered (or picked up) and each item is associated with an item-category (also referred to as product-type or commodity-type). According to the problem characteristics, item categories can have a fixed assignment, i.e., each item category is compatible only with a subset of compartments (e.g., Martins et al. 2019, Ostermeier et al. 2018) or a flexible assignment, i.e., each item category is compatible with all compartments (e.g., Christiansen et al. 2017, Lahyani et al. 2015). Most MCVRPs considered in the literature assume that compartments can be shared between customers (e.g., Kiilerich and Wøhlk 2018, Yahyaoui et al. 2020), but a few papers assume unshared compartments (e.g., Hsu, Walteros, and Batta 2020, Jetlund and Karimi 2004). In addition, single and multiple customer visits as well as split and unsplit customer demands are studied for the MCVRP.

We now point out a few exact B&P algorithms for the MCVRP. Avella, Boccia, and Sforza (2004) solve a MCVRP with fixed compartment capacities, flexible assignment of product types to compartments, and with unshared compartments. They propose a B&P algorithm where the pricing problem is solved through enumeration given that they consider routes with at most four customers. Their algorithm solves within a few seconds a real-life instance with 60 customers and vehicles with seven to nine compartments. Heßler and Irnich (2022) solve a problem with similar compartment-related characteristics through B&P. Two labeling algorithms are proposed to solve the pricing problem. In the first one, in each label all customer demands are explicitly assigned to a specific compartment and a standard one-to-one dominance test to eliminate unpromising labels is employed. In the second one, labels implicitly represent all feasible customer-to-compartment assignments for the associated path and a more sophisticated partial dominance rule is implemented. Their algorithm solves benchmark instances with up to 100 customers and four compartments within one-hour, and their results show that using partial dominance greatly reduces the total computational time. Mirzaei and Wøhlk (2019) solve a MCVRP with fixed compartment capacities, fixed assignment of product types to compartments, and shared compartments. They propose a B&P algorithm for two variants: one where a customer can receive a single visit only and one where a customer can receive different visits for each item-category. Their pricing problem is solved by a labeling algorithm which keeps track of the available capacity in the truck for each item-category. Their algorithm is tested on instances with up to 100 customers and four compartments (i.e., commodities). With four compartments, their algorithm can prove optimality for instances with up to 50 customers. Heßler (2021) solves a MCVRP with continuously and discretely flexible compartment capacities, where each item is compatible with all compartments, and different items can be loaded in a compartment only if they are of the same item-category. Therefore, item-categories are assigned to specific compartments. Three exact algorithms are proposed including a BPC algorithm in which multiple pricing problems, one for each feasible item-category-to-compartment assignment, are solved. Instances with up to 50 customers and between two and nine compartments are solved within two hours.

The PDPTWMS can be seen as a variant of the PDPTWMC where vehicles have a fixed number of compartments (referred to as stacks) with a fixed capacity, and each item is compatible with each stack and with all other items. In this problem, the stacks are operated using LIFO loading which imposes that stacks are rear-loaded and that an item can only be delivered if it is the one closest to the rear door. This problem is a generalization of the traveling salesman problem with pickup and delivery and multiple stacks (Côté et al. 2012, Pereira, Mateus, and Urrutia 2022, Pereira and Urrutia 2018) to multiple vehicles. Cherkesly et al. (2016) introduce the problem and propose a BPC algorithm with a monodirectional labeling algorithm that breaks symmetry between the stacks. Their algorithm is tested on 198 benchmark instances and instances with up to 75 requests and three stacks are solved within two hours. Using a mixed-integer programming formulation with partial route enumeration, Al-Yasiry (2020) are able to solve larger-sized instances within a few minutes. The recent work of Cherkesly and Gschwind (2022) considers variants of the PDPTWMS that relax the LIFO policy by allowing the rehandling of items, i.e., unloading and reloading, at delivery locations. The proposed BPC approach solves most instances of the benchmark considered in Al-Yasiry (2020) and Cherkesly et al. (2016).

Another related problem is the PDPTW-IC which is defined as a PDPTW where items can be (in)compatible, and incompatible items can never be loaded simultaneously in the vehicle. The problem was introduced by Deng et al. (2022) who consider two categories of items (perishable products with decaying costs and stable products without decaying costs) and aim to minimize the total costs defined as the sum of the routing, refrigeration, and decay costs. They propose a B&P algorithm with a bidirectional labeling algorithm to solve the PDPTW-IC. Their algorithm solves instances with up to 40 requests within two hours. Huang et al. (2023) study a PDP arising in the steel industry with many real-life constraints including request incompatibilities. They propose an adaptive large neighborhood search which provides good results for smaller instances when compared with the optimal solution, and the algorithm is tested on many instances with up to 450 requests. Factorovich, Méndez-Díaz, and Zabala (2020) study the single vehicle PDP with incompatibility constraints. They propose a branch-and-cut algorithm and solve instances with up to 20 requests within two hours. Ousmane, Moustapha, and Adama (2021) implement a genetic algorithm for a many-to-many PDPTW with multiple compartments and incompatible items. Results on four randomly generated instances with 25 to 100 nodes and 3 to 80 compartments are presented.

3 Mathematical formulation

The PDPTWMC is defined on a directed graph $G = (N, A)$, where $N = \{0, 2n+1\} \cup P \cup D$ is the set of vertices and A is the set of arcs. The origin and destination depot are denoted by 0 and $2n+1$, the set of pickup vertices by $P = \{1, ..., n\}$ and the set of delivery vertices by $D = \{n+1, ..., 2n\}$. Each request i requires loading an item at a pickup vertex $i \in P$, also denoted by i^+, and unloading the item at its associated delivery vertex $n+i \in D$, also denoted by i^-. For ease of notation, we also use set P to refer to the sets of items and requests. The meaning should be always clear from the context. For each vertex $i \in N$, a demand q_i is given with $q_0 = q_{2n+1} = 0$, $q_i > 0, \forall i \in P$ and $q_i = -q_{i-n}, \forall i \in D$. Furthermore, with each vertex is assigned a service time $s_i \geq 0$ and a time window $[\underline{w}_i, \overline{w}_i]$ in which the service has to start. For each pair of items $i, j \in P, i \neq j$, a binary parameter u_{ij} equals to one if items i and j are compatible, and zero otherwise. With each arc $(i, j) \in A$, a travel cost $c_{ij} \geq 0$ and a travel time $t_{ij} \geq 0$ are associated. As commonly done, we include in the travel time for arc $(i, j) \in A$ the service time s_i at vertex i. It is assumed that the triangle inequality is satisfied for the travel times. Each vehicle has an overall capacity Q and m^{max} compartments. The set of compartments is denoted M, where $|M| = m^{max}$ holds. Each compartment m has a minimum and a maximum capacity, denoted by Q_m^{min} and Q_m^{max}, such that $0 \leq Q_m^{min} \leq Q_m^{max} \leq Q$. Setting $Q_m^{min} = 0$ allows to use only a subset of the compartments. For each pair of item $i \in P$ and compartment $m \in M$, a binary parameter b_{im} equals to one if item i is compatible with compartment m, and zero otherwise.

Let Ω be the set of feasible routes with respect to pairing and precedence, time windows, item-to-item compatibility, compartment and vehicle capacity, and item-to-compartment compatibility. Each route is associated with a total cost c_r comprising a fixed vehicle cost and the sum of the travel costs of its associated arcs. For each request $i \in P$, binary parameter a_{ir} is equal to one if route r completes request i, and zero otherwise. Binary variables y_r equal one if route r is used in the solution, and zero otherwise. Then, the PDPTWMC can be formulated as

$$\text{minimize} \quad \sum_{r \in \Omega} c_r y_r \tag{1a}$$

$$\text{subject to} \quad \sum_{r \in \Omega} a_{ir} y_r = 1, \quad \forall i \in P, \tag{1b}$$

$$y_r \in \{0, 1\}, \quad \forall r \in \Omega. \tag{1c}$$

The Objective (1a) minimizes the total costs. Constraints (1b) ensure that each request is completed exactly once. The variable domains are defined by Constraints (1c).

4 Branch-price-and-cut algorithm

Formulation (1) usually contains a large number of variables, i.e., feasible routes, which cannot be enumerated. Therefore, we employ a BPC algorithm for its solution. A BPC algorithm is a branch-and-bound algorithm that uses column generation to compute the lower bounds and where cuts are added to strengthen the linear relaxation of Formulation (1), i.e., the master problem. Column generation is an iterative algorithm that starts with solving the restricted master problem (RMP), which only contains a subset of the variables. It then alternates between reoptimizing the RMP and solving the pricing problem to identify negative reduced cost columns, that will be added to the RMP. When no negative reduced cost columns exist, an optimal solution to the RMP is found.

In the remainder of this section, we present our BPC algorithm for the PDPTWMC. We first present the pricing problem and propose a bidirectional labeling algorithm for its solution. We then describe the valid inequalities and our branching strategy.

4.1 Pricing problem

Let $\pi_i, i \in P$ be the dual variables associated with Constraints (1b). The task of the pricing problem is to identify at least one feasible route $r \in \Omega$ with negative reduced cost

$$\tilde{c}_r = c_r - \sum_{i \in P} a_{ir} \pi_i \tag{2}$$

or guarantee that no such route exists. The pricing problem can be formally stated as $\min_{r \in \Omega} \tilde{c}_r$. It is a variant of an elementary shortest path problem with resource constraints (ESPPRC) that can be solved using a labeling algorithm (Irnich and Desaulniers 2005). In a labeling algorithm, labels representing partial paths are extended along the network arcs and dominance criteria are used to remove non-promising labels. To speed up the labeling process, bidirectional labeling algorithms (Righini and Salani 2006) have become a quasi standard for solving the ESPPRC pricing problems of many VRP variants. In these algorithms, forward labels are extended in forward direction starting at the origin depot and backward labels are extended in backward direction, i.e., against the orientation of the network arcs, ending at the destination depot. Both forward and backward labels are extended up to a so-called half-way point (HWP) and suitable forward and backward labels are then merged to form complete origin-destination paths. To better balance the effort of the forward and backward labeling, the HWP can be determined dynamically (Tilk et al. 2017).

For PDPs, bidirectional labeling has not been commonly applied, although its effective implementation has been recently demonstrated by Gschwind et al. (2018). The main issues are as follows. In forward labeling, items have to be picked up first and delivered later. Therefore, the algorithm needs to keep track of onboard items, i.e., items that have been picked up but not yet delivered, and also incorporate them in the dominance. Stronger dominance criteria can be realized if the reduced costs satisfy the so-called delivery triangle inequality (DTI, see Section 4.1.1). In backward labeling, on the other hand, items are delivered first and picked up later. Therefore, the meaning of onboard items is reversed, i.e., items that have been delivered but not yet picked up, and stronger dominance criteria rely on the pickup triangle inequality (PTI, see Section 4.1.2) for the reduced costs. While Ropke and Cordeau (2009) have proposed a procedure to transform an arbitrary reduced cost matrix to one that satisfies the DTI and this method has been adapted to the PTI by Gschwind et al. (2018), the DTI and the PTI can generally not be satisfied simultaneously. To enable an effective application of bidirectional labeling, Gschwind et al. (2018) propose to perform the forward and the backward labeling on two different reduced-cost matrices. This requires additional adjustments in the merge procedure in order to compute the correct reduced cost of routes.

In the following, we propose a unified bidirectional labeling algorithm for the PDPTWMC which builds on the work of Gschwind et al. (2018) and that can handle different settings for the three studied attributes (i.e., compartment capacity flexibility, item-to-compartment flexibility, and item-to-item compatibility). We start by presenting the unified forward labeling algorithm and adapt the notation to present the unified backward labeling algorithm. Then, we explain how a forward and a backward label are merged. Finally, we describe symmetry-reduction strategies and other acceleration techniques used in our labeling algorithm.

4.1.1 Forward labeling algorithm

We first define the reduced cost and describe the resources of the forward labels. Then, we propose resource extension functions (REFs) that consider only extensions that are feasible with respect to the item-to-compartment flexibility and item-to-item compatibility. Finally, we present a valid dominance rule.

Forward reduced cost. For each vertex $i \in N$, let $\tilde{\pi}_i^f := \pi_i$ if $i \in P$ and $\tilde{\pi}_i^f := 0$ otherwise. The forward reduced cost of an arc $(i, j) \in A$ is defined as

$$\tilde{c}_{ij}^f := c_{ij} - \frac{\tilde{\pi}_i^f}{2} - \frac{\tilde{\pi}_j^f}{2}. \tag{3}$$

If the travel cost c_{ij} satisfy the DTI, i.e., $c_{ij} + c_{jk} \geq c_{ik}$ for $i, k \in N, j \in D$, then the forward reduced costs as defined in Equation (3) also satisfy the DTI. Otherwise, the $\tilde{c}_{ij}^f$ matrix can be modified following the procedure proposed by Ropke and Cordeau (2009). Using $\tilde{c}_{ij}^f$, the reduced cost of a route $r \in \Omega$ can be computed as

$$\tilde{c}_r = \sum_{(i,j)\in A(r)} \tilde{c}_{ij}^f, \tag{4}$$

where $A(r)$ denotes the sequence of arcs traversed by route r.

Forward label resources. A forward partial path $R(F) = (0, \ldots, \eta(F))$ starting at the origin depot 0 and ending at vertex $\eta(F)$ is represented by a forward label $F = (\eta(F), t(F), c(F), S(F), (l^m(F))_m, (O^m(F))_m, (\psi^m(F))_m)$ that stores the following information:

- $\eta(F)$, the last visited vertex of the partial path;
- $t(F)$, the earliest feasible start of the service time at vertex $\eta(F)$;
- $c(F)$, the accumulated reduced cost;
- $S(F)$, the set of completed requests;
- $l^m(F)$, the load in compartment $m \in M$;
- $O^m(F)$, the items (open requests) in compartment $m \in M$;
- $\psi^m(F)$, the required capacity of compartment $m \in M$.

The initial label F_0 at the origin depot 0 is given by $F_0 = (0, \underline{w}_0, 0, \varnothing, (0)_m, (\varnothing)_m, (Q_m^{min})_m)$. This initial setting ensures that the required capacity of each compartment is at least its minimum capacity.

Forward REFs. The extension of a label F along arc $(\eta(F), j) \in A$ might result in multiple extensions. Each extension is characterized by the compartment it relates to, i.e., the compartment the corresponding item is loaded on or unloaded from. Let us denote $\mathcal{H}^F(j)$ as the set of potential extensions along arc $(\eta(F), j)$ which respect pairing and precedence constraints as well as item-to-compartment flexibility and item-to-item compatibility. It is defined differently depending on whether j is a pickup vertex, a delivery vertex, or the destination depot. More precisely,

$$\mathcal{H}^F(j) = \begin{cases} \{m \in M | j \notin S(F) \cup \bigcup_{s\in M} O^s(F), b_{jm} = 1, u_{ij} = 1 \ \forall i \in O^m(F)\} & \text{if } j \in P, \\ \{m \in M | j - n \in O^m(F)\} & \text{if } j \in D, \\ \{m_1 \in M | O^m(F) = \varnothing \ \forall m \in M\} & \text{if } j = 2n+1, \end{cases} \tag{5}$$

where m_1 denotes the first compartment. If j is a pickup vertex whose request is neither onboard nor has been completed before, item j can be loaded on all compartments that are compatible with it (item-to-compartment flexibility) and whose corresponding onboard items are also compatible with item j (item-to-item compatibility). If j is a delivery vertex and its corresponding item $j - n$ is onboard, there is exactly one extension to the compartment on which its corresponding item has been loaded. Finally, if j is the destination depot, there is one extension for the first compartment and the extension is allowed if all compartments are empty.

For each extension $h \in \mathcal{H}^F(j)$, the REFs creating the new label F^h are as follows:

$$\eta(F^h) = j, \tag{6a}$$

$$t(F^h) = \max\{\underline{w}_j, t(F) + t_{\eta(F)j}\}, \tag{6b}$$

$$c(F^h) = c(F) + \tilde{c}^f_{\eta(F)j}, \tag{6c}$$

$$S(F^h) = \begin{cases} S(F) \cup \{j-n\} & \text{if } j \in D, \\ S(F) & \text{otherwise,} \end{cases} \tag{6d}$$

$$l^m(F^h) = \begin{cases} l^m(F) + q_j & \text{if } m = h, \\ l^m(F) & \text{otherwise,} \end{cases} \tag{6e}$$

$$O^m(F^h) = \begin{cases} O^m(F) \cup \{j\} & \text{if } m = h \text{ and } j \in P, \\ O^m(F) \setminus \{j-n\} & \text{if } j \in D \text{ and } j-n \in O^m(F), \\ O^m(F) & \text{otherwise,} \end{cases} \tag{6f}$$

$$\psi^m(F^h) = \begin{cases} \max\{l^m(F) + q_j, \psi^m(F)\} & \text{if } m = h \text{ and } j \in P, \\ \psi^m(F) & \text{otherwise.} \end{cases} \tag{6g}$$

Equations (6a)–(6d) are standard REFs from the PDPTW literature. REFs (6e) and (6f) update the load and the set of open requests, respectively, of the compartment where item j ($j-n$) is loaded onto (unloaded from). Finally, REF (6g) updates the required capacity of a compartment as the maximum between its load (including the load of item j) and its previous required capacity if $j \in P$.

Label F^h is kept if the time windows and the capacity constraints are respected, that is if

$$t(F^h) \le \overline{w}_j, \tag{7a}$$

$$\psi^m(F^h) \le Q^{max}_m, \qquad \forall m \in M \tag{7b}$$

$$\sum_{m \in M} \psi^m(F^h) \le Q. \tag{7c}$$

Conditions (7b) ensure that the maximum capacity of each compartment is respected while Condition (7c) ensures that the total capacity of the vehicle is respected.

Forward label dominance. A label F_1 dominates another label F_2 if

$$\eta(F_1) = \eta(F_2), \tag{8a}$$

$$t(F_1) \le t(F_2), \tag{8b}$$

$$c(F_1) \le c(F_2), \tag{8c}$$

$$S(F_1) \subseteq S(F_2), \tag{8d}$$

$$O^m(F_1) \subseteq O^m(F_2) \qquad \forall m \in M, \tag{8e}$$

$$\psi^m(F_1) \le \psi^m(F_2) \qquad \forall m \in M. \tag{8f}$$

Conditions (8) ensure that that for every feasible completion of label F_2 to the destination depot, there also exists a feasible completion of label F_1 with equal or smaller reduced cost provided that the arc reduced costs $\tilde{c}^f_{ij}$ respect the DTI. A formal proof of the validity is provided in Appendix A.

4.1.2 Backward labeling algorithm

We now present the main components of our backward labeling algorithm. The differences between the forward and the backward labeling algorithm are mainly related to the reduced arc cost and the notion of the PTI, as well as the definition of the resources for time and open requests. For concision, we focus the presentation on these parts and refer to Appendix B for additional details including a proof for the validity of the dominance rule.

Backward reduced cost. For each vertex $i \in N$, let $\tilde{\pi}_i^b := \pi_i$ if $i \in D$ and $\tilde{\pi}_i^b := 0$ otherwise. The backward reduced cost of arc $(i,j) \in A$ is defined as

$$\tilde{c}_{ij}^b = c_{ij} - \frac{\tilde{\pi}_i^b}{2} - \frac{\tilde{\pi}_j^b}{2}. \tag{9}$$

The reduced cost of a route $r \in \Omega$ is then computed as in Equation (4) by replacing $\tilde{c}_{ij}^f$ with $\tilde{c}_{ij}^b$. To ensure that the backward reduced-cost matrix satisfies the PTI, i.e., $\tilde{c}_{ij}^b + \tilde{c}_{jk}^b \geq \tilde{c}_{ik}^b$ for $i,k \in N, j \in P$, an analog transformation as for the DTI in the forward case is applied (see Gschwind et al. 2018).

Backward label resources. A partial path $R(B) = (\eta(B), \ldots, 2n+1)$ starting at vertex $\eta(B)$ and ending at the destination depot is represented by a backward label $B = (\eta(B), t(B), c(B), S(B), (l^m(B))_m, (O^m(B))_m, (\psi^m(B))_m)$. Resources $c(B)$, $S(B)$, $l^m(B)$, $O^m(B)$, and $\psi^m(B)$ have the same meaning as their forward counterparts. The other resources, while similar to their corresponding forward resource, are defined slightly different:

- $\eta(B)$, the first visited vertex of the partial path;
- $t(B)$, the latest feasible start of the service time at vertex $\eta(B)$.

The backward labeling is initialized with label $B_{2n+1} = (2n+1, \overline{w}_{2n+1}, 0, \varnothing, (0)_m, (\varnothing)_m, (Q_m^{min})_m)$ at the destination depot $2n+1$.

Backward REFs. Analog to the forward case, we denote by $\mathcal{H}^B(i)$ the set of potential extensions of a label B against the orientation of arc $(i, \eta(B)) \in A$. Set $\mathcal{H}^B(i)$ is defined such that pairing and precedence, item-to-compartment flexibility, and item-to-item compatibility are respected. Formally,

$$\mathcal{H}^B(i) = \begin{cases} \{m \in M | i \in O^m(B)\} & \text{if } i \in P, \\ \{m \in M | i-n \notin S(B) \cup \bigcup_{s \in M} O^s(B), b_{i-n,m} = 1, u_{i-n,j} = 1 \ \forall j \in O^m(B)\} & \text{if } i \in D, \\ \{m_1 \in M | O^m(B) = \varnothing \ \forall m \in M\} & \text{if } i = 0. \end{cases} \tag{10}$$

Its definition is analog to that of set $\mathcal{H}^F(j)$ in Equation (5).

For each label B and each extension $h \in \mathcal{H}^B(i)$ a new label B^h is created according to the following REFs:

$$\eta(B^h) = i, \tag{11a}$$

$$t(B^h) = \min\{\overline{w}_i, t(B) - t_{i\eta(B)}\}, \tag{11b}$$

$$c(B^h) = c(B) + \tilde{c}_{i\eta(B)}^b, \tag{11c}$$

$$S(B^h) = \begin{cases} S(B) \cup \{i\} & \text{if } i \in P, \\ S(B) & \text{otherwise,} \end{cases} \tag{11d}$$

$$l^m(B^h) = \begin{cases} l^m(B) - q_i & \text{if } m = h, \\ l^m(B) & \text{otherwise,} \end{cases} \tag{11e}$$

$$O^m(B^h) = \begin{cases} O^m(B) \setminus \{i\} & \text{if } i \in P \text{ and } i \in O^m(B), \\ O^m(B) \cup \{i-n\} & \text{if } m = h \text{ and } i \in D, \\ O^m(B) & \text{otherwise,} \end{cases} \tag{11f}$$

$$\psi^m(B^h) = \begin{cases} \max\{l^m(B) - q_i, \psi^m(B)\} & \text{if } m = h \text{ and } i \in D, \\ \psi^m(B) & \text{otherwise.} \end{cases} \tag{11g}$$

Equations (11) adapt Equations (6) to backward extensions. Label B^h is kept if the time windows and the capacity constraints are respected, i.e., if it satisfies Conditions (7b) and (7c) where F^h is replaced by B^h, as well as

$$t(B^h) \geq \underline{w}_i. \tag{12}$$

Backward label dominance. A backward label B_1 dominates another backward label B_2 if Conditions (8a), (8c)–(8f), replacing F_1 and F_2 with B_1 and B_2, respectively, and

$$t(B_1) \geq t(B_2) \tag{13}$$

are fulfilled.

4.1.3 Bidirectional labeling algorithm

The basic course of our bidirectional labeling algorithm follows the ideas presented in Gschwind et al. (2018). Forward labeling is performed as described in Section 4.1.1 using reduced costs $\tilde{c}^f_{ij}$, backward labeling is performed as described in Section 4.1.2 using the different reduced costs $\tilde{c}^b_{ij}$. The merge procedure takes care that the correct reduced costs of routes are finally determined given the different cost matrices. Both forward and backward labels are extended only up to the HWP, which we define on the time resource, i.e., $t(F)$ in forward labeling and $t(B)$ in backward labeling. Furthermore, we employ a dynamic determination of the HWP.

In the PDPTWMC, the merge of a forward label F and a backward label B residing at the same vertex $i = \eta(F) = \eta(B)$ is feasible if

$$t(F) \leq t(B), \tag{14a}$$

$$S(F) \cap S(B) = \varnothing, \tag{14b}$$

$$\begin{cases} O^m(F) = O^m(B^p) & \text{if } i \in P \\ O^m(F^p) = O^m(B) & \text{if } i \in D \\ O^m(F) = O^m(B) & \text{otherwise} \end{cases} \qquad \forall m \in M, \tag{14c}$$

$$\sum_{m \in M} \max\{\psi^m(F), \psi^m(B)\} \leq Q, \tag{14d}$$

where F^p and B^p denote the predecessor labels of F and B, respectively. Condition (14a) ensures time feasibility while Condition (14b) guarantees elementarity. Conditions (14c) require that all open requests are completed and that their compartment assignments are consistent. Note that when merging at a pickup vertex $i \in P$, the information on which compartment item i has been loaded is no longer available in the backward label B and we need to resort to its predecessor label B^p. The analog is true for merging at a delivery vertex and the forward label. Condition (14d) guarantees that the route respects the total capacity.

If the merge of forward label F and backward label B is feasible, the reduced cost of the corresponding route $r \in \Omega$ can be computed as

$$\tilde{c}_r = c(F) + c(B) + \sum_{i \in \{O(F) \cap O(B)\}} \pi_i + \sum_{\substack{i \in \{O(F) \cup O(B)\} \setminus \\ \{O(F) \cap O(B)\}}} \frac{\pi_i}{2}, \tag{15}$$

where $O(F) := \bigcup_{m \in M} O^m(F)$ and $O(B) := \bigcup_{m \in M} O^m(B)$.

4.1.4 Acceleration strategies

In this section, we first propose two techniques to reduce the symmetry in the labeling. We then describe simplifications of the labeling algorithm when considering fixed compartment capacities. Finally, we mention general acceleration techniques used in our labeling algorithm.

Symmetry reduction techniques. We propose two types of symmetry-reduction techniques, one for label extensions and one for the dominance. Both types of symmetry reduction can immediately be applied in pure monodirectional forward and backward labeling. We limit the following description to the forward case, the backward case is analog. Furthermore, both techniques can be applied simultaneously. When applying bidirectional labeling, however, the merge has to be adapted accordingly. Details on this adaptation are given after the description of the symmetry-reduction techniques.

For the presentation of the symmetry-reduction techniques, some additional notation is convenient. Let $\mathcal{C}$ denote a partition of the compartments into sets of *item-to-compartment comparable* (or short *comparable*) compartments. More precisely, $\bigcup_{C\in\mathcal{C}} C = M$, $\bigcap_{C\in\mathcal{C}} C = \varnothing$, and for two different compartments $m_1 \neq m_2$ with $m_1 \in C_1$ and $m_2 \in C_2$ it holds that $C_1 = C_2$ if and only if $b_{im_1} = b_{im_2}, \forall i \in P$. We denote by $C(m) := \{m' \in M | b_{im} = b_{im'}\ \forall i \in P\}$ the set of compartments that are comparable with compartment $m \in M$.

The first symmetry reduction technique considers label extensions to empty compartments. Consider the extension of a label F along arc $(\eta(F), j) \in A$ to a pickup node $j \in P$. According to Equation (5), this generally creates multiple extensions, one for each suitable compartment. If several compartments are empty in F, some of the resulting new labels may essentially be identical except for symmetry. More precisely, two extensions $m_1, m_2 \in \mathcal{H}^F(j)$ with corresponding compartments $m_1 \in C_1$ and $m_2 \in C_2$ result in symmetric labels if compartments m_1 and m_2 are empty, i.e., $O^{m_1}(F) = O^{m_2}(F) = \varnothing$, comparable, i.e., $C_1 = C_2$, and have identical required capacity $\psi^{m_1}(F) = \psi^{m_2}(F)$. Symmetry can then be reduced by performing only a single extension for each set of extensions leading to symmetric labels.

The second symmetry reduction technique considers label dominance with comparable labels. With Conditions (8e) and (8f), the proposed dominance rule of Section 4.1.1 directly compares the status of each compartment $m \in M$ for the two labels F_1 and F_2. To strengthen the dominance rule, this strict one-by-one comparison of the compartments can be relaxed by also checking for symmetric assignments of items to compartments in label F_2 that fulfill the dominance criteria. Formally, dominance rule (8) can be improved as follows.

A forward label F_1 dominates another forward label F_2 if Conditions (8a)–(8d) hold and there exists a permutation σ of the compartments such that

$$O^m(F_1) \subseteq O^{\sigma(m)}(F_2) \quad \forall m \in M, \tag{16a}$$

$$\psi^m(F_1) \leq \psi^{\sigma(m)}(F_2) \quad \forall m \in M, \tag{16b}$$

$$\sigma(m) \in C(m) \qquad \forall m \in M. \tag{16c}$$

This improved dominance criterion is stronger than dominance rule (8) because Conditions (16a)–(16c) allow any one-to-one pairing of comparable compartments instead of the pure direct comparison of compartments in Conditions (8e) and (8f).

When using bidirectional labeling and applying any of the two symmetry reduction techniques in the forward and backward labeling, the merge procedure in (14) does no longer guarantee to provide at least one route with minimum reduced cost. This is because it relies on a pure direct comparison of compartments in Conditions (14c) and (14d) which is incompatible with the proposed symmetry reduction for label extensions and dominance. A merge condition that is valid also in these cases can be obtained by allowing the one-to-one pairing of comparable compartments with symmetric item assignments similar to the modified dominance described above. Formally, the merge of a forward label F and a backward label B residing at the same vertex i is feasible if Conditions (14a) and (14b) hold and there exists a permutation σ of the compartments such that

$$\begin{cases} O^m(F) = O^{\sigma(m)}(B^p) & \text{if } i \in P \\ O^m(F^p) = O^{\sigma(m)}(B) & \text{if } i \in D \\ O^m(F) = O^{\sigma(m)}(B) & \text{otherwise} \end{cases} \quad \forall m \in M, \tag{17a}$$

$$\sum_{m \in M} \max\{\psi^m(F), \psi^{\sigma(m)}(B)\} \leq Q, \tag{17b}$$

$$\sigma(m) \in C(m) \qquad \forall m \in M. \tag{17c}$$

Labeling simplifications for the PDPTWMC with fixed compartment capacities. If the capacities of the compartments are fixed, i.e., $Q_m^{min} = Q_m^{max}$, $\forall m \in M$, the labeling algorithm can be simplified by removing the label components $\psi^m(F)$ and $\psi^m(B)$ for all compartments $m \in M$. Accordingly, the REFs and the conditions for label feasibility, dominance, and merge feasibility related to them are also removed. More specifically, forward labels F can be extended using REFs (6a)–(6f). A newly created label F^h is feasible if Conditions (7a) and

$$l^m(F^h) \leq Q_m^{max} \qquad \forall m \in M, \tag{18}$$

are respected. Finally, a label F_1 dominates another label F_2 if Conditions (8a)–(8e) are fulfilled. The adaptations for the backward labeling are analog. Feasibility of the merge requires only Conditions (14a)–(14c) to hold.

Other acceleration techniques. To further speed up the pricing process, we use the following well-established acceleration techniques. First, we rely on pricing heuristics to generate negative reduced-cost routes quickly. Following Desaulniers, Lessard, and Hadjar (2008), we price on arc-reduced networks with a minimum of $k = 2, 5$ and 10 arcs entering and exiting each customer vertex. This approach is further adapted to the PDP case by also balancing the number of incoming (outgoing) arcs originating from (destinating to) pickup and delivery vertices. Second, in addition to completed requests, we also include unreachable requests in sets $S(F)$ and $S(B)$. In forward labeling, a request is unreachable for a label F if its pickup vertex cannot be reached time-window feasibly. The definition in the backward case is analog. The inclusion of unreachable requests improves the dominance (Feillet et al. 2004). Finally, the labeling algorithm is realized using a bucket-based implementation with one-dimensional buckets on the time resource (Sadykov, Uchoa, and Pessoa 2021).

4.2 Valid inequalities and branching

We use two well-established families of valid inequalities in our BPC algorithm: rounded capacity inequalities and subset-row inequalities.

Rounded capacity inequalities were first introduced by Laporte and Nobert (1983) for the capacitated VRP. They are robust cuts, because the corresponding duals directly relate to the reduced costs on arcs and, therefore, do not change the structure of the pricing problem. With these additional duals, the DTI and PTI of the reduced costs for forward and backward labeling, respectively, need to be restored using the transformation of Ropke and Cordeau (2009). The necessary adaptations to the computation of the reduced cost in the merge are described in Gschwind et al. (2018). To separate violated rounded capacity inequalities, we apply the heuristic of Ropke and Cordeau (2009).

Jepsen et al. (2008) were the first to introduce the subset-row inequalities for the VRP with time windows. For PDPs, they are defined on subsets of requests, and as in many other papers, we restrict ourselves to subsets of cardinality three. The subset-row inequalities are non-robust, i.e., each inequality requires an additional resource in the pricing problem making it harder to solve. For the implications of the subset-row inequalities on the bidirectional labeling for PDPs, we refer to Gschwind et al. (2018) who show how to adapt both the forward and backward labeling and the reduced cost computation for the merge.

To guarantee integer solutions, we apply the following standard hierarchical branching rule (Ropke and Cordeau 2009). We first branch on the number of vehicles, if fractional. We then branch on the outflow of a subset of vertices of size two, where we always select a subset with outflow closest to 1.5. For both branching rules, an additional constraint is added to the master problem, which results in additional dual prices to be included in the reduced costs of the corresponding arcs. This has the same implications on pricing as the rounded capacity inequalities. The search tree is explored in a best-bound first fashion.

5 Computational results

In this section, we report extensive computational experiments to assess the performance of our BPC algorithm and to derive managerial insights on the three studied attributes of the PDPTWMC. For conciseness reasons, we only report summarized results. Instance-by-instance results are provided at https://wiwi.rptu.de/fgs/logistik/pdptwmc-detailedresults. Our BPC algorithm was implemented in `C++` and compiled into 64-bit singlethread code with MS Visual Studio 2019. CPLEX 20.10 with default parameters was used to reoptimize the RMPs. All tests were conducted on RPTU Kaiserslautern-Landau's high performance computing cluster "Elwetritsch" that consists of several Intel Xeon Gold 6126 processors running at 2.60 GHz. Notice that the performance of a single thread of the cluster is comparable to that of a standard desktop processor. A time limit of 3,600 seconds was considered.

The remainder of this section is organized as follows. Section 5.1 describes the instances used in the experiments and details the different parameters related to item-to-compartment flexibility, compartment capacity flexibility, and item-to-item compatibility that we tested. Section 5.2 presents details on the performance of the proposed algorithm. Section 5.3 shows the effect of the symmetry-reduction acceleration techniques. Finally, in Section 5.4 we conduct extensive sensitivity analyses and derive managerial insights related to the three tested attributes.

5.1 Instances and attribute settings

Our instances are created by adapting the C2 instance set proposed by Cherkesly et al. (2016) for the PDPTWMS to the PDPTWMC. The set consists of 319 instances which are adapted from the TSPLIB instances. There are 99 instances based on the *a280* instance, and 220 instances based on the instances *brd14051*, *d18512*, *fnl4461*, and *nrw1379*. The number of requests ranges from 25 to 75. We apply the same rounding rule that was used in other works for these instances, i.e., travel costs are rounded to four digits and travel times are rounded up to two digits. As usual for these instances, an artificial cost of 100,000 is added to each arc $(0, i)$, $\forall i \in P$ to ensure that the primary objective is the minimization of the vehicles. We use the time windows and the demands, which are random between three and eight, as provided in the instances. Furthermore, we consider vehicles with three compartments with a total capacity of 24. In addition, we introduce 12 item categories to the instances to define the item-to-item compatibility and the item-to-compartment flexibility, and each item $i \in P$ is randomly assigned to one of the categories. We denote μ_i as the category number of item $i \in P$, i.e., $\mu_i \in \{1, ..., 12\}$. All instances are available at https://wiwi.rptu.de/fgs/logistik/pdptwmc-instances. In the following, we describe the different scenarios for the three compartment-related attributes that we include in our computational analysis. We refer to Table 10 of Appendix C for an overview of the tested scenarios and for the details on how these are obtained from the instance data.

For the compartment capacity flexibility, flexible and fixed compartment capacities are investigated. Note that for simplification reasons, we have set the minimum and maximum compartment capacities to the same values for all three compartments, i.e., $Q_m^{min} = Q^{min}, \forall m \in M$ and $Q_m^{max} = Q^{max}, \forall m \in M$, and refer to the notation Q^{min} and Q^{max} in the following. With fixed compartment capacity, denoted as `cap-[33% 33%]`, three compartments each with a third of the vehicle's capacity are considered, i.e., $Q^{min} = Q^{max} = \lfloor Q/3 \rfloor$. With flexible compartment capacity, different combinations of Q^{max} and Q^{min} are tested. The combinations considered are $Q^{max} = \{\lfloor 50\%Q \rfloor\}$ with $Q^{min} = \{0, \lceil 10\%Q \rceil, \lceil 25\%Q \rceil\}$, and $Q^{min} = \{0\}$ with $Q^{max} = \{\lfloor 50\%Q \rfloor, \lfloor 75\%Q \rfloor, \lfloor 100\%Q \rfloor\}$. These are referred to as `cap-[0% 50%]`, `cap-[10% 50%]`, `cap-[25% 50%]`, `cap-[0% 75%]`, and `cap-[0% 100%]`.

We study three different cases for the item-to-compartment flexibility. In the first case, denoted as `1 i-to-c`, each item is compatible with only one compartment. In the second case, denoted as `2 i-to-c`, partial item-to-compartment flexibility is considered where each item is compatible with exactly two compartments. Finally, in the third case, denoted as `3 i-to-c`, full item-to-compartment flexibility is considered, i.e., each item is compatible with all three compartments.

Finally, five different cases for the item-to-item compatibility are considered. For the first case, denoted as `0% i-to-i`, each item is incompatible with all other items. Then, three cases with *partial* item-to-item compatibility are considered such that 25%, 50%, and 75% of the item categories are compatible, also denoted as `25% i-to-i`, `50% i-to-i`, and `75% i-to-i`. Finally, the fifth case considers *full* item-to-item compatibility, i.e., each item is compatible with all other items, and is denoted `100% i-to-i`.

Overall, for each of the 319 instances, 90 combinations of the compartment capacity flexibility, item-to-item compatibility, and item-to-compartment flexibility attributes have been generated. Thus, 28,710 instances have been tested in total. For the remainder of this section, we use the notation provided in Table 10 of Appendix C to refer to a set of instances. For example, by referring to the *a280*-`cap-[33% 33%]` instances, we refer to all the instances based on instance *a280* with different number of requests, item-to-item compatibility characteristics, item-to-compartment flexibility characteristics, and compartment capacity of `cap-[33% 33%]`.

5.2 Summary of the computational results

In this section, we provide a discussion on the computational performance of our BPC algorithm. In general, our results show that our algorithm allows to solve 26,665 out of 28,710 instances to optimality within a reasonable computational time of 3,600 seconds. Tables 1–3 present the summarized computational results for the different cases of compartment capacity flexibility, item-to-compartment flexibility, and item-to-item compatibility, respectively. The first column reports the instance group (*Group*), e.g., *a280* or *brd14051*. The second column reports the number of instances per group and studied characteristic (*# Inst.*), e.g., in Table 1, there are 1,485 instances of group *a280* for each compartment capacity setting. Finally, for each of the studied characteristic settings, we report the number of instances solved to proven optimality (*# Opt.*) as well as the average computational time in seconds (*Sec.*).

Table 1: Summarized computational results for the PDPTWMC for different cases of compartment capacity flexibility

		cap-											
		[33% 33%]		[25% 50%]		[10% 50%]		[0% 50%]		[0% 75%]		[0% 100%]	
Group	# Inst.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.
a280	1,485	1,387	373.8	1,244	725.2	1,227	780.6	1,219	795.6	1,190	873.1	1,193	878.2
brd14051	825	820	38.0	817	78.4	817	95.7	813	99.1	811	111.4	811	123.0
d18512	825	802	140.6	809	104.0	806	108.6	807	106.9	807	114.9	808	111.4
fnl4461	825	803	193.1	784	299.3	773	356.2	768	383.8	759	439.9	757	440.1
nrw1379	825	809	105.6	808	126.2	809	132.4	807	144.7	800	183.7	800	180.4
Total	4,785	4,621	198.3	4,462	329.9	4,432	361.7	4,414	373.6	4,367	417.5	4,369	419.9

Our results in Table 1 reveal that instances with fixed compartment capacity are considerably easier to solve than instances with flexible compartment capacity. With fixed compartment capacity, 4,621 instances are solved to optimality with an average time of 198.3 seconds, compared to between 4,367 and 4,462 instances with an average time of more than 329.9 seconds with flexible compartment capacity. In addition, increasing the minimum compartment capacity allows to solve more instances to optimality and results in smaller average solution times. Likewise, decreasing the maximum compartment capacity also seems to yield easier to solve instances.

Table 2: Summarized computational results for the PDPTWMC for different cases of item-to-compartment flexibility

		i-to-c					
		1		2		3	
Group	# Inst.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.
a280	2,970	2,881	149.0	2,266	1,092.1	2,313	972.1
brd14051	1,650	1,643	21.1	1,621	142.3	1,625	109.5
d18512	1,650	1,649	6.0	1,589	188.1	1,601	149.1
fnl4461	1,650	1,636	58.8	1,500	532.0	1,508	465.4
nrw1379	1,650	1,646	17.8	1,586	236.8	1,601	181.9
Total	9,570	9,455	64.1	8,562	528.4	8,648	457.9

Our experiments indicate that increasing the item-to-compartment flexibility does not necessarily lead to instances that are harder to solve by our algorithm (see Table 2). While in the most restrictive case, i.e., `1 i-to-c`, our algorithm solves the most instances (9,455 out of 9,570) with the least average solution time (64.1 seconds), the results also show that a few additional instances are solved in less average solution time with `3 i-to-c` compared to `2 i-to-c`. These results can be explained by two factors: 1) the symmetry in the solutions, i.e., in which compartments items can be loaded, and 2) the symmetry in the labeling algorithm due to the number of extensions. The additional flexibility we get when going from `1 i-to-c` to `2 i-to-c` and from `2 i-to-c` to `3 i-to-c` increases the solution spaces because more loading possibilities exist for a vehicle route, which decreases the number of instances solved and increases the average computational time. On the other hand, solving the problem with `3 i-to-c` reduces the number of extensions compared to `2 i-to-c` because all compartments are symmetrical and the implemented symmetry-reduction techniques are effective.

Table 3: Summarized computational results for the PDPTWMC for different cases of item-to-item compatibility

		i-to-i									
		0%		25%		50%		75%		100%	
Group	# Inst.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.	# Opt.	Sec.
a280	1,782	1,753	190.3	1,544	640.4	1,483	743.3	1,412	908.3	1,268	1,206.4
brd14051	990	971	104.8	985	34.3	989	43.3	981	108.5	963	163.8
d18512	990	947	191.2	968	117.4	971	87.4	974	85.2	979	90.7
fnl4461	990	983	103.7	946	264.6	950	277.6	912	444.3	853	670.2
nrw1379	990	990	36.6	967	133.9	969	141.4	963	167.7	944	247.8
Total	5,742	5,644	134.3	5,410	293.6	5,362	325.5	5,242	420.8	5,007	576.5

Finally, while allowing more item-to-item compatibility is expected to increase the complexity of the problem as it produces, amongst others, more label extensions and symmetrical solutions, Table 3 indicates that the general trend seems to depend on the structure of the instances. For example, for the *a280* instances, the number of instances solved to optimality consistently decreases from 1,753 to 1,268 and the average solution time consistently increases from 190.3 seconds to 1,206.4 seconds when going from least (`0% i-to-i`) to most (`100% i-to-i`) flexible. Groups *fnl4461* and *nrw1379* also follow this expected trend. For the group *brd14051*, however, there does not seem to be a clear trend. Finally, we obtain unexpected results for the group *d18512* for which the number of instances solved to optimality consistently increases from 947 to 979 and the average computational time decreases from 191.2 to 90.7 when considering increasing item-to-item compatibility.

5.3 Analysis of the symmetry reduction techniques

In this section, we analyse the impact of the two symmetry reduction techniques described in Section 4.1.4, i.e., label extensions to empty compartments and label dominance of comparable compartments. Tables 4–6 present a comparison of the computational results for the PDPTWMC with and without the symmetry-reduction techniques. The tables report, for each group of instances and compartment-related attribute setting: the impact on the number of instances solved to proven optimality with and without the two symmetry reduction techniques (Δ *Opt.*) computed as # *Opt.* – # $\overline{Opt.}$, where # *Opt.* and # $\overline{Opt.}$ are the number of instances solved to optimality with and without the symmetry reduction techniques, respectively; and the average solution time ratio ($\overline{Sec.}$ / *Sec.*), where *Sec.* and $\overline{Sec.}$ are the computation time in seconds of the algorithm with and without the symmetry reduction techniques, respectively. For the latter, averages are computed as geometric means over the individual ratios for each instance.

Table 4: Summarized computational results for the PDPTWMC with and without the two symmetry reduction techniques for different cases of compartment capacity flexibility

	`cap-`											
	`[33% 33%]`		`[25% 50%]`		`[10% 50%]`		`[0% 50%]`		`[0% 75%]`		`[0% 100%]`	
Group	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.
a280	47	1.39	54	1.40	82	1.45	86	1.41	89	1.42	93	1.41
brd14051	2	1.12	6	1.25	12	1.26	9	1.37	11	1.41	13	1.41
d18512	3	1.16	8	1.25	7	1.36	8	1.32	11	1.41	11	1.40
fnl4461	12	1.35	27	1.50	35	1.50	40	1.54	43	1.60	43	1.67
nrw1379	6	1.29	9	1.41	18	1.51	22	1.49	25	1.53	27	1.60
Total	70	1.28	104	1.36	154	1.42	165	1.42	179	1.46	187	1.48

With compartment capacity flexibility (see Table 4), reducing the symmetry becomes more important when there is an increase in the compartment capacity flexibility. Recall from Table 1 that with increased compartment capacity flexibility the instances generally become harder to solve. Similar results are obtained with item-to-item compatibility as summarized in Table 6, where the symmetry-reduction techniques have a stronger impact when there are more compatible items. These instances are also generally harder to solve (see Table 3). Therefore, the relative impact of the proposed symmetry reduction is even higher for the more difficult instances, i.e., larger number of additional instances solved to optimality (Δ *Opt.*) paired with a smaller number of optima of the base algorithm (# $\overline{Opt.}$).

Table 5: Summarized computational results for the PDPTWMC with and without the two symmetry reduction techniques for different cases of item-to-compartment flexibility

	`i-to-c`					
	`1`		`2`		`3`	
Group	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.
a280	–7	0.79	–48	0.70	506	5.08
brd14051	0	0.93	–3	0.81	56	2.92
d18512	0	0.93	–1	0.81	49	2.99
fnl4461	0	0.87	–19	0.79	219	5.18
nrw1379	–1	0.89	–4	0.75	112	4.75
Total	–8	0.87	–75	0.76	945	4.18

The most interesting results are obtained with the item-to-compartment flexibility (see Table 5). Here, the impact of the symmetry-reduction techniques becomes apparent for the case of `3 i-to-c`. In fact, the algorithm with symmetry reduction solves 945 additional instances to proven optimality and is on average 4.18 times faster. With `1 i-to-c` and `2 i-to-c`, on the other hand, the symmetry reduction techniques seem to have a slightly negative effect. Recall that by definition of our instances, there are no comparable compartments for the cases `1 i-to-c` and `2 i-to-c`. Therefore, the symmetry-reduction techniques have no effect for these instances but require additional effort due to the overhead associated with them. This negative effect, however, seems to be limited.

Overall, the number of instances solved to proven optimality is 26,665 using our BPC algorithm with the symmetry-reduction techniques, which is an increase of 859 instances compared to the variant

Table 6: Summarized computational results for the PDPTWMC with and without the two symmetry reduction techniques for different cases of item-to-item compatibility

	`i-to-i`									
	`0%`		`25%`		`50%`		`75%`		`100%`	
Group	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.	Δ Opt.	$\overline{\text{Sec.}}$/Sec.
a280	55	1.44	85	1.36	99	1.45	105	1.42	107	1.41
brd14051	-1	1.16	0	1.23	20	1.34	16	1.40	18	1.37
d18512	3	1.22	5	1.23	1	1.30	17	1.40	22	1.44
fnl4461	11	1.38	28	1.49	46	1.60	60	1.52	55	1.63
nrw1379	10	1.29	16	1.35	13	1.43	26	1.61	42	1.70
Total	78	1.31	134	1.33	179	1.42	224	1.46	244	1.49

without symmetry reduction. While there is a slight decrease in the performance for `1 i-to-c` and `2 i-to-c`, we decided to keep the symmetry-reduction techniques in our algorithm for all instances for consistency reasons.

5.4 Analysis of the PDPTWMC attributes

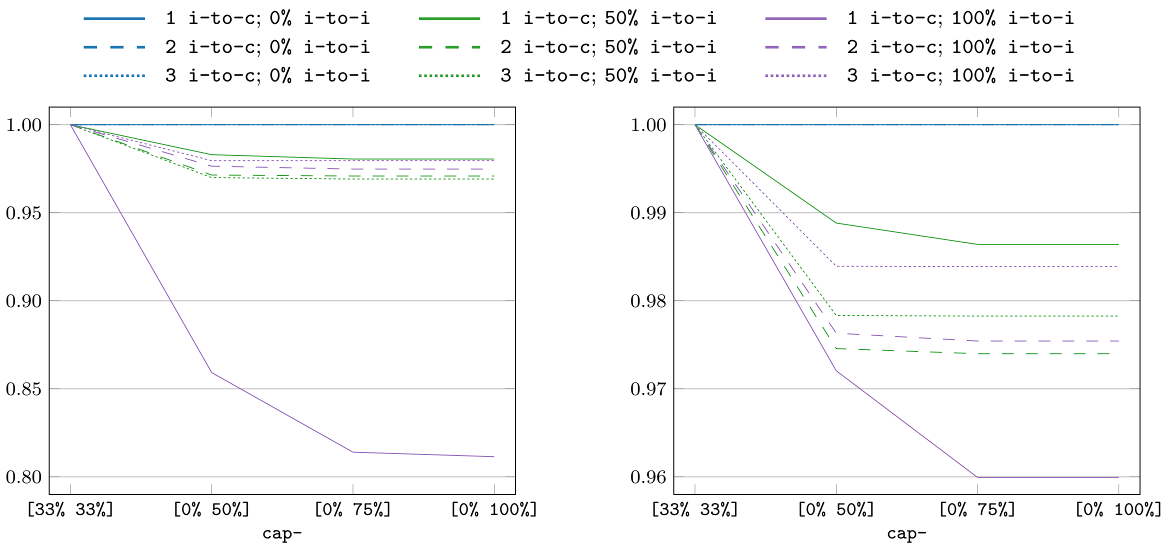
In this section, we analyse the impact of the different attributes on the PDPTWMC. We compare the results of the different compartment capacity flexibility, item-to-compartment flexibility, and item-to-item compatibility settings with respect to solution quality, i.e., number of vehicles and travel distance, and additional solution features (number of compartments used and number of simultaneous items in the compartments).

For each of the different attributes, we provide figures on the number of vehicles used and travel distance, see Figures 3–5. In each subfigure, the x-axis reports the tested values of the parameter under study and the y-axis reports the ratio between the specific parameter setting and a base parameter setting which differs according to each attribute. For the impact of the compartment capacity flexibility, the base parameter setting corresponds to the fixed compartment capacity `cap-[33% 33%]`. For the impact of the item-to-compartment flexibility, the base parameter setting corresponds to `1 i-to-c` while for the impact of the item-to-item compatibility, it corresponds to `0% i-to-i`. Each of the plots averages over those instances that are solved for all of the parameter settings of the corresponding attribute. Note that for the comparison of travel distances, we can only include instances with the same number of vehicles given that the primary objective consists of minimizing the number of vehicles. In addition, for conciseness and readability, in each figure we only show the plots of a subset of the solutions: in Figure 3, we report `0% i-to-i`, `50% i-to-i` and `100% i-to-i`; in Figure 4, we report `cap-[33% 33%]` and `cap-[0% 50%]`; in Figure 5, we report `cap-[33% 33%]`, `cap-[0% 50%]`, and `cap-[0% 100%]`. However, the results are representative also for the cases that are not shown in the figures. Detailed tables are reported at `https://wiwi.rptu.de/fgs/logistik/pdptwmc-detailedresults`. In addition, for each attribute under study, we analyze the number of compartments used as well as the average and the overall maximum number of items per compartment in the solutions (see Tables 7–9).

5.4.1 Impact of compartment capacity flexibility

In this subsection, we compare the results for different cases of compartment capacity flexibility. The analysis here is restricted to `cap-[33% 33%]`, `cap-[0% 50%]`, `cap-[0% 75%]`, and `cap-[0% 100%]`. The findings for `cap-[0% 50%]`, `cap-[10% 50%]`, and `cap-[25% 50%]`, however, are consistent.

Figure 3 illustrates that the number of vehicles and the travel distance decreases when the maximum compartment capacity increases, i.e., when there is more compartment capacity flexibility. The number of vehicles used and the distance traveled decreases by up to 18.9% and 4.0%, respectively, when comparing the different cases of $Q^{min} = 0$ with the fixed vehicle capacity. Furthermore, for both number of vehicles and travel distance, allowing more compartment capacity flexibility has by far



(a) Number of vehicles **(b) Travel distance**

Figure 3: Difference in average number of vehicles and travel distance for different cases of compartment capacity flexibility

the largest impact with `1 i-to-c` and `100% i-to-i`. This can be explained because with `1 i-to-c` there is no flexibility in the assignment of items to compartments at all and all items are compatible, hence, items can go simultaneously in the same compartment when the capacity allows for it. For fully incompatible item-to-item compatibility (`0% i-to-i`), the compartment capacity flexibility does not impact the number of vehicles used and the travel distance. This is expected, because no pair of items can be simultaneously in the same compartment.

Table 7 shows that the average number of compartments used is highest (2.6) with fixed compartment capacity, i.e., `cap-[33% 33%]`, and when increasing the maximum compartment capacity the average number of compartments used decreases. In addition, the maximum number of items per compartment is lowest (1.67 on average and 2 as a maximum) with fixed compartment capacity, and when increasing the maximum compartment capacity, there are more items per compartment.

Table 7: Characteristics of the results for the PDPTWMC for different cases of compartment capacity flexibility

	cap-			
	[33% 33%]	[0% 50%]	[0% 75%]	[0% 100%]
Average number of compartments used	2.6	2.5	2.4	2.4
Average maximum number of items per compartment	1.67	1.81	1.98	2.01
Maximum number of items per compartment	2	4	5	5

5.4.2 Impact of item-to-compartment flexibility

In this subsection, we compare the results for different cases of item-to-compartment flexibility.

Figure 4 shows that a large number of vehicles and travel distance can be saved by having partial item-to-compartment flexibility (`2 i-to-c`) compared to no item-to-compartment flexibility (`1 i-to-c`). In fact, the number of vehicles and the travel distance decrease by up to 22.0% and 9.2%, respectively. When going from partial item-to-compartment flexibility (`2 i-to-c`) to full item-to-compartment flexibility (`3 i-to-c`), the additional gain is limited.

Table 8 shows that the average number of compartments used increases when going from `1 i-to-c` to `2 i-to-c` but stabilizes when going from `2 i-to-c` to `3 i-to-c`. Finally, the maximum number of

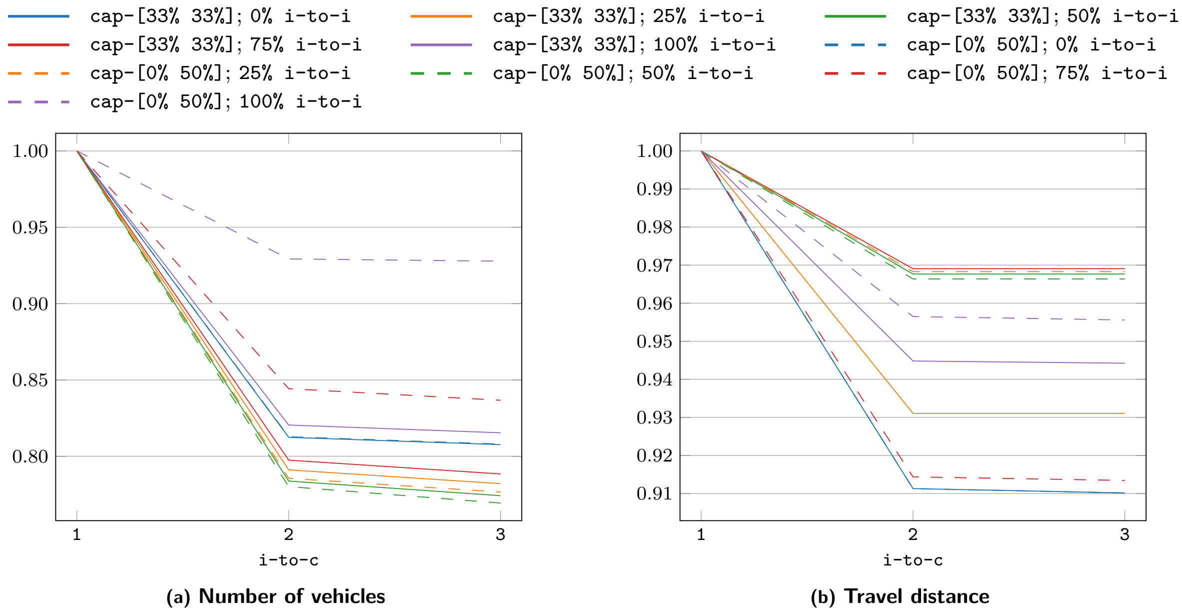


Figure 4: Difference in average number of vehicles and travel distance for different cases of item-to-compartment flexibility

items per compartment increases with an increasing item-to-compartment flexibility, when considering the average over the instances, but remains stable when considering the maximum.

Table 8: Characteristics of the results for the PDPTWMC for different cases of item-to-compartment flexibility

	i-to-c		
	1	2	3
Average number of compartments used	2.4	2.6	2.6
Average maximum number of items per compartment	1.78	1.87	1.88
Maximum number of items per compartment	5	5	5

5.4.3 Impact of item-to-item compatibility

In this subsection, the results for different cases of item-to-item compatibility are compared.

Figure 5 reveals that the average number of vehicles and travel distance decreases when the percentage of compatible items increases. The number of vehicles used and the distance traveled is decreased by up to 25.6% and 7.0%, respectively, when comparing `100% i-to-i` to `0% i-to-i`. The largest decrease in vehicles used and distance traveled is found for `1 i-to-c` (no item-to-compartment flexibility). Hence, adding more flexibility in item-to-item compatibility has a larger impact when there is no item-to-compartment flexibility, whereas adding more item-to-item compatibility has a smaller impact when there is already partial or full item-to-compartment flexibility.

Table 9 shows that when more items are compatible the average number of compartments used decreases, e.g., 2.6 and 2.4 compartments are used on average with `0% i-to-i` and `100% i-to-i`, respectively. In addition, the maximum number of items per compartment (both average and maximum over the instances) increases significantly as the compatibility between items increases. These results are expected because with increased compatibility between items, more items can be loaded simultaneously within a single compartment, thus, decreasing the average number of compartments used and increasing the number of items per compartment.

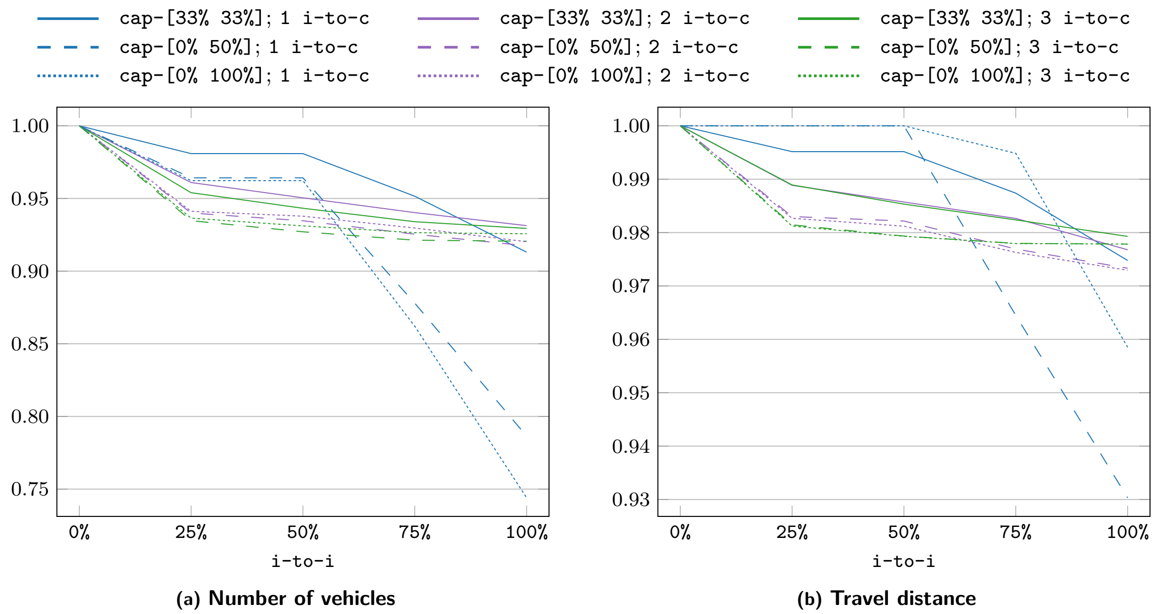


Figure 5: Difference in average number of vehicles and travel distance for different cases of item-to-item compatibility

Table 9: Characteristics of the results for the PDPTWMC for different cases of item-to-item compatibility

	i-to-i				
	0%	25%	50%	75%	100%
Average number of compartments used	2.6	2.6	2.5	2.5	2.4
Average maximum number of items per compartment	1	1.88	1.91	2.11	2.38
Maximum number of items per compartment	1	2	2	4	5

6 Conclusions

This paper introduces a new variant of PDPs coined the PDPTWMC, which generalizes the PDPTW to vehicles with multiple compartments. We consider three different compartment-related attributes, namely compartment capacity flexibility, item-to-compartment flexibility and item-to-item compatibility. The PDPTW is a special case of the PDPTWMC with fully flexible compartment capacities, full item-to-compartment flexibility, and full item-to-item compatibility. This paper contributes to the VRPMC literature by adapting the compartment-related attributes as highlighted by Ostermeier et al. (2021) to the PDP context and by developing a unified BPC algorithm that can tackle all combinations of these attributes. The pricing problems of the BPC are solved by means of a bidirectional labeling algorithm which incorporates two techniques to reduce the symmetry in the label extensions and in the dominance. Furthermore, we have introduced new benchmark instances for the PDPTWMC by adapting the C2 instances proposed by Cherkesly et al. (2016) for the PDPTWMS.

Extensive computational results show that our algorithm can solve instances with up to 75 requests within one hour. In terms of computation time and number of optimal solutions, instances become easier to solve when the flexibility in the compartment capacity decreases, with the extreme case of fixed compartment capacity being the easiest. For the item-to-compartment flexibility, our results show that most instances are solved for the most restrictive case in which items are compatible with a single compartment only (no flexibility). However, more instances are solved when items are compatible with three compartments (fully flexible) compared to two compartments. This somewhat counter-intuitive result can be explained by the symmetry-reduction techniques that, by definition of the instances, have no effect in the latter case. For the item-to-item compatibility, the results indicate that the performance of the algorithm seems to be related to the geography of the instances rather than the compatibility of the items. Finally, our results show that the symmetry-reduction techniques accelerate our algorithm especially when there are more comparable compartments and items (more item-to-compartment flexibility and item-to-item compatibility).

Finally, we have derived extensive managerial insights. In terms of compartment capacity, the number of vehicles and the travel distance is highest when it is fixed, and solution quality improves when the flexibility increases. With item-to-compartment flexibility, the number of vehicles and travel distance can be significantly reduced with `2 i-to-c` compared with `1 i-to-c`. Going from `2 i-to-c` to `3 i-to-c` has only a very small impact. Finally, solution quality improves with additional item-to-item compatibility. This effect is largest when there is no item-to-compartment flexibility, whereas the impact is smaller when there is item-to-compartment flexibility.

Appendix A Validity of the forward dominance criteria

Proposition 1. Conditions (8) constitute a valid dominance rule for forward labeling if the arc reduced cost $\tilde{c}^f_{ij}$ satisfy the DTI.

Proof. The proof applies the standard arguments first used by Dumas, Desrosiers, and Soumis (1991) to show that for each extension E_2 of $R(F_2)$ to the feasible $0-(2n+1)$-path R_2, there exists an extension E_1 of $R(F_1)$ to the feasible $0-(2n+1)$-path R_1 with better or equal reduced cost. More precisely, let $E_1 := E_2 \backslash \{i \in D | i-n \in O(F_2) \backslash O(F_1)\}$ be the subpath of E_2 in which all visits to delivery nodes whose requests are open for label F_2 but not for label F_1 are removed. Then, if R_2 is feasible, Condition (8b) ensures time-window feasibility of R_1 (recall that travel times satisfy the triangle inequality), Conditions (8d) and (8e) ensure pairing and precedence of R_1, and Conditions (8e) and (8f) ensure feasibility of R_1 with respect to item-to-item compatibility, compartment and vehicle capacity, and item-to-compartment compatibility. Finally, Condition (8c) together with the DTI guarantee $\tilde{c}_{R_1} \leq \tilde{c}_{R_2}$ for the reduced costs of R_1 and R_2. □

Appendix B Detailed backward labeling algorithm

In this section, we give additional details on the backward labeling algorithm that are not already included in Section 4.1.2.

Backward label resources. The complete backward label resources and their meaning are:

- $\eta(B)$, the first visited vertex of the partial path;
- $t(B)$, the latest feasible start of the service time at vertex $\eta(B)$;
- $c(B)$, the accumulated reduced cost;
- $S(B)$, the set of completed requests;
- $l^m(B)$, the load in compartment $m \in M$;
- $O^m(B)$, the items (open requests) in compartment $m \in M$;
- $\psi^m(B)$, the required capacity of compartment $m \in M$.

Backward label dominance. A label B_1 dominates label B_2 if

$$\eta(B_1) = \eta(B_2), \tag{19a}$$
$$t(B_1) \geq t(B_2), \tag{19b}$$
$$c(B_1) \leq c(B_2), \tag{19c}$$
$$S(B_1) \subseteq S(B_2), \tag{19d}$$

$$O^m(B_1) \subseteq O^m(B_2), \qquad \forall m \in M, \tag{19e}$$

$$\psi^m(B_1) \leq \psi^m(B_2), \qquad \forall m \in M. \tag{19f}$$

Proposition 2. Conditions (19) constitute a valid dominance rule for backward labeling if the arc reduced cost $\tilde{c}^b_{ij}$ satisfy the PTI.

Proof. The proof is completely analog to the forward case, utilizing the PTI instead of the DTI to guarantee $\tilde{c}_{R_1} \leq \tilde{c}_{R_2}$. □

Appendix C Attribute settings

Table 10 presents an overview of the tested scenarios for the compartment-related attributes and details their computation from the data specified in the instances.

Table 10: Summary of the tested item-to-item compatibility, compartment capacity flexibility, and item-to-compartment flexibility cases

Name	Description
Item-to-item compatibility	
`0% i-to-i`	$u_{ij} = 0, \forall i, j \in P, i \neq j$
`25% i-to-i`	$\forall i, j \in P, i \neq j \quad u_{ij} = \begin{cases} 0 & \mu_i, \mu_j \in \{1, \dots, 6\} \\ 0 & \mu_i, \mu_j \in \{7, \dots, 12\} \\ 0 & \mu_i \in \{1,2,3\}, \mu_j \in \{7,8,9\} \\ 0 & \mu_i \in \{4,5,6\}, \mu_j \in \{10,11,12\} \\ 1 & \text{Otherwise} \end{cases}$
`50% i-to-i`	$\forall i, j \in P, i \neq j \quad u_{ij} = \begin{cases} 0 & \mu_i, \mu_j \in \{1, \dots, 6\} \\ 0 & \mu_i, \mu_j \in \{7, \dots, 12\} \\ 1 & \text{Otherwise} \end{cases}$
`75% i-to-i`	$\forall i, j \in P, i \neq j \quad u_{ij} = \begin{cases} 0 & \mu_i, \mu_j \in \{1,2,3\} \\ 0 & \mu_i, \mu_j \in \{4,5,6\} \\ 0 & \mu_i, \mu_j \in \{7,8,9\} \\ 0 & \mu_i, \mu_j \in \{10,11,12\} \\ 1 & \text{Otherwise} \end{cases}$
`100% i-to-i`	$u_{ij} = 1, \forall i, j \in P, i \neq j$
Compartment capacity flexibility	
`cap-[33% 33%]`	$Q^{min} = Q^{max} = \lfloor Q/3 \rfloor$
`cap-[0% 50%]`	$Q^{min} = 0,\ Q^{max} = \lfloor 50\%Q \rfloor$
`cap-[10% 50%]`	$Q^{min} = \lceil 10\%Q \rceil,\ Q^{max} = \lfloor 50\%Q \rfloor$
`cap-[25% 50%]`	$Q^{min} = \lceil 25\%Q \rceil,\ Q^{max} = \lfloor 50\%Q \rfloor$
`cap-[0% 75%]`	$Q^{min} = 0,\ Q^{max} = \lfloor 75\%Q \rfloor$
`cap-[0% 100%]`	$Q^{min} = 0,\ Q^{max} = Q$
Item-to-compartment flexibility	
`1 i-to-c`	$\forall i \in P, m \in M \quad b_{im} = \begin{cases} 1 & \text{if } \mu_i = \{1,2,3,4\} \text{ and } m = 1 \\ 1 & \text{if } \mu_i = \{5,6,7,8\} \text{ and } m = 2 \\ 1 & \text{if } \mu_i = \{9,10,11,12\} \text{ and } m = 3 \\ 0 & \text{Otherwise} \end{cases}$
`2 i-to-c`	$\forall i \in P, m \in M \quad b_{im} = \begin{cases} 1 & \text{if } \mu_i = \{1,2,3,4\} \text{ and } m = \{1,2\} \\ 1 & \text{if } \mu_i = \{5,6,7,8\} \text{ and } m = \{2,3\} \\ 1 & \text{if } \mu_i = \{9,10,11,12\} \text{ and } m = \{1,3\} \\ 0 & \text{Otherwise} \end{cases}$
`3 i-to-c`	$\forall i \in P, m \in M \quad b_{im} = 1$

References

Al-Yasiry AM, 2020 Rich variants of the vehicle routing problem. Ph.D. thesis, The University of Queensland.

Avella P, Boccia M, Sforza A, 2004 Solving a fuel delivery problem by heuristic and exact approaches. European Journal of Operational Research 152(1):170–179.

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

Battarra M, Cordeau JF, Iori M, 2014 Chapter 6: Pickup-and-delivery problems for goods transportation. Toth P, Vigo D, eds., Vehicle Routing: Problems, Methods, and Applications, Second Edition, 161–191 (SIAM).

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

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

Cherkesly M, Gschwind T, 2022 The pickup and delivery problem with time windows, multiple stacks, and handling operations. European Journal of Operational Research 301(2):647–666.

Christiansen M, Fagerholt K, Rachaniotis NP, Stålhane M, 2017 Operational planning of routes and schedules for a fleet of fuel supply vessels. Transportation Research Part E: Logistics and Transportation Review 105:163–175.

Coelho LC, Laporte G, 2015 Classification, models and exact algorithms for multi-compartment delivery problems. European Journal of Operational Research 242(3):854–864.

Cornillier F, Boctor FF, Laporte G, Renaud J, 2008 An exact algorithm for the petrol station replenishment problem. Journal of the Operational Research Society 59(5):607–615.

Costa L, Contardo C, Desaulniers G, 2019 Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science 53(4):946–985.

Côté JF, Archetti C, Speranza MG, Gendreau M, Potvin JY, 2012 A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks. Networks 60(4):212–226.

Deng F, Qin H, Li J, Cheng C, 2022 The pickup and delivery problem with time windows and incompatibility constraints in cold chain transportation. Transportation Science 57(2):444–462.

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

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

Factorovich P, Méndez-Díaz I, Zabala P, 2020 Pickup and delivery problem with incompatibility constraints. Computers & Operations Research 113:104805.

Feillet D, Dejax P, Gendreau M, Gueguen C, 2004 An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44(3):216–229.

Gschwind T, Irnich S, Rothenbächer AK, Tilk C, 2018 Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research 266(2):521–530.

Henke T, Speranza MG, Wäscher G, 2015 The multi-compartment vehicle routing problem with flexible compartment sizes. European Journal of Operational Research 246(3):730–743.

Henke T, Speranza MG, Wäscher G, 2019 A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. Annals of Operations Research 275(2):321–338.

Heßler K, 2021 Exact algorithms for the multi-compartment vehicle routing problem with flexible compartment sizes. European Journal of Operational Research 294(1):188–205.

Heßler K, Irnich S, 2022 Partial dominance in branch-price-and-cut for the basic multicompartment vehicle-routing problem. INFORMS Journal on Computing 35(1):50–65.

Hsu YC, Walteros JL, Batta R, 2020 Solving the petroleum replenishment and routing problem with variable demands and time windows. Annals of Operations Research 294(1):9–46.

Huang B, Tang L, Baldacci R, Wang G, Sun D, 2023 A metaheuristic algorithm for a locomotive routing problem arising in the steel industry. European Journal of Operational Research 308(1):385–399.

Hübner A, Ostermeier M, 2019 A multi-compartment vehicle routing problem with loading and unloading costs. Transportation Science 53(1):282–300.

Irnich S, Desaulniers G, 2005 Chapter 2: Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon M, eds., Column Generation, 33–65 (Springer).

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

Jetlund AS, Karimi I, 2004 Improving the logistics of multi-compartment chemical tankers. Computers & Chemical Engineering 28(8):1267–1283.

Kiilerich L, Wøhlk S, 2018 New large-scale data instances for CARP and new variations of CARP. INFOR: Information Systems and Operational Research 56(1):1–32.

Lahyani R, Coelho LC, Khemakhem M, Laporte G, Semet F, 2015 A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. Omega 51:1–10.

Laporte G, Nobert Y, 1983 A branch and bound algorithm for the capacitated vehicle routing problem. Operations-Research-Spektrum 5:77–85.

Martins S, Ostermeier M, Amorim P, Hübner A, Almada-Lobo B, 2019 Product-oriented time window assignment for a multi-compartment vehicle routing problem. European Journal of Operational Research 276(3):893–909.

Mirzaei S, Wøhlk S, 2019 A branch-and-price algorithm for two multi-compartment vehicle routing problems. EURO Journal on Transportation and Logistics 8(1):1–33.

Mor A, Speranza MG, 2020 Vehicle routing problems over time: a survey. 4OR 18:129–149.

Ostermeier M, Henke T, Hübner A, Wäscher G, 2021 Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions. European Journal of Operational Research 292(3):799–817.

Ostermeier M, Martins S, Amorim P, Hübner A, 2018 Loading constraints for a multi-compartment vehicle routing problem. OR Spectrum 40(4):997–1027.

Ousmane B, Moustapha D, Adama C, 2021 Genetic algorithm for the pick-up and delivery problem with time window by multi-compartment vehicles. Transport and Telecommunication 22(3):343–352.

Pereira AH, Mateus GR, Urrutia SA, 2022 Valid inequalities and branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks. European Journal of Operational Research 300(1):207–220.

Pereira AH, Urrutia S, 2018 Formulations and algorithms for the pickup and delivery traveling salesman problem with multiple stacks. Computers & Operations Research 93:1–14.

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

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

Sadykov R, Uchoa E, Pessoa A, 2021 A bucket graph–based labeling algorithm with application to vehicle routing. Transportation Science 55(1):4–28.

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

Yahyaoui H, Kaabachi I, Krichen S, Dekdouk A, 2020 Two metaheuristic approaches for solving the multi-compartment vehicle routing problem. Operational Research 20(4):2085–2108.

Zbib H, Laporte G, 2020 The commodity-split multi-compartment capacitated arc routing problem. Computers & Operations Research 122:104994.