Les Cahiers du GERAD ISSN: 0711–2440

A branch-price-and-cut algorithm for multi-compartment pickup and delivery problems with incompatibility constraints and cleaning operations

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

G–2025–77

November 2025

A branch-price-and-cut algorithm for multi-compartment pickup and delivery problems with incompatibility constraints and cleaning operations

Marjolein Aerts-Veenstra [a]

Marilène Cherkesly [b, c]

Timo Gschwind [d]

[a] *Department of Operations, Faculty of Economics and Business, University of Groningen, 9700 AB Groningen, The Netherlands*

[b] *Département d'analytique, opérations et technologies de l'information, Université du Québec à Montréal, Montréal (Qc), Canada, 2X 1L7*

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

[d] *Chair of Logistics, School of Business and Economics, RPTU Kaiserslautern-Landau, 677663 Kaiserslautern, Germany*

Marilene.Cherkesly@gerad.ca

November 2025
Les Cahiers du GERAD
G–2025–77

Abstract : In this paper, we study the pickup and delivery problem with time windows, multiple compartments, incompatibility constraints and cleaning operations (PDPTWMCI). In the PDPTWMCI, four types of item incompatibilities are defined and studied. We also allow cleaning operations to remove some existing incompatibilities. The following four types of incompatibilities are considered: incompatible items cannot be loaded 1) simultaneously in the same compartment, 2) simultaneously in the same vehicle, 3) simultaneously and subsequently in the same compartment, and 4) simultaneously and subsequently in the same vehicle. For the last two types of incompatibilities, if the items are loaded subsequently, i.e., the first item is loaded and unloaded before the second item is loaded, cleaning operations can remove existing incompatibilities on the route. Each cleaning operation is done for a subset of empty compartments in a vehicle, and allows to remove all existing incompatibilities from these compartments. A set of dedicated cleaning stations are available and the number of cleaning operations is not restricted. We propose a branch-price-and-cut (BPC) algorithm to solve the PDPTWMCI. In the proposed BPC, the pricing problem is solved using a forward labeling algorithm. We use acceleration techniques to speed up the algorithm. The resulting algorithm yields a variant of selective pricing. We generate benchmark instances for the PDPTWMCI and conduct an extensive numerical campaign to assess the performance of our algorithm and to derive relevant managerial insights for the PDPTWMCI.

Keywords : Multi-compartment vehicle routing; column generation; selective pricing

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 times windows, multiple compartments, incompatibility constraints and cleaning operations (PDPTWMCI). In the PDPTWMCI, a set of items is given, where an item has a given load, and needs to be transported from its pickup location to its delivery location. Each pickup and delivery location has a predefined time window during which the service must start. A fleet of homogeneous capacitated vehicles with multiple compartments is available at the depot to pickup and deliver the set of items. In addition, items can be incompatible with other items either if they are loaded in the same compartment, or if they are loaded in the same vehicle (independently of the compartment). They can either be incompatible when they are simultaneously in the vehicle, or throughout the route (simultaneously and subsequently). Therefore, four types of incompatibilities are studied: incompatible items can not be loaded 1) simultaneously in the same compartment, 2) simultaneously in the same vehicle, 3) simultaneously and subsequently in the same compartment, and 4) simultaneously and subsequently in the same vehicle. These incompatibilities are referred to as: 1) *compartment-simultaneous incompatibility*, 2) *vehicle-simultaneous incompatibility*, 3) *compartment-simultaneous-and-subsequent incompatibility*, and 4) *vehicle-simultaneous-and-subsequent incompatibility*. For the last two types of incompatibilities, if the items are loaded subsequently, i.e., the first item is loaded and unloaded before the second item is loaded, cleaning operations can remove existing incompatibilities on the route. Moreover, we consider both symmetric and asymmetric incompatibilities for these two types. If the incompatibility is symmetric, this implies that the items cannot be loaded simultaneously or subsequently without cleaning independent of the order in which they have been picked up and delivered. If the incompatibility is asymmetric, i.e., the first item is incompatible with the second item, but the second item is compatible with the first item, this implies that the first item could be loaded and unloaded before the second item is loaded without requiring cleaning, but not the opposite. In the remainder of this section, we refer to incompatibilities as being symmetric for readability purposes. Each cleaning operation is done for a subset of empty compartments in a vehicle, and allows to remove all existing incompatibilities from these compartments. Consequently, two *compartment-simultaneous-and-subsequent incompatible* items can be loaded in the same compartment subsequently if the compartment in which the first item was loaded is cleaned after delivering that item and before picking up the second item. Similarly, two *vehicle-simultaneous-and-subsequent incompatible* items can be loaded subsequently if all the compartments in the vehicle are cleaned after delivering that item (in a single cleaning, or through multiple cleanings) and before picking up the second item. Considering cleaning in such a manner allows more flexibility than a restricted policy where only the complete vehicle can be cleaned at once, which can only occur when all compartments of the vehicle are empty. A set of dedicated cleaning stations are available and the number of cleaning operations is not restricted. Cleaning stations have predefined opening hours (time windows) during which cleaning can start. Cleaning of compartments is associated with a cleaning time. The problem consists of determining a set of least-cost (i.e., fixed vehicle cost and variable transportation cost) vehicle routes which respect the pairing and precedence constraints for pickup and delivery, the time window constraints, the capacity constraints of the vehicle and the compartments, as well as the incompatibility constraints between the items.

Table 1 summarizes the different types of incompatibilities, including "no incompatibility", as well as their restrictions for loading items. For each type of incompatibility, we report under which conditions (if any) incompatible items can be loaded in the same or in different compartments of the vehicle simultaneously (*Sim.*) and subsequently (*Sub.*). We indicate by (✓) a possible loading option. We also indicate when the loading option is possible after cleaning a compartment (*C-Clean*) or the entire vehicle (*V-Clean*). Note that the cleaning operations must be conducted after the first incompatible item has been delivered and before picking up the second incompatible item. Finally, we indicate by (✗) an infeasible loading option, even after cleaning operations.

In practice, the PDPTWMCI is an important problem occurring in different applications (see Coelho et al., 2016, for a survey on different routing applications). Incompatibilities can arise, for

Table 1: Summary of loading restrictions according to the different types of incompatibilities.

	Same compartment		Different compartments	
Type of incompatibility	Sim.	Sub.	Sim.	Sub.
Compartment-simultaneous	✗	✓	✓	✓
Vehicle-simultaneous	✗	✓	✗	✓
Compartment-simultaneous-and-subsequent	✗	C-Clean	✓	✓
Vehicle-simultaneous-and-subsequent	✗	V-Clean	✗	V-Clean
No incompatibility	✓	✓	✓	✓

example, when transporting hazardous material as there are strict rules for the transportation of hazardous products that impose incompatibilities between products (Factorovich et al., 2020), and in food distribution due to different temperature requirements for products and different product properties, e.g., the simultaneous or subsequent transportation of food and chemicals is regularly not allowed (Deng et al., 2023). In these contexts, considering cleaning is important to allow more flexibility. Cleaning operations are considered, for example, when transporting livestock (Oppen et al., 2010), olive oil (Lahyani et al., 2015), and milk (Sethanan and Pitakaso, 2016).

The PDPTWMCI has not been previously studied in the literature, but is related to the pickup and delivery problem with time windows and multiple compartments (PDPTWMC, see Aerts-Veenstra et al., 2024), which is a generalization of the pickup and delivery problem with time windows (Ropke and Cordeau, 2009) to compartmentalized vehicles (Ostermeier et al., 2021). In the PDPTWMC, three compartment-related attributes are studied, i.e., compartment capacity flexibility which allow the capacity of a compartment to be fixed or flexible, item-to-compartment flexibility which considers that items can be incompatible with compartments, and item-to-item compatibility which considers incompatible items.

In the vehicle routing literature (see Archetti et al., 2025; Mor and Speranza, 2020, for surveys on vehicle routing problems, VRPs), the concept of incompatibilities is understudied. With single compartment vehicles, incompatible items have been considered in variants of the VRP with different attributes such as multi-trips and heterogeneous vehicles. In these problems, incompatible items cannot be loaded simultaneously in the vehicle (see, e.g., Alcaraz et al., 2019; Battarra et al., 2009; Bernardino and Paias, 2022; Ceselli et al., 2009; Gendreau et al., 2016; Manerba and Mansini, 2015; Rabbani et al., 2018). This usually increases costs. Therefore, multi-compartment vehicles usually allow to decrease the costs while considering item incompatibilities. In the literature, incompatibilities between items in multi-compartment vehicles imply that two incompatible items cannot be loaded simultaneously in the same compartment (see, e.g., Agra et al., 2013; Christiansen et al., 2015; Derigs et al., 2011; Foss et al., 2016). When considering incompatible items, cleaning operations, requiring additional time or cost, are often allowed to remove existing incompatibilities. Many authors consider multi-compartment vehicles over multiple periods or multiple routes, and cleaning is then allowed between periods or routes. Some authors consider that items can soil the entire vehicle (i.e., all the compartments) and that the full vehicle must be cleaned to remove incompatibilities. Oppen et al. (2010) consider a multi-trip VRP with incompatibility between items. They also consider that some items can soil the vehicle, and cleaning is required between trips when such items have been picked up. Other authors consider that items can soil a specific compartment, and that cleaning the compartment allows to remove all existing incompatibilities from that compartment. Lahyani et al. (2015) consider a multi-period VRP with different quality of products. Therefore, products of higher quality cannot be loaded after lower quality products without cleaning, but the opposite is allowed. Sethanan and Pitakaso (2016) consider a multi-trip VRP where all items are incompatible, and cleaning is performed at the depot.

In the context of PDPs (see Battarra et al., 2014; Berbeglia et al., 2007, for surveys on PDPs), only a few papers consider item incompatibilities and cleaning operations. These attributes are studied within many-to-many PDPs in ship routing problems (SRPs) (see, e.g., Fagerholt and Christiansen, 2000; Hvattum et al., 2009; Santosa et al., 2016). However, SRPs incorporate problem characteristics quite different from truck routing, such as planning for only a few stops without round trips (i.e., not coming back to the depot), resulting in far less routing combinations (Ostermeier et al., 2021). Therefore, we now focus on papers within the one-to-one PDP setting as studied in our paper. In this context, only four papers consider item incompatibilities and none of them consider cleaning operations. In the context of single compartment PDPs, three papers (Deng et al., 2023; Factorovich et al., 2020; Huang et al., 2023) study item incompatibilities. In these papers, incompatible items can never be loaded simultaneously in the vehicle. With multi-compartment vehicles, one paper specifically considers item incompatibilities. In the paper of Aerts-Veenstra et al. (2024), incompatible items can never be loaded simultaneously in the same compartment.

In comparison with the previous literature, this paper expands the concepts of incompatibilities and cleaning in the context of pickup and delivery problems (PDPs). We define and study four types of item incompatibilities (item-to-item compatibility); considering that incompatible items cannot be loaded simultaneously and subsequently (without cleaning) either in the same compartment or the same vehicle. Moreover, we define cleaning operations in the context of multi-compartment vehicles to allow cleaning within a route (at one or multiple cleaning stations), and the resulting flexibility gained from cleaning allows to remove existing incompatibilities. These problem attributes complicate the problem setting and require state-of-the-art algorithms to solve it. Therefore, we derive a branch-price-and-cut (BPC) algorithm to solve the PDPTWMCI exactly. BPC algorithms are the leading exact solution methods for many VRP variants including PDPs (see Costa et al., 2019, for a survey on branch-and-price algorithms for VRPs). In the context of multi-compartment vehicles, few exact algorithms have been proposed and many rely on column generation and branch-and-price (see e.g., Aerts-Veenstra et al., 2024; Ceselli et al., 2009; Cherkesly and Gschwind, 2022; Deng et al., 2023; Gendreau et al., 2016). In the literature on cleaning, only a few authors use branch-and-price algorithms (Oppen et al., 2010) for problems where cleaning is mandatory between trips. Moreover, branch-and-price algorithms have proven effective for routing problems with characteristics similar to cleaning stations, referred to as VRPs with intermediate stops (see Schiffer et al., 2019, for an overview). The latter include VRPs with intermediate stops for refueling, such as the recharging of electric vehicles (e.g., Desaulniers et al., 2016; Parmentier et al., 2023) or drones (e.g., Ermağan et al., 2022) en route at dedicated recharging stations, and VRPs with intermediate stops for replenishing and unloading, such as the multidepot VRP with interdepot routes (e.g., Muter et al., 2014) which allows vehicles to be replenished at intermediate depots during the course of a route.

The contributions of this paper are fourfold. First, we introduce a new PDP variant, coined PDPTWMCI, which studies four types of item incompatibility, and considers cleaning operations to remove existing incompatibilities. Second, we model the PDPTWMCI using a set-partitioning formulation and propose an exact BPC algorithm where the pricing problem is solved by means of a monodirectional forward labeling algorithm. Our BPC algorithm differs from existing approaches in the way that we handle incompatibilities (label extensions, additional resources, and dominance). We propose different acceleration techniques and propose a variant of selective pricing that may exclude feasible non-optimal routes. Third, we generate benchmark instances for the PDPTWMCI. Fourth, we conduct an extensive numerical campaign to assess the performance of our algorithm and to derive relevant managerial insights for the PDPTWMCI. Our results also highlight that resorting to a bidirectional version of our labeling algorithm performs worse than our proposed monodirectional version due to additional resources in the labeling.

The outline of the remainder of this paper is as follows. In Section 2, we formally define the problem and provide the set-partitioning formulation. In Section 3, we detail the BPC algorithm developed to solve the PDPTWMCI. In Section 4, we create benchmark instances for the PDPTWMCI and provide computational results. Final conclusions are drawn in Section 5.

2 Problem description

The PDPTWMCI is defined on a directed graph $G = (V, A)$, where $V = N \cup W$ is the set of nodes and A is the set of arcs. The set $N = \{0, 1, ..., n, n+1, ..., 2n, 2n+1\}$ consists of the origin and destination depot, denoted by 0 and $2n+1$, the set of pickup nodes denoted by $P = \{1, ..., n\}$ and the set of delivery nodes denoted by $D = \{n+1, ..., 2n\}$. The set W denotes the set of cleaning stations. Each item i is associated with a pickup location $i \in P$ (also denoted by i^+), where the item is loaded in the vehicle, and a delivery location $n+i \in D$ (also denoted by i^-), where the item is unloaded from the vehicle. We also refer to the set P as the set of items. For each node $i \in V$, a demand q_i is given, where $q_i = 0, \forall i \in W \cup \{0, 2n+1\}$, $q_i \geq 0, \forall i \in P$, and $q_i = -q_{i-n}, \forall i \in D$. With each node $i \in V$ are associated a service time s_i, where $s_i = 0, \forall i \in W \cup \{0, 2n+1\}$ and $s_i \geq 0, \forall i \in P \cup D$, and a time window $[\underline{w}_i, \overline{w}_i]$ which corresponds to the earliest and latest time at which either the service or the cleaning needs to start. Four binary parameters u^{c1}_{ij}, u^{v1}_{ij}, u^{c2}_{ij}, and u^{v2}_{ij} are associated with each pair of items $i, j \in P,\ i \neq j$ indicating if item j is incompatible with item i. The binary parameters are equal to one to indicate an incompatibility, and zero otherwise. In the following, we define the cases when these parameters are equal to one:

- u^{c1}_{ij} is equal to one to indicate that item j cannot be loaded simultaneously in the same compartment as item i (compartment-simultaneous incompatibility);
- u^{v1}_{ij} is equal to one to indicate that item j cannot be loaded simultaneously in the same vehicle as item i (vehicle-simultaneous incompatibility);
- u^{c2}_{ij} is equal to one to indicate that if item i is loaded before item j, then item j cannot be loaded in the same compartment simultaneously and subsequently, unless the compartment has been cleaned after delivering item i and before loading item j (compartment-simultaneous-and-subsequent incompatibility);
- u^{v2}_{ij} is equal to one to indicate that if item i is loaded before item j, then item j cannot be loaded in the same vehicle simultaneously and subsequently unless every compartment has been cleaned after delivering item i and before loading item j (vehicle-simultaneous-and-subsequent incompatibility).

As explained in Section 1, compartment-simultaneous and vehicle-simultaneous incompatibilities are symmetric (i.e., $u^{c1}_{ij} = u^{c1}_{ji}$ and $u^{v1}_{ij} = u^{v1}_{ij}\ \forall i, j \in P$), whereas compartment-simultaneous-and-subsequent and vehicle-simultaneous-and-subsequent incompatibilities can be asymmetric (i.e., u^{c2}_{ij} and u^{v2}_{ij} can take different values than u^{c2}_{ji} and u^{v2}_{ji}, respectively, $\forall i, j \in P$). For each arc $(i, j) \in A$, a travel time $t_{ij} \geq 0$ and a travel cost $c_{ij} \geq 0$ are given. Note that for an arc $(i, j) \in A$, t_{ij} includes the service time s_i at node i. We assume that the travel times and travel costs satisfy the triangle inequality. An unlimited fleet of homogeneous vehicles is available at the depot to pickup and deliver the set of items. Each vehicle has a set of compartments $M = \{1, \ldots, m\}$ and each compartment has a capacity $Q > 0$. The total capacity of the vehicle is mQ. Each cleaning operation is associated with a cleaning time. The cleaning time of each cleaning operation comprises a fixed cleaning time $\delta_1 \geq 0$ and a variable time associated with cleaning one compartment $\delta_2 \geq 0$, i.e., when cleaning m' compartments the total cleaning time is $\delta_1 + m' \times \delta_2$.

Let Ω be the set of feasible routes that satisfy pairing and precedence, time windows, item incompatibilities, and compartment capacities. Let c_r be the total cost associated with route r consisting of a fixed vehicle cost and the travel cost. The binary parameter a_{ir} equals one if item i is picked up and delivered in route r and zero otherwise. Using the binary variable y_r which is equal to one if route r is selected, the PDPTWMCI can be formulated as follows

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

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

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

The objective (1a) minimizes the total costs. Constraints (1b) ensure that each item is picked up and delivered exactly once, and the domain of the variables is defined with constraints (1c).

3 Branch-price-and-cut algorithm

Formulation (1) contains an exponential number of variables. Therefore, we solve it using a BPC algorithm. BPC is a branch-and-bound algorithm, where the linear relaxations are solved using column generation, and where cuts can be added dynamically to strengthen the linear relaxations. The linear relaxation of Formulation (1) is called the master problem. Column generation starts with a restricted master problem (RMP) which is the linear relaxation of Formulation (1) for a subset of variables. Within the column-generation algorithm, a pricing problem is solved to identify negative reduced-cost columns. The algorithm iterates between reoptimizing the RMP and solving the pricing problem. If negative reduced costs columns are found, they are added to the RMP, if none exists the master problem is solved to optimality. Branching is needed to obtain integer solutions.

In this section, we first present a basic monodirectional forward labeling algorithm. Then, we propose different acceleration techniques and describe the variant of selective pricing that results when combining some of these techniques. Finally, we summarize the branching strategies as well as the implemented valid inequalities.

3.1 Basic monodirectional forward labeling algorithm

In our BPC algorithm, the pricing problem corresponds to identifying routes starting at the origin depot 0, visiting a subset of nodes $P \cup D \cup W$, and ending at the destination depot $2n+1$. A feasible route is an elementary shortest path satisfying pairing and precedence, time windows, capacity, and incompatibility constraints. More precisely, for each completed item in the path, its pickup node $i \in P$ must be visited before its corresponding delivery node $n+i \in D$, time windows must be respected at every visited node, and the capacity of each compartment can never be exceeded. In addition, an item can only be loaded in the vehicle's compartments if there are no existing incompatibilities (compartment-simultaneous, vehicle-simultaneous, compartment-simultaneous-and-subsequent, and vehicle-simultaneous-and-subsequent incompatibilities).

The objective of the pricing problem consists of identifying paths of negative reduced cost. Let $\pi_i, i \in P$ be the dual variables associated with Constraints (1b) and let $\tilde{\pi}_i^f := \pi_i$ if $i \in P$ and $\tilde{\pi}_i^f := 0$ otherwise. We define the reduced cost of an arc $(i,j) \in A$ as $\tilde{c}_{ij}^f := c_{ij} - \frac{\tilde{\pi}_i^f}{2} - \frac{\tilde{\pi}_j^f}{2}$. The reduced cost of a route r is then given by $\tilde{c}_r = \sum_{(i,j)\in A(r)} \tilde{c}_{ij}^f$, where $A(r)$ is the set of arcs used in route r.

The pricing problem is solved using a monodirectional forward labeling algorithm. In order to consider incompatibilities, our labeling algorithm has to be adapted compared to previous related work (e.g., Aerts-Veenstra et al., 2024; Cherkesly et al., 2016). More precisely, to consider all types of incompatibilities, the set of incompatible items is checked before extending a label. For compartment-simultaneous and compartment-simultaneous-and-subsequent incompatibilities, this is done when creating a set of feasible extensions (i.e., to determine the set of compartments on which an item can be loaded to). For vehicle-simultaneous and vehicle-simultaneous-and-subsequent incompatibilities, this is done before extending to a node. Finally, to keep track of compartment-simultaneous-and-subsequent and vehicle-simultaneous-and-subsequent incompatibilities, we need two additional resources. In the following, we describe our forward labeling algorithm. First, we describe the label resources (components). Second, we define the resource extension functions (REFs). Finally, in order to remove non-promising paths, a dominance criterion is presented. Note that state-of-the-art BPC algorithms developed to solve related PDPs often resort to bidirectional labeling algorithms (see Aerts-Veenstra et al., 2024; Gschwind et al., 2018, for example). For the PDPTWMCI, extensive experiments showed that using a forward monodirectional algorithm outperformed a bidirectional algorithm due to the additional algorithmic complexity of the latter. In particular, bidirectional labeling requires additional label resources in both the forward and the backward part. More details on the bidirectional algorithm are provided in Appendix C.

3.1.1 Resources

A forward label F represents a forward partial path $R(F) = (0, \dots, \eta(F))$ starting at the origin depot node and ending at node $\eta(F)$. A label $F = (\eta(F), t(F), c(F), U(F), (l^m(F))_m, (O^m(F))_m, (I_c^m(F))_m, (I_v^m(F))_m)$ is composed of the following components:

- $\eta(F)$, the *last* visited node of the partial path;
- $t(F)$, the *earliest feasible start* of the service time at node $\eta(F)$;
- $c(F)$, the accumulated reduced cost;
- $U(F)$, the set of *unreachable* items, where an item is unreachable if it has already been picked up and delivered on the partial path, or if its pickup node cannot be reached time-window feasible;
- $l^m(F)$, the load onboard compartment $m \in M$;
- $O^m(F)$, the set of items onboard compartment $m \in M$, i.e., the items that have been picked up and loaded into compartment m and are not yet delivered on the partial path;
- $I_c^m(F)$, the set of items that cannot be loaded in compartment m because of *compartment-simultaneous-and-subsequent incompatibility* between items;
- $I_v^m(F)$, the set of items that cannot be loaded in compartment m because of *vehicle-simultaneous-and-subsequent incompatibility* between items.

Note that $l^m(F)$ is redundant, because it can be determined from $O^m(F)$. It is stored in F to conveniently verify the feasibility of $R(F)$ with respect to the capacity restrictions. The initial label at the origin depot is $F_0 = (0, \underline{w}_0, 0, \emptyset, (0)_m, (\emptyset)_m, (\emptyset)_m, (\emptyset)_m)$.

3.1.2 Resource extension functions

In this section, we present the REFs along arc $(\eta(F), j)$. First, we present conditions to extend a label along an arc which consider the compatibility with the vehicle as well as the sets of empty compartments for cleaning. Second, we present the possible extensions, which consider the compartment compatibility between items as well as the compartments that are empty and can be cleaned. Third, we present the REFs according to the possible extensions. Finally, we define feasibility conditions for a label according to time windows and capacity constraints.

Before extending a label F along arc $(\eta(F), j) \in A$, we verify that extending it could yield a feasible label. Therefore, the extension can be done if one of the following conditions holds:

$$j \in P \text{ and } j \notin U(F) \cup \bigcup_{m \in M} O^m(F) \cup \bigcup_{m \in M} I_v^m(F) \text{ and } \sum_{m \in M} \sum_{i \in O^m(F)} (u_{ij}^{v1} + u_{ji}^{v1}) = 0, \tag{2a}$$

$$j \in D \text{ and } j - n \in \bigcup_{m \in M} O^m(F), \tag{2b}$$

$$j \in W \text{ and } \exists m \in M \text{ such that } O^m(F) = \emptyset, \tag{2c}$$

$$j = 2n + 1 \text{ and } \bigcup_{m \in M} O^m(F) = \emptyset. \tag{2d}$$

Condition (2a) states that if j is a pickup node then the corresponding item should neither be unreachable nor onboard any compartment, be vehicle-simultaneous-and-subsequent compatible with the set of items which have been previously picked up and delivered (according to the cleaning status of the vehicle and the compartments), and be vehicle-simultaneous compatible with all other items onboard the vehicle. Condition (2b) states that if j is a delivery node then its corresponding item should be onboard the vehicle. Condition (2c) states that if j is a cleaning station then it can only be visited if there is an empty compartment, i.e., cleaning operations can only be conducted on empty compartments. Finally, Condition (2d) states that if j is the destination depot then the vehicle must be empty.

Note that because $u_{ij}^{v1} = u_{ji}^{v1}$, Condition (2a) could also be written as

$$j \in P \text{ and } j \notin U(F) \cup \bigcup_{m \in M} O^m(F) \bigcup_{m \in M} I_v^m(F) \text{ and } \sum_{m \in M} \sum_{i \in O^m(F)} u_{ij}^{v1} = 0.$$

Extending a label F along arc $(\eta(F), j)$ might lead to multiple feasible extensions according to the compatible compartments and feasible cleaning operations. More precisely, if $j \in P \cup W$, there might be multiple feasible extensions, whereas if $j \in D \cup \{2n+1\}$, there is only one extension. This set of extensions is represented by $\mathcal{H}^f(F, j)$. Each extension is denoted by (h, H), where the first component represents the compartment on which an item is loaded or unloaded from, and the second component represents the set of cleaned compartments. Note that if j is not a cleaning station ($j \in N$), then the second term is always $H = \emptyset$. The set of possible extensions is defined as

$$\mathcal{H}^f(F,j) = \begin{cases} \{m \in M | j \notin I_c^m(F) \text{ and } \sum\limits_{i \in O^m(F)} (u_{ij}^{c1} + u_{ji}^{c1}) = 0\} \times \{\emptyset\} & \text{if } j \in P, & (3a) \\ \{m \in M | j - n \in O^m(F)\} \times \{\emptyset\} & \text{if } j \in D, & (3b) \\ \{m_1\} \times \{C \in \mathscr{P}(M) | C \neq \emptyset \text{ and } O^m(F) = \emptyset \ \forall m \in C\} & \text{if } j \in W, & (3c) \\ \{(m_1, \emptyset)\} & \text{if } j = 2n+1\ , & (3d) \end{cases}$$

where $m_1 \in M$ represents the first compartment and $\mathscr{P}(M)$ denotes the power set of M. If j is a pickup node ($j \in P$), there is one extension to each compartment for which all onboard items are compartment-simultaneous compatible and compartment-simultaneous-and-subsequent compatible with j. If j is a delivery node ($j \in D$), there is exactly one extension to the compartment from which the item is unloaded. If j is a cleaning station ($j \in W$), there are multiple extensions according to the sets of compartments that can and will be cleaned (only empty compartments can be cleaned and at least one compartment has to be cleaned). If j is the destination depot ($j = 2n+1$), there is exactly one extension to the first compartment. For simplification purposes, and because $u_{ij}^{c1} = u_{ji}^{c1}$, Equation (3a) could also be written as

$$\left\{ m \in M | j \notin I_c^m(F) \text{ and } \sum_{i \in O^m(F)} u_{ij}^{c1} = 0 \right\} \times \left\{\emptyset\right\} \text{ if } j \in P.$$

In the remainder of the paper, we continue with $u_{ij}^{c1} + u_{ji}^{c1}$ for generalizability, but everywhere this could be modified for u_{ij}^{c1} if $u_{ij}^{c1} = u_{ji}^{c1}$.

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

$$\eta(F^{(h,H)}) = j, \tag{4a}$$

$$t(F^{(h,H)}) = \begin{cases} \max\{\underline{w}_j, t(F) + t_{\eta(F),j} + \delta_1 + |H| \cdot \delta_2\} & \text{if } j \in W, \\ \max\{\underline{w}_j, t(F) + t_{\eta(F),j}\} & \text{otherwise,} \end{cases} \tag{4b}$$

$$c(F^{(h,H)}) = c(F) + \tilde{c}^f_{\eta(F),j} \tag{4c}$$

$$U(F^{(h,H)}) = \begin{cases} U(F) \cup \{j-n\} \cup \left\{i \in P | t(F^{(h,H)}) + t_{j,i} > \overline{w}_i\right\} & \text{if } j \in D, \\ U(F) \cup \left\{i \in P | t(F^{(h,H)}) + t_{j,i} > \overline{w}_i\right\} & \text{otherwise,} \end{cases} \tag{4d}$$

$$l^m(F^{(h,H)}) = \begin{cases} l^m(F) + q_j & \text{if } m = h, \\ l^m(F) & \text{otherwise,} \end{cases} \tag{4e}$$

$$O^m(F^{(h,H)}) = \begin{cases} O^m(F) \cup \{j\} & \text{if } j \in P \text{ and } m = h, \\ O^m(F) \setminus \{j-n\} & \text{if } j \in D \text{ and } j - n \in O^m(F), \\ O^m(F) & \text{otherwise,} \end{cases} \tag{4f}$$

$$I_c^m(F^{(h,H)}) = \begin{cases} I_c^m(F) \cup \left\{i \in P | u_{ji}^{c2} = 1\right\} & \text{if } j \in P \text{ and } m = h, \\ \emptyset & \text{if } j \in W \text{ and } m \in H, \\ I_c^m(F) & \text{otherwise,} \end{cases} \tag{4g}$$

$$I_v^m(F^{(h,H)}) = \begin{cases} I_v^m(F) \cup \left\{i \in P | u_{ji}^{v2} = 1\right\} & \text{if } j \in P, \\ \left\{i \in P | u_{ki}^{v2} = 1, k \in \cup_{m' \in M} O^{m'}(F)\right\} & \text{if } j \in W \text{ and } m \in H, \\ I_v^m(F) & \text{otherwise.} \end{cases} \tag{4h}$$

Equations (4a)–(4f) are adapted from the multi-compartment PDP with time windows literature, while Equations (4g)–(4h) are novel to consider the compartment-simultaneous-and-subsequent and the vehicle-simultaneous-and-subsequent incompatibilities. More precisely, Equation (4a) sets the last visited node of the partial path to j. Equation (4b) updates the time resource by making sure that it respects the earliest time at which service can start at node j, considers the travel time, and considers the cleaning time (according to the number of cleaned compartments if the extension is done to a cleaning station). Equation (4c) updates the reduced cost by considering the arc's reduced cost. Equation (4d) updates the set of unreachable items to impose elementary conditions and to include items that cannot be reached in time. The load of each compartment is updated with Equation (4e), i.e., the load of a compartment is modified only if an item is loaded on or unloaded from that compartment. The set of onboard items is updated with Equation (4f), i.e., if an item is loaded (or unloaded) in (or from) a compartment. Equation (4g) updates the set of compartment-simultaneous-and-subsequent incompatible items as follows: 1) if j is a pickup node, then all its corresponding compartment-simultaneous-and-subsequent incompatible items are incompatible with the compartment on which it is loaded, and 2) if j is a cleaning station, then all the compartment-simultaneous-and-subsequent incompatibilities are removed for the cleaned compartments. Equation (4h) updates the set of vehicle-simultaneous-and-subsequent incompatible items as follows: 1) if j is a pickup node, then all its corresponding vehicle-simultaneous-and-subsequent incompatible items are incompatible with all the compartments in the vehicle, and 2) if j is a cleaning station, then all the vehicle-simultaneous-and-subsequent incompatibilities are removed for the cleaned compartments except for the items that are vehicle-simultaneous-and-subsequent incompatible with the set of items that are still onboard the vehicle.

A label $F^{(h,H)}$ is kept if it does not violate the time windows or the capacity constraints:

$$t(F^{(h,H)}) \leq \overline{w}_j, \tag{5a}$$

$$l^m(F^{(h,H)}) \leq Q, \qquad \forall m \in M. \tag{5b}$$

These conditions are standard in the related PDPTW literature.

3.1.3 Dominance

A label F_1 dominates another label F_2 residing at the same node $\eta(F_1) = \eta(F_2)$ if

$$t(F_1) \leq t(F_2), \tag{6a}$$

$$c(F_1) \leq c(F_2), \tag{6b}$$

$$U(F_1) \subseteq U(F_2), \tag{6c}$$

$$O^m(F_1) = O^m(F_2), \qquad \forall m \in M, \tag{6d}$$

$$I_c^m(F_1) = I_c^m(F_2), \qquad \forall m \in M, \tag{6e}$$

$$I_v^m(F_1) = I_v^m(F_2), \qquad \forall m \in M. \tag{6f}$$

The validity of (6) follows immediately from the fact that the REFs for $t(F)$, $c(F)$, and $U(F)$ are non-decreasing while Conditions (6d)–(6f) require equality for the remaining resources.

3.2 Acceleration strategies

To accelerate the pricing process, we employ a well-established heuristic-pricing strategy using a series of arc-reduced networks (see, e.g., Gschwind et al., 2018; Aerts-Veenstra et al., 2024). Furthermore, we have derived several additional problem-specific acceleration techniques that are described in the following. First, we explain how to reduce the symmetry induced by the compartments in the dominance. Second, we explain how to use subset dominance in the dominance criteria. Third, we describe modifications to the arc set A. Finally, we explain how to reduce the number of label extensions. Note that defining acceleration strategies for this problem and combining them within branching and cutting schemes is not trivial as it can lead to wrongly dominated labels. Furthermore, we show that using subset dominance in our context can lead to a variant of selective pricing. Additional details related to this complexity can be found in Appendix A.3.

3.2.1 Symmetry breaking in the dominance

Because all compartments have the same characteristics, i.e., all the compartments have the same capacity, the one-to-one comparisons of the compartments in Conditions (6d)–(6f) of the dominance rule can be replaced by considering a permutation σ of the compartments. That is, we can replace them as follows:

$$O^m(F_1) = O^{\sigma(m)}(F_2), \qquad \forall m \in M, \tag{7a}$$

$$I_c^m(F_1) = I_c^{\sigma(m)}(F_2), \qquad \forall m \in M, \tag{7b}$$

$$I_v^m(F_1) = I_v^{\sigma(m)}(F_2), \qquad \forall m \in M. \tag{7c}$$

3.2.2 Using subsets in the dominance

In this section, we describe three strategies to use subsets in the dominance: (a) for the set of onboard items, (b) for the sets of incompatible items and (c) we strengthen the subsets of incompatible items by adding the set of unreachable items.

Subset dominance for the sets of onboard items. In many BPC algorithms developed for PDP problems (see e.g., Aerts-Veenstra et al., 2024; Gschwind et al., 2018; Ropke and Cordeau, 2009), subset dominance is used rather than equality (e.g., for the set of onboard items). Using subset dominance for the sets of onboard items requires that the delivery triangle inequality is respected in the reduced cost matrix, i.e., $\tilde{c}_{ij}^f + \tilde{c}_{jk}^f \geq \tilde{c}_{ik}^f$ for $i, k \in V, j \in D$. If this matrix does not respect the delivery triangle inequality, then Ropke and Cordeau (2009) propose a procedure to modify the matrix to ensure the delivery triangle inequality. In our dominance rule, we can use subset dominance for the sets of onboard items, i.e., we can replace Condition (6d) by

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

Subset dominance for the sets of incompatible items. Similar to the set of onboard items, we can use subset dominance for the sets of incompatible items, i.e., we can replace Conditions (6e) and (6f) by

$$I_c^m(F_1) \subseteq I_c^m(F_2), \qquad \forall m \in M, \tag{9a}$$

$$I_v^m(F_1) \subseteq I_v^m(F_2), \qquad \forall m \in M. \tag{9b}$$

Adding the set of unreachable items. Conditions (9a) and (9b) can be further strengthened by accounting for the unreachable items as follows:

$$I_c^m(F_1) \subseteq I_c^m(F_2) \cup U(F_2), \qquad \forall m \in M, \tag{10a}$$

$$I_v^m(F_1) \subseteq I_v^m(F_2) \cup U(F_2), \qquad \forall m \in M. \tag{10b}$$

In order to avoid costly re-computations in every dominance test, we adopt a double bookkeeping strategy (see Faldum et al., 2024) and keep $I_c^m(F) \cup U(F)$ as an additional resource in each label F.

3.2.3 Reduction of the arc set A

The current definition of the arc set A includes all arcs. Nevertheless, in an optimal solution, all arcs between a cleaning station and the depot, i.e., (i, j) such that $i = 0, j \in W$, and $i \in W, j = 2n+1$, will not be used. Therefore, we remove these arcs from the arc set A, which implies working with a subset of the feasible routes Ω. While this strategy might seem trivial, combining it with subset dominance for $O(F)$ yields a variant of selective pricing. This is discussed in Section 3.2.5.

3.2.4 Reduction of the number of label extensions

In this section, we propose two strategies to reduce the number of label extensions, i.e., when extending to empty compartments and by considering only what we refer to as Pareto-optimal cleanings.

Extensions to empty compartments. In the PDPTWMCI, each compartment of a vehicle has the same characteristics (i.e., the capacity). Compartments are said to be similar (and can be compared) when they have the same incompatibility sets. In the set of feasible extensions as defined with Equation (3), a label can be extended to multiple empty compartments. If these compartments are similar, i.e., if they have the same incompatibility sets, we can, however, restrict ourselves to extending the label to only one of these compartments. For example, let $m_1, m_2 \in M, m_1 \neq m_2$, $O^{m_1}(F) = O^{m_2}(F) = \emptyset$, be two different empty compartments with $I_c^{m_1}(F) = I_c^{m_2}(F)$ and $I_v^{m_1}(F) = I_v^{m_2}(F)$. When extending to a node $j \in P$, we perform only the extension to the compartment with the lowest index $\min_{m_1, m_2 \in M}\{m_1, m_2\}$. Equation (3a) can be written as

$$\Big\{m \in M | j \notin I_c^m(F) \text{ and } \sum_{i \in O^m(F)} (u_{ij}^{c1} + u_{ji}^{c1}) = 0 \text{ and } \big(\nexists\ m' \in M \text{ with } m' < m,$$
$$O^{m'}(F) = O^m(F) = \emptyset,\ I_c^{m'}(F) = I_c^m(F) \text{ and } I_v^{m'}(F) = I_v^m(F)\big)\Big\} \times \big\{\emptyset\big\} \text{ if } j \in P. \tag{11}$$

Pareto-optimal cleanings. In the set of feasible extensions as defined by Equation (3), we can take advantage of Pareto-optimal cleanings induced by specific values for the fixed and variable cleaning times (δ_1 and δ_2). More precisely, there are two interesting cases: when the variable cleaning time is set to zero ($\delta_2 = 0$), and when the variable cleaning time takes a strictly positive value ($\delta_2 > 0$). In the former case ($\delta_2 = 0$), cleaning any subset of compartments results in the same cleaning time. As a result, the single extension in which all empty compartments are cleaned is superior to all other extensions. It is performed if at least one of the empty compartments has an existing incompatibility. In the latter case ($\delta_2 > 0$), cleaning compartments without incompatibilities takes additional time without adding value. We, therefore, can restrict the set of extensions to those that clean only compartments with existing incompatibilities. Equation (3c) can thus be written as

$$\begin{cases} \big\{m_1\big\} \times \Big\{C \in \mathscr{P}(M) | C \neq \emptyset \text{ and } O^m(F) = \emptyset\ \forall m \in C \\ \qquad\qquad \text{and } O^m(F) \neq \emptyset\ \forall m \in M \backslash C \text{ and} \\ \qquad\qquad \exists m \in M \text{ with } I_c^m(F) \cup I_v^m(F) \neq \emptyset\Big\} & \text{if } j \in W, \delta_2 = 0, \\ \big\{m_1\big\} \times \Big\{C \in \mathscr{P}(M) | C \neq \emptyset \text{ and } \big(O^m(F) = \emptyset \\ \qquad\qquad \text{and } I_c^m(F) \cup I_v^m(F) \neq \emptyset\ \forall m \in C\big)\Big\} & \text{if } j \in W, \delta_2 > 0. \end{cases} \tag{12}$$

If we also use subset dominance for the sets of incompatible items, additional extensions are required to ensure the validity of our dominance rule (see Appendix A for details). More precisely, if there exists an empty compartment, but all empty compartments do not have any incompatibility, i.e., $I_c^m(F) \cup I_v^m(F) = \emptyset\ \forall m \in M$ with $O^m(F) = \emptyset$, no extensions to cleaning station would be performed with Pareto-optimal cleanings. In this case, we add an extra extension to the cleaning stations. This extension does not perform any cleanings and requires the fixed cleaning time (δ_1) but no variable cleaning time (δ_2). Therefore, Equation (3c) would be written as

$$\begin{cases} \Big\{(m_1, \emptyset)\Big\} & \text{if } j \in W, I_c^m(F) \cup I_v^m(F) = \emptyset \ \forall m \in M \\ & \quad \text{with } O^m(F) = \emptyset, \\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C \neq \emptyset & \text{if } j \in W, \delta_2 = 0, \exists m \in M \\ \qquad \text{and } O^m(F) = \emptyset \ \forall m \in C & \quad \text{with } O^m(F) = \emptyset \\ \qquad \text{and } O^m(F) \neq \emptyset \ \forall m \in M \backslash C \Big\} & \quad \text{and } I_c^m(F) \cup I_v^m(F) \neq \emptyset, \\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C \neq \emptyset & \text{if } j \in W, \delta_2 > 0, \exists m \in M \\ \qquad \text{and } \big(O^m(F) = \emptyset \text{ and} & \quad \text{with } O^m(F) = \emptyset \\ \qquad I_c^m(F) \cup I_v^m(F) \neq \emptyset \ \forall m \in C\big) \Big\} & \quad \text{and } I_c^m(F) \cup I_v^m(F) \neq \emptyset. \end{cases} \tag{13}$$

3.2.5 Proposed BPC with all acceleration strategies

Combining all strategies is non-trivial and yields some issues. In the following, we first describe the set of extensions and the dominance rule with all the proposed acceleration strategies. Then, we discuss two important elements: identifying the permutations of compartments and the interaction between reducing the arc set and subset dominance for the sets of onboard items.

Definition of $\mathcal{H}^f(F, j)$ with all acceleration strategies. Combining all acceleration strategies results in the following set of possible extensions:

$$\mathcal{H}^f(F, j) = \begin{cases} (11), \\ (3b), \\ (13), \\ (3d). \end{cases} \tag{14}$$

Complete dominance rule. In addition, the complete dominance rule is as follows: A label F_1 dominates another label F_2 residing at the same node $\eta(F_1) = \eta(F_2)$ if

$$t(F_1) \leq t(F_2), \tag{15a}$$
$$c(F_1) \leq c(F_2), \tag{15b}$$
$$U(F_1) \subseteq U(F_2), \tag{15c}$$
$$O^m(F_1) \subseteq O^{\sigma(m)}(F_2), \qquad \forall m \in M, \tag{15d}$$
$$I_c^m(F_1) \subseteq I_c^{\sigma(m)}(F_2) \cup U(F_2), \qquad \forall m \in M, \tag{15e}$$
$$I_v^m(F_1) \subseteq I_v^{\sigma(m)}(F_2) \cup U(F_2), \qquad \forall m \in M. \tag{15f}$$

A proof of the validity of the proposed dominance rule is given in Appendix A.

Identifying the permutations of compartments. To effectively identify an appropriate permutation σ such that (15d)–(15f) are fulfilled or to verify that none exists, we employ the following strategy (inspired from Algorithm 1 of Aerts-Veenstra et al., 2024). We first try to find for all non-empty compartments $m \in M$ of F_1, i.e., those with $O^m(F_1) \neq \emptyset$, a compartment $\sigma(m)$ such that $O^m(F_1) \subseteq O^{\sigma(m)}(F_2)$, $I_c^m(F_1) \subseteq I_c^{\sigma(m)}(F_2) \cup U(F_2)$, and $I_v^m(F_1) \subseteq I_v^{\sigma(m)}(F_2) \cup U(F_2)$. Note that these $\sigma(m)$ are unique for all non-empty $m \in M$ of F_1 (Proposition 1 of Aerts-Veenstra et al., 2024). Maintaining a sorted list (non-increasing by load $l(F)$) of the compartments for each label F helps reducing the number of tests necessary to find these pairings or to realize that none can exist. Subsequently, we actually test all possible permutations of the remaining compartments of F_2 to identify appropriate compartments $\sigma(m)$ for the empty compartments of F_1. Testing all permutations is viable, because the number of compartments is typically small.

Variant of selective pricing. Combining subset dominance for $O(F)$ (see Section 3.2.2) with the reduced arc set (see Section 3.2.3) yields a variant of selective pricing. The concept of selective pricing in branch-and-price was introduced by Desaulniers et al. (2019). It builds on the commonly used strategy of working with a relaxed set of variables, whose generation might be computationally less expensive, rather than with the set of feasible variables. A typical example is the relaxation of the elementarity condition of routes in VRP variants. The basic principle is as follows. Let Ω be the set of *feasible* variables for the problem at hand and let $\hat{\Omega} \supseteq \Omega$ be a relaxed (enlarged) set of variables. Variables $r \in \hat{\Omega} \setminus \Omega$ are called *relaxed*. In selective pricing, the pricing problem consists of finding a variable (*feasible* or *relaxed*) with negative reduced cost, if there exists a *feasible* variable with negative reduced cost. Desaulniers et al. (2019) show that the optimal RMP value obtained with the selective-pricing paradigm is a valid lower bound at any node of the search tree. Consequently, the branch-and-price algorithm remains valid even if it fails to generate some *relaxed* routes with negative reduced cost. To the best of our knowledge, selective pricing has so far exclusively been used in the context of VRPs and relaxations of the elementarity condition (Desaulniers et al., 2019; Costa et al., 2020).

Our proposed labeling with all acceleration strategies employs a variant of selective pricing that may fail to generate variables with negative reduced cost that are actually feasible for the considered problem. The general concept is as follows. Let $\bar{\Omega} \subseteq \Omega$ be a subset of the variables that are feasible for the problem at hand, such that $\bar{\Omega}$ contains all variables of at least one optimal solution. Clearly, the considered problem can immediately be modeled by Formulation (1) in which Ω is replaced by $\bar{\Omega}$. Variables in $\bar{\Omega}$ are called *optimal*, variables in $\Omega \setminus \bar{\Omega}$ are called *non-optimal*. Practically speaking, the variant of selective pricing may fail to generate *non-optimal* variables with negative reduced cost as long as it guarantees to find variables (*non-optimal* or *optimal*) with negative reduced cost whenever there exists *optimal* variables with negative reduced cost. Note that we can interpret the set of optimal routes $\bar{\Omega}$ as set of feasible routes Ω and the set of feasible routes Ω as the set of relaxed routes $\hat{\Omega}$ yielding the original selective-pricing case, i.e., validity follows immediately from the original case. However, the proposed concept is novel in the sense that it allows to miss variables that are actually feasible for the considered problem, while it is also general—one just needs to identify a restricted set of variables that guarantees to comprise at least one optimal solution. Details on the specific definition of $\bar{\Omega}$ and on which routes may be missed by our proposed labeling are presented in Appendix A.

3.3 Valid inequalities and branching

In this section, we describe the two-well known valid inequalities that we use in our BPC algorithm as well as the branching strategy.

In our BPC algorithm, we resort to rounded capacity inequalities and subset-row inequalities, which are well-known to be efficient in the contexts of the VRP and the PDP. Rounded capacity inequalities (Laporte and Nobert, 1983; Ropke and Cordeau, 2009) are known to be robust and we separate them using the heuristic proposed by Ropke and Cordeau (2009). Subset-row inequalities (Jepsen et al., 2008), on the other hand, are non-robust cuts and we limit these cuts to subsets of cardinality 3 as it is usually done in the literature.

Our branching tree is explored in a best-bound-first manner and two branching rules are implemented in an hierarchical fashion, which is typical in PDPs (Aerts-Veenstra et al., 2024; Gschwind et al., 2018; Ropke and Cordeau, 2009). If the vehicle flow is fractional, we first branch on it. If it is integer, we branch on the outflow of a subset of nodes $V \setminus \{0, 2n+1\}$ of size two. We select the subset with the outflow closest to 1.5. Note that the proposed branching rules guarantee integer arc flows, but do not guarantee integer path flows due to the multiple visits allowed at a cleaning station. Branching on consecutive flows would ensure integer path flows (Desaulniers, 2010). These branching rules are non-robust, and require to modify the pricing problem. Because we did not need these branching rules in our experiments and due to the added complexity in the pricing problem, for conciseness reasons we do not present them in our paper (similar to, e.g., Parragh et al., 2015; Yamín et al., 2025).

4 Computational results

In this section, we present our computational experiments. All variants of our BPC algorithm were implemented in C++ and compiled into 64-bit single-thread code using GCC 14.1. CPLEX 22.1.0 with default parameters (except for the time limit and allowing a single thread only) was used to reoptimize the RMPs. The tests were carried out on RPTU Kaiserslautern-Landau's high-performance computing cluster "Elwetritsch" consisting of several AMD EPYC 7262 processors running at 3.20 gigahertz. Note that that the performance of a single thread of the cluster is comparable to that of a standard desktop processor. We set a time limit of 3,600 seconds.

Our tests have been conducted on a total of 638 base instances, and we have tested many different parameter values. Therefore, to improve readability and for conciseness reasons, we only present aggregated results. In addition, some sections focus only on a few specific parameter values, and we always specify the values of each parameter in our experiments.

In this section, we first describe the set of instances on which our computational experiments have been conducted. Second, we assess the impact of different algorithmic components to determine which ones have the most impact on the performance of our algorithm. Third, we study the impact of the percentage of incompatible items for the different incompatibility types on the results. Fourth, we analyze the impact of the number of cleaning stations on the results. Last, we investigate the impact of the fixed and variable cleaning times on the results.

4.1 Instances

We adapt the instances for the PDPTWMC (Aerts-Veenstra et al., 2024) to the PDPTWMCI. We start with the 319 initial instances described by Aerts-Veenstra et al. (2024) that all originate from the TSPLIB instances. From these 319 instances, 99 are based on the $a280$ instance and 220 are based on the $brd14051$, $d18512$, $fnl4461$, and $nrw1379$ instances. The instances contain between 25 and 75 items. Original demand and time windows are used. As in previous works, the rounding of travel costs and travel times are, respectively, up to 4 and 2 digits, and in order to prioritize the minimization of the number of vehicles, a cost of 100,000 is added to each arc originating from the origin depot. Each vehicle has a total capacity of 24 and we study two different numbers of compartments, namely $|M| = 2$ and $|M| = 3$, resulting in a total of 638 instances. For each instance, we generate five cleaning stations ($|W| = 5$), numbered from $2n + 2$ to $2n + 6$. The x- and y-coordinate of each cleaning station are randomly generated between the minimal and maximal x- and y-coordinate for each of the instance with 75 items, and used for all other instances with the same name and different number of items. We also set the variable and fixed cleaning time at a station to 1 ($\delta_1 = \delta_2 = 1$). To determine these values (number of cleaning stations, δ_1 and δ_2), we have conducted a set of preliminary computational experiments. Finally, we set 25% of the item pairs as incompatible, where an equal division is made between the different types of incompatibilities, i.e., 6.25% are compartment-simultaneous incompatible, 6.25% are compartment-simultaneous-and-subsequent incompatible, 6.25% are vehicle-simultaneous incompatible, and 6.25% are vehicle-simultaneous-and-subsequent incompatible. To determine which items are incompatible, we use the 12 item categories which have been randomly assigned to each item by Aerts-Veenstra et al. (2024). More specifically, all items in categories 1–3 are compartment-simultaneous incompatible between each other, all items in categories 4–6 are compartment-simultaneous-and-subsequent incompatible between each other, all items in categories 7–9 are vehicle-simultaneous incompatible between each other, and all items in categories 10–12 are vehicle-simultaneous-and-subsequent incompatible between each other.

4.2 Impact of different algorithmic components

In this section, we study the impact of different algorithmic components on the performance of our BPC algorithm. Namely, we first report results for our proposed BPC algorithm (*Proposed BPC*) using all acceleration strategies described in Section 3.2 with the set of extensions as defined in (14) and the dominance rule (15). Then, we report results for a variant without acceleration strategies (*No acceleration*) which consists in using the set of extensions as defined in (3) and the dominance rule (6). From the variant *No acceleration*, we then add one acceleration strategy at the time: (i) modifying the dominance criteria to include symmetry breaking in the dominance (*Symmetry breaking in dominance*), i.e., dominance rule (6a)–(6c) and (7); (ii) using subsets in the dominance for the set of onboard items (*Subset dominance for* $O^m(F)$), i.e., dominance rule (6) where Condition (6d) is replaced by (8); (iii) using subsets in the dominance for the sets of incompatible items (*Subset dominance for* $I_c^m(F)$ *and* $I_v^m(F)$), i.e., dominance rule (6) where Conditions (6e) and (6f) are replaced by Conditions (9); (iv) adding the unreachable items to subset dominance for the sets of incompatible items (*Subset dominance for* $I_c^m(F)$ *and* $I_v^m(F)$ *with* $U(F)$), i.e., dominance rule (6) where Conditions (6e) and (6f) are replaced by Conditions (10); (v) removing arcs from the origin depot to a cleaning station, and from a cleaning station to the destination depot (*No depot-station arcs*); (vi) limiting the number of extensions to empty compartments as detailed in Section 3.2.4 (*Extensions to empty compartments*), i.e., using the extensions of Equations (11), (3b), (3c) and (3d); (vii) limiting the number of extensions by considering Pareto-optimal cleanings as detailed in Section 3.2.4 (*Extensions with Pareto-optimal cleanings*), i.e., using the extensions of Equations (3a), (3b), (12) and (3d). Finally, we also report the impact of solving our proposed algorithm (with all acceleration strategies) with a pure backward labeling algorithm (*Backward BPC*) and with a bidirectional labeling algorithm (*Bidirectional BPC*). Backward and bidirectional labeling are detailed in Appendix B and C, respectively. For each proposed variant, we report the number of solutions solved to proven optimality within the computational time limit (*Opt*), the average computational time in seconds (*Sec*), and a runtime ratio for all instances (*all*), as well as instances for which the linear relaxation takes more than 10 (≥ 10) and 120 seconds (≥ 120) to solve with variant *No acceleration*. To compute the average computational time, for the instances solved to proven optimality we take the required time to solve the instances, whereas for the instances that are not solved to proven optimality we take the maximum computational time of 3,600 seconds. Given that computational time, the runtime ratio for an algorithmic variant V is computed for each instance as Sec^V/Sec^{prop} where Sec^V and Sec^{prop} are the computational times in seconds for the variant V and for the proposed BPC, respectively, and we report the average over the considered instances. The results are reported in Table 2.

Table 2: Performance of our proposed algorithm compared to different algorithmic variants.

			Average runtime ratio		
Algorithmic variant	Opt	Sec	*all*	≥ 10	≥ 120
Proposed BPC	628	106.9	1.00	1.00	1.00
No acceleration	582	481.2	6.61	34.92	52.50
Symmetry breaking in dominance	622	173.1	1.69	3.55	5.42
Subset dominance for $O^m(F)$	596	391.2	4.82	22.02	34.43
Subset dominance for $I_c^m(F)$ and $I_v^m(F)$	610	245.4	2.80	8.09	13.83
Subset dominance for $I_c^m(F)$ and $I_v^m(F)$ with $U(F)$	621	168.5	2.54	5.14	6.97
No depot-station arcs	589	460.6	6.11	31.62	49.33
Extensions to empty compartments	607	274.0	2.96	9.72	17.60
Extensions with Pareto-optimal cleanings	595	403.9	4.43	25.09	42.78
Backward BPC	628	96.4	0.72	0.84	0.91
Bidirectional BPC	623	144.7	1.26	1.54	1.83

Our results show that overall each acceleration strategy is beneficial. Without any acceleration strategy, only 582 instances can be solved to proven optimality (compared to 628 for the proposed BPC) and the average computational time is more than four times the computational time of our proposed BPC (481.2 compared to 106.9 seconds). Our results show that the two most important strategies are using symmetry breaking in the dominance, and subset dominance for the sets of incompatible items when adding the set of unreachable items. For these variants, respectively, 622 and 621 instances are solved to proven optimality within an average computational time of 173.1 and 168.5. Nevertheless, combining the strategies is important and further improves BPC performance: for the harder instances, i.e., those for which the linear relaxation cannot be solved within 120 seconds, adding one strategy at the time takes on average between 5 to 49 times the computational time of our proposed BPC.

In addition, our results show that using a pure backward labeling algorithm is comparable to using a pure forward labeling algorithm, i.e., 628 instances are solved to proven optimality for both variants with average computational times of 96.4 and 106.9 seconds. Note that while the backward labeling algorithm is slightly faster for our set of instances, the problem and instance structures do not seem to suggest that one is generally better than the other. Nonetheless, resorting to a bidirectional labeling algorithm does not pay off and results in less instances being solved to proven optimality (623 instances) and the average computational time increases to 144.7 seconds. This suggests that while the instances are relatively symmetric when using a pure backward or a pure forward labeling algorithm, the complexity lies in merging the two labels. As detailed in Appendix C, resorting to a bidirectional labeling algorithm requires additional resources to allow merging of two labels. The additional resources make the labeling algorithm more complex and this added complexity has a computational cost.

4.3 Impact of the percentage of incompatible items for the different incompatibility types

In this section, we study the impact of the percentage of incompatible items for the different incompatibility types by varying the parameter value for the percentage of incompatible items. To this end, we test different parameter values representing the percentage of incompatible items. We denote the percentage of incompatible items as a quadruple ($c1 - c2 - v1 - v2$), where $c1$, $c2$, $v1$, and $v2$ represent the percentage of compartment-simultaneous, compartment-simultaneous-and-subsequent, vehicle-simultaneous, and vehicle-simultaneous-and-subsequent incompatible items, respectively. Recall, that we have initially presented results on the quadruple (6.25-6.25-6.25-6.25). For this analysis, we have tested the following additional quadruples: (0-0-0-0), (25-0-0-0), (50-0-0-0), (75-0-0-0), (100-0-0-0), (0-25-0-0), (0-50-0-0), (0-75-0-0), (0-100-0-0), (0-0-25-0), (0-0-50-0), (0-0-75-0), (0-0-100-0), (0-0-0-25), (0-0-0-50), (0-0-0-75), and (0-0-0-100). With all the quadruples that contain one non-zero value, if the value is (i) 25% then all items in categories 1–3, 4–6, 7–9, and 10–12 are incompatible between each other, (ii) 50% then all items in categories 1–6 and 7–12 are incompatible between each other, (iii) 75% then all items in categories 1–6 and 7–12 are incompatible between each other, item in categories 1–3 are incompatible with item in categories 7–9, and item in categories 4–6 are incompatible with item in categories 10–12, (iv) and 100% then all items are incompatible between each other.

Results of this analysis are provided in Table 3. The first column of this table contains the parameter value of the percentage of incompatible items of each incompatibility type, and the other columns report different statistics. To be precise, we report the total number of instances solved to optimality within the maximal computational time (*Opt*), the average computational time in seconds (*Sec*), the average impact in percentage on the number of vehicles ($\Delta\,Veh$ *(%)*), and the average number of cleanings in a solution (*# Cleanings*). The values $\Delta\,Veh$ *(%)* are obtained by first computing for each instance Veh/Veh^0 where Veh and Veh^0 are the number of vehicles with the new parameter value and without any incompatibility, respectively, and then computing the average over all instances. We do not report the average impact in percentage on the total traveled distance, because for each new parameter value, the optimal solutions of all instances result in different number of vehicles compared to the optimal solutions without incompatibility (recall that the number of vehicles is minimized first).

When there are no incompatibilities, the number of feasible routes increases which results in more complexity. With (0-0-0-0), the algorithm solves the least instances to proven optimality (575) with the largest average computational time (444.0 seconds). In addition, in general, as the percentage of incompatible items increase, the number of feasible routes reduces, which decreases the complexity. With 100% compartment-simultaneous, compartment-simultaneous-and-subsequent, vehicle-simultaneous and vehicle-simultaneous-and-subsequent incompatibilities, the number of instances

Table 3: Impact of the percentage of incompatible items for the different incompatibility types on the results.

(c1-c2-v1-v2)	Opt	Sec	ΔVeh (%)	# Cleanings
(6.25-6.25-6.25-6.25)	628	106.9	17.28%	0.37
(0-0-0-0)	575	444.0	0.00%	0.00
(25-0-0-0)	591	383.6	1.25%	0.00
(50-0-0-0)	610	261.6	4.53%	0.00
(75-0-0-0)	619	170.5	7.63%	0.00
(100-0-0-0)	625	123.8	25.90%	0.00
(0-25-0-0)	591	385.1	1.99%	0.20
(0-50-0-0)	611	237.8	8.44%	1.14
(0-75-0-0)	625	142.9	13.31%	1.63
(0-100-0-0)	623	130.8	41.12%	4.64
(0-0-25-0)	638	20.3	22.27%	0.00
(0-0-50-0)	638	0.2	70.80%	0.00
(0-0-75-0)	638	0.1	86.38%	0.00
(0-0-100-0)	638	0.1	180.10%	0.00
(0-0-0-25)	638	10.3	31.64%	0.89
(0-0-0-50)	638	0.2	96.80%	2.44
(0-0-0-75)	638	0.1	112.36%	2.53
(0-0-0-100)	638	0.1	234.86%	4.92

solved to optimality and average computational time are 625 and 123.8 seconds, 623 and 130.8 seconds, 638 and 0.1 seconds, 638 and 0.1 seconds, respectively. When comparing only compartment-simultaneous-and-subsequent incompatibility, going from (0-75-0-0) to (0-100-0-0) tends to slightly increase the problem complexity due to the increased number of cleanings.

Comparing compartment incompatibilities with vehicle incompatibilities (both simultaneous and simultaneous-and-subsequent), i.e., (X-0-0-0) with (0-0-X-0) and (0-X-0-0) with (0-0-0-X), we see that compartment incompatibilities are much more complex than vehicle incompatibilities as fewer instances are solved to optimality, i.e., up to 625 for (100-0-0-0) and 623 for (0-100-0-0) compared to 638 for (0-0-100-0) and (0-0-0-100), with larger average computational times for compartment incompatibility, e.g., 123.8 seconds for (100-0-0-0) and 130.8 seconds for (0-100-0-0) compared to 0.1 seconds for (0-0-100-0) and (0-0-0-100). This can be explained by the fact that vehicle incompatibilities lead to much fewer feasible routes. In particular, long routes are less likely to be feasible. As a result, vehicle incompatibilities tend to require more vehicles than compartment incompatibilities ($\Delta\,Veh$ goes up to 180.10% and 234.86% for vehicle incompatibilities compared to up to 25.90% and 41.12% for compartment incompatibilities). Finally, when comparing (0-X-0-0) with (0-0-0-X), i.e., the cases where cleaning can remove existing incompatibilities, we see that compartment-subsequent-and-simultaneous incompatibility results in less cleanings (up to 4.64 with (0-100-0-0) compared to up to 4.92 with (0-0-0-100)).

Comparing simultaneous incompatibilities with simultaneous-and-subsequent incompatibilities, i.e., (X-0-0-0) with (0-X-0-0) and (0-0-X-0) with (0-0-0-X), there is no clear trend in terms of complexity. Considering only simultaneous incompatibilities leads to more feasible routes, which can increase the complexity of the problem. On the other hand, considering simultaneous-and-subsequent incompatibilities might result in cleanings (e.g., on average between 0.20 and 4.92 for our instances) which also increases the complexity of the problem. Therefore, one type of incompatibility compared to another does not seem to have a specific trend related to the complexity of the problem. In addition, having incompatibilities that are simultaneous-and-subsequent compared to simultaneous increases the total costs by increasing the number of vehicles required in the solution. For compartment-related incompatibilities, going from (X-0-0-0) to (0-X-0-0) almost doubles the value of $\Delta\,Veh$, e.g., going from 4.53% to 8.44% with (50-0-0-0) and (0-50-0-0). For vehicle-related incompatibilities, the trend is similar but the relative values increase less, e.g., going from 70.80% to 96.80% with (0-0-50-0) and (0-0-0-50).

4.4 Impact of the number of cleaning stations

In this section, we study the impact of the number of cleaning stations by varying the parameter value for the number of cleaning stations ($|W|$). Recall, that we have initially set $|W| = 5$. For this analysis, we have tested three additional numbers of cleaning stations $|W| = \{1, 10, 25\}$. Therefore, we have generated 20 additional cleaning stations in a similar way as the first five (see Section 4.1) numbered as $2n + 7$ to $2n + 26$. For each tested value of $|W|$ we select the nodes $2n + 2$ to $2n + |W| + 1$ as the set of cleaning stations.

Table 4 reports the results of the analysis. The first column reports the parameter value for the number of cleaning stations, the other columns report *Opt*, *Sec*, Δ *Veh (%)*, Δ*Dist (%)*, and # *Cleanings*. The average impact in percentage on the number of vehicles (Δ *Veh (%)*) is obtained by first computing for each instance Veh/Veh^* where Veh and Veh^* are the number of vehicles with the new parameter value and the original parameter value, respectively, and then computing the average over all the instances. Similarly, the average impact in percentage on the total traveled distance (Δ*Dist (%)*) is obtained by first computing for each instance $Dist/Dist^*$ where $Dist$ and $Dist^*$ are the total traveled distance with the new parameter value and the original parameter value, respectively, and then computing the average over all the instances. Note that Δ*Dist (%)* is only computed over the instances with the same number of vehicles in the optimal solutions for the original and new parameter values due to the hierarchical nature of the objective. The other columns are similar to the columns in Table 3.

As a general trend, as the number of cleaning stations increases, the problem becomes more difficult, there are 631, 628, 622 and 617 instances solved to optimality with 1, 5, 10, and 25 cleaning stations. The average computational time also increases from 70.9 to 219.4 seconds when going from one to 25 cleaning stations. While the number of cleanings is limited, namely up to 0.59 cleanings per instance on average, adding more cleaning stations adds more feasible extensions which in turn takes more time. In addition, modifying the number of cleaning stations has a limited impact on the total cost, i.e., number of vehicles and total traveled distance. As a general trend, when there are less cleaning stations, cleaning tends to occur less often which increases the number of vehicles used and the total traveled distance. More precisely, with $|W| = 1$, the number of vehicles and the total traveled distance increase on average by 0.35% and 0.14%, and the average number of cleanings in a solution goes from 0.45 to 0.31. With $|W| = \{10, 25\}$, the number of vehicles decreases by 0.08% and 0.21%, the total traveled distance also decreases by 0.09% and 0.14%, and the average number of cleanings increase to 0.53 and 0.59 cleanings per solution. Therefore, our solutions do not seem to be very sensitive to the number of cleaning stations.

Table 4: Impact of the number of cleaning stations on the results.

$\lvert W\rvert$	Opt	Sec	ΔVeh (%)	ΔDist (%)	# Cleanings
1	631	70.9	0.35%	0.14%	0.31
5	628	106.9	0.00%	0.00%	0.45
10	622	143.8	–0.08%	–0.09%	0.53
25	617	219.4	–0.21%	–0.14%	0.59

4.5 Impact of the cleaning time

In this section, we study the impact of the cleaning time by varying the parameter values for fixed and variable cleaning times (δ_1 and δ_2). Recall, that we have initially set $\delta_1 = \delta_2 = 1$. For this analysis, we test no cleaning time, i.e., $(\delta_1,\ \delta_2) = (0, 0)$, and ten different combinations where we only deviate one of the parameter values from its original value, considering values $\{0, 2.5, 5, 10, 25\}$, and keeping the other parameter value fixed at 1. The maximal value was set in order to reach solutions without cleaning. This results in the following additional combinations of $(\delta_1,\ \delta_2)$: $(0,0)$, $(0,1)$, $(2.5,1)$, $(5,1)$, $(10,1)$, $(25,1)$, $(1,0)$, $(1,2.5)$, $(1,5)$, $(1,10)$, $(1,25)$.

Table 5 reports the results of the analysis. The first column reports the parameter values for the fixed and variable cleaning time, the other columns report, similar to the columns in Table 4, *Opt*, *Sec*, Δ*Veh (%)*, Δ*Dist (%)*, and *# Cleanings*.

When fixing $\delta_1 = \delta_2 = 0$, cleaning is likely to arise as there is no cleaning time and its additional cost is related to the distance traveled when visiting a cleaning station. Our results show that for this setting the highest number of cleanings is obtained (0.96 on average) with the fewest number of vehicles ($\Delta Veh = -1.86\%$) and the shortest traveled distance ($\Delta Dist = -0.31\%$). Nonetheless, this increases the complexity of the problem. With $(\delta_1,\ \delta_2) = (0,0)$, the least number of instances are solved to optimality (612) with the largest average computational time (233.0 seconds). In addition, when increasing the values of δ_1 and δ_2, we see an increase in the numbers of vehicles (up to 1.82%) and distance traveled (up to 0.19%) as well as a decrease in the number of cleanings. The solutions are more sensitive to the value of δ_2 than the value of δ_1, as δ_2 has a higher impact on the cost of cleaning, i.e., there is no cleaning with $\delta_1 \geq 25$ and $\delta_2 \geq 10$. Therefore, as we increase the values of δ_1 and δ_2, the symmetry induced by cleaning is reduced which results in more instances solved to optimality with a lower average computation time (up to 633 instances solved to optimality with an average computation time of 51.8 seconds for $\delta_1 = 25$ and up to 630 instances solved to optimality with an average computation time of 91.6 seconds for $\delta_2 = 25$).

Table 5: Impact of the cleaning time on the results.

(δ_1, δ_2)	Opt	Sec	ΔVeh (%)	ΔDist (%)	# Cleanings
(0,0)	612	233.0	–1.86%	–0.31%	0.96
(1,1)	628	106.9	0.00%	0.00%	0.43
(0,1)	616	191.3	–0.57%	–0.08%	0.66
(2.5,1)	630	76.0	0.53%	0.06%	0.25
(5,1)	631	63.0	1.05%	0.10%	0.15
(10,1)	632	55.6	1.57%	0.16%	0.05
(25,1)	633	51.8	1.82%	0.19%	0.00
(1,0)	623	130.0	–1.25%	–0.16%	0.71
(1,2.5)	631	95.3	0.98%	0.09%	0.18
(1,5)	630	86.7	1.64%	0.17%	0.04
(1,10)	628	90.7	1.82%	0.19%	0.00
(1,25)	630	91.6	1.82%	0.19%	0.00

5 Conclusions

In this paper, we introduced, modeled and solved the PDPTWMCI. We proposed a BPC algorithm to solve the problem and presented several acceleration strategies, i.e., symmetry breaking in the dominance, using subset dominance for the sets of onboard items, using subset dominance for the sets of incompatible items, including unreachable items to the sets of incompatible items in the subset dominance, removing depot-station arcs, reducing the number of label extensions to empty compartments and reducing the number of extension by considering Pareto-optimal cleanings. We have shown that using subsets in the dominance for the set of onboard items yields a variant of selective pricing, which could be used to other problem settings to further reduce the number of generated columns.

We generated benchmark instances for the PDPTWMCI. Our results on these instances show that our pure forward labeling algorithm outperforms the bidirectional variant, due to the increased complexity of the bidirectional labeling algorithm that requires additional resources. Comparing our algorithm to the variant without any of the proposed acceleration strategies shows that including the acceleration strategies solves to proven optimality 46 additional instances (over our set of 638 tested instances), and the computational time decreases by a factor of 6.61. This impact is even bigger for harder instances, for which the computational times decrease by a factor of more than 50. We evaluated the performance of our algorithm and derived managerial insights by studying the impact of three important parameter values, namely, the percentage of incompatible items for the different incompatibility types, the number of cleaning stations, and the cleaning time. The biggest impact in complexity, solution cost and number of cleanings is obtained when changing the percentage of incompatible items. Increasing the percentage of incompatible items decreases the complexity of the problem, where we see that compartment incompatibilities are more complex than vehicle incompatibilities. With regards to the cost of the solutions, simultaneous incompatibilities result in fewer vehicles required than simultaneous-and-subsequent incompatibilities. Comparing compartment incompatibilities with vehicle incompatibilities, we see that compartment incompatibilities require less vehicles than vehicle incompatibilities, and less cleanings in cases where cleaning can remove existing incompatibilities. Adding more cleaning stations results in an increase in complexity, a decrease in the number of vehicles and traveled distance and an increase in the number of cleanings. Increasing the fixed and variable cleaning time reduces the complexity of the problem, requires more vehicles and distance traveled and results in a decrease in the number of cleanings.

A Correctness of the labeling algorithm with acceleration strategies

In this section, we provide additional details on our labeling algorithm in combination with the different acceleration strategies. First, we discuss two special route structures which are integral to the understanding of our proof. Then, we prove that the dominance criteria for our labeling algorithm with all the acceleration strategies is correct in the sense of selective pricing. Finally, we make a note on the complexity of combining different acceleration strategies.

A.1 Special route structures

The variant of selective pricing that we employ in our algorithm for the PDPTWMCI uses the fact that the following two specific route structures cannot be part of an optimal solution (or there exists another optimal solution that does not use routes with that structure). Non-optimality of these structures relies on the triangle inequality for cost and time (c_{ij} and t_{ij}).

Structure 1: Routes with a cleaning station directly before the destination depot or a cleaning station directly after the origin depot, i.e., routes of the form $(0, ..., j, 2n+1)$ or of the form $(0, j, ..., 2n+1)$ with $j \in W$. Visiting cleaning station j is not necessary regarding feasibility of these routes and has a higher cost than the corresponding routes without that visit to cleaning station j. The set of all such routes is denoted by Ω^1.

Structure 2: Routes visiting only delivery nodes between a cleaning station and the destination depot or visiting only pickup nodes between the origin depot and a cleaning station, i.e., routes of the form $(0, ..., j, i_1^-, ..., i_l^-, 2n+1)$ with $i_1^-, ..., i_l^- \in D, j \in W$ or of the form $(0, i_1^+, ..., i_l^+, j, ..., 2n+1)$ with $i_1^+, ..., i_l^+ \in P, j \in W$. Again, visiting cleaning station j is not necessary regarding feasibility of these routes and has a higher cost than the corresponding routes without that visit to cleaning station j. The set of all such routes is denoted by Ω^2.

For the variant with selective pricing, we work with a set of routes Ω' that is a restricted set of routes compared to the problem definition, but a relaxed set of routes compared to the optimal routes, i.e., $\bar{\Omega} \subseteq \Omega' \subseteq \Omega$. In our BPC with forward labeling, the set Ω' does not contain any routes with *Structure 1* and may contain some of the routes with *Structure 2*. Routes with *Structure 1* are eliminated in the pricing by the reduction of the arc set (Section 3.2.3), while the elimination of (some of the) routes with *Structure 2* is due to the interaction between the reduced arc set and subset dominance for the sets of onboard items and related to the selective-pricing paradigm. The set $\bar{\Omega}$ does not contain any routes with *Structure 1* or *Structure 2*, i.e., $\bar{\Omega} \cap (\Omega^1 \cup \Omega^2) = \emptyset$.

A.2 Proof of the proposed labeling algorithm

Recall that our variant of selective pricing requires to identify at least one route in $r \in \Omega'$ with negative reduced cost $\tilde{c}_r < 0$, only if there exists an optimal route $r \in \bar{\Omega}$ with negative reduced cost $\tilde{c}_r < 0$.

Proposition 1. Conditions (15) yield a valid dominance rule (in the sense of selective pricing) for the forward labeling algorithm when the arc reduced costs satisfy the delivery triangle inequality.

Proof. The proof builds upon related proofs for PDP variants. Specifically, we show that for every completion E_2 of $R(F_2)$ to a path $R_2 \in \bar{\Omega}$, there exists a completion E_1 of $R(F_1)$ to a path $R_1 \in \Omega'$ such that $\tilde{c}(R_1) \leq \tilde{c}(R_2)$. Note that is sufficient to consider completions to *optimal* paths $R_2 \in \bar{\Omega}$ due to selective pricing.

We first show how we obtain E_1 from E_2. A completion E in the PDPTWMCI is characterized by three components: the sequence of nodes visited E^R, the (sequence of) compartments E^M on which items are loaded (or from which they are unloaded) at the customer nodes, and the sets of compartments that are cleaned at each visit to a cleaning station. Let $E_1^R := E_2^R \setminus \{i \in D | i-n \in O(F_2) \setminus O(F_1)\}$, with $O(F_2) = \cup_{m \in M} O^m(F_2)$ and $O(F_1) = \cup_{m \in M} O^m(F_1)$, i.e., the node sequence E_1^R is obtained from the sequence E_2^R by removing all visits to delivery nodes associated with items that are onboard for label F_2 but not for label F_1. With that definition, note that E_1^R visits all cleaning stations that are visited in E_2^R in the same sequence. The sequence of compartments E^M is defined as follows. At each customer node $i \in E_1^R$, the associated item is loaded to (pickup) or unloaded from (delivery) compartment m, where $\sigma(m)$ is the compartment this items is loaded to/unloaded from in completion E_2, i.e., items are loaded in the compartments corresponding with the permutation specified in Conditions (15). Let H_2^j be the set of compartments that are cleaned at some cleaning station $j \in E_2^R$ in completion E_2. The set of compartments H_1^j that are cleaned at the corresponding visit to j in completion E_1 is defined as follows. If for R_1 there are no existing incompatibilities in any of the compartments upon arriving at j, then we do not clean any compartment, i.e., $H_1^j = \emptyset$ (the corresponding cleaning time equals δ_1). Otherwise, if (i) $\delta_2 = 0$ we clean all compartments that are empty and (ii) if $\delta_2 > 0$ we clean all compartments (according to permutation σ) that are cleaned in H_2^j and that have an existing incompatibility upon arriving at j on R_1. With this definition, the set of cleaned compartments in H_1^j is a subset of the cleaned compartment in H_2^j, and ensures that whatever is feasible in terms of incompatibility for R_2 is also feasible for R_1. In addition, $|H_1^j| \leq |H_2^j|$, i.e., the time required for cleaning at j is never smaller for E_2 than for E_1, and the corresponding extension to cleaning station j can be performed by the labeling algorithm, i.e., it is in the respective set of extensions $\mathcal{H}^f(F, j)$ with F representing the partial path up to the predecessor of j according to E_1.

We can now show that if $R_2 \in \bar{\Omega}$, then $R_1 \in \Omega'$. Given Condition (15a), the fact that $|H_1^j| \leq |H_2^j|$ for each cleaning station $j \in E_2^R$, and the triangle inequality for travel times, path R_1 respects time windows. Conditions (15c) and (15d) ensure that on R_1 each item is completed at most once and its pickup node is visited before its corresponding delivery node with. Conditions (15d) together with the definition of the sequence E^M guarantee that the capacity constraints are respected. Conditions (15e) and (15f) together with the definition of H_1^j for all cleaning stations $j \in E_2^R$ ensure that whatever is compatible for R_2 is also compatible for R_1, so that R_1 respects all restrictions related to item incompatibilities.

Finally, the reduced cost of R_1 does not exceed that of R_2, i.e., $\tilde{c}(R_1) \leq \tilde{c}(R_2)$, due to Condition (15b) and the delivery triangle inequality. □

There is one case that requires additional commenting. Let R_2 be a feasible path, but $R_2 \in \Omega^2$, i.e., $R_2 := (0, ..., j, i_1^-, ..., i_l^-, 2n+1)$, $i_1^-, ..., i_l^- \in D, j \in W$. Thus, $R_2 \notin \bar{\Omega}$. Consider now the situation that $\{i_1^+, ..., i_l^+\} \subseteq (O(F_2) \setminus O(F_1))$. Then, $R_1 := (0, ..., j, 2n+1)$, i.e., $R_1 \in \Omega^1$. Accordingly, R_1 cannot be generated by the forward labeling algorithm because arcs between cleaning stations and the destination depot are not contained in the reduced arc set as defined in Section 3.2.3. Therefore, it might seem that F_2 is wrongly dominated by F_1. Indeed, R_2 may have negative reduced cost but there exists no negative reduced cost path resulting from any completion of F_1 to a feasible path in Ω'. Nevertheless, while R_2 could be in the set Ω', it is not an optimal route, i.e., $R_2 \notin \bar{\Omega}$. Therefore, path R_2 is properly discarded according to selective pricing.

A.3 A note on the complexity of developing strategies in the labeling algorithm

For the studied problem, developing strategies in the labeling algorithm is not trivial. In fact, it appears logical to remove some additional arcs, namely those between two cleaning stations, and to eliminate extensions to cleaning stations when no empty compartment has any incompatibility. However, these two strategies are not compatible with our proposed labeling algorithm and we explain why in the following.

Inter-station arcs. In an optimal solution, routes do not contain two cleaning stations subsequently (or there exists a corresponding optimal solution in which one of these stations is removed). Therefore, it seems straightforward to also remove inter-station arcs, i.e., (i, j) such that $i, j \in W$. However, in combination with subset dominance for the sets of onboard items, this yields wrongly dominated labels. In particular, routes of the following form may be wrongly discarded in the forward labeling: $(0, ..., j, i_1^-, ..., i_l^-, k, ..., 2n+1)$ with $i_1^-, ..., i_l^- \in D, j, k \in W, j \neq k$. Correspondingly, routes of the form $(0, ..., j, i_1^+, ..., i_l^+, k, ..., 2n+1)$ with $i_1^+, ..., i_l^+ \in P, j, k \in W, j \neq k$ may be wrongly discarded in the backward labeling. Preliminary tests have shown that our algorithm performed best when using subset dominance for the sets of onboard items, rather than without it and removing inter-station arcs.

Extensions to empty compartments with no incompatibility. In the set of pareto-optimal cleanings, if there are empty compartments and all of them have no incompatibility, we still perform an extension to cleaning stations. While this might not seem necessary, it yields incorrectly dominated labels when using subset dominance for the sets of incompatible items. In particular, a completion of such a label may visit additional cleaning stations that could not be visited by a completion to the dominating label due to less (possibly zero) incompatibilities of the latter. Such completions may yield routes $r \in \bar{\Omega}$ that can be part of an optimal solution, but might never be generated if we remove extensions to cleaning stations when there are no incompatibilities. Preliminary tests have shown that using subset dominance for the sets of incompatible items together with these additional extensions performed much better than the variant that disregards the extensions and uses equality dominance for the sets of incompatible items.

B Backward labeling algorithm

There is a lot of analogy between the forward and backward labeling algorithms. The differences are presented here.

Backward reduced costs. In the backward labeling algorithm, we define the reduced cost of an arc $(i, j) \in A$ as $\tilde{c}_{ij}^b := c_{ij} - \frac{\tilde{\pi}_i^b}{2} - \frac{\tilde{\pi}_j^b}{2}$, where $\tilde{\pi}_i^b := \pi_i$ if $i \in D$ and $\tilde{\pi}_i^b := 0$ otherwise. Then, the reduced cost of a route r can be computed as $\tilde{c}_r = \sum_{(i,j)\in A(r)} \tilde{c}_{ij}^b$. If the reduced cost matrix does not respect the pickup triangle inequality, which is defined as $\tilde{c}_{ij}^b + \tilde{c}_{jk}^b \geq \tilde{c}_{ik}^b$ for $i, k \in V, j \in P$, we use the procedure proposed by Ropke and Cordeau (2009) adapted by Gschwind et al. (2018) to modify the matrix to ensure the pickup triangle inequality.

Backward label resources. A backward label B represents a backward partial path $R(B) = (\eta(B), \ldots, 2n+1)$ starting at node $\eta(B)$ and ending at the destination depot. In a label $B = (\eta(B), t(B), c(B), U(B), (l^m(B))_m, (O^m(B))_m, (I_c^m(B))_m, (I_v^m(B))_m)$, the following components differ from their forward counterpart:

- $\eta(B)$, the *first* visited node of the partial path;

- $t(B)$, the *latest feasible start* of the service time at node $\eta(B)$;
- $U(B)$, the set of *unreachable* items, where an item is unreachable if it has already been delivered and picked up on the partial path, or if its delivery node cannot be reached time-window feasible;
- $O^m(B)$, the set of items onboard compartment $m \in M$, i.e., the items that have been delivered and loaded into compartment m and are not yet picked up on the partial path.

The initial backward label is $B_{2n+1} = (2n+1, \overline{w}_{2n+1}, 0, \emptyset, (0)_m, (\emptyset)_m, (\emptyset)_m, (\emptyset)_m)$.

Backward resource extension functions. Extending a label B against the orientation of arc $(i, \eta(B)) \in A$ only leads to a feasible solution if, analogous to Equations (2) for the forward counterpart, one of the following conditions holds:

$$i \in P \text{ and } i \in \bigcup_{m\in M} O^m(B), \tag{16a}$$

$$i \in D \text{ and } i-n \notin U(B) \cup \bigcup_{m\in M} O^m(B) \cup \bigcup_{m\in M} I_v^m(B) \text{ and } \sum_{m\in M}\sum_{j\in O^m(B)} (u^{v1}_{i-n,j} + u^{v1}_{j,i-n}) = 0, \tag{16b}$$

$$i \in W \text{ and } \exists m \in M \text{ such that } O^m(B) = \emptyset, \tag{16c}$$

$$i = 0 \text{ and } \bigcup_{m\in M} O^m(B) = \emptyset. \tag{16d}$$

Extending a label B against the orientation of arc $(i, \eta(B))$ might lead to multiple feasible extensions. This set of extensions is given by $\mathcal{H}^b(i,B)$ and is defined analogously to the forward counterpart in (14) as

$$\mathcal{H}^b(i,B) = \begin{cases} \{m \in M | i \in O^m(B)\} \times \{\emptyset\} & \text{if } i \in P, \quad (17a)\\ \{m \in M | i-n \notin I_c^m(B) \\ \quad \text{and } \sum_{j\in O^m(B)} (u^{c1}_{i-n,j} + u^{c1}_{j,i-n}) = 0 \\ \quad \text{and } (\nexists\, m' \in M \text{ with } m' < m, \\ \quad O^{m'}(B) = O^m(B) = \emptyset, \\ \quad I_c^{m'}(B) = I_c^m(B) \\ \quad \text{and } I_v^{m'}(B) = I_v^m(B))\} \times \{\emptyset\} & \text{if } i \in D, \quad (17b)\\ \{(m_1, \emptyset)\} & \text{if } i \in W, \\ & \quad I_c^m(B) \cup I_v^m(B) = \emptyset \\ & \quad \forall m \in M \text{ with } O^m(B) = \emptyset, \quad (17c)\\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C \neq \emptyset & \text{if } i \in W, \delta_2 = 0, \exists m \in M \\ \qquad \text{and } O^m(B) = \emptyset\ \forall m \in \text{C} & \quad \text{with } O^m(B) = \emptyset \\ \qquad \text{and } O^m(B) \neq \emptyset\ \forall m \in M \backslash C\Big\} & \quad \text{and } I_c^m(B) \cup I_v^m(B) \neq \emptyset, \quad (17d)\\ \{m_1\} \times \{C \in \mathscr{P}(M) | C \neq \emptyset & \text{if } i \in W, \delta_2 > 0, \exists m \in M \\ \qquad \text{and } (O^m(B) = \emptyset \text{ and} & \quad \text{with } O^m(B) = \emptyset \\ \qquad I_c^m(B) \cup I_v^m(B) \neq \emptyset\ \forall m \in C)\} & \quad \text{and } I_c^m(B) \cup I_v^m(B) \neq \emptyset, \quad (17e)\\ \{(m_1, \emptyset)\} & \text{if } i = 0. \quad (17f) \end{cases}$$

For each extension $(h,H) \in \mathcal{H}^b(i, \eta(B))$, the REFs that create label $B^{(h,H)}$ are analogous to their forward counterpart (4) and write as follows:

$$\eta(B^{(h,H)}) = i, \tag{18a}$$

$$t(B^{(h,H)}) = \begin{cases} \min\{\overline{w}_i, t(B) - t_{i,\eta(B)} - \delta_1 - |H| \cdot \delta_2\} & \text{if } i \in W, \\ \min\{\overline{w}_i, t(B) - t_{i,\eta(B)}\} & \text{otherwise,} \end{cases} \tag{18b}$$

$$c(B^{(h,H)}) = c(B) + \tilde{c}^b_{i,\eta(B)}, \tag{18c}$$

$$U(B^{(h,H)}) = \begin{cases} U(B) \cup \{i\} \cup \{j \in P | t(B^{(h,H)}) - t_{j+n,i} < \underline{w}_{j+n}\} & \text{if } i \in P, \\ U(B) \cup \{j \in P | t(B^{(h,H)}) - t_{j+n,i} < \underline{w}_{j+n}\} & \text{otherwise,} \end{cases} \tag{18d}$$

$$l^m(B^{(h,H)}) = \begin{cases} l^m(B) - q_j & \text{if } m = h, \\ l^m(B) & \text{otherwise,} \end{cases} \tag{18e}$$

$$O^m(B^{(h,H)}) = \begin{cases} O^m(B) \setminus \{i\} & \text{if } i \in P \text{ and } i \in O^m(B), \\ O^m(B) \cup \{i-n\} & \text{if } i \in D \text{ and } m = h, \\ O^m(B) & \text{otherwise,} \end{cases} \tag{18f}$$

$$I^m_c(B^{(h,H)}) = \begin{cases} I^m_c(B) \cup \{j \in P | u^{c2}_{j,i-n} = 1\} & \text{if } i \in D \text{ and } m = h, \\ \emptyset & \text{if } i \in W \text{ and } m \in H, \\ I^m_c(B) & \text{otherwise,} \end{cases} \tag{18g}$$

$$I^m_v(B^{(h,H)}) = \begin{cases} I^m_v(B) \cup \{j \in P | u^{v2}_{j,i-n} = 1\} & \text{if } i \in D \\ \{j \in P | u^{v2}_{jk} = 1, k \in \cup_{m' \in M} O^{m'}(B)\} & \text{if } i \in W \text{ and } m \in H, \\ I^m_v(B) & \text{otherwise.} \end{cases} \tag{18h}$$

A label $B^{(h,H)}$ is kept if it does not violate the time windows or the capacity constraints.

Backward dominance. The dominance conditions for the backward labeling algorithm correspond to Conditions (15) of the forward labeling algorithm, where the forward labels F_1 and F_2 are replaced by the backward labels B_1 and B_2, respectively, and Inequality (15a) is replaced by $t(B_1) \geq t(B_2)$. The validity of the dominance can straightforwardly be adapted from the proof of the forward labeling algorithm.

C Bidirectional labeling algorithm

The pricing problem can also be solved using a bidirectional labeling algorithm instead of a pure forward or a pure backward labeling algorithm (see Aerts-Veenstra et al., 2024; Gschwind et al., 2018, for state-of-the-art bidirectional algorithms for PDPs). In this section, we present a bidirectional labeling algorithm. First, we present the necessary modifications to the mono-directional variants. Then, we provide the merge procedure.

C.1 Modifications to the mono-directional labeling algorithms

In this section, we explain the modifications required to the two mono-directional variants, i.e., to the forward labeling algorithm as presented in Section 3.1 and to the backward labeling algorithm as presented in Appendix B, when applying a bidirectional labeling algorithm. We refer to F and B for the forward and backward labels.

In the the bidirectional labeling algorithm, we need the following additional resources:

- $\mathcal{P}^m_c(F)$, the set of items *previously onboard compartment* $m \in M$, i.e., the set of items that have been onboard compartment $m \in M$ (still onboard or already *delivered* in the partial path) since the last cleaning of compartment m on the partial path or the start of the partial path if compartment m has not been cleaned on the partial path;
- $\mathcal{P}^m_v(F)$, the set of items *previously onboard the vehicle*, i.e., the set of items that have been onboard the vehicle (still onboard or already *delivered* in the partial path) since the last cleaning

of compartment m on the partial path or the start of the partial path if compartment m has not been cleaned on the partial path;

- $\mathcal{P}_c^m(B)$, the set of items *previously onboard compartment* $m \in M$, i.e., the set of items that have been onboard compartment $m \in M$ (still onboard or already *picked up* in the partial path) before the first cleaning of compartment m on the partial path or the end of the partial path if compartment m has not been cleaned on the partial path;
- $\mathcal{P}_v^m(B)$, the set of items *previously onboard the vehicle*, i.e., the set of items that have been onboard the vehicle (still onboard or already *picked up* in the partial path) before the first cleaning of compartment m on the partial path or the end of the partial path if compartment m has not been cleaned on the partial path;
- $S(B)$, the set of *completed* items, where an item is completed if it has already been delivered and picked up on the partial path;
- $\mathcal{W}(F)$, the set of *cleaned compartments* at the last visited node $\eta(F)$ of the partial path (trivially $\emptyset$ whenever $\eta(F)$ is not a cleaning station);
- $\mathcal{W}(B)$, the set of *cleaned compartments* at the first visited node $\eta(B)$ of the partial path (trivially $\emptyset$ whenever $\eta(B)$ is not a cleaning station).

Note that resources $\mathcal{W}(F)$ and $\mathcal{W}(B)$ are not necessary and can be obtained from the extension, but they are kept to facilitate the notation for the merge procedure. In the initial labels F_0 and B_{2n+1}, we have $\mathcal{P}_c^m(F_0) = \mathcal{P}_v^m(F_0) = \mathcal{P}_c^m(B_{2n+1}) = \mathcal{P}_v^m(B_{2n+1}) = (\emptyset)_m$ and $S(B_{2n+1}) = \mathcal{W}(F_0) = \mathcal{W}(B_{2n+1}) = \emptyset$.

Again, extending a label leads to multiple feasible extensions. However, the set of extensions as defined in the monodirectional versions needs to be modified. In particular, we always need to allow extensions to cleaning stations in which the set of cleaned compartments is empty, because it might be necessary to merge with a partial path of the opposite direction that did not have any incompatibility in any of its empty compartments. Formally, the set $\mathcal{H}^f(i, F)$ in the bidirectional variant is defined as

$$\mathcal{H}^f(i,F) = \begin{cases} (11), & \\ (3b), & \\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C = \emptyset & \text{if } \; i \in W, \delta_2 = 0, \\ \qquad \text{or } \big(O^m(F) = \emptyset \; \forall m \in C & \\ \qquad \text{and } O^m(F) \neq \emptyset \; \forall m \in M \backslash C\big)\Big\} & \\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C = \emptyset & \text{if } \; i \in W, \delta_2 > 0, \\ \qquad \text{or } \big(\forall m \in C : O^m(F) = \emptyset \text{ and} & \\ \qquad I_c^m(F) \cup I_v^m(F) \neq \emptyset \;\big)\Big\} & \\ (3d). & \end{cases}$$

Analogously,

$$\mathcal{H}^b(i,B) = \begin{cases} (17a), & \\ (17b), & \\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C = \emptyset & \text{if } \; i \in W, \delta_2 = 0, \\ \qquad \text{or } \big(O^m(B) = \emptyset \; \forall m \in C & \\ \qquad \text{and } O^m(B) \neq \emptyset \; \forall m \in M \backslash C\big)\Big\} & \\ \{m_1\} \times \Big\{C \in \mathscr{P}(M) | C = \emptyset & \text{if } \; i \in W, \delta_2 > 0, \\ \qquad \text{or } \big(\forall m \in C : O^m(B) = \emptyset \text{ and} & \\ \qquad I_c^m(B) \cup I_v^m(B) \neq \emptyset\big)\Big\} & \\ (17f). & \end{cases}$$

For each forward extension $(h, H) \in \mathcal{H}^f(F, j)$, the REFs of the additional resources are given by

$$\mathcal{P}_c^m(F^{(h,H)}) = \begin{cases} \mathcal{P}_c^m(F) \cup \{j\} & \text{if } j \in P \text{ and } m = h, \\ \emptyset & \text{if } j \in W \text{ and } m \in H, \\ \mathcal{P}_c^m(F) & \text{otherwise,} \end{cases} \tag{21a}$$

$$\mathcal{P}_v^m(F^{(h,H)}) = \begin{cases} \mathcal{P}_v^m(F) \cup \{j\} & \text{if } j \in P, \\ \cup_{m' \in M} O^{m'}(F) & \text{if } j \in W \text{ and } m \in H, \\ \mathcal{P}_v^m(F) & \text{otherwise,} \end{cases} \tag{21b}$$

$$\mathcal{W}(F^{(h,H)}) = H. \tag{21c}$$

REF (21a) updates the set of items previously onboard compartment $m \in M$, i.e., if an item is picked up it is added to the set of previously onboard items of compartment m if it is loaded into compartment m and if compartment m is cleaned all previously onboard items are removed from the set. REF (21b) adds items that are picked up to the sets of items previously onboard the vehicle $\mathcal{P}_v^m(F^{(h,H)})$ for all compartments $m \in M$, while it resets $\mathcal{P}_v^m(F^{(h,H)})$ for all compartments that are cleaned to the set of items currently onboard any of the compartments. REF (21c) updates the cleaned compartments. Similarly, for each backward extension $(h, H) \in \mathcal{H}^b(i, B)$, the additional REFs are

$$\mathcal{P}_c^m(B^{(h,H)}) = \begin{cases} \mathcal{P}_c^m(B) \cup \{i - n\} & \text{if } i \in D \text{ and } m = h, \\ \emptyset & \text{if } i \in W \text{ and } m \in H, \\ \mathcal{P}_c^m(B) & \text{otherwise,} \end{cases} \tag{22a}$$

$$\mathcal{P}_v^m(B^{(h,H)}) = \begin{cases} \mathcal{P}_v^m(B) \cup \{i - n\} & \text{if } i \in D, \\ \cup_{m' \in M} O^{m'}(B) & \text{if } i \in W \text{ and } m \in H, \\ \mathcal{P}_v^m(B) & \text{otherwise,} \end{cases} \tag{22b}$$

$$S(B^{(h,H)}) = \begin{cases} S(B) \cup \{i\} & \text{if } i \in P, \\ S(B) & \text{otherwise,} \end{cases} \tag{22c}$$

$$\mathcal{W}(B^{(h,H)}) = H. \tag{22d}$$

In the forward case, dominance of a label F_1 over another label F_2 residing at the same node $\eta(F_1) = \eta(F_2)$ requires

$$\mathcal{P}_c^m(F_1) \subseteq \mathcal{P}_c^{\sigma(m)}(F_2), \qquad \forall m \in M, \tag{23a}$$

$$\mathcal{P}_v^m(F_1) \subseteq \mathcal{P}_v^{\sigma(m)}(F_2), \qquad \forall m \in M, \tag{23b}$$

in addition to Conditions (15). The backward dominance is modified accordingly.

C.2 Merge procedure

The forward and backward labels are extended only up to a half-way point, which is defined on the time resources $t(F)$ and $t(B)$. The determination of the half-way point is done in a dynamic fashion (see Tilk et al. (2017)).

The merge of a forward label F and a backward label B at the same node $i = \eta(F) = \eta(B)$ is feasible if

$$\begin{cases} t(F) \le t(B) + \delta_1 + |\mathcal{W}(F)| \cdot \delta_2 & \text{if } i \in W, \\ t(F) \le t(B) & \text{otherwise,} \end{cases} \tag{24a}$$

$$U(F) \cap S(B) = \emptyset, \tag{24b}$$

$$\begin{cases} O^m(F) = O^{\sigma(m)}(B_p) & \text{if } i \in P \\ O^m(F_p) = O^{\sigma(m)}(B) & \text{if } i \in D \\ O^m(F) = O^{\sigma(m)}(B) & \text{otherwise} \end{cases} \quad \forall m \in M, \tag{24c}$$

$$I_c^m(F) \cap \mathcal{P}_c^{\sigma(m)}(B) = \emptyset, \quad \forall m \in M, \tag{24d}$$

$$\mathcal{P}_c^m(F) \cap I_c^{\sigma(m)}(B) = \emptyset, \quad \forall m \in M, \tag{24e}$$

$$I_v^m(F) \cap \mathcal{P}_v^{\sigma(m)}(B) = \emptyset, \quad \forall m \in M, \tag{24f}$$

$$\mathcal{P}_v^m(F) \cap I_v^{\sigma(m)}(B) = \emptyset, \quad \forall m \in M, \tag{24g}$$

$$\begin{cases} \sigma(m) \in \mathcal{W}(B) & \text{if } m \in \mathcal{W}(F) \\ \sigma(m) \notin \mathcal{W}(B) & \text{otherwise} \end{cases} \quad \forall m \in M, \tag{24h}$$

where F_p and B_p are the predecessors of labels F and B, respectively. Here, (24a) ensures feasibility with respect to the time resource, (24b) guarantees that an item is picked up and delivered at most once, (24c) requires that all onboard items are completed, (24d)–(24g) guarantees feasibility with respect to item incompatibilities, and (24h) requires that the same compartments (according to the permutation σ) are cleaned if the merge node i is a cleaning station.

The reduced cost of a feasible route $r \in \Omega$ is computed as

$$\tilde{c}_r = c(F) + c(B) + \sum_{i \in \{O(F) \cap O(B)\}} \pi_i + \sum_{\substack{i \in \{O(F) \cup O(B)\} \setminus \\ \{O(F) \cap O(B)\}}} \frac{\pi_i}{2}, \tag{25}$$

where $O(F) := \bigcup_{m \in M} O^m(F)$ and $O(B) := \bigcup_{m \in M} O^m(B)$. The last two terms tackle the difference in the reduced costs of the arcs in the forward and backward labelings (see Gschwind et al., 2018).

References

Marjolein Aerts-Veenstra, Marilène Cherkesly, and Timo Gschwind. A unified branch-price-and-cut algorithm for multicompartment pickup and delivery problems. Transportation Science, 58(5):1121–1142, 2024.

Agostinho Agra, Marielle Christiansen, and Alexandrino Delgado. Mixed integer formulations for a short sea fuel oil distribution problem. Transportation Science, 47(1):108–124, 2013.

Juan J Alcaraz, Luis Caballero-Arnaldos, and Javier Vales-Alonso. Rich vehicle routing problem with last-mile outsourcing decisions. Transportation Research Part E: Logistics and Transportation Review, 129:263–286, 2019.

C Archetti, LC Coelho, MG Speranza, and P Vansteenwegen. Beyond fifty years of vehicle routing: Insights into the history and the future. European Journal of Operational Research, 2025.

M. Battarra, M. Monaci, and D. Vigo. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. Computers & Operations Research, 36(11):3041–3050, 2009.

Maria Battarra, Jean-François Cordeau, and Manuel Iori. Chapter 6: Pickup-and-delivery problems for goods transportation. In Vehicle Routing: Problems, Methods, and Applications, Second Edition, pages 161–191. SIAM, 2014.

Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. TOP, 15(1):1–31, 2007.

Raquel Bernardino and Ana Paias. The family traveling salesman problem with incompatibility constraints. Networks, 79(1):47–82, 2022.

Alberto Ceselli, Giovanni Righini, and Matteo Salani. A column generation algorithm for a rich vehicle-routing problem. Transportation Science, 43(1):56–69, 2009.

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

Marilène Cherkesly, Guy Desaulniers, Stefan Irnich, and Gilbert Laporte. 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, 2016.

Marielle Christiansen, Kjetil Fagerholt, Nikolaos P Rachaniotis, Ingeborg Tveit, and Marte Viktoria Øverdal. A decision support model for routing and scheduling a fleet of fuel supply vessels. In Computational Logistics: 6th International Conference, ICCL 2015, Delft, The Netherlands, September 23-25, 2015, Proceedings 6, pages 46–60. Springer, 2015.

Leandro C Coelho, Jacques Renaud, and Gilbert Laporte. Road-based goods transportation: a survey of real-world logistics applications from 2000 to 2015. INFOR: Information Systems and Operational Research, 54 (2):79–96, 2016.

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

Luciano Costa, Claudio Contardo, Guy Desaulniers, and Diego Pecin. Selective arc-ng pricing for vehicle routing. International Transactions in Operational Research, 28(5):2633–2690, 2020.

Faheng Deng, Hu Qin, Jiliu Li, and Chun Cheng. The pickup and delivery problem with time windows and incompatibility constraints in cold chain transportation. Transportation Science, 57(2):289–572, 2023.

Ulrich Derigs, Jens Gottlieb, Jochen Kalkoff, Michael Piesche, Franz Rothlauf, and Ulrich Vogel. Vehicle routing with compartments: applications, modelling and heuristics. OR Spectrum, 33:885–914, 2011.

Guy Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations Research, 58(1):179–192, 2010.

Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. Operations Research, 64(6):1388–1405, 2016.

Guy Desaulniers, Diego Pecin, and Claudio Contardo. Selective pricing in branch-price-and-cut algorithms for vehicle routing. EURO Journal on Transportation and Logistics, 8(2):147–168, 2019.

Umut Ermağan, Barış Yıldız, and F Sibel Salman. A learning based algorithm for drone routing. Computers & Operations Research, 137:105524, 2022.

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

Kjetil Fagerholt and Marielle Christiansen. A combined ship scheduling and allocation problem. Journal of the Operational Research Society, 51(7):834–842, 2000.

Stefan Faldum, Sarah Machate, Timo Gschwind, and Stefan Irnich. Partial dominance in branch-price-and-cut algorithms for vehicle routing and scheduling problems with a single-segment tradeoff. OR Spectrum, 46(4): 1063–1097, 2024.

Elise Foss, Trine N. Myklebust, Henrik Andersson, and Marielle Christiansen. A multi-product maritime inventory routing problem with undedicated compartments. In Ana Paias, Mario Ruthmair, and Stefan Voß, editors, Computational Logistics, pages 3–17, Cham, 2016. Springer International Publishing.

Michel Gendreau, Daniele Manerba, and Renata Mansini. The multi-vehicle traveling purchaser problem with pairwise incompatibility constraints and unitary demands: A branch-and-price approach. European Journal of Operational Research, 248(1):59–71, 2016.

Timo Gschwind, Stefan Irnich, Ann-Kathrin Rothenbächer, and Christian Tilk. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research, 266(2):521–530, 2018.

Baobin Huang, Lixin Tang, Roberto Baldacci, Gongshu Wang, and Defeng Sun. A metaheuristic algorithm for a locomotive routing problem arising in the steel industry. European Journal of Operational Research, 308 (1):385–399, 2023.

Lars Magnus Hvattum, Kjetil Fagerholt, and Vinícius Amaral Armentano. Tank allocation problems in maritime bulk shipping. Computers & Operations Research, 36(11):3051–3060, 2009.

Mads Jepsen, Björn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research, 56(2):497–511, 2008.

Rahma Lahyani, Leandro C Coelho, Mahdi Khemakhem, Gilbert Laporte, and Frédéric Semet. A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. Omega, 51:1–10, 2015.

Gilbert Laporte and Yves Nobert. A branch and bound algorithm for the capacitated vehicle routing problem. OR Spektrum, 5:77–85, 1983.

Daniele Manerba and Renata Mansini. A branch-and-cut algorithm for the multi-vehicle traveling purchaser problem with pairwise incompatibility constraints. Networks, 65(2):139–154, 2015.

Andrea Mor and Maria Grazia Speranza. Vehicle routing problems over time: a survey. 4OR, 18:129–149, 2020.

Ibrahim Muter, Jean-François Cordeau, and Gilbert Laporte. A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. Transportation Science, 48(3):425–441, 2014.

Johan Oppen, Arne Løkketangen, and Jacques Desrosiers. Solving a rich vehicle routing and inventory problem using column generation. Computers & Operations Research, 37(7):1308–1317, 2010.

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

Axel Parmentier, Rafael Martinelli, and Thibaut Vidal. Electric vehicle fleets: scalable route and recharge scheduling through column generation. Transportation Science, 57(3):631–646, 2023.

Sophie N. Parragh, Jorge Pinho de Sousa, and Bernardo Almada-Lobo. The dial-a-ride problem with split requests and profits. Transportation Science, 49(2):311–334, 2015.

Masoud Rabbani, Razieh Heidari, Hamed Farrokhi-Asl, and Navid Rahimi. Using metaheuristic algorithms to solve a multi-objective industrial hazardous waste location-routing problem considering incompatible waste types. Journal of Cleaner Production, 170:227–241, 2018.

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

Budi Santosa, Rita Damayanti, and Biswajit Sarkar. Solving multi-product inventory ship routing with a heterogeneous fleet model using a hybrid cross entropy-genetic algorithm: a case study in Indonesia. Production & Manufacturing Research, 4(1):90–113, 2016.

Maximilian Schiffer, Michael Schneider, Grit Walther, and Gilbert Laporte. Vehicle routing and location routing with intermediate stops: a review. Transportation Science, 53(2):319–343, 2019.

Kanchana Sethanan and Rapeepan Pitakaso. Differential evolution algorithms for scheduling raw milk transportation. Computers and Electronics in Agriculture, 121:245–259, 2016.

Christian Tilk, Ann-Kathrin Rothenbächer, Timo Gschwind, and Stefan Irnich. 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, 2017.

Daniel Yamín, Guy Desaulniers, and Jorge E. Mendoza. The electric vehicle routing and overnight charging scheduling problem on a multigraph. INFORMS Journal on Computing, 37(4):808–830, 2025.