Partial Dominance in Branch-Price-and-Cut Algorithms for Vehicle Routing and Scheduling Problems with a Single-Segment Tradeoff

Stefan Faldum[a], Sarah Machate[b], Timo Gschwind[b,*], Stefan Irnich[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*
[b] *Chair of Logistics, Faculty of Business Studies and Economics, RPTU Kaiserlautern-Landau, Gottlieb-Daimler-Straße 42, D-67663 Kaiserlautern, Germany.*

Abstract

For many variants of vehicle routing and scheduling problems solved by a branch-price-and-cut (BPC) algorithm, the pricing subproblem is an elementary shortest-path problem with resource constraints (SPPRC) typically solved by a dynamic-programming labeling algorithm. Solving the SPPRC subproblems consumes most of the total BPC computation time. Critical to the performance of the labeling algorithms and thus the BPC algorithm as a whole is the use of effective dominance rules. Classical dominance rules rely on a pairwise comparison of labels and have been used in many labeling algorithms. In contrast, partial dominance describes situations where several labels together are needed to dominate another label, which can then be safely discarded. In this work, we consider SPPRCs, where a linear tradeoff describes the relationship between two resources. We derive a unified partial dominance rule to be used in ad hoc labeling algorithms for solving such SPPRCs as well as insights into its practical implementation. We introduce partial dominance for two important variants of the vehicle routing problem, namely the electric vehicle routing problem with time windows with a partial recharge policy and the split-delivery vehicle routing problem with time windows (SDVRPTW). Computational experiments show the effectiveness of the approach, in particular for the SDVRPTW, leading to an average reduction of 20% of the total BPC computation time, with savings of 30% for the more difficult instances requiring more than 600 seconds of computation time.

Keywords: branch-price-and-cut, partial dominance, column generation, labeling algorithm, vehicle routing and scheduling

1. Introduction

For many variants of the *vehicle routing problem* (VRP, Toth and Vigo, 2014), *branch-and-price* (BP, Desaulniers *et al.*, 2005) based algorithms constitute the leading exact solution methodology (Costa *et al.*, 2019). A BP algorithm is a branch-and-bound algorithm in which the lower bounds are computed by column generation. Column generation is an iterative procedure that can tackle linear programs containing a huge number of variables. At each iteration, it solves a *restricted master problem* (RMP) comprising only a subset of the variables of the original linear program and one or several pricing subproblems to dynamically generate missing variables with negative reduced cost or to prove that no such variable exists. Cutting planes are added to strengthen the linear relaxations giving rise to a *branch-price-and-cut* (BPC) algorithm. For details on the theory of BPC, we refer to (Barnhart *et al.*, 1998; Lübbecke and Desrosiers, 2005).

The master program is often an extended set-partitioning or set-covering formulation for selecting the best routes, while the pricing subproblem is an elementary *shortest-path problem with resource constraints*

*Corresponding author.
Email addresses: stfaldum@uni-mainz.de (Stefan Faldum), sarah.machate@rptu.de (Sarah Machate), gschwind@rptu.de (Timo Gschwind), irnich@uni-mainz.de (Stefan Irnich)

(SPPRC) typically solved by dynamic-programming labeling algorithms (for an overview of SPPRCs and labeling algorithms, see Irnich and Desaulniers, 2005). In a labeling algorithm, partial paths are gradually extended in a network from a given source o to a sink d (the origin and destination depot in the context of VRPs) seeking for a resource feasible minimum-cost o-d-path. The partial paths are represented by labels that store information on the accumulated resource consumption up to the endpoint of the partial paths. Herein, resources are quantities necessary to compute the reduced cost and, e.g., the load onboard and the start of the service, at the end of the partial path. In particular, resources are used to decide on the feasibility of partial paths. The propagation of the labels along the arcs of the network is performed with the help of *resource extension functions* (REFs, Irnich, 2008). Crucial for the performance of a labeling algorithm is the dominance relation between partial paths. The dominance relation is a relation between labels (i.e., partial paths) used to identify and discard those that cannot lead to an optimal solution of the SPPRC. Dominance is typically realized by comparing the resource values of the labels and avoids the enumeration of all feasible partial paths that can be found in the network.

Classical dominance relations that have been used in many labeling algorithms rely on a pairwise comparison of labels. Informally speaking, if one of the labels is *worse* than the other, it can be safely discarded. Formally, we assume that the network (V, A) with source $o \in V$ and sink $d \in V$ are given. For a partial path p, i.e., an o-i-path ending at some vertex $i \in V$, any i-d-path q path that provides a feasible o-d-path $r = (p, q)$ is called a *feasible extension* of p. Now, pairwise dominance between two partial paths can be characterized with the help of feasible extensions as follows:

Proposition 1. *(Pairwise Dominance)*
Let p_1 and p_2 be two different partial paths ending at the same vertex $i \in V$. If, for each feasible extension q_2 of p_2,

(i) the path q_2 is also a feasible extension of p_1, i.e., $r_1 = (p_1, q_2)$ is feasible

(ii) or there exists a feasible extension q_1 of p_1 (the extension q_1 is allowed to differ from q_2) so that $r_1 = (p_1, q_1)$ is feasible,

and r_1 has a smaller or identical reduced cost than r_2, then the partial path p_2 is dominated.

Because the conditions (i) and (ii) in Proposition 1 are difficult to verify for two arbitrary partial paths p_1 and p_2, labeling algorithms typically test sufficient conditions, so-called *dominance rules*. Dominance rules do not directly consider extensions, but they compare of the resource values of the labels of p_1 and p_2.

In contrast to pairwise dominance, *partial dominance* describes the situation that several labels together are, informally speaking, *better* than another label which can then be safely discarded. A formal characterization of partial dominance is given in the following proposition.

Proposition 2. *(Partial Dominance)*
Let $\mathcal{P}$ be a set of partial paths and $p_2 \notin \mathcal{P}$ be another partial path all ending at the same vertex $i \in V$. If, for each feasible extension q_2 of p_2, there exists a $p_1 \in \mathcal{P}$ such that the condition (i) or the condition (ii) of Proposition 1 is fulfilled and the resulting o-d-path r_1 has a smaller or identical reduced cost than $r_2 = (p_2, q_2)$.

The notion of partial dominance reflects the concept that any path $p_1 \in \mathcal{P}$ generally fulfills the dominance conditions in Proposition 1 only for some extensions q_2 of p_2, i.e., p_1 partially dominates p_2. Considering all the paths in $\mathcal{P}$ together, all extensions q_2 of p_2 are dominated so that p_2 can be discarded.

In the VRP literature, partial dominance has been employed in different situations like SPPRCs with a tradeoff between resources, handling relaxations of the elementarity condition in SPPRCs, and other use cases (see Section 2). The focus of this paper is on partial dominance for SPPRCs with a tradeoff between resources and, in particular, on the most basic type of such a tradeoff, i.e., a linear tradeoff with a single linear piece as depicted in Figure 1. It shows the feasible domain of two resources x and y for a partial path represented by label F. For the resource x, the minimum feasible resource consumption is F^a and maximum feasible resource consumption is F^b. Likewise, it is F^l and F^u for the resource y. For both resources, smaller values are preferable. The tradeoff function f characterizes the nature of the tradeoff: Whenever we allow resource x to increase by one unit (recall that smaller values are preferable), the consumption of resource y can be decreased by m units, i.e., the slope of f is $-m$.

The contributions of this paper are the following. We provide a formal characterization of linear tradeoffs between resources with a single linear piece within SPPRCs and derive a unified partial dominance rule to be used in ad hoc labeling algorithms for the solution of corresponding SPPRCs as well as insights on its practical implementation. Furthermore, we exemplify the application of partial dominance for two important variants of the VRP with a linear tradeoff in their SPPRC pricing subproblems, namely the *electric vehicle routing problem with time windows* (EVRPTW) with a partial recharge policy and the *split-delivery vehicle routing problem with time windows* (SDVRPTW). Finally, we report an extensive computational analysis of using partial dominance compared to the classical pairwise dominance for the EVRPTW and the SDVRPTW on their standard benchmarks.

The remainder of the paper is structured as follows. In Section 2, we categorize different use cases in which partial dominance has been used within SPPRC subproblems and review the corresponding literature. Section 3 provides the theoretical analysis of partial dominance for linear tradeoffs between resources and details its application to the EVRPTW and the SDVRPTW. Our computational study on the EVRPTW and the SDVRPTW is reported in Section 5. Final conclusions are drawn in Section 6.

2. Literature Review

Partial dominance has been used for different types of SPPRC subproblems. We categorize into *linear tradeoff functions* and *non-linear tradeoff functions* describing the relationship of two resources, partial dominance in SPPRC with *elementarity* constraints, and *other forms* of partial dominance.

Linear Tradeoff Functions. First, SPPRC subproblems that have a tradeoff between two resources can benefit from partial dominance. This type of dominance started with the works (Ioachim *et al.*, 1998, 1999) for airplane flight scheduling. In this application, the only resources are time and reduced cost such that a partial path can be completely characterized by a continuous, piecewise linear tradeoff function. The seminal paper of Desaulniers *et al.* (1998) clarifies when and how such linear tradeoffs between resources result from resource constraints in extensive path-based formulations solved with column-generation algorithms: Resource constraints in the master program referring to resource levels at vertices or at arcs result in linear node/vertex and arc costs. For continuous, piecewise linear tradeoff functions, Desaulniers and Villeneuve (2000) developed an implicit form of partial dominance for the respective SPPRC subproblems: Whenever the minimum of several tradeoff functions of a set of paths is completely below the tradeoff function of another path, this path can be eliminated. Such continuous, piecewise linear tradeoff functions between two resources arise in several vehicle scheduling and routing problems. Examples of this form of partial dominance used for solving SPPRC subproblems are the VRP with soft time windows (Liberatore *et al.*, 2010), the time-window assignment VRP (Spliet and Gabor, 2015) and (Spliet *et al.*, 2018), the active-passive VRP (Tilk *et al.*, 2018), the time-dependent *VRP with time windows* (VRPTW) (Lera-Romero *et al.*, 2020), and SDVRPTW and linear weight-related cost (Luo *et al.*, 2017).

The handling of piecewise defined functions per label is somewhat inconvenient, in particular when the number of pieces is not bounded (by a small instance-independent number). For the EVRPTW with partial recharges (Desaulniers *et al.*, 2016b) and the SDVRPTW (Desaulniers, 2010), there is only one linear piece which includes the case of a single point. The EVRPTW and the SDVRPTW are the two VRP variants that we consider in the computational analysis, see Section 4. In a follow up publication on a variant of the inventory-routing problem, Desaulniers *et al.* (2016a) apply similar solution methods, because also this problem can be modeled with REFs with not more than one linear piece.

Even if we did not find works suggesting partial dominance for some applications, it should also be applicable when solving SPPRC subproblems of the following problems: the dial-a-ride problem with ride-time constraints (Gschwind and Irnich, 2015), the synchronized pickup and delivery problem (Gschwind, 2015), the truck-and-trailer VRP with quantity-dependent transfer times (Rothenbächer *et al.*, 2018), and the VRP with partial outsourcing (Baller *et al.*, 2020). This list should not be considered complete.

Non-Linear Tradeoff Functions. Also non-linear tradeoffs between some resource and reduced cost have been considered. Comparing piecewise defined non-linear tradeoff functions and herewith establishing a partial dominance is more intricate but still beneficial. In the VRPTW with convex inconvenience cost functions (He *et al.*, 2019), a customer-specific tradeoff function is defined over the customer's time windows.

Elementarity. Partial dominance improves dominance for relaxations of the elementary SPPRC: For the SPPRC with 2-cycle elimination (Houck *et al.*, 1980; Kohl *et al.*, 1999), i.e., short cycles of the form (i, j, i), two labels that have mutually different predecessor vertices can dominate another label by resource values. For the SPPRC with k-cycle elimination, i.e., all cycles of length up to k are forbidden, Irnich and Villeneuve (2006) generalize this partial dominance. Up to six labels with different predecessor sequences are needed for 3-cycle elimination. For the general case, Irnich and Villeneuve prove that not more than $k!(k-1)!$ different predecessor sequences are needed. For a SPPRC with combined 2-cycle and k-cycle elimination, Bode and Irnich (2014) further generalize the partial dominance. Such a combined cycle elimination occurs in BPC algorithms for the capacitated arc routing problem when branching decisions and k-cycle free subproblems defined on the street network are considered simultaneously. For the SPPRC subproblem of the minimum latency problem, Bulhões *et al.* (2018) derive a partial dominance for the *ng*-route relaxation of the subproblem. In this context, a first label that dominates a second label with respect to all resources but not with respect to the *ng*-route restrictions (it cannot be extended to one or several customers to which the second label can be extended) does however partially dominate the second. This partial dominance results in so-called dominated extensions, i.e., vertices to which an extension of the second label can be safely avoided. Full dominance occurs when a label cannot be extended either due to the set of dominated extensions or due to *ng*-route restrictions. These ideas were further exploited by Costa *et al.* (2021) where even stronger arc-*ng*-route dominance rules were proven in the context of selective pricing (Desaulniers *et al.*, 2019).

Other Forms of Partial Dominance. We found some works using a partial dominance that does not fall into one of the above categories. For the basic multi-compartment VRP, Heßler and Irnich (2023) use labels that represent a partial paths for which (at least) one feasible packing of delivery items into compartment exists. The decision about the concrete packing finally used is, however, postponed until the partial path reaches the destination. In particular, one label generally represents many alternative packings. Partial dominance is used to allow that packings of one label are dominated by some packings of a second label.

Two-arc fixing using reduced costs as suggested by Desaulniers *et al.* (2020) is an acceleration technique for solving SPPRC pricing problems faster without compromising optimality of the overall BPC approach. Sequences of two consecutive arcs that cannot occur in an optimal solution are identified and the labeling algorithm ensures that after traversing the first arc of the sequence, the label is not extended along the second arc. In order to maintain optimality of the pricing, a label must be extended by the information which (second) arcs must not be used for an extension. This information must be taken into account in the dominance. The authors rely on partial dominance, since otherwise the majority of the labels is not comparable with the standard pairwise dominance.

3. Partial Dominance for Linear Decreasing Tradeoff Functions

In this section, we formally define the linear tradeoff between two resources and describe the basic theory for partial dominance between labels. Furthermore, we address general issues for the application of partial dominance within labeling algorithms for SPPRCs. Finally, we provide some details of our implementations.

3.1. Basic Theory

We assume that the linear tradeoff exists between two resources with real-valued resource values x and y. For a partial path and its label F, a proper tradeoff can be described by a segment S in $\mathbb{R}^2$ with endpoints (F^a, F^u) and (F^b, F^l) with $F^a < F^b$ and $F^u > F^l$, see Figure 1. Formally, the segment is the convex hull of its endpoints, i.e., $S = \text{conv}(\{(F^a, F^u), (F^b, F^l)\})$. The intuitive interpretation of the tradeoff

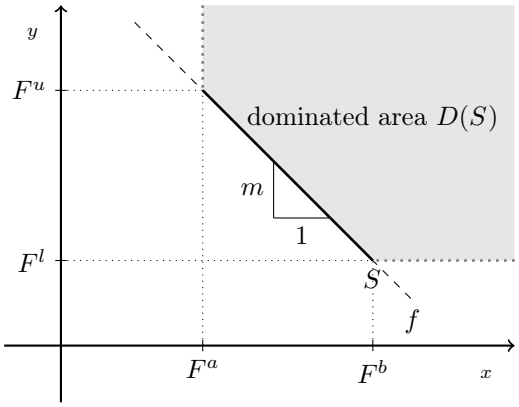


Figure 1: Tradeoff with a single linear piece between two resources x and y.

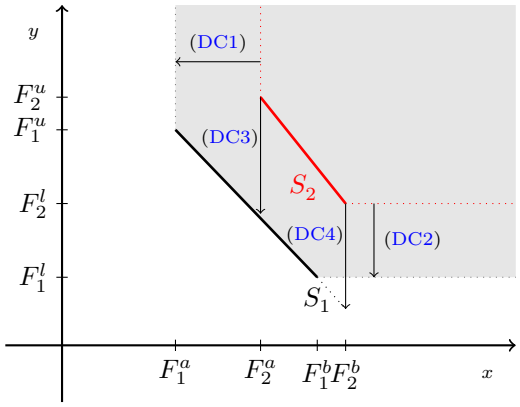


Figure 2: Full dominance of segment S_1 over S_2.

described by the segment is that if resource x is allowed to increase by one unit, the consumption of resource y can be decreased by $m = (F^u - F^l)/(F^b - F^a) > 0$ units.

Every point (x, y) on the segment S can be described with the help of a linear function $f : \mathbb{R}^1 \to \mathbb{R}^1$ defined by

$$y = f(x) = F^u - m \cdot (x - F^a)\,. \tag{1}$$

Note that we have intentionally defined this underlying function f on the entire real space, and not only on the interval $[F^a, F^b]$. Moreover, we can also cope with the case of a one-point segment, i.e., that the two endpoints (F^a, F^u) and (F^b, F^l) are identical, i.e., $F^a = F^b$ and $F^l = F^u$. In this case, we still use (1) by defining the (negative) slope to be $m = 1$. With this definition, also the inverse of f is well-defined, it is $x = f^{-1}(y) = F^a - 1/m \cdot (y - F^u)$.

For both resources, smaller values are preferable so that a combination (x, y) of two resource values (hereafter referred to as a *point*) *dominates* (x', y') if $x \leq x'$ and $y \leq y'$ hold. Graphically speaking, a point (x, y) dominates all points that lie on the top right of it. This dominance between points can be extended to sets. In particular, for the segment S, the *dominated area* is

$$D(S) = \{(x', y') \in \mathbb{R}^2 : \exists (x, y) \in S \text{ with } (x, y) \text{ dominates } (x', y')\},$$

see also see Figure 1.

3.1.1. Full Dominance

We first describe the standard full dominance with respect to the two resources that are traded against each other. Note that a partial path typically has more resources than just these two. As a consequence, the following conditions for full dominance regarding the two resources are only necessary conditions for dominance between labels (see also the dominance for the EVRPTW and the SDVRPTW described in Sections 4.1 and 4.2).

We assume that two labels F_1 and F_2 with corresponding segments S_1 defined by (F_1^a, F_1^u) and (F_1^b, F_1^l) and S_2 defined by (F_2^a, F_2^u) and (F_2^b, F_2^l) are given. Label F_1 *fully dominates* F_2 if $S_2 \subset D(S_1)$. An example where S_2 lies completely in the dominated area of S_1 is illustrated in Figure 2.

Full dominance can be verified using the following four conditions, which must all hold true:

$$F_1^a \leq F_2^a, \tag{DC1}$$
$$F_1^l \leq F_2^l, \tag{DC2}$$
$$F_1^u - (F_2^a - F_1^a) \cdot m_1 \leq F_2^u, \tag{DC3}$$
$$F_1^u - (F_2^b - F_1^a) \cdot m_1 \leq F_2^u - (F_2^b - F_2^a) \cdot m_2, \tag{DC4}$$

Condition (DC1) compares the minimum x-values and checks whether F_1^a lies further on the left than F_2^a. Likewise, condition (DC2) compares the minimum y-values and checks whether F_1^l is below F_2^l. Condition (DC3) compares the values of f_1 and f_2 at $x = F_2^a$. Finally, Condition (DC4) compares the values of f_1 and f_2 at $x = F_2^b$. The four dominance conditions are visualized in Figure 2.

If both segments S_1 and S_2 consist of a single point each, i.e., $F_1^a = F_1^b$ and $F_2^a = F_2^b$, these conditions collapse into the standard dominance of the form $F_1^a \leq F_2^a$ and $F_1^l \leq F_2^l$ between independent resources, i.e., (DC1) and (DC2).

3.1.2. Partial Dominance

We now characterize partial dominance between the two resources for which a tradeoff occurs. Reusing the same notation as in Section 3.1.1, we define that a label F_1 *partially dominates* F_2 if

$$D(S_1) \cap S_2 \neq \varnothing. \tag{2}$$

Recalling that full dominance requires $S_2 \subset D(S_1)$, we speak of a *proper* partial domination if (2) and $S_2 \not\subset D(S_1)$ are fulfilled. In the spirit of Proposition 2, a set $\mathcal{F}$ of labels together *dominates* a label F_2, if

$$S_2 \subset \bigcup_{F_1 \in \mathcal{F}} D(S_1). \tag{3}$$

An important question is now how condition (3) can be tested effectively within a labeling algorithm. Therefore, we analyze the sets $S_2^d := D(S_1) \cap S_2$ of dominated points and their complements $S_2^u := S_2 \backslash D(S_1)$, i.e., the sets of undominated points, in more detail. To this end, we categorize proper partial domination into

- partial dominance *from the left* with $(F_2^a, F_2^u) \in D(S_1)$ and $(F_2^b, F_2^l) \notin D(S_1)$,
- partial dominance *from the right* $(F_2^a, F_2^u) \notin D(S_1)$ and $(F_2^b, F_2^l) \in D(S_1)$, and
- partial dominance *over a central piece* with $(F_2^a, F_2^u) \notin D(S_1)$ and $(F_2^b, F_2^l) \notin D(S_1)$.

Moreover, we distinguish the cases in which $S_1 \cap S_2$ is either

(i) the empty set (*without intersection*),
(ii) a single point (x_s, y_s) (with proper *intersection*), or
(iii) a segment consisting of more than one point (*overlapping*).

Note that in case of overlapping segments, the underlying functions are identical, i.e., $f_1 = f_2$ in case (iii).

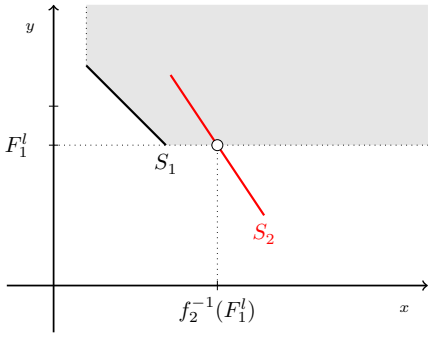
This results in full dominance and seven different types of proper partial dominance as exemplified in Figure 3 and further explained below. We provide identification features regarding the dominance conditions (DC1)–(DC4).

FD: *Full dominance.* All dominance conditions (DC1)–(DC4) are fulfilled.

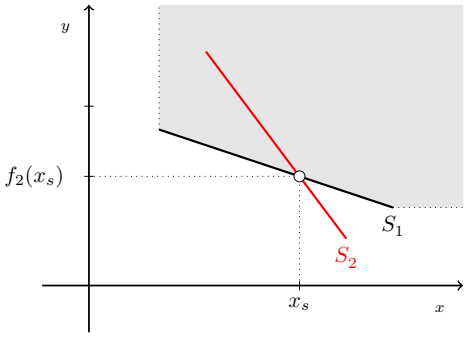
PD1: *Partial dominance from the left without intersection.* Dominance conditions (DC1) and (DC3) are fulfilled, dominance condition (DC2) is violated. In addition, either (DC4) is fulfilled (which includes overlapping segments), or (DC4) is violated and $S_1 \cap S_2 = \varnothing$. See Figure 3a.

PD2: *Partial dominance from the left with intersection.* Dominance conditions (DC1) and (DC3) are fulfilled, dominance condition (DC4) is violated and $S_1 \cap S_2 \neq \varnothing$. Dominance condition (DC2) can either be fulfilled or violated. See Figure 3b.

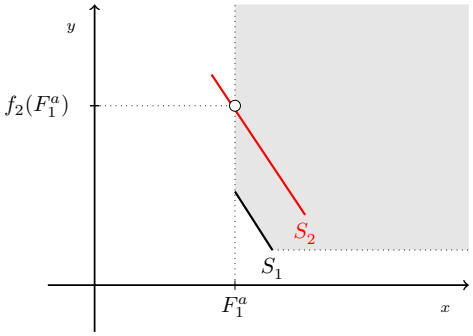
PD3: *Partial dominance from the right without intersection.* Dominance conditions (DC2) and (DC4) are fulfilled, and dominance condition (DC1) is violated. In addition, either (DC3) is fulfilled (which includes overlapping segments), or (DC3) is violated and $S_1 \cap S_2 = \varnothing$. See Figure 3c.



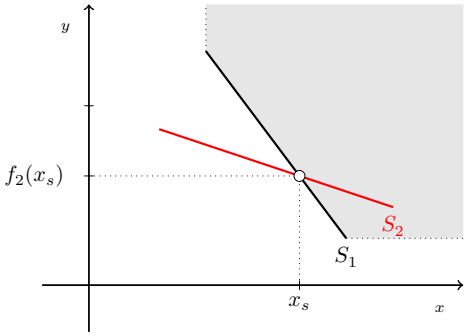
(a) PD1: Dominance from the left w/o intersection.



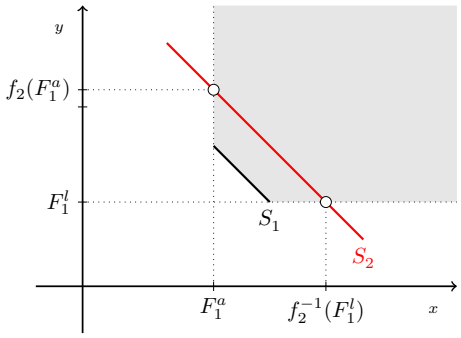
(b) PD2: Dominance from the left w. intersection.



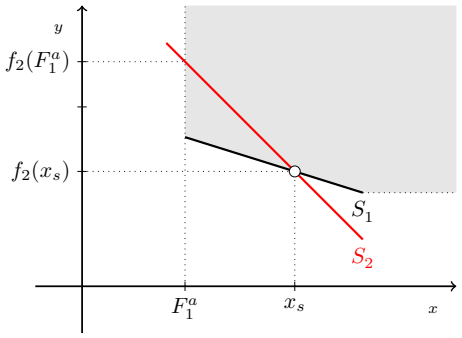
(c) PD3: Dominance from the right w/o intersection.



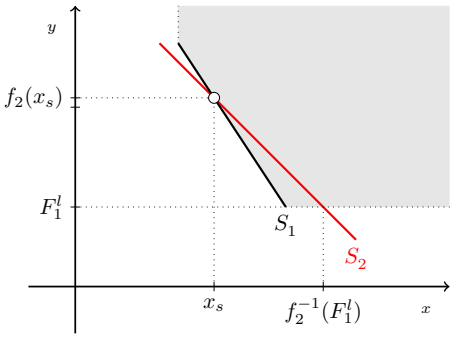
(d) PD4: Dominance from the right w. intersection.



(e) PD5: Dominance over a central piece w/o intersection.



(f) PD6a: Dominance over a central piece w. intersection.



(g) PD6b: Dominance over a central piece w. intersection.

Figure 3: Different cases of partial dominance.

		From the left		*From the right*		*Over a central piece*		
Condition	CD	PD1	PD2	PD3	PD4	PD5	PD6a	PD6b
DC1	✓	✓	✓	✗	–	✗	✗	✗
DC2	✓	✗	–	✓	✓	✗	✗	✗
DC3	✓	✓	✓	–	✗	–[a]	✓	✗
DC4	✓	–	✗	✓	✓	–[a]	✗	✓
$S_1 \cap S_2$	–	(i) ∨ (iii)	(ii)	(i) ∨ (iii)	(ii)	(i) ∨ (iii)	(ii)	(ii)

Table 1: Fulfillment of dominance conditions of different types of (partial) dominance. Symbols have the following meaning: ✓ fulfillment; ✗ non-fulfillment; – fulfillment is not relevant for identification of dominance type; [a] at least one of the conditions (DC3) and (DC4) needs to be fulfilled.

PD4: *Partial dominance from the right with intersection.* Dominance conditions (DC2) and (DC4) are fulfilled, dominance condition (DC3) is violated and $S_1 \cap S_2 \neq \varnothing$. Dominance condition (DC1) can either be fulfilled or violated. See Figure 3d.

PD5: *Partial dominance over a central piece without intersection.* Dominance conditions (DC1) and (DC2) are violated. In addition, either dominance conditions (DC3) and (DC4) are both fulfilled (which includes overlapping segments), or only one of them is fulfilled and $S_1 \cap S_2 = \varnothing$. See Figure 3e.

PD6a and PD6b: *Partial dominance over a central piece with intersection.* The following two cases are possible: In case PD6a, only dominance condition (DC3) is fulfilled, dominance conditions (DC1), (DC2) and (DC4) are violate $S_1 \cap S_2 \neq \varnothing$. See Figure 3f.
In case PD6b, only dominance condition (DC4) is fulfilled, dominance conditions (DC1)–(DC3) are violated $S_1 \cap S_2 \neq \varnothing$. See Figure 3g.

An overview of the relation between the four dominance conditions (DC1)–(DC4) and the above types is given in Table 1.

A label F_2 with a single-point segment S_2 is either fully dominated (FD) or not dominated at all. Proper partial domination is impossible. For testing full dominance, conditions DC1 and DC2 suffice, since conditions DC3 and DC4 are implied by the former. However, labels with a single-point segment can dominate other labels both fully or proper partially.

Two labels with segments that have identical slopes m cannot intersect. Under this assumption, the dominance types PD2, PD4, PD6a, and PD6b with intersection are impossible.

The set of dominated points S_2^d is always a closed interval (segment), i.e., the endpoints belong to the set. In contrast, the complement with respect to the segment, which is the set of undominated points S_2^u (of the segment), is either the empty set, a half-open interval in $\mathbb{R}^2$, or a union of the latter. The seven cases are summarized in Table 2. Hence, the remaining undominated area comprises one or two half-open intervals in $\mathbb{R}^2$ (in the following, we use the term half-open segment). If S_2 is further proper partially dominated by other segments, the undominated set continues to decompose into a union of sub-segments, each of which could be open, half-open (from the left or right), or closed.

It is cumbersome to always distinguish between the four cases. Moreover, additional binary attributes (flags) would be needed to properly classify the cases. Instead, we assume that each undominated set S^u is a closed set. Note first that simply replacing all (half-)open sub-segments by their closure does not impact the correctness on the labeling algorithm, since it does not lead to incorrectly discarded labels. However, partial dominance becomes slightly weaker, since a single point sub-segment might be left over although it is already redundant.

Second, in many applications, the underlying feasible domains of x and y are discrete or can be reduced to discrete sets, e.g., the integers (see examples in Section 4). For example, if all travel times, service times, and time windows are defined by integer values, then a feasible schedule can also be found in the integer domain. In particular, x, F^a, and F^b can be assumed integer in this case. Then, the (half-)open

Types of partial dominance	Set S_2^u of undominated points
PD1	$\text{conv}(\{(f_2^{-1}(F_1^l), F_1^l), (F_2^b, F_2^l)\}) \setminus \{(f_2^{-1}(F_1^l), F_1^l)\}$
PD2	$\text{conv}(\{(x_s, f_2(x_s)), (F_2^b, F_2^l)\}) \setminus \{(x_s, f_2(x_s))\}$
PD3	$\text{conv}(\{(F_2^a, F_2^u), (F_1^a, f_2(F_1^a))\}) \setminus \{(F_1^a, f_2(F_1^a))\}$
PD4	$\text{conv}(\{(F_2^a, F_2^u)), (x_s, f_2(x_s))\}) \setminus \{(x_s, f_2(x_s))\}$
PD5	$\text{conv}(\{(f_2^{-1}(F_1^l), F_1^l), (F_2^b, F_2^l)\}) \setminus \{(f_2^{-1}(F_1^l), F_1^l)\} \uplus \text{conv}(\{(F_2^a, F_2^u), (F_1^a, f_2(F_1^a))\}) \setminus \{(F_1^a, f_2(F_1^a))\}$
PD6a	$\text{conv}(\{(F_2^a, F_2^u), (F_1^a, f_2(F_1^a))\}) \setminus \{(F_1^a, f_2(F_1^a))\} \uplus \text{conv}(\{(x_s, f_2(x_x)), (F_2^b, F_2^l))\}) \setminus \{(x_s, f_2(x_x))\}$
PD6b	$\text{conv}(\{(F_2^a, F_2^u), (x_s, f_2(x_s))\}) \setminus \{(x_s, f_2(x_s))\} \uplus \text{conv}(\{(f_2^{-1}(F_1^l), F_1^l), (F_2^b, F_2^l))\}) \setminus \{(f_2^{-1}(F_1^l), F_1^l)\}$

Table 2: Resulting set of undominated points S_2^u of a label F_2 after proper partial dominance of a label F_1 with segment S_1. For the cases PD2, PD4, PD6a, and PD6b with intersection, let $(x_s, f_1(x_s)) = (x_s, f_2(x_s))$ denote the intersection point of the two segments S_1 and S_2.

intervals $(F^a, F^b]$, $[F^a, F^b)$, and (F^a, F^b) can be replaced by the closed intervals $[F^a + 1, F^b]$, $[F^a, F^b - 1]$, and $[F^a + 1, F^b - 1]$, respectively. Likewise, if the other resource y is integer, a similar procedure can be applied to the (half-)open intervals formed with F^l and F^u. The adaptation of the sets of undominated points for the types PD1–PD6b of partial dominance is straightforward for the integer case. We refer to this as *rounding* in the following.

3.2. Storage and Use of Partial Dominance within Labeling Algorithms

The advantage of the standard full dominance is that, for each pair of labels F_1 and F_2, it suffices to test $S_2 \subset D(S_1)$, i.e., (DC1)–(DC4) just once. If the test fails, the two partial paths are incomparable and this relation does not change in the course of the labeling algorithm. In contrast, if the test (3) fails at one point, the labeling algorithm may generate additional labels such that a new set $\mathcal{F}$ may result in a true test (3), which allows to eliminate F_2. However, repeating the test (3) multiple times in the course of the labeling algorithm is highly undesirable and typically prohibitively time-consuming. Therefore, a labeling algorithm using partial dominance needs to have a mechanism that sequentially compares every two labels F_1 and F_2 only once, and stores in a cumulative fashion the (intermediate) result within the labels.

In general, there are at least the following possibilities to maintain and utilize the dominance information for each partial path:

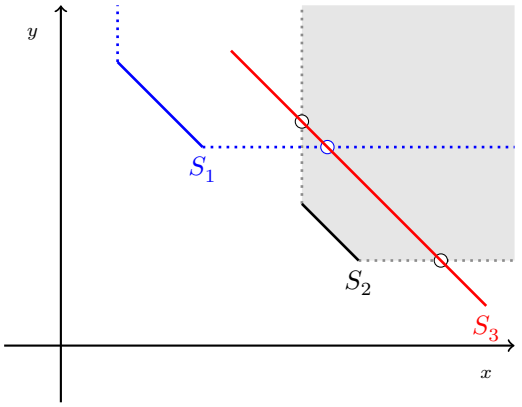
(1) either store one segment,
(2) or store two segments,
(3) or store a variable number of segments; this can be realized by
  (a) either storing and updating the information in the original label F_2,
  (b) or creating (possibly multiple) new labels.

We first elaborate on possibilities (1)–(3) and discuss advantages and disadvantages. We then discuss the two variants (3a) and (3b) to store variable numbers of segments for partial paths.

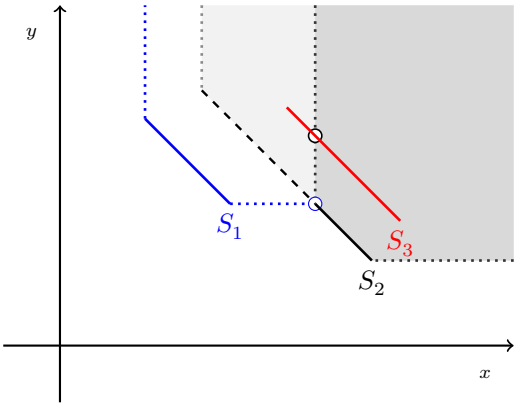
Possibility (1). When a partial path, i.e., the corresponding label, is freshly generated (at initialization) the segment S describes the undominated set $S^u = S$ of points with the help of the attributes F^a, F^b, F^l, and F^u. A consequence of storing a single segment only is that, if a label F_1 partially dominates F_2, then F_2^a, F_2^b, F_2^l, and F_2^u should again describe the resulting set of undominated points S_2^u. This can be realized by directly modifying the values F_2^a, F_2^b, F_2^l, and F_2^u accordingly (see Table 2).

Overall, such an implementation of partial dominance gradually reduces the set of undominated points S_2^u of a label F_2 by replacing S_2^u with $S_2^u \setminus D(S_1)$ whenever a label F_1 partially dominates it (and $S_2^u \setminus D(S_1)$ is a single segment). The new criterion to test whether F_2 becomes redundant, i.e., completely dominated by (one or) several labels in the sense of Proposition 2, is $S_2^u \subset D(S_1)$ for some label F_1.

An immediate advantage of this type of implementation is that no additional information needs to be maintained. However, it has the following disadvantages:



(a) Impact of the order in which S_1 and S_2 are compared with S_3 when partial dominance over a central piece is not performed.



(b) Impact of double bookkeeping when S_1 is first compared with S_2 and afterwards S_2 is compared with S_3.

Figure 4: Examples of partial dominance

(i) It can only rely on dominance from the left (PD1 and PD2) and from the right (PD3 and PD4), because only in these cases a single segment describes the set of undominated points S_2^u. Dominance over central pieces (PD5, PD6a, and PD6b) produces two segments of undominated points and is therefore not compatible with possibility (1). Neglecting partial dominance over central pieces obviously weakens the overall dominance. An additional undesirable effect is the following: The order of pairwise comparisons influences the overall outcome. More precisely, the resulting set of undominated points S^u of a label F can be different for two different orders of comparison with other labels. Figure 4a shows an example with three segments S_1, S_2, and S_3. Segment S_1 partially dominates S_3 from the left, and S_2 partially dominates S_3 over a central piece. If the labeling algorithm first compares S_2 and S_3, it neglects this fact. The subsequent comparison of S_1 and S_3 results in a partial dominance of S_1 over S_3 from the left. The resulting set of undominated points comprises more than half of the original segment S_3. If the order of the two pairwise comparisons is swapped, partial dominance from the left of S_1 over S_3 allows the subsequent partial dominance from the left of S_2 over S_3. In comparison, the remaining set of undominated points of S_3 is much smaller.

(ii) The capability to partially dominate another label can be reduced or lost. Consider the example shown in Figure 4b. It depicts three segments S_1, S_2, and S_3, where segment S_2 fully dominates S_3, and segment S_1 partially dominates S_2 and S_3 from the left. If first S_1 partially dominates S_2, then S_2 is updated to describe the remaining set of undominated points as highlighted by the thick line. Now, the reduced segment S_2 only partially dominates S_3 from the right. Both S_1 and S_2 together are needed to completely dominate S_3.

Possibility (2). A second type of implementation stores two segments: the initial segment S as well as the remaining set S^u of undominated points. We assume that the initial segment is represented by F^a, F^b, F^l, and F^u. Storing S^u can be accomplished by introducing four *additional* attributes $F^{\underline{a}}, F^{\underline{b}}, F^{\underline{l}}$, and $F^{\underline{u}}$. Note that the two attributes $F^{\underline{a}}$ and $F^{\underline{b}}$ are sufficient, because $F^{\underline{u}} = f(F^{\underline{a}})$ and $F^{\underline{l}} = f(F^{\underline{b}})$ and the function f can be expressed with the original attributes.

This type of implementation is denoted *double bookkeeping* in the following, because it keeps the information about both the largest possible segment for dominating other labels as well as the segment of undominated points that we want to become as small as possible (recall that the label is completely dominated once this set is the empty set). Hence, double bookkeeping overcomes the above disadvantage (ii).

However, because the stored information is limited to only two segments, no dominance over central pieces can be performed so that disadvantage (i) remains.

Possibility (3). Storing a variable number of segments allows to exactly describe the initial segment S as well as the sets of dominated and undominated points of a partial path resulting from all possible types of partial dominance (PD1–PD6b), in particular including dominance over a central piece. Formally, if partial domination has been performed with a subset $\mathcal{F}$ of labels $F_1 \in \mathcal{F}$, then the set of undominated points is

$$S_2^u = S_2 \setminus \bigcup_{F_1 \in \mathcal{F}} D(S_1)$$

and may consist of up to $|\mathcal{F}| + 1$ segments. Alternatively, storing the complement $S_2 \cap \bigcup_{F_1 \in \mathcal{F}} D(S_1)$, i.e., the set of dominated points, may require representing $|\mathcal{F}|$ segments. If S_2 is given, dominated points can be reproduced from undominated points, and vice versa. We can formalize the set of undominated points of a partial path with the help of a variable number K of sub-segments $S^{1,u}, \dots, S^{K,u}$ with $S^u = \bigcup_{k=1}^{K} S^{k,u}$.

Partial dominance can then be realized similar to possibilities (1) and (2) by gradually reducing the set of undominated points S^u. This requires updating all sub-segments $S_2^{1,u}, \dots, S_2^{K,u}$ accordingly. In particular, in each update, some sub-segments may remain unchanged, some sub-segments may become empty sets (and can be eliminated), some sub-segments may be reduced (if dominated from the left or from the right), and some sub-segments may decompose into two new sub-segments (if dominated over central pieces).

Overall, this type of representation overcomes both of the above disadvantages (i) and (ii). The drawback is that the number of sub-segments is variable, which is more complex to implement. Even more, it requires the allocation and deallocation of memory during the dominance algorithm slowing down the computational speed in which the dominance algorithm can be completed. Note that the dominance is typically the most time-critical part of a labeling algorithm.

We now comment on the two possibilities to represent variable numbers of segments for partial paths.

Possibility (3a). A first type of implementation maintains the classical one-to-one correspondence between partial paths and labels. All initial and intermediate information about a partial path, its capability to dominate other labels, and its own state of being partially dominated are stored within a single label (the *original* label created at initialization) whose attributes are updated whenever it is partially dominated by other labels. In particular, in each label the segment S is represented and maintained using the original attributes F^a, F^b, F^l, and F^u. Furthermore, the K sub-segments $(S^{k,u})_{k=1}^K$ can be described with $4K$ additional values $(F^{k,\underline{a}}, F^{k,\underline{b}}, F^{k,\underline{l}}, F^{k,\underline{u}})_{k=1}^K$ (again $2K$ values $(F^{k,\underline{a}}, F^{k,\underline{b}})$ suffice, since $F^{k,\underline{l}} = f(F^{k,\underline{b}})$ and $F^{k,\underline{u}} = f(F^{k,\underline{a}})$ and f is known). At initialization, the assignments $K = 1$ and $S^{1,u} = S$ have to be made, i.e., $F^{1,\underline{a}} = F^a, F^{1,\underline{b}} = F^b, F^{1,\underline{l}} = F^l$, and $F^{1,\underline{u}} = F^u$.

The main advantage of possibility (3a) is that all information of a partial path is stored only once (non-redundant, memory efficient). A disadvantage is that it requires a variable-sized representation of the sub-segments of either dominated or undominated points of the segment S. This may be time-critical when labels themselves are stored in dynamically allocated memory and their variable-sized attributes are also stored in dynamically allocated memory. Moreover, the pairwise comparison is more intricate from a implementation point of view.

Possibility (3b). In contrast, some works mention that they create several labels per partial path when convenient for storing the information about partial domination or in the presence of multiple extensions per arc, e.g., when items can alternatively be loaded in different compartments of a vehicle (Cherkesly and Gschwind, 2022; Aerts-Veenstra *et al.*, 2023). An alternative type of implementation could, for example, use labels like with double bookkeeping in possibility (2). Instead of restricting partial dominance to PD1–PD4, partial dominance over central pieces can be established by creating a new copy F_2' of a label F_2 whenever two sub-segments are created from one sub-segment of undominated points. One sub-segment is then stored in the original label F_2 and the other sub-segment in the new copy F_2'. In general, with this implementation, a partial path with set of undominated points $S^u = \bigcup_{k=1}^{K} S^{k,u}$ is represented by K labels, one for each sub-segment $(S^{k,u})_{k=1}^K$. In the spirit of double bookkeeping, each label maintains the initial segment S

using the original attributes F^a, F^b, F^l, and F^u and its corresponding sub-segment $S^{k,u}$ with the additional attributes $F^{\underline{a}}, F^{\underline{b}}, F^{\underline{l}}$, and $F^{\underline{u}}$.

3.3. Implementation Details of Our Labeling Algorithm

Concerning possibilities (3a) and (3b) to represent variable numbers of segments in partial dominance, we see no strong point in possibility (3b) except for being rather simple to code. In comparison, possibility (3a) allows to keep all information about a partial path within a single label.

Regarding the alternatives (1)–(3) to maintain and utilize dominance information, the main question is whether or not partial dominance over a central piece is essential. To answer this question, we recommend to conduct pretests to quantify how often the partial dominance types PD5, PD6a, and PD6b occur compared to partial dominance from the left and from the right. On the basis of such pretests (see also Section 4), we decided not to consider partial dominance over a central piece in our BPC algorithms. The additional computational overhead to cope with a variable number of segments is high. In comparison, the impact caused by additional dominance was low, i.e., we did not see a strong reduction in the number of labels and dominance tests when omitting partial dominance over a central piece.

It is clear that double bookkeeping, i.e., possibility (2), leads to a stronger dominance compared to possibility (1). The necessary addition of the (often integer) attributes $F^{\underline{a}}, F^{\underline{b}}, F^{\underline{l}}$, and $F^{\underline{u}}$ to a label F is rather harmless for a computer implementation; the main effect is a slightly higher memory consumption and, as a consequence, a possible higher number of cache misses.

There is one more refinement related to double bookkeeping that we would like to discuss now. Recall that the resources whose dependency is described by the segment are problem-specific. Independent from a specific problem, the segment of a given label F_i referring to a vertex i is typically used to define the segment of an extension to a vertex j represented by label F_j. REFs $\textsc{Ref}_{ij}$ are used for this purpose (see Section 4 for examples), i.e., $(F_j^a, F_j^b, F_j^l, F_j^u) = \textsc{Ref}_{ij}(F_i^a, F_i^b, F_i^l, F_i^u)$. This offers the opportunity to initialize the set of undominated points S_j^u for a newly created label F_j based on information about partial dominance of its predecessor F_i. The related attributes are computed with the same REFs as

$$(F_j^{\underline{a}}, F_j^{\underline{b}}, F_j^{\underline{l}}, F_j^{\underline{u}}) = \textsc{Ref}_{ij}(F_i^{\underline{a}}, F_i^{\underline{b}}, F_i^{\underline{l}}, F_i^{\underline{u}}).$$

The advantage is that labels F_j with a smaller set of undominated points S_j^u can be eliminated sooner, i.e., with less applications of partial dominance.

In summary, we have chosen possibility (2), i.e., double bookkeeping, together with the refinement that undominated sets of points are propagated with REFs.

4. Two Variants of the VRPTW

In this section, we present two variants of the VRPTW with tradeoff resources and discuss the application as well as specific adaptations of the theory from Section 3 to these variants. First, we formally define the VRPTW as the base problem of the considered variants and introduce the common notation, followed by a description of the two variants, namely the EVRPTW and the SDVRPTW. For convenience, we use the same notation, e.g., for index sets, for all variants even though their definition may be slightly different. The correct meaning should be clear from the context. We then describe the use of partial dominance within the labeling algorithms to solve the pricing subproblems of column-generation approaches for the EVRPTW and the SDVRPTW.

The VRPTW can be defined on a directed graph $G = (V, A)$. The vertex set $V = N \cup \{o, d\}$ comprises the customer vertices N and two copies o and d of the depot. A travel time t_{ij} and routing cost c_{ij} are associated with each arc $(i, j) \in A$. Both travel times and routing costs are assumed to satisfy the triangle inequality. For each vertex $i \in V$, a non-negative demand q_i, a service duration s_i, and a time window $[e_i, l_i]$ in which the service must start are given. We assume $q_o = q_d = s_i = s_i = 0$ for the depot copies o and d. A fleet K of homogeneous vehicles each with a capacity of Q is available at the common depot to serve the customers. The VRPTW is the problem of finding at most $|K|$ vehicle routes such that the total routing cost is minimized, each customer is served by exactly one route, and each route is feasible, i.e., it

is an elementary o-d-path in G satisfying the capacity and time-window constraints. For further details on the VRPTW and its solution by BPC-based approaches, we refer to (Costa *et al.*, 2019).

The EVRPTW is an extension of the VRPTW that takes into account the limited driving range of electric commercial vehicles and the possibility of recharging the vehicle's battery en route at recharging stations. We consider two variants of the EVRPTW that employ a so-called *partial recharging* policy where any amount of energy can be recharged when visiting a recharging station. The two variants differ in the number of allowed recharges per route: at most a *single* (S) or *multiple* (M) recharges are allowed. They are referred to as EVRPTW-S and EVRPTW-M, respectively, in the following. To model these EVRPTW variants, the set R of recharging stations can be added to the vertex set V, i.e., $V = N \cup \{o, d\} \cup R$. Each vehicle in the homogeneous fleet is equipped with an electric battery of maximum capacity B. Moreover, each arc $(i, j) \in A$ is associated with an energy consumption b_{ij}. As described by Desaulniers *et al.* (2016b), we assume that the battery capacity and the energy consumptions are given in recharging time units so that, e.g., B equals the time to fully recharge a completely empty battery. In particular, there is a one-to-one relationship between the recharging duration and the amount of recharged energy, i.e., recharging for Δ units of time increases the *state of charge* (SoC) by exactly Δ units. A route is feasible in the EVRPTW, if it is feasible for the underlying VRPTW (timing constraints now include also recharging times) and the SoC is never negative or greater than the battery's capacity B. In addition, the number of recharges per route must comply with the recharging policy (S or M). Also the EVRPTW has the objective of minimizing the total routing cost. A detailed description of the considered EVRPTW variants and corresponding BPC algorithms for their solution can be found in (Desaulniers *et al.*, 2016b, 2020).

The SDVRPTW is a relaxation of the VRPTW in which customers may be visited more than once, i.e., the customer demands can be satisfied through multiple visits by different vehicles. This also allows finding feasible solutions to instances with customer demands $q_n > Q$. Contrary to the VRPTW, a route in the SDVRPTW is not only characterized by a sequence of customers but also by a corresponding *delivery pattern* ρ which specifies the quantities $d_{\rho n}$ delivered to each customer n visited on the route. A route is capacity-feasible if the sum the quantities $d_{\rho n}$ does not exceed the vehicle capacity Q. Each customer's demand must be fulfilled by the sum of received delivery quantities of all routes. Again, the objective is the minimization of the total routing cost.

Desaulniers (2010) proposed a BPC approach to the SDVRPTW in which only routes with *extreme delivery pattern* have to be explicitly considered. In an extreme delivery pattern, only one customer n receives a *split delivery*, i.e., $0 < d_{\rho n} < q_n$. All other customers receive either a *zero delivery* ($d_{\rho n} = 0$) or a *full delivery* ($d_{\rho n} = q_i$). Routes with general delivery patterns are considered implicitly in the RMP by convex combinations of routes with extreme delivery patterns. For details we refer to (Desaulniers, 2010; Archetti *et al.*, 2011).

4.1. Labeling with Partial Dominance for the EVRPTW

We now show how partial dominance can be used in the labeling algorithm proposed by Desaulniers *et al.* (2016b) to solve the elementary SPPRC pricing problem of the EVRPTW. After visiting a recharging station, there is a tradeoff between time and SoC: the more energy to charge, the later the time, but the higher the SoC. Since the amount to be recharged, i.e., the recharging duration, depends on the part of the partial path after this recharging station, it can only be determined a posteriori. Consequently, the labels must maintain the relevant tradeoff segment of feasible times, which implies different recharging durations and achievable SoCs. If no recharging station has been visited on the partial path, there is no tradeoff, i.e., the segment reduces to a single point. For the sake of brevity, we only briefly discuss the resources and dominance rules used in the original algorithm. We also limit the presentation to the forward part of the labeling, since the backward labeling is similar. For more details on the original labeling algorithm, see (Desaulniers *et al.*, 2016b).

A partial path p starting at the origin depot o and ending at vertex $i \in V$ is represented by a label $F_i = (F_i^{cost}, F_i^{load}, (F_i^{cust_n})_{n \in N}, F_i^{rch}, F_i^{tMin}, F_i^{tMax}, F_i^{rtMax})$. The $6 + |N|$ components of the label have the following meaning:

F_i^{cost}**:** the reduced cost of path p;

F_i^{load}: the accumulated customer demands along path p;

$F_i^{cust_n}$: for each $n \in N$, the number of times customer n has been visited along path p;

F_i^{rch}: the number of recharges performed along path p;

F_i^{tMin}: the earliest service start time at vertex i assuming that, if a recharging station is visited prior to i along p, a minimum recharge that ensures battery feasibility up to i has been performed;

F_i^{tMax}: the earliest service start time at vertex i assuming that, if a recharging station is visited prior to i along p, a maximum recharge respecting time-window feasibility up to i has been performed;

F_i^{rtMax}: with the artificial assumption that recharging is possible at all vertices, F^{rtMax} denotes the maximum possible recharging duration at vertex i assuming that, if a recharging station is visited prior to i along p, a minimum recharge that ensures battery feasibility up to i has been performed. Note that this assumption is used only to propagate the information along the path, but a real recharge never occurs at a customer.

To simplify the notation, we omit the index i of the residing vertex so that we write, e.g., F^{cost} instead of F_i^{cost}.

Instead of modeling the SoC directly, Desaulniers *et al.* implicitly represented it by the maximum feasible additional recharging duration F^{rtMax} (=maximum feasible amount of energy to be recharged) with respect to the earliest service start time F^{tMin} at the current vertex. With this representation, for both tradeoff resources time and maximum possible recharging duration, smaller values are preferable. Therefore, the theory of Section 3 is immediately applicable to realize partial dominance. Moreover, by the above definition of the battery related data (capacity, consumption, and recharging rate), the negative slope of all tradeoff functions is $m = 1$, which simplifies the representation and dominance conditions. As a result,

$$\begin{aligned} F^a &= F^{tMin} \\ F^b &= F^{tMax} \\ F^l &= F^{rtMax} - (F^{tMax} - F^{tMin}) \\ F^u &= F^{rtMax} \end{aligned}$$

so that only the three attributes $F^{tMin}, F^{tMax}, F^{rtMax}$ are needed to describe the tradeoff and the corresponding segment. The classical dominance rule of Desaulniers *et al.* (2016b) for pairwise dominance is as follows. Let F_1 and F_2 be two labels representing partial paths ending at the same vertex i. Label F_1 dominates F_2, if

$$\begin{aligned} F_1^{cost} &\leq F_2^{cost}, \\ F_1^{load} &\leq F_2^{load}, \\ F_1^{cust_n} &\leq F_2^{cust_n} \quad \forall\, n \in N, \\ \text{and (DC1)–(DC3)}&. \end{aligned}$$

In the EVRPTW-S, $F_1^{rch} \leq F_2^{rch}$ is additionally required. Since segments have identical slope with $m_1 = m_2 = 1$, conditions (DC1)–(DC3) imply condition (DC4). Alternatively, one could require conditions (DC1), (DC2), and (DC4), which imply condition (DC3).

Having identical slopes also simplifies partial dominance between two tradeoff segments. First, the two segments can never have a proper intersection (see Section 3.1.2) meaning that only PD1 and PD3 are relevant (recall that we do not employ partial dominance *over a central piece*). Second, also the identification of them is simplified. For PD1, conditions (DC1) and (DC3) are fulfilled (implying (DC4)), while condition (DC2) is violated. For PD3, conditions (DC2) and (DC3) are fulfilled (implying (DC4)), while condition (DC1) is violated. Furthermore, the determination of the undominated points is simplified (see Table 2), since the tradeoff function and its inverse are identical with (negative) slopes given by $m = 1$.

Finally, for time windows and travel times defined by integer values, all battery related data can also be assumed to be integer. Thus, the feasible domains of the tradeoff resources time and SoC can be reduced to integers, and rounding can be applied to the segments in partial dominance as described in Section 3.1.2. The integer assumption is satisfied for the standard benchmark instances of the EVRPTW used in the computational experiments presented in Section 5.

4.2. Labeling with Partial Dominance for the SDVRPTW

In the following, we detail the application of partial dominance to the labeling algorithm of Desaulniers (2010) for pricing routes and associated extreme delivery patterns for the SDVRPTW. Here, the pricing subproblem is an elementary SPPRC combined with the linear relaxation of a bounded knapsack problem. Since there is a dual price for each unit to be delivered to a customer, there is a tradeoff between the two resources vehicle load and reduced cost: The question of how much to deliver to the (unique) customer receiving a split delivery is, in turn, the question of how much to earn from each unit delivered. Similar to the recharging duration in the EVRPTW, the quantity to be delivered is not known when visiting the split customer. Therefore, the label represents this tradeoff with a segment indicating the possible vehicle load (implying the delivery quantity to the split customer) and the achievable reduced cost. If no split delivery has been made along a partial path, there is no tradeoff, resulting in a single-point segment.

For brevity, we only present the attributes of a label and the dominance rules of the original algorithm only for the case of forward labeling (backward labeling is symmetric with respect to the tradeoff). A partial path p from source o to a vertex $i \in V$ is represented by a label $F_i = (F_i^{cost}, F_i^{time}, F_i^{load}, (F_i^{cust_n})_{n\in N}, F_i^{split}, F_i^{sMax}, F_i^{s\pi})$ with $6 + |N|$ components defined as follows:

F_i^{cost}: the reduced cost of path p without the cost for a potential split delivery;

F_i^{time}: the earliest service start time at vertex i;

F_i^{load}: the accumulated customer demands of full deliveries along path p;

$F_i^{cust_n}$: for each $n \in N$, the number of times customer n has been visited along path p;

F_i^{split}: binary indicator whether or not a split delivery has occurred on path p;

F_i^{sMax}: the maximum quantity that can be delivered to the split delivery customer on path p;

$F_i^{s\pi}$: dual price per unit delivered to the split delivery customer on path p.

Smaller values are preferable for both tradeoff resources load and reduced cost. As a result, the theory of Section 3 is also applicable here, and the segment describing the tradeoff in the SDVRPTW is defined by

$$\begin{aligned} F^a &= F^{load} \\ F^b &= F^{load} + F^{sMax} \\ F^l &= F^{cost} - F^{sMax} F^{s\pi} \\ F^u &= F^{cost} \end{aligned}$$

which are given by the four components F^{load}, F^{cost}, F^{sMax}, and $F^{s\pi}$ where the latter describes the negative slope m. However, unlike for the EVRPTW, the negative slopes given by $m = F^{s\pi}$ are generally different for different labels so that a proper intersection is possible. Consequently, no direct simplifications can be made.

The classical rule of Desaulniers (2010) for pairwise dominance is as follows: Let F_1 and F_2 be two labels representing partial paths ending at the same vertex i, then label F_1 dominates F_2, if

$$\begin{aligned} F_1^{time} &\le F_2^{time}, \\ F_1^{split} &\le F_2^{split}, \\ F_1^{cust_n} &\le F_2^{cust_n} \quad \forall\, n \in N, \\ \text{and (DC1)–(DC4)}. & \end{aligned}$$

For partial dominance between labels, all types PD1–PD6b are relevant. Finally, it is known that, for each SDVRPTW instance, there exists an optimal solution in which all delivery quantities are integer, under the precondition that all demand values and the vehicle capacity are integer (Archetti *et al.*, 2006). The integer assumption is satisfied for the standard benchmark instances used in Section 5, so that rounding is applicable as described in Section 3.1.2.

5. Computational Results

In this section, we report an extensive computational study of partial dominance within the BPC algorithms for the EVRPTW and the SDVRPTW. The two BPC algorithms were implemented in `C++` and compiled into 64-bit single-thread code with MS Visual Studio 2019. CPLEX 20.10 with default parameters (except for the time limit and allowing only a single thread) was used to reoptimize the RMPs. All computations were performed on the high performance computing cluster Elwetritsch of the RPTU Kaiserslautern-Landau, consisting of several Intel Xeon Gold 6126 processors running at 2.60GHz. Note that the performance of a single thread on the cluster is comparable to that of a standard desktop processor.

5.1. BPC Algorithms

The BPC algorithms for the EVRPTW and the SDVRPTW share the same code basis, which is adapted to each of the problem variants. The base code uses several acceleration techniques that are well established in the literature. For the EVRPTW, the computational setup of the BPC algorithm is completely identical to that used in (Desaulniers *et al.*, 2020). For the SDVRPTW, a modified setup seems more favorable. Both computational setups are summarized below.

Pricing. To speed up the pricing, we apply bidirectional labeling with a dynamic half-way point using the time as the monotone resource (Tilk *et al.*, 2017), i.e., F^{tMin} for the EVRPTW and F^{time} for the SDVRPTW. Furthermore, instead of solving the strongly $\mathcal{NP}$-hard elementary versions of the SPPRCs, we rely on the well-known ng-path relaxations of them (Baldacci *et al.*, 2011). In our implementation, the neighborhood sizes are set to 14 (EVRPTW) and 4 (SDVRPTW). In addition, we use the concept of unreachable customers who cannot be reached due to their resource levels (Feillet *et al.*, 2004). We use pricing heuristics based on families of arc-reduced networks as proposed by Desaulniers *et al.* (2008), where the networks have a limited number of incoming and outgoing arcs for each customer vertex. The number of arcs chosen in our implementation are 2, 5, 10, 15 (EVRPTW) and 3, 10 (SDVRPTW). For the SDVRPTW, we additionally consider another heuristic pricing strategy that only allows full deliveries and zero deliveries to customers, i.e., split deliveries are disregarded, resulting in an SPPRC without tradeoffs. The latter strategy is combined with the arc-reduced networks. Finally, our labeling algorithms are based on a bucket-based implementation using a one-dimensional bucketing on the time resource (Sadykov *et al.*, 2021).

Valid Inequalities. To strengthen the linear relaxation of the master program, *rounded capacity inequalities* (RCIs, Naddef, 2002) and *subset-row inequalities* (SRIs, Jepsen *et al.*, 2008) are added as two families of valid inequalities. Violated inequalities are separated only at the root node (level zero) for the EVRPTW and up to level one of the search tree for the SDVRPTW. For the SDVRPTW, additional inequalities that upper bound the flow by one on every pair of anti-parallel customer arcs are also dynamically added. The separation procedures add inequalities to the RMP only if they are violated by at least 0.05.

RCIs are robust cuts because their dual prices can be incorporated by modifying the reduced cost of the associated arcs; they do not change the structure of the pricing problem. For the SDVRPTW, RCIs are critical to the overall performance of our BPC algorithm and are, therefore, extensively separated using the extended and greedy shrinking heuristics of Ralphs *et al.* (2003), the route-based algorithm of Archetti *et al.* (2011) for the SDVRPTW, and using an exact mixed-integer programming (MIP) formulation following the ideas of Martinelli *et al.* (2013). The route-based algorithm is invoked only when the shrinking heuristics fail to find a violated RCI. Similarly, the MIP is only used if the route-based algorithm also fails to identify a

violated RCI. To avoid prohibitively long computation times, CPLEX is given a hard time limit of 10 seconds for solving the MIP. For the EVRPTW, RCIs are less important and only the shrinking heuristics are used.

SRIs are non-robust cuts so that, for each active SRI, an additional resource needs to be added in the labeling algorithms making the pricing problem more difficult to solve. Therefore, we use the limited memory variant of the SRIs as proposed by Pecin *et al.* (2017a). Furthermore, as done in most works, we limit ourselves to SRIs with row sets of size three. Our implementation uses the same separation algorithm and vertex memory as presented by (Pecin *et al.*, 2017b).

Branching. Branching is necessary to finally ensure integer solutions. For the EVRPTW, we use the hierarchical branching scheme of Desaulniers *et al.* (2016b), i.e., we branch on (i) the total number of routes, (ii) the total number of recharges, (iii) the total number of recharges at a recharging station, and (iv) the total flow on an arc.

For the SDVRPTW, we use the scheme of Desaulniers (2010), i.e., we branch on (i) the total number of routes, (ii) the total number of visits to a customer, (iii) the total flow on an arc, and (iv) whether two arcs are used in succession.

On all levels, we select a branching variable with fractional value closest to 0.5. The search tree is explored using a best-bound first node selection strategy.

For the SDVRPTW, we additionally apply strong branching as follows. In each node of the search tree, a set of branching candidates is selected by considering a given number of branching variables with the largest fractional values. For each candidate, a rough evaluation of both child nodes is performed by solving the RMP with the corresponding branching constraint without any column generation. The most promising candidate is selected according to the product rule proposed by Achterberg (2007). At the root node, the maximum size of the candidate set is 15, and the size of the candidate set decreases by two for each level of the search tree. However, the minimum size of the candidate set is five. Note that this implementation of strong branching, i.e., no pricing in the evaluation of the candidates, decreases the share of the total time that is spent in pricing and, thus, reduces the impact of partial dominance on the overall performance.

5.2. Instances

Our computational study uses the standard benchmark instances for the EVRPTW and SDVRPTW from the literature. Both instance sets extend the well-known (Solomon, 1987) benchmark for the VRPTW. The original benchmark consists of 56 instances with 100 customers each. For each instance, two smaller instances were derived by considering only the first 25 and 50 customers, respectively. The benchmark consists of six groups with instances characterized by different geographical distributions of the customers, namely clustered (C), random (R), and a mixture of both (RC). Also, different lengths of the scheduling horizon, namely short horizon with narrow time windows (series 1) and long horizon with wide time windows (series 2). In all instances, the common depot hosts an unlimited fleet of homogeneous vehicles with capacities varying between 200 and 1000.

EVRPTW Instances. The Solomon instances were adapted to the EVRPTW by Schneider *et al.* (2014). They introduced 21 randomly positioned recharging stations together with battery capacities for the vehicles, consumption and recharging rates, and modified time windows for some customers. For more details we refer to (Schneider *et al.*, 2014). The EVRPTW benchmark comprises a total of $56 \times 3 = 168$ instances. The online supplement of (Desaulniers *et al.*, 2016b) provides instance files that directly include all necessary parameters in a precomputed and rounded form.

SDVRPTW Instances. Desaulniers (2010) adapted the original Solomon benchmark to the SDVRPTW. To foster split deliveries, the modified instance set considers three smaller values $Q = \{30, 50, 100\}$ for the vehicle capacity for each original instance. The total number of SDVRPTW instances is therefore $56 \times 3 \times 3 = 504$. Travel times and routing costs between all pairs of vertices are set to the Euclidean distance, rounded down to one decimal place. Since the triangle inequality is assumed to hold for both, the resulting travel time and routing cost matrices must be further processed. We follow the approach of Bianchessi and Irnich (2019), who compute shortest paths between all pairs of vertices using the travel times (including customer service times at the tail of each arc) and independently the routing costs.

	Instance size n			
Problem	25	50	100	All
EVRPTW-S	31.07	33,96	33.66	32.87
EVRPTW-M	44.96	47.78	45.98	46.23
SDVRPTW	56.85	63.83	66.13	62.14

Table 3: Percentage of the created labels that have a proper tradeoff.

5.3. Analysis of Ratio of Labels with Tradeoff

In a first experiment, we evaluate the potential of partial dominance for the three VRPTW variants. To this end, we analyze how often feasible labels with a tradeoff are created. Table 3 provides an overview for the three VRPTW variants aggregated by instance size n. It shows the geometric mean of the percentage of labels with a proper tradeoff out of all created labels over the respective instances. The numbers reveal that the differences are mainly between the problem variants rather than between instances of different sizes. For the EVRPTW-S, the share of labels with a tradeoff is only about one-third, while this number is nearly double for the SDVRPTW. With a percentage of labels with a tradeoff of 46.23%, the EVRPTW-M is in between the two variants.

These differences are due to the nature of the considered problems. They become particularly pronounced because of the bidirectional labeling: For the EVRPTW-S, a proper tradeoff only results if the single allowed visit to a recharging station has been performed. With multiple recharging visits allowed, longer routes – all containing visits to recharging stations – become battery feasible in the EVRPTW-M, which increases the share of tradeoff labels. In the bidirectional labeling approach, these long routes are always broken down into a shorter forward and a shorter backward partial path. As a result, tradeoffs occur less often. For the SDVRPTW, the extension to any customer may result in a tradeoff.

Overall, we expect that partial dominance should have a much more positive effect on the performance of the BPC algorithm for the SDVRPTW than for the EVRPTW variants. Of the two latter variants, the EVRPTW-M is expected to benefit more than the EVRPTW-S. On the other hand, the differences in the percentages of labels with a tradeoff are negligible for different instance sizes within the same problem variant. Note that the same holds true also when considering alternative groupings by instance characteristics (e.g., we analyzed the groups C, R, and RC as well as series 1 and 2 without finding significant dependencies).

5.4. Comparison on Identical Pricing Instances

In a second experiment, we evaluate the impact of using partial dominance over classical pairwise dominance within SPPRC labeling algorithms. There are two main opposing effects that affect the performance: On the one hand, partial dominance allows more labels to be eliminated, resulting in fewer total labels and fewer dominance comparisons, the number of which is quadratic in the number of labels. On the other hand, the individual dominance comparisons between labels are more complex in the case of partial dominance. Furthermore, the implementation of partial dominance comes with some additional overhead due to the additional resources required in the labeling. In general, it is not clear a priori, which of the two effects prevails and whether partial dominance pays off at the end.

A fair comparison of two different SPPRC labeling algorithms is delicate, because already one different route generated by one of the algorithms is likely to result in a completely different overall trajectory of the BPC algorithm. To eliminate such effects, which may distort the true behavior, we implemented a special version of the BPC algorithm in which the same dual solution is always passed to both SPPRC labeling algorithms, i.e., to the classical labeling algorithm using only pairwise dominance and to the labeling algorithm using partial dominance. Only the routes generated by one of the algorithms are passed to the RMP. To further reduce possible side effects due to the order in which the different labeling algorithms are executed, we perform an additional *warm-up run* of each algorithm before executing the run for which data is tracked (in particular, we noticed that warm-up runs improve the stability and replicability of the recorded

computation times). In this setting, each individual pricing problem is thus solved four times in a row, and extend the maximal total computation time to four hours per instance. Note that the total computation times achieved in this particular setting do not reflect real-world solution times.

Figure 5 summarizes the comparison for the EVRPTW-S, the EVRPTW-M, and the SDVRPTW. It shows the number of feasible *labels* generated (-L), the number of *dominance* tests performed (-D), and the solution *time* (-T). All three indicators are presented as the ratio of the number of the labeling with partial dominance divided by those of the labeling with classical pairwise dominance. The information in Figure 5 is aggregated as follows. For each instance, we compute the geometric means over the pricing iterations so that each iteration contributes equally. We then take the geometric means over the instances (again, each instance contributes equally). In addition, we present results not only for *all pricing* (AP) iterations but also for *exact pricing* (EP) iterations only. Typically, only very few EP iterations are necessary within the BPC algorithm, but they often consume the majority of the total computation time. Therefore, the information related to EP iterations is better suited to reflect the impact of partial dominance on the BPC algorithms.

In addition, Figure 5 groups the results for different subsets of instances according to their solution time. For example, there are $n = 362$ SDVRPTW instances with a runtime of at least one second (≥ 1) shown in Figure 5c. Finally, the results shown in Figure 5 refer to the root node solution only. The results for the extended root node (root plus cuts) and the full BPC algorithm are very similar so that we omit them.

We highlight three main observations from Figure 5: First, partial dominance is never detrimental for any of the considered VRPTW variants and labeling algorithms. Second, the positive effect of partial dominance is much stronger for the SDVRPTW (Figure 5c) and weaker for the two EVRPTW variants, with the EVRPTW-M (Figure 5b) benefiting more than the EVRPTW-S (Figure 5a). This is in line with the analysis performed in the previous section. Third, the general trends are the same for all three problem variants: The effect of partial dominance is stronger for EP than for AP. More difficult instances, i.e., those with longer solution times, benefit more. The smallest effect (=largest ratios) is obtained for the number of labels, the strongest for the number of dominance tests (recall that their number is quadratic in the number of labels), and the actual savings in computation time lies between the latter two. The explanation is that with the more complex dominance tests, labeling with partial dominance cannot translate the savings in dominance tests to savings in computation times in a one-to-one fashion.

We can quantify the benefits of partial dominance for solving individual pricing problem instances: For the SDVRPTW , about 20% (30%) of the computation time can be saved when considering AP iterations (EP iterations). These numbers increase to approximately 30% (AP) and 40% (EP) for the most difficult instances. For the EVRPTW variants, the savings are much smaller, as only about 2% to 5% of the computation time can be saved on average across all EVRPTW instances. For the more difficult instances, savings of up to 6% (EVRPTW-S) and 10% (EVRPTW-M) are possible in the EP iterations.

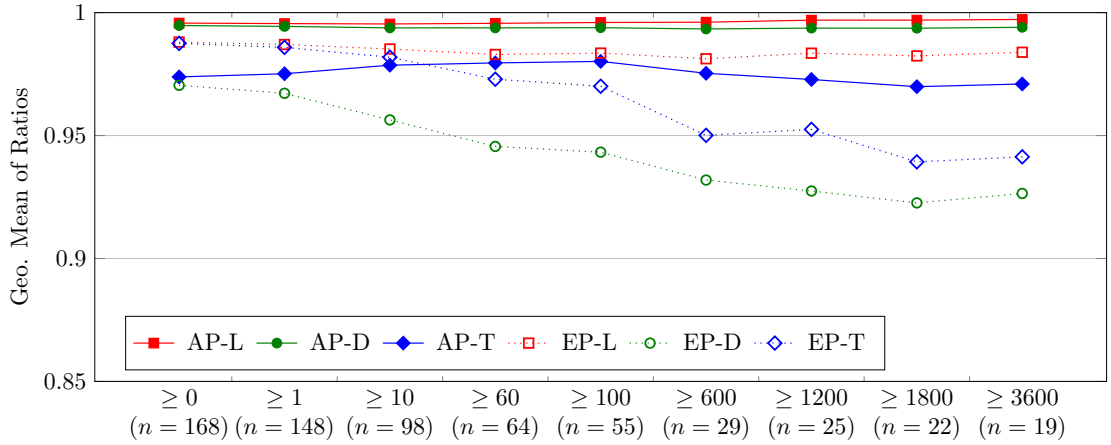
In summary, our results on identical pricing instances suggest that the application of partial dominance should lead to a noticeable/significant improvement in the overall BPC algorithm for the SDVRPTW. For the EVRPTW, however, the effect is expected to be small.

5.5. Comparison on Individual BPC Runs

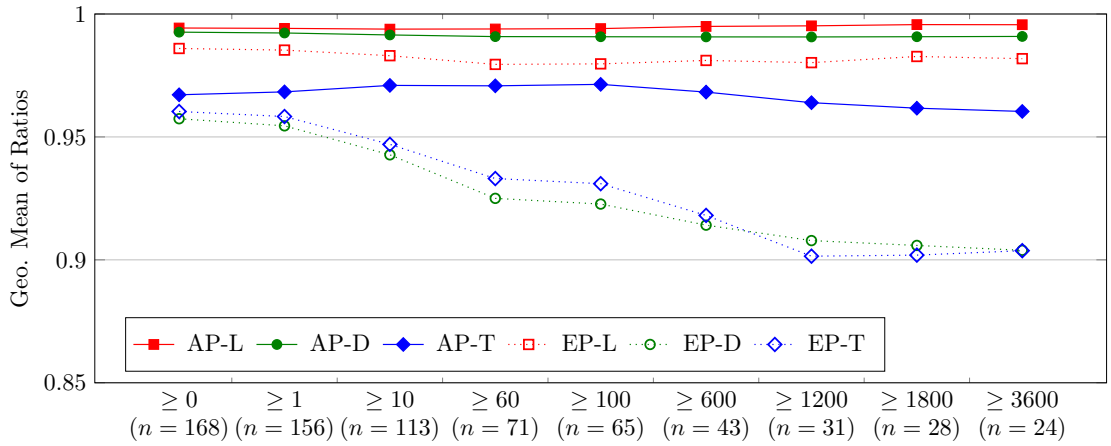
In a final experiment, we evaluate the performance of the BPC algorothm with partial dominance compared to an otherwise identical BPC algorithm that uses only the classical pairwise dominance. To be conform with other works from the literature, we allow up to two hours of computation time for the EVRPTW and one hour for the SDVRPTW. In the light of Section 5.4, we restrict the evaluation to the EVRPTW-M and the SDVRPTW.

The performance profiles of the two BPC algorithms are shown in Figure 6a for the EVRPTW-M and in Figure 6b for the SDVRPTW. The performance profile of an algorithm specifies the number of instances solved by that algorithm within τ times the time taken by the fastest algorithm in an instance-by-instance comparison (Dolan and Moré, 2002).

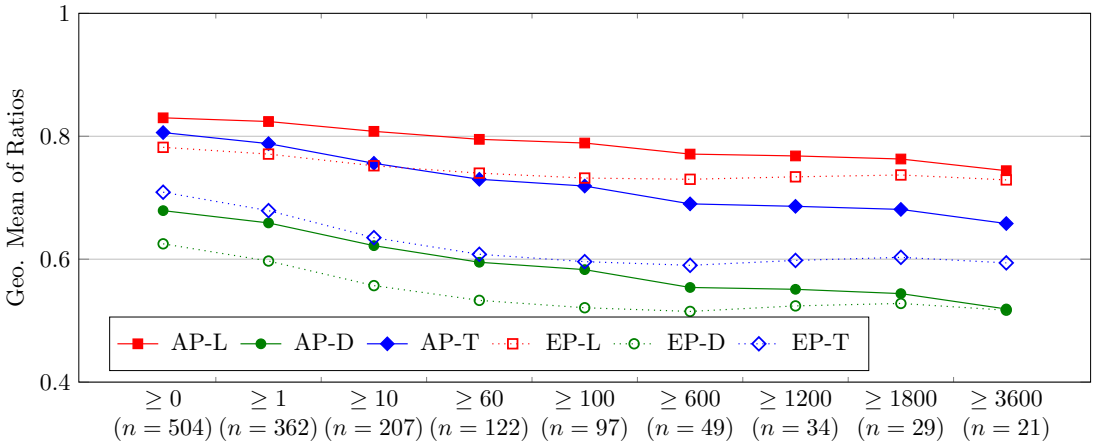
For both variants, the performance profiles reveal that the BPC using partial dominance is superior to the one using only classical pairwise dominance. While the difference is rather small for the EVRPTW-M, a substantial advantage of partial dominance is evident for the SDVRPTW. For the latter, an average



(a) EVRPTW-S

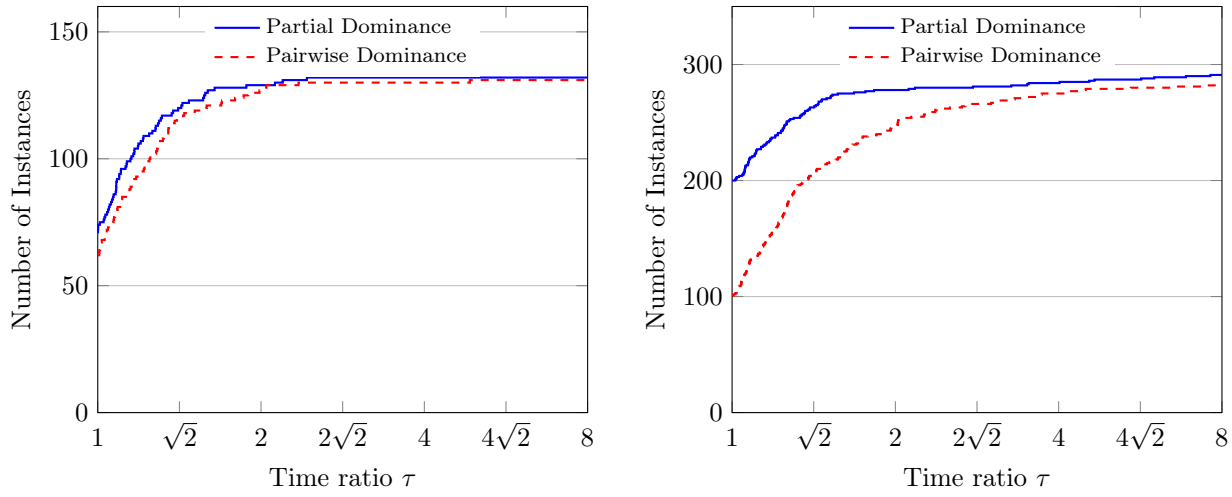


(b) EVRPTW-M



(c) SDVRPTW

Figure 5: Geometric mean of ratios of the number of created labels (-L), the number of dominance tests (-D), and the solution time (-T) for either all pricing problems solved (AP) or only exact pricing over the full network (EP). Results are grouped according to total runtimes $\geq t$.



(a) EVRPTW-M

(b) SDVRPTW

Figure 6: Performance profiles of individual runs of the BPC algorithms with partial dominance and with only classical pairwise dominance. Note that the abscissa has a logarithmic scale.

reduction of 20% of the total BPC computation time is achieved. Even higher savings of 30% result for the more difficult instances requiring more than 600 seconds of computation time. This result is consistent with what we reported for identical pricing instances in the previous section. The results also show that the BPC algorithm with partial dominance is able to solve more instances to proven optimality (132 versus 131 for the EVRPTW-M and 293 versus 288 for the SDVRPTW) than the BPC algorithm with only classical pairwise dominance. Note that unlike in the identical pricing case, where partial dominance was consistently always better, the picture here is more mixed, i.e., there are several instances where the BPC algorithm with classical pairwise dominance is the faster algorithm. This can be explained by generally different trajectories of the algorithms (caused by different dual solutions, the use of separation heuristics, and different branching decisions) resulting from different columns being priced out in some iterations.

6. Conclusions

A crucial building block of column generation based solution approaches like BPC for many VRPs is the effective solution of instances of the SPPRC, which constitute the pricing subproblems of the overall approach. The focus of this paper has been on SPPRCs with tradeoffs between resources. For these type of problems, partial dominance can improve the classical pairwise dominance between labels that is typically employed in labeling algorithms to solve variants of SPPRCs. Partial dominance allows labels to be dominated by sets of other labels, where each of these labels dominates the former only partially, i.e., only for a certain subset of label extensions. The two main (opposing) effects of partial dominance are (i) the potential to dominate more labels implying that overall less labels are created and less dominance comparisons are necessary; and (ii) a more complex dominance rule to be tested.

We have studied in detail the most basic tradeoff between two resources, namely a tradeoff with a single linear piece: We have provided a formal characterization of the corresponding tradeoff segments and have derived a unified partial dominance rule to be used in ad hoc labeling algorithms for solving SPPRCs with such a tradeoff. Furthermore, we have discussed issues related to the practical implementation of partial dominance. The application of partial dominance has been exemplified for two variants of the EVRPTW with a partial recharge policy and the SDVRPTW, two important variants of VRP.

Computational results on standard benchmark instances for the considered problems have revealed the following main insights. On a per pricing problem basis, i.e., on truly identical instances of SPPRCs, we have found that the labeling algorithms applying partial dominance never performed worse than their counterparts with classical pairwise dominance. Furthermore, partial dominance has proved more beneficial for the pricing problems of the more difficult EVRPTW and SDVRPTW instances. It has also proved more beneficial for the EP instances rather than for heuristic pricing iterations on reduced networks. Overall, for the SDVRPTW, average speedups of up to 40% (EPs of the more difficult instances) could be realized by using partial dominance. For the two EVRPTW variants, savings were much smaller. This behavior can be explained by the fact that much less labels actually show a tradeoff in the EVRPTW variants (33% and 46%) compared to 62% in the SDVRPTW. Such a straightforward analysis can, thus, help to a priori estimate a potential gain from implementing partial dominance.

Finally, we have found that the results for the individual pricing instances do translate also to an improvement of an overall BPC, which employs many well-established acceleration techniques and whose performance is influenced also by many other effects that are not immediately related to the solution of the pricing problems. Again, these benefits are substantial for the SDVRPTW while they are rather minor for the EVRPTW.

Acknowledgement

This research was supported by the Deutsche Forschungsgemeinschaft (DFG) under grants GS 83/1-1 and IR 122/10-1 of project 418727865. This support is gratefully acknowledged.

References

Achterberg, T. (2007). *Constraint Integer Programming.* Ph.D. thesis, Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften, Berlin, Germany.

Aerts-Veenstra, M., Cherkesly, M., and Gschwind, T. (2023). A unified branch-price-and-cut algorithm for multi-compartment pickup and delivery problems. Les Cahiers du GERAD G-2023-26, Groupe d'études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada.

Archetti, C., Speranza, M. G., and Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, **40**(1), 64–73.

Archetti, C., Bouchard, M., and Desaulniers, G. (2011). Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science*, **45**(3), 285–298.

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

Baller, A. C., Dabia, S., Dullaert, W. E. H., and Vigo, D. (2020). The vehicle routing problem with partial outsourcing. *Transportation Science*, **54**(4), 1034–1052.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Bianchessi, N. and Irnich, S. (2019). Branch-and-cut for the split delivery vehicle routing problem with time windows. *Transportation Science*, **53**(2), 442–462.

Bode, C. and Irnich, S. (2014). The shortest-path problem with resource constraints with $(k, 2)$-loop elimination and its application to the capacitated arc-routing problem. *European Journal of Operational Research*, **238**(2), 415–426.

Bulhões, T., Sadykov, R., and Uchoa, E. (2018). A branch-and-price algorithm for the minimum latency problem. *Computers & Operations Research*, **93**, 66–78.

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

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

Costa, L., Contardo, C., Desaulniers, G., and Pecin, D. (2021). Selective arc-ng pricing for vehicle routing. *International Transactions in Operational Research*, **28**(5), 2633–2690.

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

Desaulniers, G. and Villeneuve, D. (2000). The shortest path problem with time windows and linear waiting costs. *Transportation Science*, **34**(3), 312–319.

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

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

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

Desaulniers, G., Rakke, J. G., and Coelho, L. C. (2016a). A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, **50**(3), 1060–1076.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016b). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, **64**(6), 1388–1405.

Desaulniers, G., Pecin, D., and Contardo, C. (2019). Selective pricing in branch-price-and-cut algorithms for vehicle routing. *EURO Journal on Transportation and Logistics*, **8**, 147–168.

Desaulniers, G., Gschwind, T., and Irnich, S. (2020). Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Science*, **54**(5), 1170–1188.

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

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

Gschwind, T. (2015). A comparison of column-generation approaches to the synchronized pickup and delivery problem. *European Journal of Operational Research*, **247**(1), 60–71.

Gschwind, T. and Irnich, S. (2015). Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, **49**(2), 335–354.

He, Q., Irnich, S., and Song, Y. (2019). Branch-and-cut-and-price for the vehicle routing problem with time windows and convex node costs. *Transportation Science*, **53**(5), 1409–1426.

Heßler, K. and Irnich, S. (2023). Partial dominance in branch-price-and-cut for the basic multi-compartment vehicle routing problem. *INFORMS Journal on Computing*, **35**(1), 50–65.

Houck, D., Picard, J., Queyranne, M., and Vemuganti, R. (1980). The travelling salesman problem as a constrained shortest path problem: theory and computational experience. *Opsearch*, **17**, 93–109.

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

Ioachim, I., Desrosiers, J., Soumis, F., and Bélanger, N. (1999). Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, **119**(1), 75–90.

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

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York.

Irnich, S. and Villeneuve, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, **18**(3), 391–406.

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

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

Lera-Romero, G., Miranda Bront, J. J., and Soulignac, F. J. (2020). Linear edge costs and labeling algorithms: The case of the time-dependent vehicle routing problem with time windows. *Networks*, **76**(1), 24–53.

Liberatore, F., Righini, G., and Salani, M. (2010). A column generation algorithm for the vehicle routing problem with soft time windows. *4OR*, **9**(1), 49–82.

Luo, Z., Qin, H., Zhu, W., and Lim, A. (2017). Branch and price and cut for the split-delivery vehicle routing problem with time windows and linear weight-related cost. *Transportation Science*, **51**(2), 668–687.

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

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Naddef, D. (2002). Polyhedral theory. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, chapter 2, pages 29–116. Kluwer, Dordrecht.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017a). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, **9**(1), 61–100.

Pecin, D., Contardo, C., Desaulniers, G., and Uchoa, E. (2017b). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, **29**(3), 489–502.

Ralphs, T., Kopman, L., Pulleyblank, W., and Trotter, L. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, **94**(2-3), 343–359.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2018). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, **52**(5), 1174–1190.

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

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, **48**(4), 500–520.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**(2), 254–265.

Spliet, R. and Gabor, A. F. (2015). The time window assignment vehicle routing problem. *Transportation Science*, **49**(4), 721–731.

Spliet, R., Dabia, S., and Van Woensel, T. (2018). The time window assignment vehicle routing problem with time-dependent travel times. *Transportation Science*, **52**(2), 261–276.

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

Tilk, C., Bianchessi, N., Drexl, M., Irnich, S., and Meisel, F. (2018). Branch-and-price-and-cut for the active-passive vehicle-routing problem. *Transportation Science*.

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