Subset-Row Inequalities and Unreachability in Path-based Formulations for Routing and Scheduling Problems

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

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

Abstract

This work considers branch-price-and-cut algorithms for variants of the vehicle-routing problem in which subset-row inequalities (SRIs) are used to strengthen the linear relaxation. SRIs often help to substantially reduce the size of the branch-and-bound search tree. However, their use is computationally costly because SRIs modify the structure of the respective column-generation subproblem which is a shortest-path problem with resource constraints (SPPRC). Each active SRI requires the addition of a resource to the labeling algorithm that is invoked for solving the SPPRC in every iteration. In the context of time-window constraints, the concept of unreachable customers has been used for preprocessing (time-window reduction, arc elimination, precedence identification) as well as for improving the dominance between labels in the elementary SPPRC and its relaxations. We show that the identification of unreachable customers can also help to improve the dominance due to a modified comparison of SRI-related resources. Computational experiments with a fully-fledged branch-price-and-cut algorithm for the (standard and electric) vehicle routing problem with time windows demonstrate the effectiveness of the approach: Overall computation times decrease, for some difficult instances they may even be cut in half, while the required modifications of a computer implementation for combining SRIs with unreachable customers is minor.

Keywords: Routing, subset-row inequalities, labeling algorithm, unreachability, branch-price-and-cut

1. Introduction

One of the great success stories of *branch-price-and-cut* (BPC) algorithms for the exact solution of *vehicle-routing problems* (VRPs, Toth and Vigo, 2014) has been the invention of non-robust cutting planes starting with the work of Jepsen et al. (2008) introducing *subset-row inequalities* (SRIs). A BPC algorithm is, according to Costa et al. (2019), "a branch-and-bound algorithm where the lower bounds are computed by column generation and the cutting planes are added to strengthen the linear relaxations encountered in the search tree." The column-generation procedure iteratively solves a *restricted master program* (RMP), which is the linear relaxation of a route-based formulation over a restricted subset of all potential routes, and a pricing subproblem, which is in almost all VRP applications a *shortest-path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). The task of the SPPRC is to generate negative-reduced cost routes (if any). A dynamic-programming labeling algorithm is typically used for this purpose.

During the last two decades, research on SPPRC labeling algorithms has been intense, which can be explained by the fact that the largest share of the overall BPC solution time is typically consumed by the labeling algorithm. Several powerful algorithmic components have been invented to accelerate labeling. To

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

name a few milestones, the concept of unreachable customers improved dominance in *elementary SPPRCs* (ESPPRCs, Feillet et al., 2004), relaxations of elementarity have led to well-solvable SPPRC relaxations (Irnich and Villeneuve, 2006; Desaulniers et al., 2008; Baldacci et al., 2011, 2012; Bode and Irnich, 2014), and important acceleration techniques include decremental state space relaxation (Boland et al., 2006; Righini and Salani, 2008), bidirectional labeling (Righini and Salani, 2006; Tilk et al., 2017), label bucketing (Sadykov et al., 2021), and arc fixing (Irnich et al., 2010; Desaulniers et al., 2020; Bianchessi et al., 2023).

The above-mentioned SRIs have such a high impact on BPC performance because they often significantly strengthen the master-program formulations so that they help to provide very tight dual bounds. At the same time, the SRI-related dominance rule proposed by Jepsen et al. (2008) still allows an effective solution of SPPRC subproblems. Note that the subproblems have to be extended by one additional resource per active non-robust cut. This resource has to be properly considered in the reduced-cost computation and dominance procedure. Work on non-robust cuts has been expanded in two directions, where one stream of works considers easier to solve relaxations using a limited memory (Pecin et al., 2017b,a), and the second stream of works invented labeling algorithms integrating alternative non-robust cuts such as clique inequalities (Spoorendonk and Desaulniers, 2010), non-robust strengthened capacity and k-path cuts (Baldacci et al., 2008), and subset-row cuts for set-covering formulations (Balster et al., 2023).

In this paper, we introduce an improved dominance rule for SPPRCs that consider SRIs. To this end, we reconsider the concept of unreachable customers, which has been used for preprocessing (time-window reduction, arc elimination, precedence identification) in the context of time-window constraints (Desrochers et al., 1992) as well as for strengthening the dominance in elementary SPPRCs and their relaxations (Feillet et al., 2004). We show that for an SRI defined by a subset S, dominance comparisons between two labels can ignore the SRI-related resource as soon as the potentially weaker label represents a path that cannot feasibly fulfill further tasks of S. As a result, the stronger label does not need to be penalized with the respective dual price making the dominance comparison stronger. This leads to a smaller number of labels to be considered, reduced pricing times, and finally faster BPC algorithms. In a computational study on the *vehicle routing problem with time windows* (VRPTW, Desaulniers et al., 2014) and the *electric VRPTW* (EVRPTW, Desaulniers et al., 2016), we quantify the resulting speedups. As can be expected, the speedup typically increases with the number of SRIs that are added and active as well as with the size of the branch-and-bound search tree. The well-known Solomon benchmark instances serve for the experiments.

The improved dominance rule is part of the regular pairwise dominance comparison performed very often in the course of the BPC algorithm. Therefore, it has to be ensured that the new improved dominance runs seamlessly without significantly slowing down dominance. We elaborate the computation of auxiliary information during each label extension step that helps to mitigate the additional effort needed in the improved dominance comparison.

The remainder of this work is structured as follows: Section 2 presents the theoretical background of SPPRC labeling including a generic formulation of standard as well as limited-memory based SRIs, general dominance principles, and an analysis which type of dominance is compatible with which type of SRIs. The improved dominance is introduced in Sections 3 and possibilities to define and identify unreachable vertices in Section 4. In Section 5, computational results compare two fully-fledged BPC algorithms where one is equipped with the stronger dominance and the other one with the standard dominance in the dynamic-programming labeling algorithm solving the SPPRC pricing subproblem. The paper closes with final conclusions given in Section 6.

2. Subset-Row Inequalities in Labeling Algorithms

We start with the description of an *extensive formulation* in the sense of (Desaulniers et al., 2005; Lübbecke and Desrosiers, 2005). Let M represent a set of *tasks* to be performed, e.g., the set of customers to be visited in the capacitated VRP and VRPTW variants, the set of pickup-and-delivery requests in VRPs concerned with point-to-point transportation, or the set of arcs to be serviced in arc-routing problems etc. Moreover, let Ω denote the set of all feasible *routes*. The relationship between tasks and routes is given by the non-negative integer coefficients a_{ir} for $i \in M$ and $r \in \Omega$ describing how often the route r performs task i. Let $c_r \in \mathbb{R}$ denote the cost of route $r \in \Omega$. The extensive formulation has one variable λ_r for each $r \in \Omega$ indicating how often route r is selected in the solution:

$$
\begin{aligned}
z = \min \quad & \sum_{r\in\Omega} c_r\lambda_r && && \text{(1a)}\\
\text{subject to} \quad & \sum_{r\in\Omega} a_{ir}\lambda_r = 1 && \forall i \in M && \text{(1b)}\\
& \lambda_r \geq 0 \qquad \text{integer} && \forall r \in \Omega && \text{(1c)}
\end{aligned}
$$

The objective (1a) minimizes the total cost of all selected routes. Constraints (1b) ensure that all tasks are performed once by exactly one route. Note that in relaxed formulations routes may perform the same service more than once ($a_{ir} \geq 2$), but these routes r can never be part of a feasible integer solution due to the integrality constraints (1c). Besides the set-partitioning constraints (1b), many models contain further constraints of the form $B\boldsymbol{\lambda} \geq \boldsymbol{d}$, where B is the arc-route incidence matrix of the digraph $D = (V, A)$ over which the routing problem is defined. Examples are fleet-size constraints, balancing constraints, and service-level constraints. We do not need to explicitly consider these constraints because they are robust, i.e., their presence does not change the structure of the SPPRC subproblem that we describe below. They only modify the reduced costs of the arcs (for a more detailed discussion of the impact of robust constraints we refer to Desrosiers et al., 2024, Sect. 7.2, Note 7.3).

When used in column generation, the linear relaxation of formulation (1) is restricted to a small subset $\Omega' \subset \Omega$. The resulting linear model over the variables $\lambda_r \geq 0$ for $r \in \Omega'$ is known as the RMP. The pricing subproblem receives a dual solution $\boldsymbol{\pi} = (\pi_i)_{i\in M}$ of the RMP and must determine at least one route r with negative reduced cost $\tilde{c}_r = c_r - \sum_{i\in M} a_{ir}\pi_i$, if one exists. The column-generation process is then repeated with a new subset Ω' to which at least one negative reduced-cost route has been added. Column generation terminates when all routes have non-negative reduced cost. The primal solution $\bar{\boldsymbol{\lambda}}$ of the respective RMP is then the solution to the linear relaxation of formulation (1).

In general, the lower bound $LB = \boldsymbol{c}^\top\bar{\boldsymbol{\lambda}}$ provided by the linear relaxation is strictly smaller than z. In such a situation, strengthening the linear relaxation with valid inequalities can help to reduce the size of the branch-and-bound tree. SRIs are rank-1 Gomory cuts for models containing set-packing constraints $\sum_{r\in\Omega} a_{ir}\lambda_r \leq 1$ for $i \in M$, as fulfilled for set-partitioning models of type (1). An SRI is defined by a subset S of the rows M and non-negative rational weights $\boldsymbol{w} = (w_i)_{i\in S}$:

$$\sum_{r\in\Omega} \left\lfloor \sum_{i\in S} w_i a_{ir} \right\rfloor \lambda_r \leq \left\lfloor \sum_{i\in S} w_i \right\rfloor .$$

An SRI defined for $(S, \boldsymbol{w})$ can be written more shortly by defining the coefficients $a_r(S, \boldsymbol{w}) = \lfloor \sum_{i\in S} w_i a_{ir} \rfloor$ and the *right-hand side* (RHS) $b(S, \boldsymbol{w}) = \lfloor \sum_{i\in S} w_i \rfloor$:

$$\sum_{r\in\Omega} a_r(S, \boldsymbol{w})\lambda_r \leq b(S, \boldsymbol{w}) \tag{2}$$

In practice, the subsets S that are used to strengthen the RMP are small. Table 1 lists weights that lead to undominated SRIs for subsets S with cardinality between three and five (see Pecin et al., 2017c).

Example 1. *For a subset $S = \{i_1, i_2, i_3\}$ with $|S| = 3$ elements, the associated SRI uses weights $w_i = 1/2$ for $i \in S$. The SRI enforces that there is at most one route $r \in \Omega$ in every feasible solution that serves two or more requests from S. Indeed, with the definition $a(S, \boldsymbol{w}) = \lfloor \sum_{i\in S} a_{ir}/2 \rfloor$ the inequality is*

$$\sum_{\substack{r\in\Omega:\\ 2\leq\sum_{i\in S} a_{ir}\leq 3}} \lambda_r \quad + \quad 2\cdot \sum_{\substack{r\in\Omega:\\ 4\leq\sum_{i\in S} a_{ir}\leq 5}} \lambda_r \quad + \quad 3\cdot \sum_{\substack{r\in\Omega:\\ 6\leq\sum_{i\in S} a_{ir}\leq 7}} \lambda_r \quad + \dots \qquad \leq 1.$$

The first summand means that at most one route performing more than one task of S can be part of an integer solution. The additional summands can only contribute if non-elementary routes are considered. For example, the condition $4 \leq \sum_{i\in S} a_{ir} \leq 5$ implies that at least one request of S is covered at least twice.

Size $\|S\|$	Weights $\boldsymbol{w} = (w_i)_{i \in S}$	RHS $b(S, \boldsymbol{w})$
3	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	1
4	$(\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1
5	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $(\frac{2}{4}, \frac{2}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, $(\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, $(\frac{3}{5}, \frac{2}{5}, \frac{2}{5}, \frac{1}{5}, \frac{1}{5})$,	1
	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, $(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3})$, $(\frac{3}{4}, \frac{3}{4}, \frac{2}{4}, \frac{2}{4}, \frac{1}{4})$	2

Table 1: Undominated combinations of weights for subset-row inequalities with $3 \leq |S| \leq 5$.

2.1. SPPRC and Labeling

Jepsen et al. (2008) have introduced SRIs for SPPRC pricing subproblems. Recall that $D = (V, A)$ denotes the digraph over which the SPPRC is defined. We assume w.l.o.g. that each feasible route starts at the given origin vertex $o \in V$ and ends at the destination vertex $o' \in V$. Moreover, let $\mathscr{R}$ denote the set of attributes (a.k.a. resources) that define the SPPRC. Examples of attributes are the earliest service time, the accumulated demand, the traveled distance etc. (further modeling examples are provided in Irnich and Desaulniers, 2005). Also binary attributes related to *ng*-route relaxation (Baldacci et al., 2011) are to be included if this type of SPPRC relaxation is used. In addition, we assume that one attribute rdc $\in \mathscr{R}$ represents the accumulated reduced cost. At this point, we explicitly *exclude* attributes that are introduced to handle the impact of SRIs. This exclusion enables us to later distinguish between subproblems that disregard SRIs and subproblems that consider SRIs.

The prevalent solution approach for SPPRCs is by dynamic-programming labeling algorithms. Starting from the origin vertex o associated with the trivial path $p = (o)$, o-i-paths (a.k.a. partial paths) are systematically extended in a vertex-by-vertex fashion towards the destination vertex o'. More precisely, a partial path $p = (o, \ldots, i)$ ending at vertex i is extended along all outgoing arcs $(i, j) \in A$ of i producing new partial paths $p' = (o, \ldots, i, j)$, which are immediately checked for feasibility and dismissed if infeasible.

We further assume that $\mathbf{T} = (T^{\text{res}})_{\text{res} \in \mathscr{R}}$ denotes a problem-specific vector of attributes. *Resource extension functions* (REFs, Desaulniers et al., 1998; Irnich, 2007) are used to propagate the attributes through the network. Let f_{ij} denote the REF associated with arc $(i, j) \in A$. The REF f_{ij} propagates a vector $\mathbf{T}$ of attributes referring to a vertex i to resulting attributes $\mathbf{T}'$ for the vertex j via $\mathbf{T}' = f_{ij}(\mathbf{T})$. For convenience, we assume that infeasibility is indicated via $f_{ij}^{\text{rdc}}(\mathbf{T}) = \infty$. Moreover, we do not consider parallel arcs and multiple REFs per arc to keep the notation simple (even though this extension bears no conceptual difficulty, see Desaulniers, 2010; Goel and Irnich, 2017). Let $\mathbf{T}_0$ denote the initial values of the attributes associated with the origin o. For any path $r = (i_0, i_1, \ldots, i_\ell)$, we recursively define

$$\mathbf{T}_k = f_{i_{k-1} i_k}(\mathbf{T}_{k-1}) \qquad \text{for all } k = 1, 2, \ldots, \ell.$$

For convenience, we define $\mathbf{T}(r)$ as the attribute vector at the last vertex i_ℓ, i.e., $\mathbf{T}(r) = \mathbf{T}_\ell$. The path r is *feasible* if and only if $\mathbf{T}(r)^{\text{rdc}} = T_\ell^{\text{rdc}} < \infty$ (assuming $\infty + a = \infty$ for any real value $a \in \mathbb{R}$).

Example 2. *The VRPTW is defined by a demand d_i and a time window $[e_i, \ell_i]$ for all vertices $i \in V$ together with a routing cost c_{ij} and a travel and service time t_{ij} for all arcs $(i, j) \in A$ as well as a vehicle capacity Q (for details we refer to, e.g., Desaulniers et al., 2014). Attributes for the following resources are relevant: reduced cost, load, earliest service time, and ng-related binary attributes. The latter ng-route relaxation is defined by neighborhoods $(N_j)_{j \in V}$ with $j \in N_j \subseteq V$. The initial attributes at the origin vertex o are given by $\mathbf{T}_o = (0, 0, e_o, \mathbf{0})$. The propagation of attributes $\mathbf{T}$ from a vertex i along an outgoing arc $(i, j) \in A$ to the* *vertex j gives the new attributes $\mathbf{T}' = f_{ij}(\mathbf{T})$, where the REF f_{ij} is componentwise defined as:*

$$T'^{rdc} = f_{ij}^{rdc}(\mathbf{T}) = T^{rdc} + \tilde{c}_{ij} \tag{3a}$$

$$T'^{load} = f_{ij}^{load}(\mathbf{T}) = T^{load} + d_i \tag{3b}$$

$$T'^{time} = f_{ij}^{time}(\mathbf{T}) = \max\{e_j, T^{time} + t_{ij}\} \tag{3c}$$

$$T'^{ng,v} = f_{ij}^{ng,v}(\mathbf{T}) = \begin{cases} T^{ng,v} + 1, & \text{if } v = j \\ T^{ng,v}, & \text{if } v \in N_j, v \neq j \\ 0, & \text{otherwise} \end{cases} \qquad \forall v \in V \tag{3d}$$

The extension is feasible if $T'^{time} \leq \ell_j$, $T'^{load} \leq Q$, and $T'^{ng,v} \leq 1$ for all $v \in V$. Otherwise, we set $T'^{rdc} = \infty$ (not formalized in (3a) for the sake of simplicity).

The reduced costs $\tilde{c}_{ij}$ on the arcs $(i, j) \in A$ in (3a) result from dual prices $\boldsymbol{\pi}$ of task fulfillment constraints (1b) (and possibly other robust constraints, e.g., limiting the fleet size) to be consistently subtracted from the original arc costs c_{ij}.

What we have described so far is the path/label extension part of a labeling algorithm. Repeating extensions without using a pruning mechanism would be a brute-force enumeration of all feasible o-i paths for all $i \in V$. Dominance identifies paths that are useless for finding a most negative reduced cost o-o'-path. It is crucial for the effectiveness of a labeling algorithm. Refining the analysis of Desaulniers et al. (2020), we now categorize the principles behind dominance. For any o-i-path p and i-o'-path q, the concatenation is the o-o'-path with the vertex sequence (p, q) (without repeating i). If (p, q) is a feasible o-o'-path, the path q is called a *feasible completion* of p.

Definition 1. *Let p_1 and p_2 be two different o-i-paths (partial paths). If, for each feasible completion q_2 of p_2 to a path $r_2 = (p_2, q_2)$, there exists a feasible completion q_1 of p_1 to path $r_1 = (p_1, q_1)$ with smaller or identical reduced cost compared to r_2, then the partial path p_2 is* dominated. *We distinguish between the following three types of completions q_1 of p_1:*

*(i) either $q_1 = q_2$ is the feasible completion of p_1 (*identical completion dominance*, ICD),*

*(ii) or $q_1 = q_2'$ is an i-o'-subpath of q_2 (including $q_2' = q_2$ as a non-proper possible subpath) such that $r_1 = (p_1, q_2')$ is the feasible completion of p_1 (*subpath completion dominance*, SCD),*

*(iii) or there exists another i-o'-path q_1 that is the feasible completion of p_1 (here, the completion q_1 may or may not be identical to q_2 or a subpath of it) (*possibly different completion dominance*, PDCD).*

Many SPPRC labeling algorithms rely on the ICD principle, such as those for the CVRP, the VRPTW, and almost all of their extensions in which goods are shipped from depot to customers. Likewise, all goods may be collected from customers and shipped to the depot. The attributes $\mathbf{T}$ introduced above serve for testing ICD. We assume that the relation $\preceq$ compares two partial paths p^1 and p^2 via $\mathbf{T}(p^1) \preceq \mathbf{T}(p^2)$. The validity of the relation indicates that every feasible completion of p^2 is a feasible completion of p^1 with identical or smaller reduced cost. In the VRPTW case, ICD can be proven directly because the REFs defined in (3) are non-decreasing and resources are bounded from above (see, e.g., Desaulniers et al., 1998; Irnich, 2007).

Example 3. *In the VRPTW, the relation $\mathbf{T}_1 = \mathbf{T}(p^1) \preceq \mathbf{T}(p^2) = \mathbf{T}_2$ can be written out explicitly as $T_1^{rdc} \leq T_2^{rdc}$, $T_1^{time} \leq T_2^{time}$, $T_1^{load} \leq T_2^{load}$, and $T_1^{ng,v} \leq T_2^{ng,v}$ for all $v \in V$. Since the REFs (3) are non-decreasing and feasibility constraints only impose upper bounds, every feasible completion q of p^2 is also a feasible completion of p^1. All vertices are reached with better or identical attribute values, i.e., $\mathbf{T}(p^1, q) \preceq \mathbf{T}(p^2, q)$. Note that this relationship is not only true for completion q but also for extensions with partial paths of q, i.e., in every intermediate label extension step.*

We have seen that ICD completely relies on the attributes $\mathbf{T}$. Therefore, labeling algorithms do not store and manipulate partial paths directly but use labels. A *label* L is a (convenient) representation of a feasible o-i-path p, and we write $p = p(L)$ and $L = L(p)$ to express the mutual relationship. Typically, the label L stores the attributes $\mathbf{T} = \mathbf{T}(p(L))$ indicating the resource state at the last vertex i. In addition, the label includes a reference to the predecessor label $pred(L)$ when $p(L)$ results from the extension of $p(pred(L))$. In case of multiple REFs per arc, also a reference to the generating REF is recorded. Herewith, the partial path can be uniquely reconstructed from the label.

SCD has been applied in BPC algorithms for the pickup-and-delivery problem (Dumas et al., 1991; Battarra et al., 2014) and the dial-a-ride problem (Doerner and Salazar-González, 2014), i.e., goods and passenger transportation is between pairs of points. It is often the superior dominance rule for these problems and related variants, e.g., in the presence of additional time-window and ride-time constraints (Gschwind and Irnich, 2015; Gschwind et al., 2018).

Example 4. *For the* pickup-and-delivery problem with time windows *(PDPTW), we assume that the n given transportation requests are represented by pickup-delivery pairs of the form $(i, i+n)$ with $i \in P$ referring to the pickup points and $i+n \in D$ to the delivery points. Associated demands fulfill $d_i = -d_{i+n} > 0$. The remaining data is identical to the VRPTW so that REFs (3a)–(3b) and the corresponding feasibility conditions can be used here, too.*

To model the pairing and precedence constraints, additional attributes are needed: for each request $(v, v+n)$ with $v \in P$, the attribute $T^{open,v}$ indicates whether the request is already picked up but not delivered yet. The initial values are $T^{open,v}_o = 0$ for all $v \in P$. Propagation of the attributes along arcs $(i,j) \in A$ is defined by

$$T'^{open,v} = f^{open,v}_{ij}(\mathbf{T}) = \begin{cases} T^{open,v} + 1, & \text{if } v = j \in P \\ T^{open,v} - 1, & \text{if } v = n + j \in D \\ T^{open,v}, & \text{otherwise} \end{cases} \qquad \forall v \in V. \tag{4}$$

The extension to $j \neq o'$ is feasible w.r.t. the open request, if $0 \leq T'^{open,v} \leq 1$ for all $v \in P$. For $j = o'$, $T'^{open,v} = 0$ for all $v \in P$ is required.

Reduced costs on arcs can here be defined as $\tilde{c}_{ij} = c_{ij} - \pi_j$ for all $j \in P$ where π_j is the dual price of the fulfillment constraint of request $(j, j+n)$, and $\tilde{c}_{ij} = c_{ij}$, otherwise. With this definition, the delivery triangle inequality *(DTI) holds (for a deeper discussion, see Gschwind et al., 2018) so that the following* strong dominance *can be defined. For two partial paths p_1 and p_2 ending at the same vertex, the relation $\mathbf{T}_1 = \mathbf{T}(p^1) \preceq \mathbf{T}(p^2) = \mathbf{T}_2$ can be defined with a componentwise $\leq$ between $\mathbf{T}_1$ and $\mathbf{T}_2$. In particular, labels with different sets of open requests can be compared as long as the one associated with p_1 is included in the one associated with p_2. The proof of the validity of the strong dominance relies on the DTI and SCD. A feasible completion q_2 of p_2 must necessarily include all delivery vertices $D_2 = \{(j+n) \in D : T^{open,j}_2 = 1\}$. However, if $T^{open,j}_1 = 0$ for at least one $(j+n) \in D_2$, the completion q_2 is not a feasible completion of p_1, because only deliveries $D_1 = \{(j+n) \in D : T^{open,j}_1 = 1\}$ are valid for a feasible completion of p_1. Removing all vertices $(j+n) \in D_2 \setminus D_1$ from q_2 creates a proper subpath of q_2 which can be used as a feasible completion q_1 of p_1 (which is a case of SCD). The DTI guarantees that the reduced cost of q_1 is not greater than the reduced cost of q_2.*

Another example for SCD is the labeling-based BPC algorithm for the soft-clustered VRP (Hintsch and Irnich, 2019).

We finally discuss PDCD now. Even if PDCD is not often found in the literature, we sketch an important example leaving out the technical details of describing the complete instance data and REFs.

Example 5. *The truck-and-trailer routing problem with time windows (Rothenbächer et al., 2018) is another extension of the VRPTW. Here, the fleet consists of trucks to which trailers can be attached in order to extend the capacity. Some customers are not accessible with a truck-and-trailer combination but can however be serviced by a truck alone if its trailer is previously detached and parked at a suitable location. The truck must attach the trailer at some later point because only a truck-and-trailer combination is allowed to return to the destination depot.*

The BPC algorithm represents routes of a truck-and-trailer combination by the vertices that the truck visits. The labeling approach stores for each feasible partial path the position where the trailer has been parked (and not yet coupled again to the truck) in an attribute $T^{pos} \in V \cup \{\perp\}$ ('$\perp$' for trailer not parked, i.e., trailer attached). While a straightforward dominance can only compare paths p_1 and p_2 with identical trailer location $T^{pos}(p_1) = T^{pos}(p_2)$, an improved dominance uses the PDCD for cases with $T^{pos}(p_1) \neq T^{pos}(p_2)$.

As explained in (Rothenbächer et al., 2018, p. 1180), the idea is that the potentially dominating path p_1 may dominate p_2 with $T^{pos}(p_1) \neq T^{pos}(p_2)$, 'if the first vehicle can be transferred into the same trailer status as the second without violating the dominance constraints'. This is tested 'by hypothetically moving the first vehicle in up to three steps':
(1) If $T^{pos}(p_1) \neq \perp$, the first truck must pick up its trailer at position $T^{pos}(p_1)$,
(2) If $T^{pos}(p_2) \neq \perp$, the first truck must park its trailer at position $T^{pos}(p_2)$,
(3) The first truck must move back to the current customer, say j.
This creates a detour q^+, i.e., a cycle over vertex j. Any completion q_2 of p_2 is transformed into the completion $q_1 = (q^+, q_2)$ of p_1 (this is PDCD). Note that the higher cost and time attributes of q_1 compared to q_2 must be considered leading to modified rules (3a) and (3c).

2.2. Relaxed Subset-Row Inequalities

SRIs have been relaxed in order make the labeling less computationally expensive compared to the original SRIs as stated in (2). Formally, any integer coefficients $a_r^{\mathscr{M}}(S, \boldsymbol{w})$ on the *left-hand side* (LHS) that fulfill $a_r^{\mathscr{M}}(S, \boldsymbol{w}) \leq a_r(S, \boldsymbol{w})$ for all $r \in \Omega$ give rise to a valid inequality

$$\sum_{r \in \Omega} a_r^{\mathscr{M}}(S, \boldsymbol{w}) \lambda_r \leq b(S, \boldsymbol{w}), \tag{5}$$

in the following denoted as *relaxed SRI* or simply SRI also.

Consistent SRI relaxations result from modified labeling procedures in which SRI-related attributes are disregarded under certain conditions. Concepts to control which attributes are considered and disregarded ('forgotten') have turned out very powerful and successful in SPPRC labeling: For example, the definition of a neighborhood in the *ng*-route relaxation controls, per vertex, how much information about elementary is kept or forgotten. This can be seen as a short-term *memory*. Pecin et al. (2017b,a) have transferred the idea of a memory to the SRI case. The most flexible relaxation that they define is the following *arc-based memory*: For each SRI $(S, \boldsymbol{w})$, the arc subset $\mathscr{M}(S, \boldsymbol{w}) \subset A$ defines, for which arcs the information regarding SRI $(S, \boldsymbol{w})$ is maintained: if $(i, j) \in \mathscr{M}(S, \boldsymbol{w})$, then the labeling procedure considers all $(S, \boldsymbol{w})$-related information during a label extension over the arc $(i, j) \in A$, otherwise, the labeling procedure disregards and forgets any $(S, \boldsymbol{w})$-related information.

Three types of memory have been used in the literature:

1. The origin SRI definition of Jepsen et al. (2008) uses a *full memory* $\mathscr{M}(S, \boldsymbol{w}) = A$ for all SRIs $(S, \boldsymbol{w})$.
2. The limited *node-based memory* of Pecin et al. (2017b) is defined per SRI $(S, \boldsymbol{w})$ with the help of a vertex subset $V(S, \boldsymbol{w}) \subset V$. The memory is then defined as $\mathscr{M}(S, \boldsymbol{w}) = \bigcup_{j \in V(S, \boldsymbol{w})} \delta^-(j)$, where $\delta^-(j)$ denotes the set of ingoing arcs of j (for backward labeling, $\mathscr{M}(S, \boldsymbol{w}) = \bigcup_{j \in V(S, \boldsymbol{w})} \delta^+(j)$, where $\delta^+(j)$ denotes the set of outgoing arcs of j).
3. The limited *arc-based memory* of Pecin et al. (2017a) can be defined with any subset of arcs.

Typically, the limited memory is not chosen a priori but results from the violated SRIs that have been identified. The routes with $\lambda_r > 0$ in the RMP solution that contribute to a violated SRI defined by $(S, \boldsymbol{w})$ are inspected. Inserting the subsequence between the first and last occurrence of a vertex included in S suffices to guarantee that the LHS in (5) (relaxed version) coincides with the LHS in (2) (full memory version). In the case of a node-based memory, the ingoing/outgoing arcs of the vertices of the subsequence are added to $\mathscr{M}(S, \boldsymbol{w})$. Likewise, the arcs of the subsequence are added to $\mathscr{M}(S, \boldsymbol{w})$ for an arc-based memory.

Next, we describe how the propagation of attributes works in the presence of possibly relaxed SRIs. We assume that the initial partial path p_0 has the label $L_0 = L(p_0)$ without predecessor and initial resource state $\mathbf{T}(L_0)$. Let $\mathscr{S}$ denote the set of active SRIs $(S, \boldsymbol{w})$, i.e., those with non-zero dual price $\sigma_{S,w} < 0$. For a partial path $p = p(L)$, its reduced cost including the contribution of the dual prices $\sigma_{S,w}$ is then:

$$\tilde{c}_p = c_p - \sum_{i \in M} a_{ip} \pi_i - \sum_{(S, \boldsymbol{w}) \in \mathscr{S}} a_p^{\mathscr{M}}(S, \boldsymbol{w}) \sigma_{S,w}$$

For each label L, besides the problem-specific attributes $\mathbf{T} = \mathbf{T}(L)$, additional integer attributes $W_{S,w} = W_{S,w}(L)$ accumulate the weights w_i whenever one of the vertices $i \in S$ is reached. The extension of a label L

for the partial path $p(L)$ ending at vertex $i \in V$ over an arc $(i,j) \in A$ produces the following new label L' with attributes $\mathbf{T}' = \mathbf{T}(L')$, $W'_{S,w}$, and coefficients ${a'}^{\mathscr{M}}_{S,w}$:

$$\mathbf{T}'^{\text{rdc}} = f^{\text{rdc}}_{ij}(\mathbf{T}) - \sum_{\substack{(S,\boldsymbol{w})\in\mathscr{S}\,:\,j\in S,\\ W_{S,w}+w_j\geq 1,\\ (i,j)\in\mathscr{M}(S,\boldsymbol{w})}} \sigma_{S,w} \tag{6a}$$

$$\mathbf{T}'^{\text{res}} = f^{\text{res}}_{ij}(\mathbf{T}) \qquad \forall \text{res} \in \mathscr{R} \setminus \{\text{rdc}\} \tag{6b}$$

$$W'_{S,w} = \begin{cases} 0, & \text{if } (i,j) \notin \mathscr{M}(S,\boldsymbol{w}) \\ W_{S,w} + w_j, & \text{if } j \in S,\ W_{S,w}+w_j < 1, \text{ and } (i,j) \in \mathscr{M}(S,\boldsymbol{w}) \\ W_{S,w} + w_j - 1, & \text{if } j \in S,\ W_{S,w}+w_j \geq 1, \text{ and } (i,j) \in \mathscr{M}(S,\boldsymbol{w}) \\ W_{S,w}, & \text{otherwise} \end{cases} \qquad \forall (S,\boldsymbol{w}) \in \mathscr{S} \tag{6c}$$

$${a'}^{\mathscr{M}}_{S,w} = a^{\mathscr{M}}_{S,w} + \begin{cases} 1, & \text{if } j \in S,\ W_{S,w}+w_j \geq 1 \text{ and } (i,j) \in \mathscr{M}(S,\boldsymbol{w}) \\ 0, & \text{otherwise} \end{cases} \tag{6d}$$

We explain the updates in (6a)–(6d) from simple to more difficult. The update of all problem-specific resources (except for the reduced cost) is straightforward with the REF f_{ij} as shown in (6b). The update of the accumulated weights in (6c) distinguishes four cases: First, if the extension arc (i,j) is not in the memory, the accumulated value is reset to 0, while all other cases require $(i,j) \in \mathscr{M}(S,\boldsymbol{w})$. Second, if the vertex j is the set S and the new accumulated weight $W_{S,w} + w_j$ does not increase to the value 1 or greater, then this value is stored. Third, if $W_{S,w} + w_j \geq 1$, then its fractional part, which is equal to $W_{S,w} + w_j - 1$, is stored (a value in the half-open interval $[0,1)$). Fourth and last, in all other cases the current value $W_{S,w}$ is just transferred to $W'_{S,w}$.

The integer coefficient $a^{\mathscr{M}}_{S,w}$ of the route variable for the SRI $(S,\boldsymbol{w})$, computed by (6d), increases by one if and only if the accumulated weight increases to or exceeds 1. This exactly corresponds with the third case in the computation of $W'_{S,w}$. Note that the coefficients $a^{\mathscr{M}}_{S,w}$ do not necessarily need to be computed and stored within each label. It is however convenient for our later arguments to be able to refer to (6d).

With the already given explanation, it is now simpler to describe the reduced cost update in (6a). Note first that in many VRP applications, the propagation of the reduced cost is done with an REF of the type $f^{\text{rdc}}_{ij}(\mathbf{T}) = \mathbf{T}^{\text{rdc}} + \tilde{c}_{ij}$ where $\tilde{c}_{ij} - (\pi_i + \pi_j)/2$ and π_i are the dual prices of the partitioning constraints and $\pi_o = \pi_{o'}$ the dual price of the fleet size constraint. However, to be generic, we also allow more involved cost updates (some examples are He et al., 2019; Liberatore et al., 2010; Bektaş and Laporte, 2011). The crucial modification related to the dual prices of the SRIs are captured in the sum in (6a). Under the same conditions for which the integer coefficient ${a'}^{\mathscr{M}}_{S,w}$ is increased, the dual price $\sigma_{S,w}$ is incorporated.

Dominance. The seminal paper of Jepsen et al. (2008) defines the following modified dominance rule that incorporates the dual prices of the active SRIs: Sufficient conditions for a label L^1 dominating a label L^2, both resident at the same vertex, are

$$\tilde{c}(L^1) - \sum_{\substack{(S,\boldsymbol{w})\in\mathscr{S},\\ W_{S,w}(L^1) > W_{S,w}(L^2)}} \sigma_{S,w} \leq \tilde{c}(L^2). \tag{7}$$

and $\mathbf{T}(L^1) \preceq \mathbf{T}(L^2)$. Their proof covers the case of a full memory (refined versions of memory were not yet invented) and argues with ICD. More precisely, the condition $\mathbf{T}(L^1) \preceq \mathbf{T}(L^2)$ was used to argue with ICD. The intuition is that, for SRIs $(S,\boldsymbol{w})$ with $W_{S,w}(L^1) > W_{S,w}(L^2)$, the label L^1 is closer to be penalized with $-\sigma_{S,w} \geq 0$ than the other label L^2. The worst case is taken into account with including all those penalties in the dominance relation (7). In particular, this allows dominance between two labels for which the $W_{S,w}$-attributes are not directly comparable (with $\leq$).

The works of Pecin et al. (2017b,a) prove that the same dominance rule is valid for ICD with the refined memories, i.e., for any type of memory. It is straightforward to prove the validity of the dominance (7) for SCD in combination with a full memory. The following example shows that SCD in combination with any type of limited memory is invalid.

	Full memory SRIs	Node-based memory SRIs	Arc-based memory SRIs
ICD	✓	✓	✓
SCD	✓	invalid	invalid
PDCD	invalid	invalid	invalid

Table 2: Valid (✓) and invalid combinations of dominance principles and types of SRI-related memory.

Example 6. *We consider the PDPTW from Example 4. For an SRI $(S, \boldsymbol{w})$ with associated set $S = \{j, k, l\}$, let the delivery vertex $i+n$ not be in the node-based limited memory of the SRI. Two labels L^1 and L^2 refer to the same vertex and have identical attributes $W_{S,w}(L^1) = W_{S,w}(L^2) = 0$. The label L^1 has no open requests. The label L^2 has the open request $(i, i+n)$, i.e., i is visited and $i+n$ is not. Consider the completion $q_2 = (j, i+n, k, o')$ of label L^2. With the limited memory, the attribute $W_{S,w}(L^2)$ is disregarded at vertex $i+n$ so that no penalty $-\sigma_{S,w}$ is added for the completion of the second path. SCD assumes the completion $q_1 = (j, k, o')$ of L^1. Here, the penalty $-\sigma_{S,w}$ is added when visiting vertex k. As a result, L^2 may have a negative reduced cost, while L^1 may a positive reduced cost. Therefore, the first label does not dominate the second. It follows that SCD is not compatible with a limited memory for SRIs.*

Table 2 summarizes which type of dominance is compatible with the different types of SRI memory.

3. Improved Dominance

In this section, we focus on the role of unreachable vertices and subsets of unreachable vertices for an improved SRI-related dominance. For each vertex $i \in V$ and each state $\mathbf{T}$ of the resources at i, let $U_i(\mathbf{T})$ denote the set of vertices u that cannot be feasibly reached, i.e., there exists no resource-feasible i-u-path with initial resource state $\mathbf{T}$ at vertex i to reach the vertex $u \in V$.

Example 7. *(cont'ed from Example 2) In variants of the VRPTW, a possible definition of the set $U_i(\mathbf{T})$ only depends on the attribute T^{time}. Compute, for each pair $(i, u) \in V \times V$, the* latest departure time *$LDT(i, u)$, i.e., the latest point in time to leave vertex i so that one can feasibly reach u. These values can be computed with an i-to-all shortest-path algorithm with a non-decreasing update function. Then, the set of unreachable vertices is*

$$U_i^{time}(\mathbf{T}) = \{u \in V : T^{time} > LDT(i, u)\}. \tag{8}$$

The idea is now to test, for a label L resident at a vertex i, whether all vertices $u \in S$ for an SRI $(S, \boldsymbol{w})$ are unreachable. In the positive case, i.e., if $U_i(\mathbf{T}) = U_i^{time}(\mathbf{T}(L)) \supseteq S$ holds, the attribute referring to this SRI $(S, \boldsymbol{w})$ is irrelevant in the following sense.

First, it follows that the path $p(L)$ cannot be extended to any vertex in $u \in S$ and therefore never collects another penalty $\sigma_{S,w}$. Second, if another label L^1 is trying to dominate $L = L^2$, every feasible completion q^2 of $p(L^2)$ does not visit vertices $u \in S$. Therefore, extending $p(L^1)$ with the same completion q^2 (i.e., ICD) or with a subpath of q^2 (i.e., SCD) does not impose any additional penalty $\sigma_{S,w}$ to the first path. Third, if label $L = L^1$ is trying to dominate another label L^2, then the necessary precondition $\mathbf{T}(L^1) \preceq \mathbf{T}(L^2)$ imposes $S \subseteq U_i(\mathbf{T}(L^1)) \subseteq U_i(\mathbf{T}(L^2))$. Thus, the same argument as in the second case applies here, too. Summarizing, the attribute is irrelevant for extending label L and for dominance with label L whether it is the dominating or the dominated label.

We therefore suggest the following *improved dominance rule* that only differs from (7) in the additional precondition $S \not\subseteq U_i(\mathbf{T}(L^2))$, i.e.,

$$\tilde{c}(L^1) - \sum_{\substack{(S,\boldsymbol{w}) \in \mathscr{S}, \\ W_{S,w}(L^1) > W_{S,w}(L^2), \\ S \not\subseteq U_i(\mathbf{T}(L^2))}} \sigma_{S,w} \leq \tilde{c}(L^2), \tag{9}$$

assuming that both labels L^1 and L^2 are resident at vertex $i \in V$. Compared to condition (7), the new condition (9) is simpler to fulfill, which means that dominance based on condition (9) is stronger.

Note that the condition $S \not\subseteq U_i(\mathbf{T}(L^2))$ is independent of label L^1 and can therefore be pre-computed, e.g., when label L^2 is created. Consequently, we suggest to build and store within each label a second bit vector $\boldsymbol{W}^{weak}$ componentwise defined as

$$W^{weak}_{S,w}(L) = \begin{cases} w^{max}_{S,w}, & \text{if } S \subseteq U(\mathbf{T}(L)) \\ W_{S,w}(L), & \text{otherwise} \end{cases}, \tag{10}$$

where $w^{max}_{S,w}$ is the largest cumulative value smaller than 1 for the SRI $(S, \boldsymbol{w})$, i.e., $w^{max}_{S,w} = 1/2$ for weights with denominator 2, $w^{max}_{S,w} = 2/3$ for denominator 3, $w^{max}_{S,w} = 3/4$ for denominator 4, and $w^{max}_{S,w} = 4/5$ for denominator 5, see Table 1. The following and final form of the improved dominance rule is equivalent to the rule (9) but computationally simpler to check within the summation:

$$\tilde{c}(L^1) - \sum_{\substack{(S,\boldsymbol{w})\in\mathscr{S}, \\ W_{S,w}(L^1) > W^{weak}_{S,w}(L^2)}} \sigma_{S,w} \leq \tilde{c}(L^2). \tag{11}$$

To see the equivalence, note that $\{(S, \boldsymbol{w}) \in \mathscr{S} : W_{S,w}(L^1) > W^{weak}_{S,w}(L^2)\} = \{(S, \boldsymbol{w}) \in \mathscr{S} : W_{S,w}(L^1) > W_{S,w}(L^2)\}$ and $S \not\subseteq U_i(\mathbf{T}(L^2))\}$.

Implementation Details. We consider now the special case of SRIs with $|S| = 3$ as used in most implementations. Here the attributes $W_{S,w}$ and $W^{weak}_{S,w}$ are binary (assuming that values 0 and $1/2$ are encoded with 0 and 1) and all SRI-related attributes are typically stored in one bit vector, say $\boldsymbol{W} = (W_{S,w})$ and $\boldsymbol{W}^{weak} = (W^{weak}_{S,w}) \in \{0,1\}^{|\mathscr{S}|}$. The SRIs $(S, \boldsymbol{w})$ with $W_{S,w}(L^1) > W^{weak}_{S,w}(L^2)$ are then effectively identified by bit vector operations

$$\boldsymbol{W}(L^1) \wedge \sim \boldsymbol{W}^{weak}(L^2)$$

(with '$\wedge$' for the bitwise `and`-operator and '$\sim$' the bitwise `not`-operator). Modern programming languages allow to store bit vectors in a compressed format, e.g., the `bitset` template class in `C++`. Our experience is that using such a compressed representation has a very positive effect on the performance of the labeling algorithm.

We propose to compute both $\boldsymbol{W}(L)$ and $\boldsymbol{W}^{weak}(L)$ during the construction of each new label L, i.e., in the label extension procedure of the labeling algorithm. The computation of $\boldsymbol{W}(L)$ can be performed as described by the REF (6c) or in (Pecin et al., 2017a, p. 493). For the efficient computation of $\boldsymbol{W}^{weak}(L)$, we propose to a priori create a *lookup table* that stores the unreachable information regarding all relevant subsets S for all $(S, \boldsymbol{w})$ in the following way: For each vertex $i \in V$ and each possible attribute value T^{res} used in the definition of the set of unreachable vertices, e.g., the time attribute T^{time} in Example 2 for the VRPTW variants, $\mathbf{Z}(i, T^{\text{res}}) = (Z(i, T^{\text{res}})_{S,w}) \in \{0,1\}^{|\mathscr{S}|}$ is a binary vector that indicates whether $S \subseteq U_i(T^{\text{res}})$ holds. Precisely, $S \subseteq U_i(T^{\text{res}})$ if and only if $Z(i, T^{\text{res}})_{S,w} = 1$. Since this lookup table $\mathbf{Z}$ has dimension $|V|$ times the size of the domain of T^{res}, e.g., the time window width, the binary vector should be stored in compressed format. We can now directly compute $\boldsymbol{W}^{weak}(L)$ as

$$\boldsymbol{W}^{weak}(L) := \boldsymbol{W}(L) \vee \mathbf{Z}(i, T^{\text{res}}(L)) \tag{12}$$

where $\vee$ is the bitwise `or`-operator and i is the vertex that the label L resides at.

4. Subsets of Unreachable Vertices

In this section, we present different possibilities for the definition of the sets of unreachable vertices. Advantages and disadvantages of the respective definitions are discussed. As examples we consider the VRPTW and the EVRPTW. In addition, we discuss how large lookup tables can be discretized in order to save memory and make the overall approach faster.

4.1. Unreachability for the VRPTW

In the VRPTW, the definition of unreachable vertices can rely on the time attribute (see above, Eq. (8)), on the load attribute, or both. We now discuss the latter possibility:

Example 8. *(cont'ed from Example 7) We can compute, for all $i, u \in V$, a minimum demand i-u-path with cumulative minimum demand $CMD(i, u)$ so that the set of unreachable vertices is*

$$U_i^{load}(\mathbf{T}) = \{u \in V : T^{load} + CMD(i, u) - d_i > Q\}. \tag{13}$$

Combining both sets, we can define $U_i(\mathbf{T}) = U_i^{time}(\mathbf{T}) \cup U_i^{load}(\mathbf{T})$.

The implementations of improved dominance rules based on $U_i^{time}(\mathbf{T})$, $U_i^{load}(\mathbf{T})$, and $U_i(\mathbf{T})$ are worth being discussed in more detail. For the time-related sets $U_i^{time}(\mathbf{T})$, we proposed implementing a two-dimensional lookup table $\mathbf{Z}(i, T^{time})$ for quickly retrieving the unreachability information and relating it to the SRIs, see end of Section 3. For the load-related sets $U_i^{load}(\mathbf{T})$, a perfectly similar approach is viable. Here, the lookup table $\mathbf{Z}(i, T^{load})$ is again two-dimensional and indexed by vertices $i \in V$ and the domain $\{0, 1, \ldots, Q\}$ of the load attribute. Computing $\boldsymbol{W}^{weak}(L)$ can in both cases (time and load) be done with Eq. (12).

Instead of Eq. (12) using only one lookup table, it is possible to combine both lookup tables $\mathbf{Z}^{time}(i, T^{time})$ and $\mathbf{Z}^{load}(i, T^{load})$ defining $\boldsymbol{W}^{weak}(L) := \boldsymbol{W}(L) \vee \mathbf{Z}^{time}(i, T^{time}(L)) \vee \mathbf{Z}^{load}(i, T^{load}(L))$. A vertex set S is here considered unreachable if either all vertices $u \in S$ have a too early time-window end or if all have a too large demand. This is not the strongest possible definition of unreachability combining time- and load-attributes, because all vertices of S have to fall into either category: unreachable because of the time attribute or because of the load attribute. For the combined sets $U_i(\mathbf{T}) = U_i^{time}(\mathbf{T}) \cup U_i^{load}(\mathbf{T})$, the unreachable vertices are, in general, supersets of $U_i^{time}(\mathbf{T})$ and of $U_i^{load}(\mathbf{T})$. For example, an SRI for the set $S = \{i_1, i_2, i_3\}$ may have $i_1 \in U_i^{time}(\mathbf{T}) \setminus U_i^{load}(\mathbf{T})$ and $i_2, i_3 \in U_i^{load}(\mathbf{T}) \setminus U_i^{time}(\mathbf{T})$. Then, $S \not\subseteq U_i^{time}(\mathbf{T})$ and $S \not\subseteq U_i^{load}(\mathbf{T})$, but $S \subseteq U_i(\mathbf{T})$.

The use of the sets $U_i(\mathbf{T})$ therefore leads to a stronger dominance compared to $U_i^{time}(\mathbf{T})$ and $U_i^{load}(\mathbf{T})$, respectively. However, the stronger dominance comes at a high cost regarding computer memory. We would have to use a three-dimensional lookup table depending on i and both attributes T^{time} and T^{load}. We do not follow this approach, since it would only work for instances with a few customers and relatively tight resource windows.

4.2. Unreachability for the EVRPTW

The EVRPTW extends the VRPTW by considering a homogeneous fleet of battery powered electric vehicles characterized by a limited driving range that can be extended by recharging the vehicle at dedicated recharging stations. The EVRPTW exists in various variants. We present the definition of Desaulniers et al. (2016) who assume a linear battery charge and consumption. They focus on the following alternative recharging policies: On the one hand, either (S) at most a *single recharge* per route is allowed, or (M) *multiple recharges* per route are allowed. On the other hand, (F) batteries are always *fully* recharged when visiting a recharging station or (P) *partial* battery recharges are possible. The result is four variants named EVRPTW-SF, EVRPTW-SP, EVRPTW-MF, and EVRPTW-MP.

Let the vertex set be $V = \{o, o'\} \cup N \cup R$ where N denotes the set of customers and R the set of recharging stations. Desaulniers et al. (2016) model the battery-capacity constraint with the help of the time b_{ij} required to recharge the consumed energy when traveling between locations i and j, i.e., for each arc $(i, j) \in A$. Let B be the corresponding battery capacity of a vehicle (in time units). For a route $(i_0, i_1, \ldots, i_p)$, the amount to recharge at every visited recharging station must be decided. Furthermore, the resulting recharging time needs to be incorporated into the time-window constraints. A necessary condition for the feasibility of a route is that there exist a schedule $(T_0, T_1, \ldots, T_p)$ and a (battery-)loading plan $(X_0, X_1, \ldots, X_p)$ that fulfill the following conditions: First, loading is only possible at recharging stations R, i.e., $X_j = 0$ if $i_j \in N \cup \{o, o'\}$ and $0 \leq X_j \leq B$ for $i_j \in R$. Second, the time-window constraints are fulfilled, i.e., $T_j \in [e_{i_j}, \ell_{i_j}]$ for all $j = 0, \ldots, p$ and $T_{j-1} + s_{i_{j-1}} + X_{j-1} + t_{i_{j-1}, i_j} \leq T_j$ for all $j \in \{1, \ldots, p\}$, where it is assumed that there

Table 3: Resources in VRPTW and Variants of EVRPTW

Problem	VRPTW	EVRPTW-SP/MP	
Attribute(s)	forward/backward	forward/backward	Description
T^{rdc}	•	•	accumulated reduced cost
T^{load}	•	•	accumulated load
T^{time}	•		service start time
T^{rch}		•	binary: recharged yes/no
T^{tMin}		•	earliest time start of service
T^{tMax}		•	latest time start of service
T^{rtMax}		•	maximum amount to be recharged
No. of Attr.	3	6	

are no service times at recharging stations, i.e., $s_{i_j} = 0$ for $i_j \in R$. Third, the loading plan must be feasible, i.e., $B - \sum_{j=1}^{q} b_{i_{j-1},i_j} + \sum_{j=1}^{q-1} X_j \geq 0$ and $B - \sum_{j=1}^{q} b_{i_{j-1},i_j} + \sum_{j=1}^{q} X_j \leq B$ for all $q \in \{1, \ldots, p\}$.

For the EVRPTW-MP (multiple, partial recharges), the given conditions are sufficient. For the single recharge policy, i.e., the EVRPTW-SF and EVRPTW-SP, at most one of the vertices $i_0, i_1, \ldots, i_p$ can be a recharging station. For the full recharge policy, i.e., the EVRPTW-SF and EVRPTW-MF, the battery must always be completely recharged at recharging stations, i.e., $B - \sum_{j=1}^{q} b_{i_{j-1},i_j} + \sum_{j=1}^{q} X_j = B$ if $i_q \in R$.

For EVRPTW variants with partial recharge, forward and backward labeling can be based on the same type of attributes and REFs, because any time-window and recharging feasible forward o-o'-path is, if reversed, a feasible o-o'-path in the transposed network, and vice versa, i.e., there exists a possibly different but feasible schedule and battery-load plan for the reversed path if and only if one exists for the original path (see Desaulniers et al., 2016, p. 1398). Table 3 lists the attributes used to model EVRPTW variants with partial recharge. For the sake of brevity, we restrict our analysis to partial recharging because this case is probably more interesting. The attributes for reduced cost T^{rdc} and load T^{load} from the VRPTW need to be complemented with additional four attributes that we briefly describe now. Three additional attributes T^{tMin}, T^{tMax}, and T^{rtMax} are needed to describe the linear tradeoff between the maximum amount of energy that can be recharged (also expressed as a recharging time) and the earliest service time. The earliest start of service is no longer a single point in time (as T^{time} in the VRPTW) but can lie in the time interval $[T^{tMin}, T^{tMax}]$ over which the tradeoff-curve with slope -1 is described by the initial maximum amount of energy T^{rtMax} (referring time T^{tMin}). A feasible schedule and load plan exist by construction of the tradeoff-curve. All details about the labeling algorithms including the precise definitions of REFs can be found in (Desaulniers et al., 2016).

We now describe problem-tailored definitions of sets of unreachable vertices for the EVRPTW-SP/MP. As in Example 7 for the VRPTW, we can pre-compute and use the latest departure time $LDT(i, u)$ from i to feasibly reach vertex u. Replacing T^{time} by the corresponding attribute T^{tMin} of the EVRPTW-SP/MP, we get

$$U_i^{time}(\mathbf{T}) = \{u \in V : T^{tMin} > LDT(i, u)\}. \tag{14}$$

Recall that $LDT(i, u)$ is the latest start time at i to feasibly reach u. Moreover, the distance between i and u may require recharging. Recharging is necessary if $T^{rtMax} + b(i, u) > B$ for a path between i and u with minimum recharging time $b(i, u)$. The additional time to be reserved for recharging is therefore $\max\{0, T^{rtMax} + b(i, u) - B\}$. Hence, we would like to replace the condition $T^{tMin} > LDT(i, u)$ in the above definition of $U_i^{time}(\mathbf{T})$ by the condition $T^{tMin} + \max\{0, T^{rtMax} + b(i, u) - B\} > LDT(i, u)$. However, the latter condition depends in a non-additive way on T^{tMin} and T^{rtMax} which makes the implementation with a lookup table practically impossible (see Examples 8 and discussion at the end of Section 3). Instead, we propose to use

$$U_i^{bat}(\mathbf{T}) = \{u \in V : (T^{tMin} + T^{rtMax}) + b(i,u) - B > LDT(i,u)\}. \tag{15}$$

in combination with $U_i^{time}(\mathbf{T})$ as defined in (14). The reasoning is as follows: If $T^{rtMax} + b(i,u) - B < 0$, then $U_i^{bat}(\mathbf{T}) \subseteq U_i^{time}(\mathbf{T})$ so that nothing is wrong when using $U_i^{bat}(\mathbf{T})$. Otherwise, for $T^{rtMax} + b(i,u) - B \geq 0$, we have $U_i^{time}(\mathbf{T}) \subseteq U_i^{bat}(\mathbf{T})$ and using the latter set is not only feasible but, in general, produces a stronger dominance.

Overall, we propose to build two lookup tables, $\mathbf{Z}^{time}(i, T^{tMin})$ based on $U_i^{time}(\mathbf{T})$ and $\mathbf{Z}^{bat}(i, T^{tMin} + T^{rtMax})$ based on $U_i^{bat}(\mathbf{T})$. Building the latter lookup table is viable, since the sum $T^{tMin} + T^{rtMax}$ is as good as any single attribute. In the labeling algorithm, we compute $\boldsymbol{W}^{weak}(L)$ as

$$\boldsymbol{W}^{weak}(L) := \boldsymbol{W}(L) \ \vee \ \mathbf{Z}^{time}(i, T^{tMin}(L)) \ \vee \ \mathbf{Z}^{bat}(i, T^{tMin}(L) + T^{rtMax}(L)). \tag{16}$$

4.3. Discretization of Unreachability

The sizes of the lookup tables $\mathbf{Z}$ grow linearly with the number $|V|$ of vertices and the number of attribute values (for the respective vertex). For the time attribute, the lookup table $\mathbf{Z}^{time}$ has $\sum_{i \in V}(\ell_i - e_i + 1)$ entries, where each entry is a bit-vector of size $|\mathscr{S}|$. As a consequence, the pre-computation of lookup tables can consume substantial memory and computation time when instances with many customers and wide time windows are considered (likewise for resources with a large domain).

To alleviate this problem, we propose to discretize attribute values and to use slightly relaxed criteria for the computation of unreachable customers. For this purpose, let $\delta \in \mathbb{Z}_{>0}$ be a *discretization factor*. The function

$$g^\delta(T) = \delta \cdot \lfloor T/\delta \rfloor$$

assumes only integer values that are multiples of δ and underestimates any resource value T, i.e., $g^\delta(T) \leq T$. Instead of using definitions (8), (13), and (15), we can define new unreachable sets with

$$\begin{aligned} U_i^{\delta,time}(\mathbf{T}) &= \{u \in V : g^\delta(T^{time}) > LDT(i,u)\} \\ U_i^{\delta,load}(\mathbf{T}) &= \{u \in V : g^\delta(T^{load} + CMD(i,u) - d_i) > Q\} \\ U_i^{\delta,bat}(\mathbf{T}) &= \{u \in V : g^\delta(\,(T^{tMin} + T^{rtMax}) + b(i,u) - B) > LDT(i,u)\} \end{aligned}$$

which are subsets of the original sets $U_i^{time}(\mathbf{T})$, $U_i^{load}(\mathbf{T})$, and $U_i^{bat}(\mathbf{T})$, respectively. Note that for $\delta = 1$, the new and original definitions coincide (assuming integer-valued attributes, see also the description of the benchmark instances in Section 5). The larger the discretization factor δ, the less accurate is the information provided by the new sets of unreachable customers.

The point is now that only entries at positions $g^\delta(T)$, i.e., multiples of δ, of the associated lookup tables $\mathbf{Z}$ are needed, i.e., when retrieving $\mathbf{Z}(i, g^\delta(T))$. Hence, the lookup tables can be condensed by the factor δ. This tradeoff between accuracy and size makes it important to find a compromise value for δ, possibly depending on the resource(s) under consideration.

5. Computational Results

In this section, we report the results of our computational study on the use of the improved dominance rule. Results were computed with a standard PC running Windows 10 equipped with an Intel(R) Core(TM) i7-6900k processor clocked at 3.2 GHz with 64 GB RAM main memory. The BPC algorithms were implemented in `C++` and compiled into 64-bit single-thread code with MS Visual Studio 2022. The callable library of CPLEX 22.1.0 was used for solving the RMPs.

Setup of BPC Algorithms. To facilitate comparisons between VRPTW and EVRPTW, we use the same setup for our BPC algorithms independent of whether VRPTW or EVRPTW instances are solved (as done by Desaulniers et al. (2020), even if better results could be obtained with problem-tailored setups). We sketch the main algorithmic components and their parameterization:

- The labeling algorithm uses a bidirectional strategy with a dynamic half-way point (Tilk et al., 2017) with the monotone attribute T^{time} for the VRPTW and T^{tMin} for the EVRPTW-SP/MP. For further details we refer to (Desaulniers et al., 2020, p. 1177).
- As mentioned before, we solve relaxations of the elementary SPPRC which are based on the ng-route relaxation of Baldacci et al. (2011). With a neighborhood size of $|N_j| = 14$ for all $j \in V$ we try to exploit the tradeoff between the difficulty of subproblem relaxation and the size of the branch-and-bound tree.
- Arc fixing allows to eliminate provably redundant arcs from the network over which the SPPRC labeling algorithm is defined. We use the standard version as first described in (Irnich et al., 2010) where reduced costs are computed with a complete forward and a complete backward labeling at the root node only (including cuts) of the branch-and-bound tree.
- Heuristic a.k.a partial pricing can help to further reduce the total time spent in pricing. We sequentially apply four pricing heuristics that use arc-reduced networks with a minimum of 2, 5, 10, and 15 arcs, respectively, that enter and exit each customer vertex. The exact pricer defined over the complete network is only called if all heuristics fail. Further details are explained in (Desaulniers et al., 2008).
- Before adding SRIs, violated robust *capacity cuts* (CC) are added to the RMP to strengthen the linear relaxation of the master program. The associated separation problem is solved by employing two variants of the shrinking heuristic (extended and greedy shrinking) as first presented by Ralphs et al. (2003). For the SRIs, we use the separation algorithm and vertex memory as described in the work of Pecin et al. (2017b). An SRI or CC is considered violated, if the violation is at least $\varepsilon_{cut} = 0.05$. Moreover, the maximum number of SRIs to add is limited to 320.
- Branching is required whenever the addition of valid inequality is not yet sufficient to produce an integer solution. For the VRPTW, we apply the standard two-level branching strategy: branching on (V1) the total number of routes and (V2) the total flow on an arc. For the EVRPTW, we apply the same four-level branching strategy as in (Desaulniers et al., 2016, 2020): branching on (E1) the total number of routes, (E2) the total number of recharges, (E3) the total number of recharges at each recharging station $i \in R$, and (E4) the total flow on an arc. (V1), (E1), (E2), and (E3) are enforced by adding an inequality to the RMP, while (V2) and (E4) are implemented by removing arcs from the pricing network. In (V2), (E2), (E3), and (E4), the specific branching variable is chosen as the one with fractional value closest to 0.5. The two resulting branches bound the branching variable from above (below) by the rounded-down (rounded-up) value. The search tree is explored with a best-bound first strategy.

VRPTW Instances. Solomon's benchmark of VRPTW instances consists of 56 instances with 100 customers grouped by customer distributions random (R), clustered (C), or mixed (RC) as well as by tight (series 1 with subsets R1, C1, RC1) or loose (series 2 with subsets R2, C2, RC2) constraints. 25- and 50-customer instances result from dropping the last 75 and 50 customers, respectively. Optimal solutions have been computed for all $56 \cdot 3 = 168$ instances and a complete table with exact results can be found in the Online Supplement of (He et al., 2019). As in several other works like (Pecin et al., 2017a,b; Pessoa et al., 2018), an upper bound of $UB = opt + 1$ is provided to the BPC algorithms where *opt* is the cost of an optimal solution. Metaheuristics (e.g., Vidal et al., 2013) routinely find these optimal solutions, too. Note that routing costs and travel times are rounded to one decimal unit. In our implementation, we multiply them (and also the time window bounds) with 10 to obtain integer values for all attributes.

EVRPTW Instances. Schneider et al. (2014) constructed the 100-customer EVRPTW benchmark instances from Solomon's benchmark. We refer to (Desaulniers et al., 2016) for a detailed description of how recharging stations were added, battery capacities were set, and some time windows were modified in order to ensure feasibility in all cases. With the four recharging variants (SF, SP, MF, MP) this leads to an overall benchmark of $56 \cdot 3 \cdot 4 = 672$ instances. Recall that we only consider the recharging variants SP and MP with partial recharging (see Section 4.2), i.e., 336 of these instances. Note that EVRPTW instances use a different rounding rule for routing costs and travel times, leading to 2-digit values. In our implementation, we multiply them (and also the time window bounds) with 100 to obtain integer values for all attributes.

Reduction and Grouping Instances. As in (Desaulniers et al., 2020), we restrict the experiments to those (E)VRPTW instances that require cutting and/or branching to compute an integer optimal solution. In the other cases, the modified dominance rule has no impact leading to identical results for the original and improved dominance. The consideration of these instances would otherwise bias the statistical analysis. For the VRPTW, 86 instances are already integer optimal when solving the linear relaxation leading to 82 VRPTW instances for the experiments. For the EVRPTW with partial recharging, $50 + 52 = 102$ instances are dropped due to optimality of the linear relaxation. Moreover, we restrict ourselves to those instances for which the standard BPC algorithm solves the linear relaxation within 2 hours. This excludes another $7 + 9 = 16$ instances. For the experiments, $111 + 107 = 218$ EVRPTW-SP and EVRPTW-MP instances remain for the computational study.

5.1. Comparison of the Original and Improved Dominance Rules

In a first experiment, we compare the original dominance rule (7) and the improved dominance rule (11). It is important to perform such a comparison on true instances of the pricing problem, i.e., with a sequence of dual prices and associated reduced costs as they occur in the course of a BPC algorithm. Otherwise, instances with, e.g., randomly generated dual prices and, in particular, with randomly chosen SRIs would not reflect the true diversity and difficulty of SPPRC subproblems. Furthermore, recall that due to partial pricing the SPPRC instances are either defined by an arc-reduced network or the complete network.

What complicates the comparison is that any modification on the labeling algorithm typically leads to different computed routes due to degeneracy, i.e., non-unique optima resulting from labels with identical reduced costs. As a result, we see very different trajectories of pricing iterations if an original implementation is replaced by a new one. In our case, when the first violated SRI is added and active, the two labeling algorithms with the original and improved dominance rules produce a completely different series of pricing iterations. From this point on, the pricing iterations are no longer directly comparable between the two labeling algorithms. We master these complications in the following way:

- We always run the original and the improved labeling algorithm on identical SPPRC pricing instances, i.e., with exactly the same input in the form of identical dual values. Only the routes that are computed by one of the two labeling algorithms are added to the RMP. Without loss of generality, we take the routes from the improved labeling algorithm.
- We also observed that measured computation times (we use the precise `chrono` STL library of `C++`) differ depending on whether the original or the improved labeling algorithm is solved first or second. This effect is not fully understood by us, but may be explained with effects that memory allocation and deallocation have on modern PCs. However, we also observed that the recorded computation times become very stable when the exactly same algorithm is called a second time. Therefore, we twice solve the same SPPRC instance (defined by identical dual values) with the original labeling algorithm and twice with the improved labeling algorithm. Recorded computation times are those from the second and fourth call.
- We compensate the increased computational effort (solving each pricing problem four times) by extending the standard computation time limit of 2 hour to 8 hours.
- For better comparison and reproducibility, full memory and no arc fixing is used.

In addition, the experiments are intended to highlight the impact of the chosen attribute (or combination of attributes) for the definition of unreachable customers and of the discretization factor δ.

- For the VRPTW, the vehicle capacity is hardly binding in the Solomon-based instances. Thus, the *load* resource is ineffective for defining the sets of unreachable customers (this was also confirmed in pretests) and therefore our improved dominance considers only the *time* resource with the set of unreachable vertices U_i^{time}, see Eq. (8). We compare improved dominance without discretization (i.e., $\delta = 1$) and discretization with the factor $\delta = 10$.
- For the EVRPTW variants with partial recharging, different combinations of the set of unreachable vertices U_i^{load} (defined by Eq. (13)) for the *load* attribute, U_i^{tMin} (defined by Eq. (14)) for the *time* attribute, and U_i^{bat} (defined by Eq. (15)) for the *bat* attribute with different discretization factors δ could be compared.

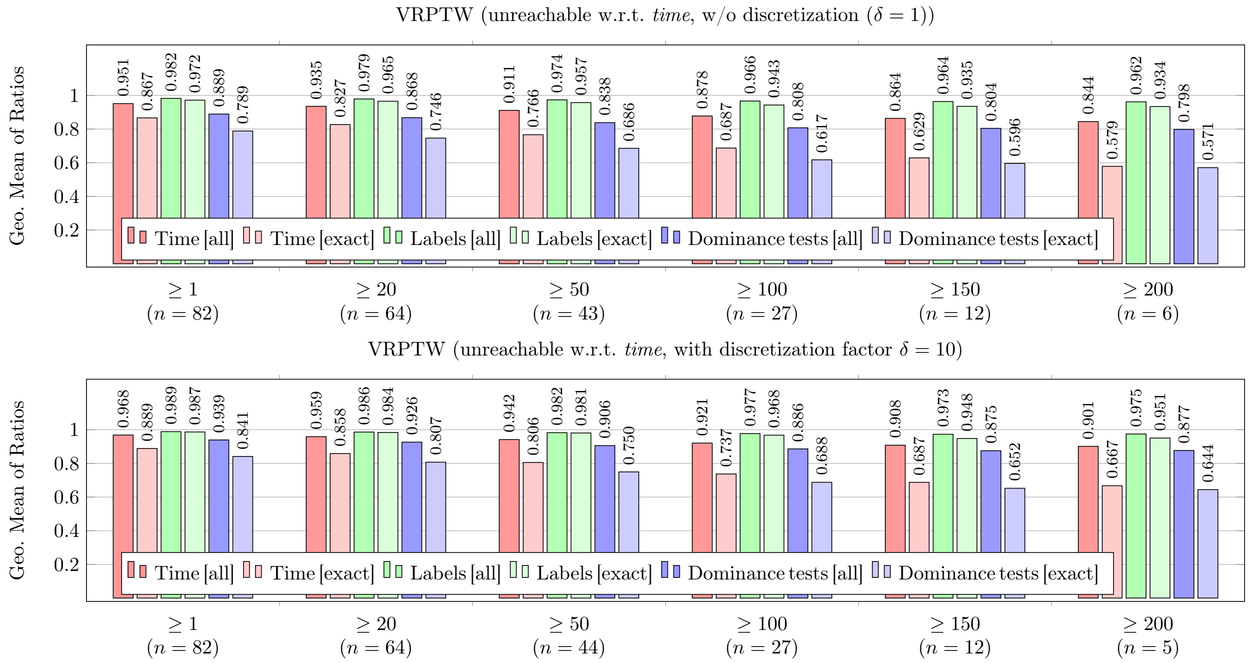


Figure 1: Geometric mean of the computation time ratios, number of generated labels, and number of dominance tests performed; we distinguish between all pricing iterations (partial and exact: [all]) and iterations with exact pricing using the complete network ([exact]); results are grouped by $\geq n_{SRI}$ where only instances for which the total number of added SRIs is at least n_{SRI} are considered. The upper and lower diagram show results without discretization and with discretization using the factor $\delta = 10$, respectively.

For the same reason as before, it is not promising to consider the *load* attribute for unreachability. Similarly, the attribute *bat* is only competitive in combination with *time*. Therefore, we consider the *time* attribute and the combination of the attributes *time* and *bat* (referred to as *time+bat* in the following).

For the EVRPTW-SP and EVRPTW-MP, we compare the improved dominance without discretization (i.e., $\delta = 1$) and discretization with the factor $\delta = 100$. The latter larger value of $\delta = 100$ reflects the different rounding rules for travel times in the definition of the EVRPTW instances, see beginning of Section 5.

For the VRPTW, EVRPTW-SP, and EVRPTW-MP, Figures 1–3 summarize the comparison of the original and improved dominance rule in the form of geometric means of the following criteria:

Time: The ratio of the computation times of the labeling algorithm equipped with the improved and the original dominance rule;

Labels: The ratio of the number of generated labels;

Dominance tests: The ratio of the number of dominance tests that have been performed.

We distinguish between:

all: All calls to the labeling algorithm are taken into account (partial pricing using arc-reduced networks and exact pricing using complete networks);

exact: Only calls to the labeling algorithm using the complete networks are considered.

Note that the latter filtering step focusses on the more difficult SPPRC subproblems that require, in comparison, substantially longer computation times. The combination with the above three criteria with *all* and *exact* gives six ratios displayed in different colors in the bar charts of Figures 1–3.

The geometric means of all ratios are first computed per instance. Afterwards, geometric means are computed for the VRPTW (Figure 1), EVRPTW-SP (Figure 2), and EVRPTW-MP (Figure 3) subsets. For analyzing the impact of the number n_{SRI} of SRIs, we filter over instances that have at least 1, 20, 50,

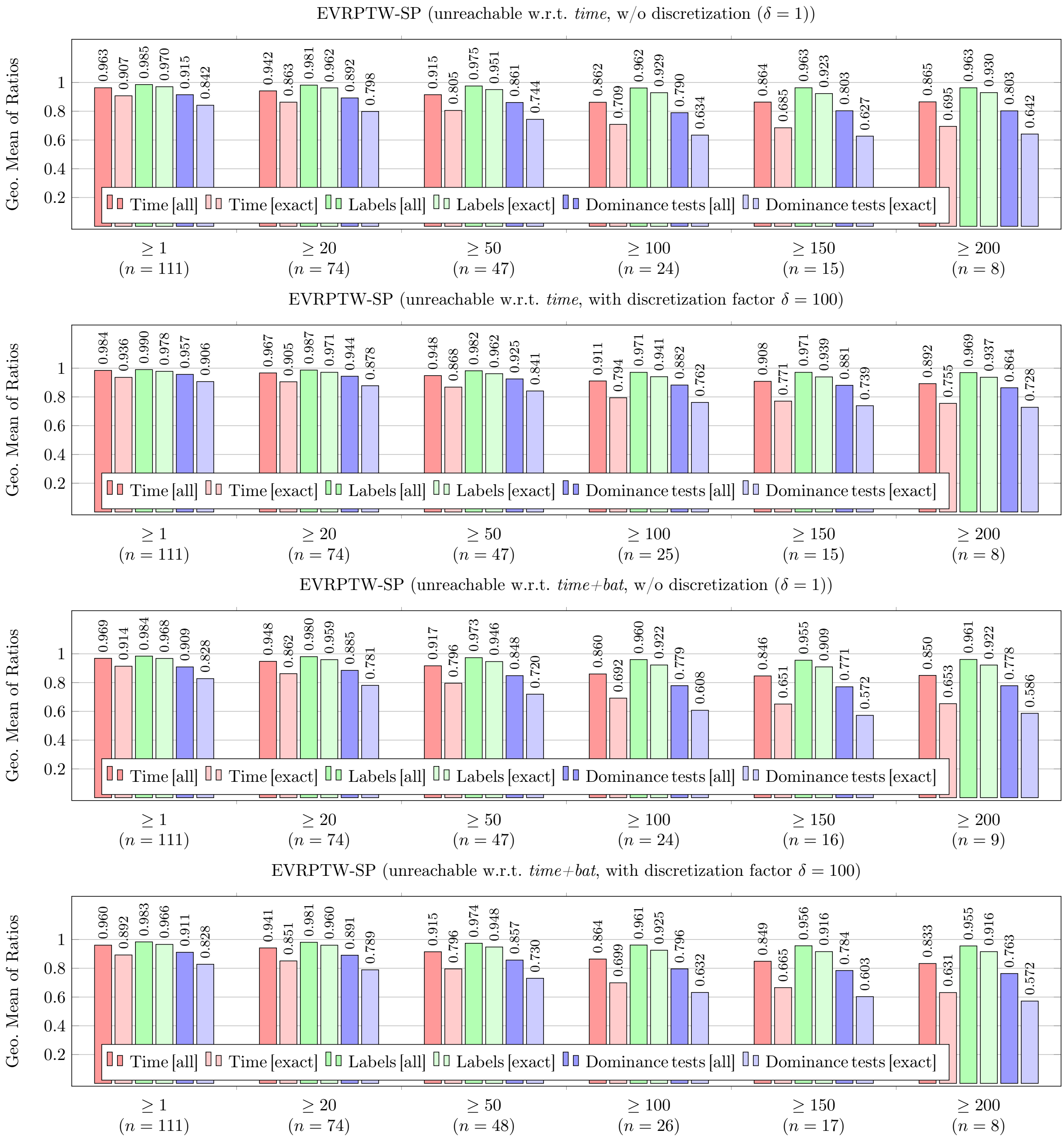


Figure 2: Geometric mean of the computation time ratios, number of generated labels, and number of dominance tests performed; we distinguish between all pricing iterations (partial and exact: [all]) and iterations with exact pricing using the complete network ([exact]); results are grouped by $\geq n_{SRI}$ where only instances for which the total number of added SRIs is at least n_{SRI} are considered. The four diagrams show results using attributes *time* and *time+bat* to define unreachable customers as well as results without discretization and with discretization using the factor $\delta = 100$.

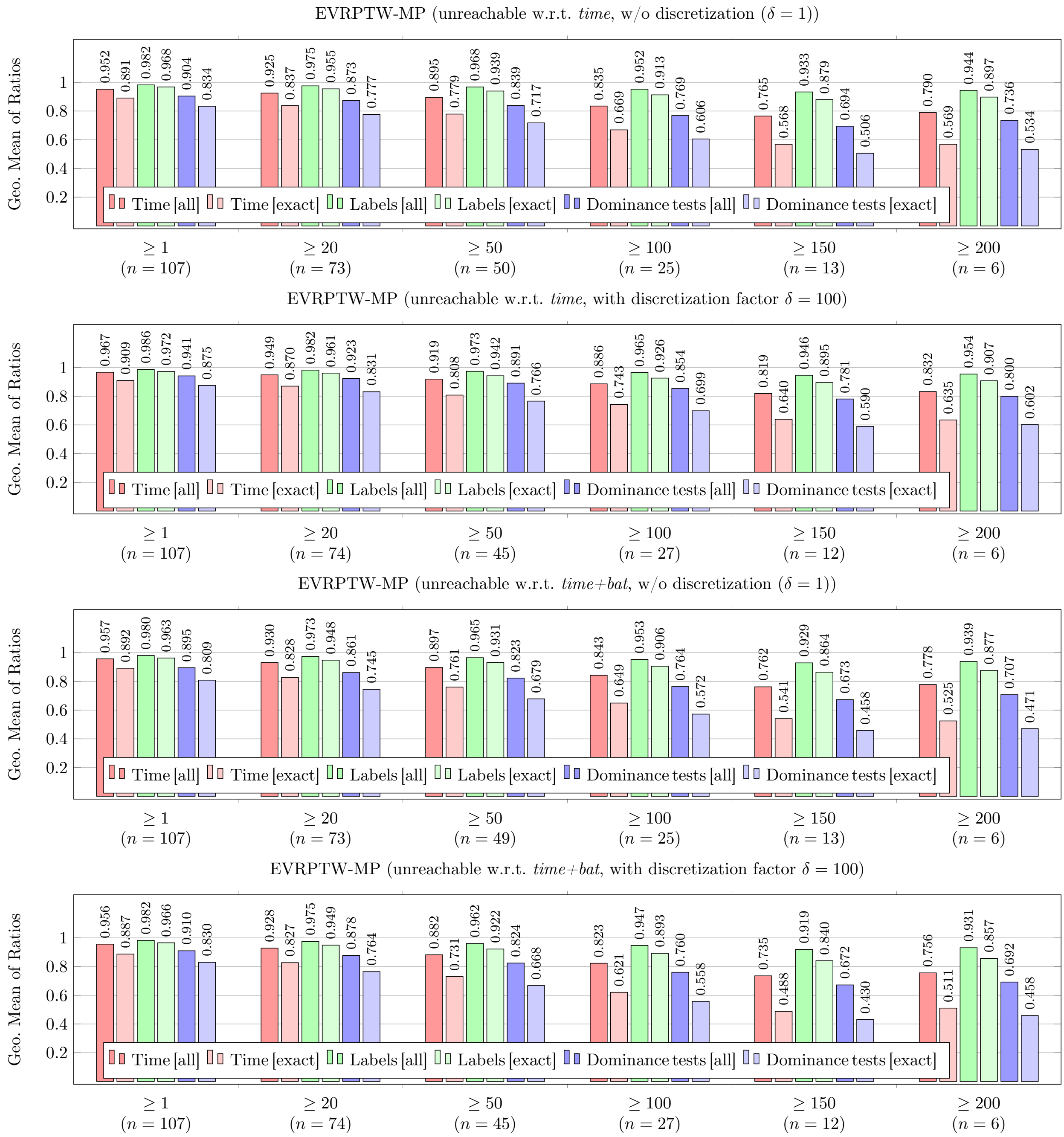


Figure 3: Geometric mean of the computation time ratios, number of generated labels, and number of dominance tests performed; we distinguish between all pricing iterations (partial and exact: [all]) and iterations with exact pricing using the complete network ([exact]); results are grouped by $\geq n_{SRI}$ where only instances for which the total number of added SRIs is at least n_{SRI} are considered. The four diagrams show results using attributes *time* and *time+bat* to define unreachable customers as well as results without discretization and with discretization using the factor $\delta = 100$.

100, 150, and 200 SRIs in the RMP. The resulting number of instances is indicated as '($n = \square$)'.

All average ratios are strictly smaller than one showing that the improved dominance is always beneficial. For each of the $12+24+24 = 60$ groups, the decrease in the number of dominance comparisons is larger than the decrease in computation time. In turn, the decrease in computation time is larger than the decrease in the number of generated labels. Note that the number of dominance comparisons is quadratic in the number of labels. It is responsible for a large share of the consumed computation time but other steps such as label generation, label extension, and memory management also contribute. This explains why results are even better regarding the number of dominance tests compared to SPPRC computation times.

Figures 1–3 also show that effects tend to be more pronounced for increasing values of n_{SRI}. Likewise, effects are more pronounced for *exact* and less for *all*. This means that the improved dominance is more helpful for more difficult and more time-consuming SPPRC subproblems. It is noticeable that these results are consistent over all 60 groups.

We discuss problem-specific results now, starting with the VRPTW. Figure 1 shows the results without discretization (i.e., $\delta = 1$) compared to with discretization using the factor $\delta = 10$:

- Here, discretization does not improve average results, neither in computation time, nor the number of labels, nor the number of dominance tests.
- Without discretization, the SPPRC computation times drop below 60% on average in the best case, i.e., for $n_{SRI} \geq 200$ and *exact*.
- For the remaining experiments with the VRPTW, we use the setting without discretization, i.e., factor $\delta = 1$.

Next, we discuss results for the two EVRPTW variants. Figure 2 shows results for the EVRPTW-SP (single recharge only) and Figure 3 for the EVRPTW-MP (multiple recharges possible). In both cases, we compare results for the attribute combinations *time* versus *time+bat* and without discretization (i.e., $\delta = 1$) versus with discretization using the factor $\delta = 100$:

- Comparing the SP and MP variant, the EVRPTW-MP has almost consistently smaller values in the corresponding groups. We interpret the results in the sense that multiple recharges allow slightly longer routes which makes the EVRPTW instances and the respective SPPRC subproblems more difficult. The improved dominance is more useful in these cases.
- The setting using the attribute combinations *time+bat* is slightly better than *time* for both EVRPTW variants SP and MP.
- Regarding discretization, there is no clear winner comparing the corresponding groups of instances.
- For the remaining experiments with the EVRPTW, we use the attribute combination *time+bat* and we further investigate the impact of the discretization.

5.2. Selection of the Discretization Factor and Type of Memory in EVRPTW Variants

With the following experiments, we determine for the two EVRPTW variants (MP and SP) which type of memory to be used (see Section 2.2) and whether discretization with the factor $\delta = 100$ is beneficial (see Section 4.3). In contrast to the experiments presented in Section 5.1, each run is now completely independent of the others. We impose a time limit of two hours (7200 seconds) and allow arc fixing to make the BPC algorithms competitive with state-of-the-art implementations. For the comparison, the two types of SRI memory (*full* and *limited*) are each tested against discretization factors of $\delta = 1$ and $\delta = 100$. Table 4 summarizes the results in the form of aggregate indicators that have the following meaning:

#opt: Number of instances solved to optimality within the time limit;

#fastest: Number of instances for which the BPC is faster than the three other BPC algorithms (ties are possible due to rounding using a precision of 1 milliseconds);

BPC time: Average computation time in seconds of the BPC algorithm (unsolved instances are counted with 7200 seconds);

% time prep.+cutting: Relative time spent with preparation (computation of the sets U_i^{time} and, for *time+bat*, of the sets U_i^{bat}) as well as the time for SRI separation in percent;

#SRIs: Average number of SRIs added to the RMP.

Discret. factor δ	Variant / SRI memory	EVRPTW-SP ($n = 111$) full	limited	EVRPTW-MP ($n = 107$) full	limited
1	#fastest	9	3	16	4
	#opt	90	**92**	**89**	88
	BPC time	1517.2	**1453.4**	1567.2	1541.7
	% time prep.+cutting	7.9	12.2	4.8	8.6
	#SRIs	63.7	103.2	64.8	103.4
100	#fastest	**54**	27	**54**	15
	#opt	90	**92**	88	88
	BPC time	1542.9	1463.3	**1517.1**	1534.7
	% time prep.+cutting	**1.0**	1.3	**0.8**	1.1
	#SRIs	63.8	103.3	65.0	103.5

Note: Better values are highlighted in **bold**.

Table 4: Comparison of the BPC algorithms using the improved dominance with the attribute combination *time+bat* w.r.t. full versus limited memory as well as without discretization ($\delta = 1$) and with discretization using factor $\delta = 100$.

In Table 4, the results for the EVRPTW-SP must be explained further. With limited memory, the two variants solve the same 92 instances optimally, while the variant with full memory and discretization using $\delta = 100$ solves another instance (`R208_21_25`). Hence, 93 instances are compared w.r.t. their runtime, which is consistent with the four variants having 9+3+54+27 = 93 runs that are the fastest. The following conclusions can now be drawn from Table 4:

- The fastest BPC variant for most instances is the one that combines a full memory with discretization using the factor $\delta = 100$. Also for the factor $\delta = 1$, the variant with full memory is faster than the one with limited memory. This is clear indication that the limited memory is less effective for accelerating the BPC for the EVRPTW with partial recharges compared to the VRPTW. For the latter, excellent results were reported in (Pecin et al., 2017a) that we can also confirm in Section 5.3.
  This outcome is somewhat unexpected, but can be explained with the indicator presented in Table 4: The limited memory clearly lowers the computational effort per SRI, but it strongly increases the number of generated SRIs on average (see *#SRIs*). This effect is less pronounced in the VRPTW explaining the inferior performance of the limited-memory BPC variants for the EVRPTW. (We observed a similar effect with the original dominance.)
- For both the EVRPTW-SP and the EVRPTW-MP, the above fastest variant fails to find two of the 92 optima and one of the 89 optima, respectively. This is probably due to unfavorable branching decisions. It seems that these results are independent of the use of a limited or full memory.
- The BPC times do not vary substantially among the four BPC variants: for the EVRPTW-SP they differ by not more than 6 percent and for the EVRPTW-MP by not more than 4 percent. Note that arithmetic averages mainly reflect long computation times of instances that are difficult to solve. This perfectly explains that the variants that are the fastest most of the time are not necessarily those that also minimize BPC times.
- The indicator *% time prep.+cutting* shows that without discretization, a substantial share of the total computation time is spent with the computation of the lookup tables (between 4.8 and 12.2 percent on average; for some otherwise fast-to-solve instances, the share can grow up to 70 percent). With discretization using the factor $\delta = 100$, these times vary around one percent so that they are negligible. This partly explains the large difference in *#fastest* values between variants with and without discretization.

For the final experiments on the EVRPTW, we choose the full memory and discretization with the factor $\delta = 100$.

5.3. Branch-Price-and-Cut Results

Finally, we present the results obtained with the BPC algorithms that only differ in the dominance rule which is either the original or the improved dominance. Recall that many important algorithmic design decisions have been determined in the above experiments: For the VRPTW, we use the attribute *time* for the determination of unreachable customers and no discretization, while we still have to find out which type of memory (full or limited) is beneficial. For the EVRPTW-SP and EVRPTW-MP, we use the attribute combination *time+bat*, discretization with $\delta = 100$, and full memory for the SRIs.

The final results are presented with the help of performance profiles (Dolan and Moré, 2002). Given a set of algorithms $\mathcal{A} = \{A_1, A_2, \ldots, A_p\}$ (e.g., $p = 4$ for the four BPC algorithms for the VRPTW), the performance profile $\rho_A(\tau)$ of an algorithm $A \in \mathcal{A}$ describes the ratio of instances that can be solved by A within a factor τ compared to the fastest algorithm, i.e., $\rho_A(\tau) = \left|\left\{I \in \mathcal{I} : t_I^A/t_I^* \leq \tau\right\}\right| / |\mathcal{I}|$, where $\mathcal{I}$ is the benchmark set, t_I^A is the computation time of algorithm A when applied to instance $I \in \mathcal{I}$, and t_I^* is the minimal computation time of all algorithms $\mathcal{A}$ solving the instance I. It follows that $\rho_A(1)$ is the percentage of instances for which algorithm A is the fastest. Unsolved instances are taken into account with a time of $t_I^A = \infty$ so that, for large values of τ, $\rho_A(\tau)$ is the percentage of instances solved by A within the time limit (we assume that ∞/∞ gives ∞).

Separate diagrams with performance profiles for the VRPTW, EVRPTW-SP, and EVRPTW-MP are presented in Figure 4. In all three cases, the BPC algorithms that use the improved dominance are the ones that perform best, i.e., the respective profiles indicate that a larger number of the instances are solved in a time not exceeding the time of fastest algorithm by a factor of τ.

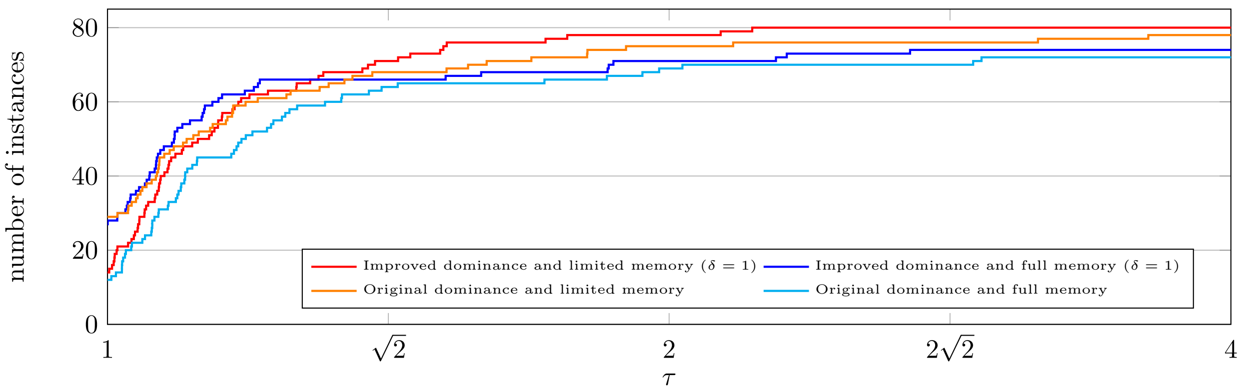
For the VRPTW (see Figure 4a), the two BPC algorithm, one with the improved dominance and a full memory and one with the original dominance and a limited memory, are the ones that are the fastest in 27 and 29 of the cases (see $\tau = 1$), respectively. However, starting from $\tau = 1.3$, the BPC algorithm with improved dominance and a limited memory solves more instances in the corresponding extended time; it is the only BPC algorithm terminating within the time limit for all 80 solved VRPTW instances. The point here seems to be that a limited memory is crucial only for some more difficult instances (e.g., instance `R209` is not solved with the full memory, but both variants with limited memory solves it within 400 seconds).

For the EVRPTW (see Figures 4b and 4c), the results are very clear: the improved dominance is superior to the original dominance. The variant with the improved dominance is the fastest for a larger number of instances (see $\tau = 1$). As already discussed in the previous section, unfavorable branching is probably responsible for the slightly lower number of proven optima in case of the EVRPTW-MP (88 instead of 89, see $\tau > 2$).

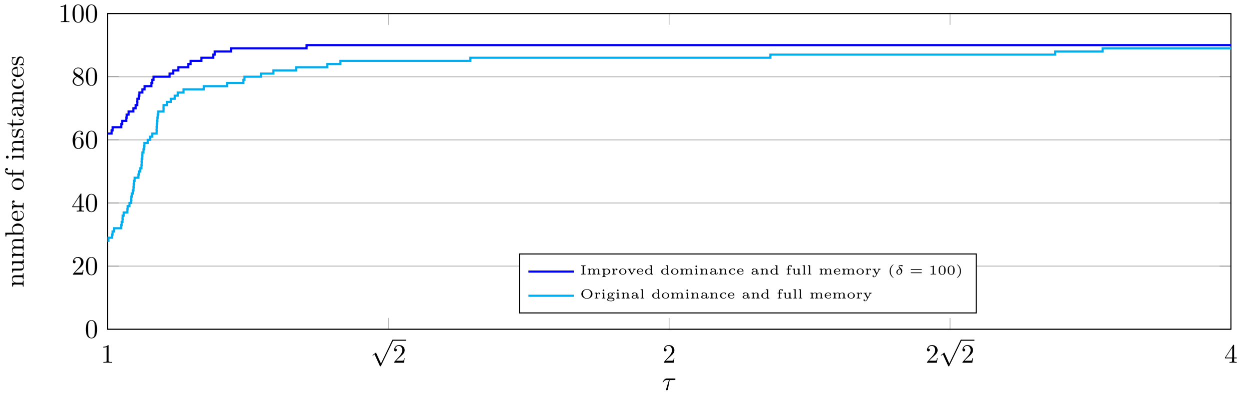
6. Conclusions

In this work, we have introduced an acceleration technique for BPC algorithms that use SRIs and a labeling-based solution approach to solve the column-generation subproblems. Essentially, the speedup results from an improved dominance rule for comparing labels. Using the concept of unreachable customers, the improved dominance leads, on average, to a smaller number of labels to store, extend, and compare in the labeling algorithm. This, in turn, accelerates pricing, which consumes most of the computational time of BPC algorithms in vehicle routing (and beyond). For some instances in our testbed, the total BPC computation time is more than halved by using the improved dominance rules.

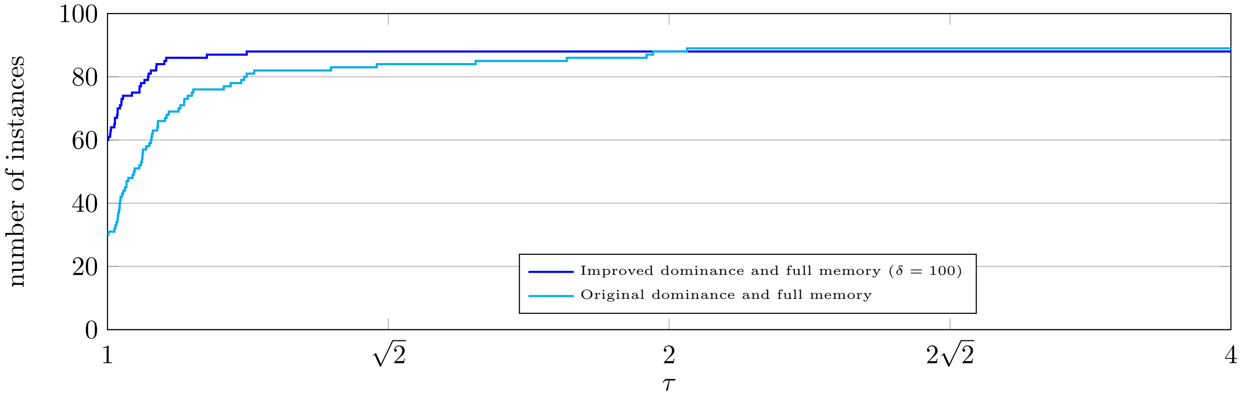
From the application side, we have shown how to compute sets of unreachable customers for the standard *load* and *time* resources as we find them in the VRPTW. An example of more complicated resource updates and resource dependencies is the EVRPTW with partial recharging, for which we have shown how to construct sets of unreachable customers with the help of the sum of two attribute values (one for the earliest service start time and the other for the maximum amount to be recharged). Which of the attributes or combinations of attributes is finally used should be decided by computational tests. For the EVRPTW, we have seen that using a single *time* attribute alone is inferior to the more sophisticated attribute combination (*time+bat*). The reason for this is that using multiple attributes in combination provides stronger improved dominance rules and the necessary pre-computation of the unreachable customer sets can be accelerated



(a) Four BPC algorithms for the VRPTW.



(b) Two BPC algorithms for the EVRPTW-SP.



(c) Two BPC algorithms for the EVRPTW-MP.

Figure 4: Performance profiles of BPC algorithms using either the original or improved dominance.

with the discretization technique that we introduced. Note that such a result is certainly not generalizable, as it strongly depends on the chosen instance set.

What is generalizable, however, is the principle of improving the SRI-related dominance by considering sets of unreachable customers. We have classified the SRI memories (full and limited, vertex-based or arc-based) as well as the dominance principles (identical completion, subpath completion, and possibly different completion) with respect to their compatibility. The compatible cases cover a wide range of problems from the family of VRPs (Irnich et al., 2014). In summary, there are several points that speak in favor of improving SRI-related dominance by unreachability: It is widely applicable, the implementation is rather straightforward because it affects a clear-cut and limited component of the BPC algorithm, and it can significantly accelerate the overall solution process.

Acknowledgement

This research was supported by the Deutsche Forschungsgemeinschaft (DFG) under grants GS 83/1-1 and IR 122/10-1 project no. 418727865. This support is gratefully acknowledged.

References

Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi:10.1007/s10107-007-0178-5.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011. doi:10.1287/opre.1110.0975.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(3):356–371, 2012. doi:10.1287/ijoc.1110.0456.

Isaac Balster, Teobaldo Bulhões, Pedro Munari, Artur Alves Pessoa, and Ruslan Sadykov. A new family of route formulations for split delivery vehicle routing problems. *Transportation Science*, 57(5):1359–1378, 2023. ISSN 1526-5447. doi:10.1287/trsc.2022.0085.

Maria Battarra, Jean-François Cordeau, and Manuel Iori. Pickup-and-delivery problems for goods transportation. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing*, chapter 6, pages 161–191. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi:10.1137/1.9781611973594.ch6.

Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8): 1232–1250, 2011. doi:10.1016/j.trb.2011.02.004.

Nicola Bianchessi, Timo Gschwind, and Stefan Irnich. Resource-window reduction by reduced costs in path-based formulations for routing and scheduling problems. *INFORMS Journal on Computing*, 36(1):224–244, 2023. doi:10.1287/ijoc.2022.0214.

Claudia Bode and Stefan Irnich. 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, 2014. doi:10.1016/j.ejor.2014.04.004.

Natashia Boland, John Dethridge, and Irina Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58 – 68, 2006. doi:10.1016/j.orl.2004.11.011.

Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985, 2019. doi:10.1287/trsc.2018.0878.

Guy Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192, 2010. doi:10.1287/opre.1090.0713.

Guy Desaulniers, Jacques Desrosiers, Irina Ioachim, Marius M. Solomon, François Soumis, and Daniel Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Teodor G. Crainic and Gilbert Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Springer, 1998. ISBN 978-0-7923-8161-7.

Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors. *Column Generation*. Springer, New York, NY, 2005. ISBN 0-387-25485-4.

Guy Desaulniers, François Lessard, and Ahmed Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi:10.1287/trsc.1070.0223.

Guy Desaulniers, Oli B.G. Madsen, and Stefan Ropke. The vehicle routing problem with time windows. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing*, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi:10.1137/1.9781611973594.ch5.

Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016. doi:10.1287/opre.2016.1535.

Guy Desaulniers, Timo Gschwind, and Stefan Irnich. Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Science*, 54(5):1170–1188, 2020. doi:10.1287/trsc.2020.0988.

Martin Desrochers, Jacques Desrosiers, and Marius M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi:10.1287/opre.40.2.342.

Jacques Desrosiers, Marco Lübbecke, Guy Desaulniers, and Jean Bertrand Gauthier. *Branch-and-Price*. Unpublished, 2024. https://www.researchgate.net/publication/381918232_Branch-and-Price.

Karl F. Doerner and Juan-José Salazar-González. Pickup-and-delivery problems for people transportation. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing*, chapter 7, pages 193–212. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi:10.1137/1.9781611973594.ch7.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi:10.1007/s101070100263.

Yvan Dumas, Jacques Desrosiers, and François Soumis. The pick-up and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991. doi:10.1016/0377-2217(91)90319-Q.

Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Guéguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi:10.1002/net.20033.

Asvin Goel and Stefan Irnich. An exact method for vehicle routing and truck driver scheduling problems. *Transportation Science*, 51(2):737–754, 2017. doi:10.1287/trsc.2016.0678.

Timo Gschwind and Stefan Irnich. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science*, 49(2):335–354, 2015. doi:10.1287/trsc.2014.0531.

Timo Gschwind, Stefan Irnich, Ann-Kathrin Rothenbächer, and Christian Tilk. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. *European Journal of Operational Research*, 266(2):521–530, 2018. doi:10.1016/j.ejor.2017.09.035.

Qie He, Stefan Irnich, and Yongjia Song. Branch-cut-and-price for the vehicle routing problem with time windows and convex node costs. *Transportation Science*, 2019. doi:10.1287/trsc.2019.0891.

Timo Hintsch and Stefan Irnich. Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, 280:164–178, 2019. doi:10.1016/j.ejor.2019.07.019.

Stefan Irnich. Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2007. doi:10.1007/s00291-007-0083-6.

Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, New York, 2005. doi:10.1007/0-387-25486-2_2.

Stefan Irnich and Daniel Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18(3):391–406, 2006. doi:10.1287/ijoc.1040.0117.

Stefan Irnich, Guy Desaulniers, Jacques Desrosiers, and Ahmed Hadjar. Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, 22(2):297–313, 2010. doi:10.1287/ijoc.1090.0341.

Stefan Irnich, Paolo Toth, and Daniele Vigo. The family of vehicle routing problems. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing*, chapter 1, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi:10.1137/1.9781611973594.ch1.

Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. doi:10.1287/opre.1070.0449.

Federico Liberatore, Giovanni Righini, and Matteo Salani. A column generation algorithm for the vehicle routing problem with soft time windows. *4OR*, 9(1):49–82, 2010. doi:10.1007/s10288-010-0136-6.

Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi:10.1287/opre.1050.0234.

Diego Pecin, Claudio Contardo, Guy Desaulniers, and Eduardo Uchoa. New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 29(3):489–502, 2017a. doi:10.1287/ijoc.2016.0744.

Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, 2017b. doi:10.1007/s12532-016-0108-8.

Diego Pecin, Artur Pessoa, Marcus Poggi, Eduardo Uchoa, and Haroldo Santos. Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, 45(3):206–209, 2017c. doi:10.1016/j.orl.2017.02.006.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and Francois Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018. doi:10.1287/ijoc.2017.0784.

Ted K. Ralphs, Leonid Kopman, William R. Pulleyblank, and Leslie E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2-3):343–359, 2003. doi:10.1007/s10107-002-0323-0.

Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi:10.1016/j.disopt.2006.05.007.

Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008. doi:10.1002/net.20212.

Ann-Kathrin Rothenbächer, Michael Drexl, and Stefan Irnich. Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, 52(5):1174–1190, 2018. doi:10.1287/trsc.2017.0765.

Ruslan Sadykov, Eduardo Uchoa, and Artur Pessoa. A bucket graph-based labeling algorithm with application to vehicle routing. *Transportation Science*, 55(1):4–28, 2021. doi:10.1287/trsc.2020.0985.

Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014. doi:10.1287/trsc.2013.0490.

Simon Spoorendonk and Guy Desaulniers. Clique inequalities applied to the vehicle routing problem with time windows. *INFOR*, 48(1):53–67, 2010. doi:10.3138/infor.48.1.053.

Christian Tilk, Ann-Kathrin Rothenbächer, Timo Gschwind, and Stefan Irnich. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, 261(2):530–539, 2017. doi:10.1016/j.ejor.2017.03.017.

Paolo Toth and Daniele Vigo, editors. *Vehicle routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. ISBN 978-1-61197-358-7.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013. doi:10.1016/j.cor.2012.07.018.