An Exact Algorithm for the Vehicle Routing Problem with Time Windows and Route Duration Minimization

Burak Koksal[a], Ayse Akbalik[a], Simon Belieres[b], Jean-François Cordeau[c], Nicolas Jozefowiez[a,*], Jorge E. Mendoza[c], Anass Nagih[d]

[a] *Université de Lorraine, CNRS, LORIA, F-57000 Metz, France*
[b] *Department of Information, Operations and Decision Sciences, TBS Business School, 20 Bd Lascrosses, Toulouse, 31000, France*
[c] *Department of Logistics and Operations Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7, Canada*
[d] *LCOMS, University of Lorraine, 7 rue Marconi, Metz, 57070, France*

Abstract

We study a variant of the Vehicle Routing Problem with Time Windows (VRPTW) in which the objective is to minimize the total route duration rather than the total travel distance or time. In this variant, departure times from the depot are part of the main decision variables: by delaying vehicle departure times, waiting times at customer locations can be reduced, thereby decreasing the total route duration. To solve this problem, we develop an exact method based on column generation. Because departure time flexibility introduces a fundamental challenge in the pricing problem, we propose a dynamic programming labeling algorithm that extends an existing procedure for the Minimum Tour Duration Problem. We further extend this framework to compute completion bounds and solve the problem using an enumeration algorithm. Our main contribution is a new set of dominance rules linking route duration attributes and accumulated dual values. We prove that these rules are necessary and sufficient for dominance between labels, meaning that every dominance relation that can be inferred from the label attributes is detected by our criterion. We compare our approach with the only two exact methods currently available for the VRPTW with duration minimization (VRPTWDM). We show theoretically that our dominance rules are strictly stronger than those used in the literature. Computational experiments confirm that the resulting pricing algorithm achieves substantially stronger pruning and significantly fewer label extensions than previous dominance criteria. More generally, the proposed dominance framework can be applied to other vehicle routing problems with duration-minimizing objectives.

Keywords: vehicle routing, route duration minimization, column generation, enumeration, labeling algorithm

*Corresponding author
Email address: nicolas.jozefowiez@univ-lorraine.fr (Nicolas Jozefowiez)

1. Introduction

This study investigates a variant of the Vehicle Routing Problem with Time Windows (VRPTW) that involves an important yet less-studied objective function. In most VRPTW variants, the objective is to minimize the total travel distance or the total travel time (Desaulniers et al., 2014), assuming that all routes start at a common time that is fixed a priori. In this study, we consider a variant where the objective is to minimize the total route duration. For a given route, the duration is defined as the time elapsed between the vehicle's departure from the depot and its return. As such, the departure time of each vehicle is a decision variable, as postponing the departure may potentially decrease waiting times at customer locations, and reduce the route duration. Column generation is well established in the literature as an effective approach for the VRPTW. However, one of the main challenges of applying column generation to this specific problem lies in the pricing subproblem, for which an efficient dynamic-programming algorithm must be designed. We propose a dynamic programming labeling algorithm based on the work of Tilk and Irnich (2017). We also propose completion bounds that enable the solution of the problem through an enumeration process (Baldacci et al., 2011). We address the most natural VRPTW extension in this study; however, the dynamic programming algorithm proposed for the pricing problem is adaptable to related routing problems with duration minimization objectives.

We refer to this problem as the VRPTW with route duration minimization (VRPTWDM), which was introduced by Savelsbergh (1992). In his seminal paper, Savelsbergh (1992) focused on local search operators. He proposed efficient edge-exchange improvement methods specifically tailored to minimize the total route duration and demonstrated that the influence of departure times on route durations must be considered when evaluating local search moves. Although the VRPTWDM is relevant to many applications, notably in the context of city logistics, it has received significantly less attention in the literature than VRPTW variants aimed at minimizing the total travel distance or time. Next, we review articles relevant to our study.

Some studies have investigated variants of the Traveling Salesman Problem with Time Windows (TSPTW) aimed at minimizing the tour duration. Favaretto et al. (2006) develop an ant-colony metaheuristic for TSPTW variants in which departure times and waiting times are explicitly considered. Kara et al. (2013) propose an Integer Linear Programming formulation for a TSPTW variant that aims to minimize the sum of travel and waiting times. Their objective is to minimize the tour duration of a single vehicle. Tilk and Irnich (2017) study the same problem and refer to it as the Minimum Tour Duration Problem (MTDP). They design an exact dynamic programming algorithm that propagates several interdependent temporal resources and derive dominance rules that remain valid despite the non-additive nature of tour durations when departure times are flexible. Ultimately, these contributions highlight the algorithmic difficulties involved in handling a minimum-duration objective, even in the context of a single vehicle.

Next, we focus on VRP-related studies that involve the management of multiple vehicles to satisfy demand. We first note that route duration plays a central role in contexts involving drivers' working hour regulations and integrated planning settings. Goel (2009)

studies vehicle routing and scheduling under European drivers' working hours, which involve limited driving times, mandatory breaks, and rest periods. He proposes an optimization-based approach that embeds duration-aware and regulation-aware feasibility checks into a large neighborhood search framework. Rijal et al. (2023) consider the joint planning of warehouse operations and vehicle routing and show that minimizing individual route durations significantly impacts warehouse dispatching and order picking efficiency. However, in both of these studies, route durations were constrained by sophisticated feasibility requirements or monitored to synchronize some operational resources rather than being treated as the primary objective.

Dabia et al. (2013) propose a branch-and-price algorithm for a time-dependent VRPTW with duration minimization, in which vehicles may delay their departure from the depot to avoid congestion and thereby reduce route duration. Their set-partitioning master problem is coupled with a pricing problem solved by a labeling algorithm that propagates piecewise linear ready time functions and uses specialized dominance and merging strategies to achieve its goals. Lera-Romero et al. (2020) revisit the same problem, proposing a branch-cut-and-price algorithm that integrates acceleration techniques and devotes particular care to the labeling algorithm used in the pricing problem. From a modeling perspective, the classical VRPTWDM with time-independent travel times can be considered as a special case of these time-dependent formulations, which is obtained by assuming constant travel times on all arcs.

Recently, duration-minimizing objectives have been studied for VRPTW variants. Schmidt (2024) develops a branch-price-and-cut algorithm for a VRP with drones and time windows in which the objective is to minimize the duration of combined truck-and-drone tours. Her formulation couples vehicle routes and drone flights and relies on a duration-aware labeling algorithm in the pricing subproblem, illustrating that duration-based objectives require carefully designed label states and dominance rules when departure times and waiting decisions are flexible. A study addressing the same problem as this article is Michelini et al. (2026), which provides a natural benchmark for this study. The authors propose a dynamic programming algorithm to solve the pricing problems. This algorithm is embedded within a branch-and-price framework, resulting in an exact solution to the problem addressed. Lichau (2026) proposes a branch-and-cut-and-price algorithm for the VRPTWDM. This algorithm is an adaptation to the duration minimization of an algorithm initially designed for a problem where the duration is constrained.

Although minimizing the total route duration in the context of the VRPTW is practically important, it poses significant algorithmic challenges. Specifically, introducing flexible departure times yields a fundamental difference from most VRPTW variants found in the literature. Typically, extending two paths with the same sequence of nodes does not change their relative cost. In other words, for two comparable labels, if one has a lower cost than the other, performing the same extension should preserve this difference between the two labels. However, this property does not hold when the route duration is minimized and the departure times can be delayed. Indeed, postponing the departure from the depot enables the reduction of waiting times incurred when arriving at customer locations before their

earliest service times, thereby reducing the route duration. This has important implications for dynamic programming approaches used to solve pricing problems in column-generation methods. In particular, dominance between labels cannot rely solely on a direct comparison of reduced costs, as this may lead to the discarding of labels that could still generate optimal routes. Michelini et al. (2026) explicitly identify this issue and introduce an additional dominance rule to address it, which is essential for the practical convergence of the pricing algorithm.

The contributions of this study are fourfold. First, we propose a dynamic programming algorithm to solve the pricing problem in the column generation process. Our approach is based on the labeling scheme proposed by Tilk and Irnich (2017). To enforce the dominance rules, we introduce rules that link accumulated dual values to key attributes governing route duration. We also define a backward pricing algorithm and use it to define completion bounds that can be used in the enumeration approach proposed by Baldacci et al. (2011). We prove that the proposed rules are necessary and sufficient for dominance between labels, i.e., they enable detecting every dominance relation that can be inferred from the label attributes. Second, we show theoretically that the proposed set of rules is stronger than that proposed by Michelini et al. (2026) and Lichau (2026). Third, to isolate the effect of the dominance rules on the solution of the pricing, we compare the three dynamic programming algorithms using an identical implementation, i.e., dominance criterion is the only varying component. Our experiments computationally validate that the proposed criterion leads to substantially fewer label extensions and significantly faster solution times. Fourth, we evaluate the proposed approach on Solomon benchmark instances with up to 100 customers. Our approach outperforms that of Michelini et al. (2026), both in terms of the number of instances solved and computational time. It also achieves competitive results with respect to Lichau (2026), despite relying on a substantially simpler framework. This suggests that the gains obtained from the proposed dominance criterion translate into strong computational performance even without the advanced acceleration techniques employed in state-of-the-art branch-cut-and-price implementations.

The remainder of this paper is organized as follows. Section 2 formally defines the VRPTWDM and presents its set-partitioning formulation. Section 3 details the pricing problem and labeling algorithm, with particular attention to the new dominance rules. Section 4 describes the enumeration algorithm. Section 5 presents the computational results. Finally, Section 6 presents the conclusions and outlines directions for future research.

2. Problem description and formulation

The VRPTWDM is defined on a directed graph $G = (N, A)$ where N and A denote the node and arc sets, respectively. Nodes 0 and $n+1$ both represent the depot, that is, $n+1$ is a copy of 0, while the subset $V = \{1, \ldots, n\} \subset N$ represents the customer nodes. Each node $i \in N$ is associated with a demand q_i ($q_0 = q_{n+1} = 0$), service time s_i ($s_0 = s_{n+1} = 0$), and time window $[a_i, b_i]$ ($a_0 = a_{n+1}$ and $b_0 = b_{n+1}$), where a_i and b_i denote the earliest and latest times at which the service can begin. Specifically, a vehicle that arrives at node i before a_i must wait until a_i to start the service, and the service at i cannot start after b_i.

For each arc $(i, j) \in A$, let t_{ij} denote the travel time from node i to node j. To simplify the notation, we assume that the service times are incorporated into the travel times without loss of generality. The graph is assumed to be complete and to satisfy the triangle inequality.

A fleet of M vehicles with a homogeneous capacity Q is available at the depot. Each vehicle performs at most one route, which starts and ends at the depot, and the sum of the demands served by each vehicle cannot exceed Q. The demand of each customer i must be served by exactly one vehicle. The duration of a route is defined as the time elapsed between the moment the vehicle departs from the depot and the moment it returns to the depot. The objective is to determine a set of routes along with their schedules, ensuring that all constraints are satisfied while minimizing the total duration of the routes.

We formulate the VRPTWDM as a set-partitioning problem. Let R denote the set of feasible routes. For each route $r \in R$, let a_{ir} denote the number of times customer $i \in V$ is visited by route r, and let c_r denote the minimum duration of route r. The binary decision variable λ_r indicates whether route r is selected. The master problem is formulated as follows:

$$\min \quad \sum_{r \in R} c_r \lambda_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ir} \lambda_r = 1, \qquad \forall i \in V \tag{2}$$

$$\sum_{r \in R} \lambda_r \leq M, \tag{3}$$

$$\lambda_r \in \{0, 1\}, \qquad \forall r \in R \tag{4}$$

The objective function (1) minimizes the total duration. Constraints (2) ensure that each customer is served exactly once. Constraint (3) imposes an upper bound on the number of vehicles.

The number of feasible routes $|R|$ grows exponentially with the number of customers, so enumerating R and solving the master problem directly is impractical. We therefore solve the linear relaxation of (1)–(4) by column generation. Rather than handling the full set R, the method works with a *restricted master problem* (RMP), which is the linear relaxation of the master problem restricted to a subset of routes $R' \subseteq R$. The RMP is solved, and a pricing problem searches for routes that improve its solution. Any such route is added to R', and the process repeats. It terminates when no improving route remains, yielding an optimal solution of the linear relaxation of the master problem.

3. Pricing algorithm

Section 3.1 presents the pricing problem, while the remaining sections describe the proposed labeling algorithm, which constitutes the main contribution of this study. Our presentation focuses on duration-related labels, which play a key role in the efficiency of the solution. These attributes, together with their extension functions, are based on the work

of Tilk and Irnich (2017) for the MTDP and are summarized in Section 3.2. We discuss in Section 3.3 different possibilities for extending the dominance rules of Tilk and Irnich (2017) for the duration to consider the reduced cost. Then, we propose the dominance rules and prove their correctness in Section 3.4. We further prove in Section 3.5 that these rules are necessary as well as sufficient. In Sections 3.6 and 3.7, we compare our dominance rules with those of Michelini et al. (2026) and Lichau (2026), respectively. We also provide examples in which our approach allows pruning labels that would not be discarded by their method. Finally, Section 3.8 recalls the additional label attributes required for the VRPTWDM, such as vehicle capacity and visited-node tracking.

3.1. Pricing problem definition

The pricing problem identifies feasible routes with negative reduced costs that can improve the current RMP solution. Let π_i denote the optimal dual value associated with Constraint (2) for customer $i \in V$. Similarly, let π^u denote the optimal dual value associated with Constraint (3). The reduced cost of route r is

$$\nu_r = c_r - \sum_{i \in V} a_{ir}\pi_i - \pi^u. \tag{5}$$

The pricing problem is formulated as an elementary shortest path problem with resource constraints (ESPPRC) on the graph $G = (N, A)$ with dual costs associated with the nodes in addition to the travel times of the arcs.

3.2. Label attributes for duration minimization

A label represents a partial path from the origin depot and the resources consumed along that path. For duration minimization, we adopt the label formulation of Tilk and Irnich (2017), who developed it for the MTDP. Each label ℓ at node i has the following attributes:

- i: the last visited node,
- e: the earliest possible departure time from node i (Note that it can also be seen as the earliest possible service time as the service time for a customer is included in the travel times of the outbound arcs),
- d: the minimum duration from the origin depot to node i,
- h: the negative of the latest feasible departure time from the origin depot that achieves duration d.

A label $\ell = (i, e, d, h)$ can be extended to an adjacent node j if and only if $e + t_{ij} \leq b_j$. At the origin depot, the initial label is $(0, a_0, 0, -b_0)$. If the extension is feasible, a new label $\ell' = (j, e', d', h')$ is created with the following attributes:

$$e' = \max\{e + t_{ij}, a_j\}, \tag{6}$$
$$d' = \max\{d + t_{ij}, h + a_j\}, \tag{7}$$
$$h' = \max\{d + t_{ij} - b_j, h\}. \tag{8}$$

Equation (6) computes the earliest departure time from j. Equation (7) updates the minimum route duration. If the vehicle arrives before a_j, the route duration is updated to $h + a_j$, reflecting the route duration obtained by delaying the departure time from the origin depot to arrive at j at time a_j, thereby absorbing the waiting times. Equation (8) updates the negative of the latest feasible departure time.

3.3. Extension of the dominance rules of Tilk and Irnich (2017)

As the number of labels grows exponentially, dominance rules are required to prune them. When considering the duration, Tilk and Irnich (2017) used the following three rules to test whether a label $\ell_1 = (e_1, d_1, h_1)$ dominates a label $\ell_2 = (e_2, d_2, h_2)$:

$$e_1 \leq e_2, \tag{9}$$
$$d_1 \leq d_2, \tag{10}$$
$$h_1 \leq h_2. \tag{11}$$

To identify routes with minimum reduced costs, the labels must track dual contributions. Let σ refer to the dual value accumulated along the path. At the origin depot, we have $\sigma = -\pi^u$. When extending from node i to customer node j, attribute σ' is computed as follows:

$$\sigma' = \sigma - \pi_j. \tag{12}$$

The reduced cost of a label is equal to $d + \sigma$. A natural approach is to add the following rule to Rules (9)–(11):

$$\sigma_1 \leq \sigma_2. \tag{13}$$

However, Rule (13) is very strict and does not allow a compromise between d and σ. It would be better to replace Rules (10) and (13) with a rule on the reduced cost as follows:

$$d_1 + \sigma_1 \leq d_2 + \sigma_2. \tag{14}$$

Unfortunately, this rule is not preserved when extending the labels in conjunction with Rules (9) and (11).

Example 3.1 (Rule (14) is not preserved under extension)**.** Consider two labels ℓ_1 and ℓ_2 at the same node i with $e_1 = e_2$.

Label	d	h	σ	$d + \sigma$
ℓ_1	10	-80	0	10
ℓ_2	20	-75	-10	10

Label ℓ_1 satisfies the dominance rules, that is, (9), (11), and (14) with respect to ℓ_2. Specifically, we have $e_1 = e_2$, $h_1 = -80 \leq -75 = h_2$, and $d_1 + \sigma_1 = 10 \leq 10 = d_2 + \sigma_2$. Under the standard dominance criteria, ℓ_1 dominates ℓ_2 and ℓ_2 is discarded.

Now, consider extending both labels to node j with $a_j = 100$, $b_j = 200$, $t_{ij} = 5$, and $\pi_j = 100$ applying the extension formulas as follows:

$$d'_1 = \max\{d_1 + t_{ij}, h_1 + a_j\} = \max\{10 + 5, -80 + 100\} = \max\{15, 20\} = 20,$$
$$d'_2 = \max\{d_2 + t_{ij}, h_2 + a_j\} = \max\{20 + 5, -75 + 100\} = \max\{25, 25\} = 25.$$

The reduced costs after the extension are as follows.

$$d'_1 + \sigma'_1 = d'_1 + (\sigma_1 - \pi_j) = 20 + (0 - 100) = -80,$$
$$d'_2 + \sigma'_2 = d'_2 + (\sigma_2 - \pi_j) = 25 + (-10 - 100) = -85.$$

Label ℓ_2 now yields a lower reduced cost than ℓ_1 and should not be pruned.

The counterexample reveals the underlying issue: although ℓ_1 and ℓ_2 have equal reduced costs before extension, label ℓ_1 achieves this with a worse (higher) dual contribution σ compensated by a better duration d while also having a significantly lower h value. Upon extension to a node with a late time window opening ($a_j = 100$), the h attribute becomes binding in the duration computation using Equation (7). The advantage that ℓ_1 held in duration vanishes, but the disadvantage in σ remains the same.

To address this issue, we analyze the evolution of the reduced cost gap between the two labels under extension. Suppose ℓ_1 satisfies the reduced cost condition with respect to ℓ_2, that is, $d_1 + \sigma_1 \leq d_2 + \sigma_2$, or, equivalently, $d_1 - d_2 \leq \sigma_2 - \sigma_1$. We wish to ensure that this inequality is preserved after any extension, that is, that the inequality $d'_1 - d'_2 \leq \sigma'_2 - \sigma'_1$ holds.

Because the dual contributions are updated by subtracting the same value π_j from both labels, we have $\sigma'_2 - \sigma'_1 = \sigma_2 - \sigma_1$. Thus, $d_1 - d_2 \leq \sigma_2 - \sigma_1$ is preserved after the extension if $d'_1 - d'_2 \leq \sigma_2 - \sigma_1$. To enforce this condition, we first look for an upper bound Γ such that $d'_1 - d'_2 \leq \Gamma$. Next, we add a rule $\Gamma \leq \sigma_2 - \sigma_1$ that must be satisfied by our labels, thus ensuring that $d'_1 - d'_2 \leq \sigma_2 - \sigma_1$ and that the rule $d_1 - d_2 \leq \sigma_2 - \sigma_1$ is preserved by the extension. Recalling that the extension formula (7) computes the new duration as $d' = \max\{d + t_{ij}, h + a_j\}$, we investigate the four possible cases.

Case 1: $d_1 + t_{ij} \geq h_1 + a_j$ and $d_2 + t_{ij} \geq h_2 + a_j$.
Then $d'_1 - d'_2 = d_1 + t_{ij} - d_2 - t_{ij} = d_1 - d_2 \leq \sigma_2 - \sigma_1$.

Case 2: $d_1 + t_{ij} \geq h_1 + a_j$ and $d_2 + t_{ij} < h_2 + a_j$.
Then, $d'_1 - d'_2 = d_1 + t_{ij} - h_2 - a_j$. Because $h_2 > d_2 + t_{ij} - a_j$, we have $d'_1 - d'_2 \leq d_1 + t_{ij} - d_2 - t_{ij} + a_j - a_j = d_1 - d_2 \leq \sigma_2 - \sigma_1$.

Case 3: $d_1 + t_{ij} < h_1 + a_j$ and $d_2 + t_{ij} \geq h_2 + a_j$.
Then, $d'_1 - d'_2 = h_1 + a_j - d_2 - t_{ij}$. Because $d_2 \geq h_2 + a_j - t_{ij}$, we have $d'_1 - d'_2 \leq h_1 + a_j - h_2 - a_j + t_{ij} - t_{ij} = h_1 - h_2$. In this case, $h_1 - h_2$ constitutes an upper bound on $d'_1 - d'_2$, such that Γ should be at least equal to $h_1 - h_2$.

Case 4: $d_1 + t_{ij} < h_1 + a_j$ and $d_2 + t_{ij} < h_2 + a_j$.

Then, $d'_1 - d'_2 = h_1 + a_j - h_2 - a_j = h_1 - h_2$. In this case, $h_1 - h_2$ again reflects an upper bound on $d'_1 - d'_2$, such that Γ should be at least equal to $h_1 - h_2$.

Ultimately, we note that for Cases 1 and 2, the rule is preserved by the extension. For both Cases 3 and 4, we computed an upper bound $\Gamma = h_1 - h_2$ on $d'_1 - d'_2$. Therefore, we introduce the following dominance rule:

$$h_1 - h_2 \leq \sigma_2 - \sigma_1. \tag{15}$$

Together, these two conditions guarantee that the reduced cost advantage of ℓ_1 over ℓ_2 is preserved under any feasible extension.

3.4. Proposed set of dominance rules

Following the previous analysis, we restate the proposed dominance rules for convenience. A label ℓ_1 dominates a label ℓ_2 at a node i if

$$e_1 \leq e_2, \tag{9}$$
$$d_1 + \sigma_1 \leq d_2 + \sigma_2, \tag{14}$$
$$h_1 - h_2 \leq \sigma_2 - \sigma_1. \tag{15}$$

Rule (9) is standard for VRPTW-type problems, as it guarantees that if ℓ_2 can be extended to node j with respect to its time windows, then ℓ_1 can also be extended to j. Rule (14) compares the reduced costs of partial paths. Rule (15) is a cost-timing trade-off: if ℓ_1 has a worse (larger) h value, the difference must be compensated by a sufficient advantage in the accumulated dual cost.

Next, we prove that these rules hold for extensions.

Proposition 3.1. *Let ℓ_1 and ℓ_2 be two labels at node i such that ℓ_1 satisfies the dominance Rules* (9), (14), *and* (15) *with respect to ℓ_2. Then, Rules* (9), (14), *and* (15) *are preserved while extending labels ℓ_1 and ℓ_2 to any node j.*

Proof. This is well known for Rule (9). In the previous analysis of the reduced cost, we established that Rule (14) is preserved by extension whenever the labels additionally satisfy Rule (15). We next prove that Rule (15) holds after the extension.

Let ℓ'_1 and ℓ'_2 denote the labels obtained by extending ℓ_1 and ℓ_2 to node j. We must show that $h'_1 - h'_2 \leq \sigma'_2 - \sigma'_1$. Because $\sigma'_1 = \sigma_1 - \pi_j$ and $\sigma'_2 = \sigma_2 - \pi_j$, this is equivalent to showing $h'_1 - h'_2 \leq \sigma_2 - \sigma_1$. We next define $g_1 = d_1 + t_{ij} - b_j$ and $g_2 = d_2 + t_{ij} - b_j$. By equation (8), we have $h'_1 = \max\{g_1, h_1\}$ and $h'_2 = \max\{g_2, h_2\}$. We consider four cases.

Case 1: $h_1 \geq g_1$ and $h_2 \geq g_2$.

Then $h'_1 = h_1$ and $h'_2 = h_2$. By Rule (15) for ℓ_1 and ℓ_2,

$$h'_1 - h'_2 = h_1 - h_2 \leq \sigma_2 - \sigma_1.$$

Case 2: $h_1 \geq g_1$ and $h_2 < g_2$.

Then $h'_1 = h_1$ and $h'_2 = g_2 > h_2$. Thus:

$$h'_1 - h'_2 = h_1 - g_2 < h_1 - h_2 \leq \sigma_2 - \sigma_1.$$

Case 3: $h_1 < g_1$ and $h_2 \geq g_2$.
Then $h'_1 = g_1$ and $h'_2 = h_2 \geq g_2$. From the reduced cost Rule (14), we obtain

$$d_1 + \sigma_1 \leq d_2 + \sigma_2 \implies d_1 - d_2 \leq \sigma_2 - \sigma_1.$$

We know that

$$g_1 - g_2 = (d_1 + t_{ij} - b_j) - (d_2 + t_{ij} - b_j) = d_1 - d_2.$$

Because $h'_2 \geq g_2$, we have $h'_1 - h'_2 = g_1 - h'_2 \leq g_1 - g_2 \leq \sigma_2 - \sigma_1$.

Case 4: $h_1 < g_1$ and $h_2 < g_2$.
Then $h'_1 = g_1$ and $h'_2 = g_2$. As in Case 3:

$$h'_1 - h'_2 = g_1 - g_2 \leq \sigma_2 - \sigma_1. \qquad \square$$

Because these rules also hold at the final depot in a route, we ensure that we find the minimum reduced cost route.

3.5. Necessity of the dominance rules

In the previous section, we established that the Rules (9), (14), and (15) are sufficient to detect if a label ℓ_1 dominates a label ℓ_2. We next prove that these rules are necessary, i.e., if ℓ_1 dominates ℓ_2 then the rules are respected.

We denote a *completion* of a label ℓ as a sequence of feasible extensions of ℓ that ends at the destination depot $n+1$, thereby producing a feasible route. Let d^{n+1}, σ^{n+1}, and $d^{n+1} + \sigma^{n+1}$ denote the duration, the accumulated dual value, and the reduced cost of such completion, respectively. A label ℓ_1 dominates a label ℓ_2 if (i) every completion of ℓ_2 is also feasible for ℓ_1, and (ii) the completion yields a lower reduced cost when associated to ℓ_1, i.e., $d_1^{n+1} + \sigma_1^{n+1} \leq d_2^{n+1} + \sigma_2^{n+1}$, or equivalently $d_1^{n+1} - d_2^{n+1} + \leq \sigma_2^{n+1} - \sigma_1^{n+1}$. When this condition holds, ℓ_2 can be discarded without losing any optimal route. Recalling that a completion applies the same sequence of arcs to both labels, we have that $\sigma_2^{n+1} - \sigma_1^{n+1} = \sigma_2 - \sigma_1$. As such, ℓ_2 can be discarded if we have that

$$d_1^{n+1} - d_2^{n+1} \;\leq\; \sigma_2 - \sigma_1 \qquad \text{for every completion,} \tag{16}$$

where σ_1 and σ_2 refer to the accumulated dual values of ℓ_1 and ℓ_2, respectively.

The earliest-departure rule (9) guarantees that any extension feasible for ℓ_2 is also feasible for ℓ_1. We next prove that the duration gap $d_1^{n+1} - d_2^{n+1}$ is necessarily bounded by $\sigma_2 - \sigma_1$. Recall that Rule (14) ensures that $d_1 - d_2 \leq \sigma_2 - \sigma_1$ and Rule (15) ensures that $h_1 - h_2 \leq \sigma_2 - \sigma_1$.

Lemma 3.1. *Assuming two labels ℓ_1 and ℓ_2 such that ℓ_1 dominates ℓ_2, then $d_1^{n+1} - d_2^{n+1} \leq \max\{d_1 - d_2,\, h_1 - h_2\}$ for all completions, and this bound is attained by some completion.*

Proof. We recall that the duration after an extension to a node j with cost t is $d' = \max\{d + t, h + a_j\}$. Therefore, we have the four possibilities:

	$d_2 + t \geq h_2 + a_j$	$d_2 + t < h_2 + a_j$
$d_1 + t \geq h_1 + a_j$	$d'_1 - d'_2 = d_1 - d_2$	$d'_1 - d'_2 \leq d_1 - d_2$
$d_1 + t < h_1 + a_j$	$d'_1 - d'_2 \leq h_1 - h_2$	$d'_1 - d'_2 = h_1 - h_2$

We can deduce that $d'_1 - d'_2 \leq \max\{d_1 - d_2, h_1 - h_2\}$. Applying this inequality along the completion gives, by induction, $d_1^{n+1} - d_2^{n+1} \leq \max\{d_1 - d_2,\ h_1 - h_2\}$. It remains to show that the bound is attained, that is, that both $d_1 - d_2$ and $h_1 - h_2$ are realized by some completion. Suppose we extend both labels to a node j and then back to the depot such that a_j is small enough that for both extended labels we have $d + t \geq h + a_j$ then $d'_1 - d'_2 = d_1 - d_2$. Similarly, if a_j is large enough, we have for both labels $d + t < h + a_j$ and then $d'_1 - d'_2 = h_1 - h_2$. Thus each of the two values is realized by some completion, and so is their maximum. □

Theorem 3.2. *ℓ_1 dominates ℓ_2 if and only if Rules* (9)*,* (14)*, and* (15) *hold.*

Proof. $\Leftarrow$ This was established in Subsection 3.4.

$\Rightarrow$ We show that violating any single rule yields a completion on which ℓ_1 does not dominate ℓ_2. If Rule (9) fails, then $e_1 > e_2$, and a node j with $e_2 + t_{ij} \leq b_j < e_1 + t_{ij}$ admits a completion feasible for ℓ_2 but not for ℓ_1. Combining (16) with Lemma 3.1, the cost condition holds for every completion if and only if $\max\{d_1 - d_2,\ h_1 - h_2\} \leq \sigma_2 - \sigma_1$, that is, if and only if both Rule (14) ($d_1 - d_2 \leq \sigma_2 - \sigma_1$) and Rule (15) ($h_1 - h_2 \leq \sigma_2 - \sigma_1$) hold. □

Each rule is, moreover, independently active: the earliest-departure rule because e alone governs feasibility and appears in neither cost rule, and the two cost rules because the attainment in Lemma 3.1 shows that each of $d_1 - d_2$ and $h_1 - h_2$ is the binding term for some completion, so neither cost rule implies the other. It also implies that these rules cannot be reinforced by an additional rule.

3.6. Comparison with Michelini et al. (2026)

The algorithm of Michelini et al. (2026) uses the labels (i, d, h, e, σ). They do not use the same recursive formulas for d and h as Tilk and Irnich (2017). The extension rules along an arc (i, j) for e, h, and d are

$$e' = \max\{e + t_{ij},\ a_j\}, \tag{17}$$

$$h' = \max\{h,\ \theta_j - b_j\}, \tag{18}$$

$$d' = \max\{d + t_{ij},\ e' + h'\}, \tag{19}$$

with θ_j being the cumulative travel time from the depot to j. The extension is feasible if and only if $e' \leq b_j$.

In their algorithm, a label ℓ_1 dominates a label ℓ_2 if they are at the same node and

$$e_1 \leq e_2, \tag{20}$$

$$d_1 \leq d_2, \tag{21}$$

$$\max\{0, h_1 - h_2\} \leq \sigma_2 - \sigma_1. \tag{22}$$

Rule (22) is similar to Rule (15) and allows a trade-off between the dual cost and possible postponement of the departure time. Together, Rules (21) and (22) are designed to identify routes with minimum reduced cost. However, because the rules insist on component-wise dominance in d and σ, they cannot capture the trade-off between the route duration and the dual cost: a label with larger d but sufficiently smaller σ may yield a better reduced cost yet fail Rule (21). We demonstrate that integrating this trade-off into our dominance criteria leads to a strictly stronger dominance, enabling more pruning while preserving optimality.

Proposition 3.2. *If Rules* (20)–(22) *hold, then Rules* (9), (14), *and* (15) *also hold.*

Proof. First, let us express Rule (22) as

$$\sigma_1 \leq \sigma_2, \tag{23}$$

$$h_1 - h_2 \leq \sigma_2 - \sigma_1. \tag{24}$$

Rule (20) is identical to Rule (9). Recalling that Rule (21) is $d_1 \leq d_2$ and that Rule (23) is $\sigma_1 \leq \sigma_2$, adding these two inequalities yields $d_1 + \sigma_1 \leq d_2 + \sigma_2$, that is, their presence ensures that Rule (14) holds. Finally, the rules in Equations (24) and (15) are identical. □

Proposition 3.3. *There exist labels satisfying Rules* (9), (14), *and* (15), *but not Rules* (20)–(22).

Proof. The difference lies in how the two algorithms manage the compromise between duration and dual costs. Let $d_1 = 5$, $\sigma_1 = 1$, $h_1 = 3$, $d_2 = 3$, $\sigma_2 = 4$ and $h_2 = 1$, with $e_1 \leq e_2$. Rules (9), (14), and (15) are verified, and ℓ_2 can be discarded from the list of candidates. The algorithm in Michelini et al. (2026) retains both labels because Rule (21) blocks the dominance of ℓ_2 by ℓ_1, and the swapped version of Rule (23) ($\sigma_2 \leq \sigma_1$) blocks the dominance of ℓ_1 by ℓ_2. The other rules are common to both algorithms and do not influence the decision. □

These two results establish that the set of dominance rules proposed in this paper is strictly stronger than that of Michelini et al. (2026).

3.7. Comparison with Lichau (2026)

Lichau (2026) represents a partial path from the depot to node i with a label $\ell = (i, \phi, e, d, r)$, with:

- ϕ: the accumulated reduced cost;
- e: the earliest time such that the route schedule is feasible;
- d: the minimum duration from the origin depot to node i for all feasible schedules;
- r: the time buffer, i.e., the largest value such that e can be increased without increasing d.

The initial label is $(0,\ 0,\ 0,\ 0,\ b_0)$. They extend a label $\ell = (i, \phi, e, d, r)$ to a node j via arc (i, j) with travel time $t = t_{ij}$ and obtain a $\ell' = (j, \phi', e', d', r')$ using the following extension rules:

$$\begin{aligned}
\phi' &= \phi + t_{ij} - \pi_j + \max\{0, e' - e - t - r\} \\
e' &= \max\{e + t,\ a_j\} \\
r' &= \max\{0,\ \min\{r - e' + e + t,\ b_j - e'\}\} \\
d' &= d + t + \max\{0, e' - e - t - r\}.
\end{aligned}$$

Note that the extension is feasible if and only if $e' \leq b_j$.

The attributes e and d are the same in our labels and those of Lichau (2026). Additionally, one can express their remaining attributes ϕ and r as $\phi = d + \sigma$ and $r = d - e - h$, respectively. The dominance rules used by Lichau (2026) to determine whether a label ℓ_1 dominates a label ℓ_2 are:

$$\begin{aligned}
&e_1 \leq e_2, \\
&\phi_1 \leq \phi_2 - \max\{0,\ (e_2 + r_2) - (e_1 + r_1)\}, \\
&d_1 \leq d_2.
\end{aligned}$$

By substituting ϕ and r using the fomulas above, we obtain the set of rules:

$$e_1 \leq e_2, \tag{9}$$
$$d_1 + \sigma_1 \leq d_2 + \sigma_2, \tag{14}$$
$$h_1 - h_2 \leq \sigma_2 - \sigma_1, \tag{15}$$
$$d_1 \leq d_2. \tag{10}$$

Proposition 3.4. *If the dominance rules* (9)*,* (14)*,* (15)*, and* (10) *hold, then the dominance rules* (9)*,* (14)*, and* (15) *also hold.*

Proof. The proof is immediate since Rules (9), (14), and (15) are a subset of the dominance rules of Lichau (2026). □

Proposition 3.5. *There exist labels satisfying Rules* (9)*,* (14)*, and* (15)*, but not Rule* (10)*.*

Proof. Consider two labels such that $e_1 = e_2, h_1 = h_2, d_1 = 15, d_2 = 10$, and $\sigma_1 = 0, \sigma_2 = 10$. Rule (9) holds as $e_1 = e_2$. Additionally, $d_1 + \sigma_1 = 15 \leq 20 = d_2 + \sigma_2$, and therefore Rule (14) holds. Finally, Rule (15) also holds as $h_1 - h_2 = 0 \leq \sigma_2 - \sigma_1 = 10$. However, Rule (10) is violated since $d_1 = 15 > 10 = d_2$. □

These two results establish that the set of dominance rules proposed in this paper is strictly stronger than that of Lichau (2026).

3.8. Other attributes

In addition to duration-related attributes, the labeling algorithm must track resources related to vehicle capacity and maintain path elementarity. Each label includes:

- w: the cumulative load along the path;
- U: the set of visited customer nodes.

At the origin depot, $w = 0$ and $U = \emptyset$. An extension from node i to node j is feasible only if $w + q_j \leq Q$ and $j \notin U$. When feasible, the attributes are updated as follows:

$$w' = w + q_j, \tag{25}$$
$$U' = U \cup \{j\}. \tag{26}$$

The corresponding dominance rules are as follows.

$$w_1 \leq w_2, \tag{27}$$
$$U_1 \subseteq U_2. \tag{28}$$

In the pricing problem, we relax the elementary constraint, as is usual in VRP pricing algorithms, and use an ng-route relaxation (Baldacci et al., 2011). The ng-route relaxation is defined as follows: for each customer $i \in V$, a neighborhood set $N_i \subseteq V$ of cardinality $\Delta = 8$ is predefined, containing i and the $\Delta - 1$ customers closest to i in terms of travel times such that it is possible to reach j from i while respecting the time window constraints and vice versa. For a partial path $p = (0, i_1, i_2, \ldots, i_k)$ starting at the origin depot and ending at customer i_k, the ng-set $\Pi_p \subseteq N_{i_k}$ is defined as

$$\Pi_p = \left\{ i_m : m \in \{1, \ldots, k-1\},\ i_m \in \bigcap_{j=m+1}^{k} N_{i_j} \right\} \cup \{i_k\}.$$

The ng-set includes customer locations that have been visited and are still considered "nearby" during the subsequent path; these customers cannot be revisited. However, customer locations that have been visited but are no longer contained in the ng-set Π_p can be revisited, allowing for the identification of certain non-elementary routes. The dominance rule for the visited nodes is then replaced by the following rule:

$$\Pi_1 \subseteq \Pi_2. \tag{29}$$

4. Enumeration algorithm

As the column generation process terminates with an optimal solution to the linear relaxation of the master problem, the integrality of the solution cannot be guaranteed. To obtain integer solutions, we use an enumeration algorithm along the lines of Baldacci et al. (2011). Section 4.1 describes the overall algorithm, and Sections 4.2 and 4.3 detail the forward and backward labeling procedures, respectively. Section 4.4 describes the merging of forward and backward labels. Finally, Section 4.5 presents the completion bound used to prune the forward labels during enumeration.

4.1. Algorithm

First, we use column generation to compute a lower bound LB on the master problem. We also obtain the dual values π^* associated with the constraints, which will be used for the enumeration phase and the computation of the completion bound. Then, we remove the non-elementary routes from the master problem, restore integrality of the variables, and solve the resulting integer program. Thus, we obtain an upper bound UB. We also define an artificial upper bound $UB' = \epsilon_{lb} \cdot LB$. Initially, we set $\epsilon_{lb} = 1.01$. We define $\gamma = \min\{UB, UB'\} - LB$ as the gap between the upper and lower bounds. Then, we generate all the routes with a reduced cost computed according to π^* smaller than γ using a dynamic programming labeling algorithm with the labels described in Subsection 4.2. The obtained routes are added to the master problem, and we solve it again using integer variables. If the optimal solution value is lower than or equal to $\min\{UB, UB'\}$, then we have obtained the optimum for the global problem. Otherwise, we increase ϵ_{lb} (by 0.01) and repeat the process until we find the optimal solution or reach a time limit. At each iteration, the enumeration does not restart from scratch; instead, we use a warm start with the labels generated at the previous iteration.

4.2. Forward enumeration label

The labels are similar to those used during the pricing phase, except for the set of visited nodes. Here, we must keep track of the exact set of visited nodes U (and not the ng-sets) to ensure that the routes are elementary. We also need to replace the dominance rule (28) with a weaker rule:

$$U_1 = U_2. \tag{30}$$

This rule requires the set of visited customers to be equal as a prerequisite for dominance comparisons. Since demands are fixed for each customer, this rule implies $w_1 = w_2$; rule (27) therefore becomes inactive. We keep it in the label implementation for consistency with the pricing phase. With the rule (30), the dominance becomes weaker and may not prune as many labels. A usual solution is to compute a bound on the reduced cost of any extension of the label and prune it if this bound is larger than γ. In the next subsection, we define backward labels representing a path from a node to the depot and explain how we use them to define a completion bound for the forward labels.

4.3. Backward labels

A backward label represents a path from node i to the depot. Following the work of Tilk and Irnich (2017), we link the following values to a backward label:

- e^b: the latest feasible time at the node i,
- d^b: UB minus the minimum path duration from i to $n+1$,
- h^b: UB minus the earliest arrival time at $n+1$ achieving duration $UB - d^b$,

- w^b: the cumulative load along the path without the demand of i,
- σ^b: the cumulative dual cost along the path without considering π_i,
- U^b: the set of customer nodes visited along the path.

Here, UB is any upper bound on the route duration (e.g., $b_0 - a_0$).

The backward extension rules for these attributes from label ℓ^b at node j to label $\ell^{b'}$ at node i via arc (i, j) are as follows:

$$e^{b'} = \min\{e^b - t_{ij},\ b_i\}, \tag{31}$$
$$d^{b'} = \min\{d^b - t_{ij},\ h^b - t_{ij} + b_j\}, \tag{32}$$
$$h^{b'} = \min\{d^b - a_j,\ h^b\}, \tag{33}$$
$$w^{b'} = w^b + q_j, \tag{34}$$
$$\sigma^{b'} = \sigma^b - \pi_j, \tag{35}$$
$$U^{b'} = U^b \cup \{i\}. \tag{36}$$

The extension is feasible if $e^{b'} \geq a_i$ and $w^{b'} \leq Q$. The initial label is $(n+1, b_{n+1}, UB, UB - a_{n+1}, 0, 0, \emptyset)$.

A backward label ℓ_1^b at node i dominates a backward label ℓ_2^b at node i if

$$e_1^b \geq e_2^b, \tag{37}$$
$$d_1^b \geq d_2^b, \tag{38}$$
$$h_1^b \geq h_2^b, \tag{39}$$
$$w_1^b \leq w_2^b, \tag{40}$$
$$\sigma_1^b \leq \sigma_2^b, \tag{41}$$
$$U_1^b \subseteq U_2^b. \tag{42}$$

Similar to the forward label, it is possible to replace Rules (38), (39), and (41) with the following two rules:

$$\sigma_1^b - d_1^b \leq \sigma_2^b - d_2^b, \tag{43}$$
$$\sigma_1^b - h_1^b \leq \sigma_2^b - h_2^b. \tag{44}$$

The correctness of these rules is proved in Appendix A.

4.4. Merging of forward and backward labels

A forward label ℓ^f at node i can be merged with a backward label ℓ^b at node i if the following conditions are satisfied:

$$e^f \leq e^b, \tag{45}$$
$$w^f + w^b \leq Q, \tag{46}$$
$$U^f \cap U^b = \{i\}. \tag{47}$$

Following the merging formula of Tilk and Irnich (2017), the reduced cost of the complete route obtained by joining a forward and a backward label is

$$\nu = UB + \max\{d^f - d^b, h^f - h^b\} + \sigma^f + \sigma^b \tag{48}$$

$$= UB + \max\{d^f - d^b + \sigma^f + \sigma^b, h^f - h^b + \sigma^f + \sigma^b\} \tag{49}$$

$$= UB + \max\{x + \rho^d, y + \rho^h\}, \tag{50}$$

with $x = d^f + \sigma^f, y = h^f + \sigma^f, \rho^d = \sigma^b - d^b$, and $\rho^h = \sigma^b - h^b$. Note that x and y are the values that the forward dynamic algorithm tracks, whereas ρ^d and ρ^h are computed by the backward dynamic programming algorithm.

4.5. Completion bounds

We define a completion bound that allows pruning a forward label whenever it cannot be extended to a route whose reduced cost is smaller than γ. To compute this bound, we use a dynamic programming algorithm over the backward labels. For a tuple (i, e, w, U), where $e \in [a_i, b_i]$, $w \in [q_i, Q]$, and $U \in \mathcal{P}_i$, the algorithm produces a list κ of pairs (ρ^d, ρ^h) that are non-dominated in the Pareto sense. Here, $\mathcal{P}_i$ denotes the family of visited node sets of feasible open paths starting at node i and ending at the depot.

Let ℓ^f be an elementary path from 0 arriving at i at time e^f with load w^f, having visited the set of customers U^f. The bound on the reduced cost possible by extending this route is given by

$$lb = \min_{e^f \le e \le b_i, 0 \le w \le Q - w^f, U^b \in \mathcal{P}_i, U^f \cap U^b = \{i\}, (\rho^d, \rho^h) \in \kappa[i,e,w,U^b]} \max\{x + \rho^d, y + \rho^h\}$$

The label ℓ^f can be pruned if $UB + lb \ge \gamma$.

To keep the computational time and queries of the bound reasonable, we use the following relaxations. First, the backward dynamic algorithm does not consider elementary routes but ng-routes replacing $U^b \in \mathcal{P}_i$ by $ng \in N_i$. Subsequently, we do not track the load of the routes for the bounds. Finally, we replace the complete Pareto set of (ρ^d, ρ^h), for each e and ng, by a single pair containing the two extreme values for ρ^d and ρ^h.

5. Computational results

This section reports on the computational performance of the proposed approach. We first describe the acceleration techniques used in the pricing algorithm by means of a multi-phase heuristic pricing strategy. We then provide the implementation details of the dynamic programming algorithm for the pricing problem and the dynamic algorithm for the enumeration phase, followed by a description of the experimental setup and benchmark instances. We next recall the two solution methods used in our experiments and outline how each is evaluated. We then assess the impact of the proposed dominance rules on the pricing algorithm's performance, and report the performance of the complete enumeration algorithm on Solomon's VRPTW benchmark instances (Solomon, 1987), comparing our results with those obtained by Michelini et al. (2026). Finally, we show that the pricing algorithm of Michelini et al. (2026) can incorrectly discard labels when a bound on the route duration is enforced.

5.1. Multiphase heuristic pricing

Solving the pricing problem to optimality at every iteration of column generation is unnecessary; an exact solution is required only at the final iteration to ensure that no negative reduced cost route exists. We exploit this observation using a multiphase heuristic pricing strategy that progressively strengthens the dominance criteria.

In the first phase, we apply only the reduced cost Rule (14). If this phase identifies negative reduced cost routes, they are added to the RMP, and after re-optimization, pricing restarts from the first phase. If no negative reduced cost route is found, we proceed to the second phase, which incorporates the full set of dominance rules (9), (14), (15), and (27), but omits the visited dominance rule (29). Again, if negative reduced cost routes are found, they are added to the RMP, and pricing restarts in the first phase. If the second phase fails to find any improved route, we proceed to the final phase, which applies all dominance rules, including the ng-set dominance. The column generation is terminated when the final phase finds no negative reduced cost route.

5.2. Implementation details

This section provides additional details on the implementation of the dynamic programming algorithm for solving the pricing problem.

Our pricing dynamic programming algorithm for the subproblem follows the framework proposed by Feillet et al. (2004). During the early iterations of the column generation, the pricing algorithm may identify a large number of negative reduced cost routes. To limit the computational overhead, we retain at most 30 routes per pricing call during the heuristic phase and at most 150 routes during the exact phase.

For the enumeration phase, as two labels can only dominate each other if they visit the same set of customers, the labels are stored in lists defined by pairs (w, i) with w being the load of the route and i the last visited node. They are also extended by increasing the values of w.

5.3. Setup and instances

All algorithms are implemented in Julia 1.12.4, with mathematical models formulated using JuMP and linear programs solved by Gurobi 13. The experiments are executed on a single thread on an Apple M4 Pro processor with 24 GB of memory. We evaluate our algorithm on Solomon's benchmark instances with up to 100 customers. A time limit of 2 h is imposed for all experiments unless specified differently in the paper.

5.4. Overview of the solution methods

Our experiments rely on two methods: column generation and route enumeration. In this section, we briefly recall them and outline how each is evaluated in our experiments.

Column generation solves the linear relaxation of the master problem. At each iteration, the RMP is reoptimized and the pricing problem of Section 3 searches for routes of negative reduced cost through the multiphase heuristic pricing of Section 5.1; this repeats until no such route remains. Upon termination, it returns a lower bound LB together with the associated dual values. The efficiency of this phase depends on the dominance rules applied

in the labeling algorithm. We therefore evaluate the impact of the new dominance rules we propose in isolation in Section 5.5.

Since this relaxation may be fractional, an optimal integer solution is recovered by route enumeration. Given the gap between LB and an upper bound UB, the routes whose reduced cost falls within this gap are enumerated following Section 4, with the completion bounds used to prune the search. These routes are added to the master problem, which is solved as an integer program by a general-purpose MIP solver, and the gap is tightened until optimality is certified or the time limit is reached. The complete method is evaluated in Sections 5.6 and 5.7.

Our method is significantly simpler than the branch-cut-and-price of Lichau (2026), which augments a comparable pricing-and-enumeration core with generic acceleration techniques such as cutting planes, strong branching, and stabilization, implemented in the state-of-the-art solver of Sadykov and Vanderbeck (2021). Our algorithm nonetheless remains competitive, thanks to the dominance rules of Section 3.4, which we prove to be stronger than those of Lichau (2026) in Section 3.7; these accelerators could moreover be incorporated to speed it up further.

5.5. Impact of the proposed dominance rules

In Section 3.6, we proved that the dominance rules we propose are theoretically stronger and allow stricter pruning than those proposed in Michelini et al. (2026). In this section, we evaluate their practical impact. To isolate the effect of the dominance rules, a common algorithm must be used; we therefore use our column generation implementation for the comparison. The only difference between the two variants is the dominance criteria applied within the labeling algorithm; all other components (master problem, heuristic pricing, and ng-route relaxation) are identical. The time limit is 1 h.

Table 1 reports the results for three different approaches when solving the linear relaxation of the master problem. Each row corresponds to an instance class (Class) for a given number of customers (n), and #Inst. denotes the number of instances in that group. For each method, we report the shifted geometric mean of the CPU time (in seconds), the number of label extensions (#Ext.), and the number of instances for which the lower bound is obtained within the time limit. For the rules of Michelini et al. (2026), the means were computed only for the instances solved within the time limit. Two sets of results are reported for the rules proposed in this study. Namely, the column group "Partial" restricts the means to the subset of instances also solved by the rules of Michelini et al. (2026), enabling a direct comparison on a common instance set, while the column group "Full" reports results over all instances in each class.

Compared to Michelini et al. (2026), the results demonstrate substantial computational improvements for all instance classes. On the common instance set, the proposed rules consistently reduced the computation times and label extension counts by one to two orders of magnitude compared to the baseline. For example, for the C1 instances with 100 customers, the mean computation time decreases from 93 s to 0.88 s, whereas the number of label extensions decreases from 3.9 M to 157 k. The gains in solution rates are equally striking: whereas the rules of Michelini et al. (2026) leave 15 of the 87 instances unsolved within the

Table 1: Comparison of the dynamic programming algorithm using the dominance rules of Michelini et al. (2026), those of Lichau (2026), and the dynamic programming algorithm using the rules proposed in this paper.

			Michelini et al. (2026)			Lichau (2026)			Proposed (Partial)		Proposed (Full)		
Class	n	#Inst.	Time (s)	#Ext.	Solved	Time (s)	#Ext.	Solved	Time (s)	#Ext.	Time (s)	#Ext.	Solved
C1	25	9	10.5	264 651	7/9	0.84	34 077	9/9	0.08	14 310	0.61	28 495	9/9
C1	50	9	26.6	1 235 397	6/9	3.2	179 267	9/9	0.22	49 300	2.5	144 876	9/9
C1	100	9	93.0	3 918 626	4/9	23.0	1 334 182	9/9	0.88	156 993	12.1	798 996	9/9
R1	25	12	0.33	45 210	12/12	0.02	5 792	12/12	0.02	5 562	0.02	5 562	12/12
R1	50	12	7.6	486 523	12/12	0.52	69 860	12/12	0.43	60 739	0.43	60 739	12/12
R1	100	12	163	4 605 238	8/12	18.3	1 216 538	12/12	3.7	435 903	9.4	835 390	12/12
RC1	25	8	0.74	74 005	8/8	0.09	21 783	8/8	0.08	21 594	0.08	21 594	8/8
RC1	50	8	6.6	508 136	8/8	1.1	156 097	8/8	0.81	117 166	0.81	117 166	8/8
RC1	100	8	103	5 249 112	7/8	9.2	1 034 904	8/8	4.2	659 362	6.4	823 585	8/8

time limit, the proposed rules solve all. These improvements are consistent across the three instance classes, confirming that the advantage stems from the dominance criterion itself rather than from any instance-specific structure of the rule.

Now, we investigate the impact of including the non-necessary Rule (10) of Lichau (2026). The penalty it imposes is most visible on the 100-customer instances. There, using our set of dominance rules instead of the ones Lichau (2026) proposes nearly halves (factor of about 1.9) the shifted geometric mean CPU time on the R1 (9.4 s vs. 18.3 s) and C1 (12.1 s vs. 23.0 s) classes and reduces it by a factor of about 1.4 on RC1 (6.4 s vs. 9.2 s). The number of label extensions falls by roughly 30% on average, ranging from about 20% for RC1 to 40% for C1 (31% for R1). We therefore conclude that adding Rule (10) is not only unnecessary but also detrimental, it significantly degrades the performance of the pricing algorithm.

5.6. Evaluation of the complete algorithm

Table 2 reports the following values for each class of instances and instance sizes (n): the number of instances solved to optimality (n_{opt}), the shifted geometric means ($s = 1$) of the computation times to compute the lower bound $\bar{t}_{\text{lb}}$ in seconds, the shifted geometric means ($s = 10$) of the number of columns generated during the enumeration phase ($\overline{\#\text{cols}}$), the shifted geometric means ($s = 1$) of the total computational times, and the arithmetic means of the optimality gaps computed using the lower bound found at the start of the enumeration and the upper bound found at the end (i.e. $UB/LB - 1$). The instances for which the time limit was reached are excluded from all averages.

The algorithm solved 82 instances out of 87. Only one of the solved instances required more than 1 h. Moreover, 79 instances were solved in under 30 min, and 70 instances in under 5 min. A total of 66 instances were solved in under a minute. Regarding the instances not solved to optimality, upper bounds of 1.7% and 2.3% were found for the instances RC104_100 and RC108_100, respectively. For the instances R104_100, R108_100, and R112_100, the enumerations for $\epsilon_{lb} = 1.01$ do not terminate, and the initial upper bounds returned by solving the integer master program are far from the lower bound (11.1%, 14.4%, and 15%, respectively).

Table 2: Performance of the enumeration algorithm on Solomon instances.

Class	n	n_{opt}	$\bar{t}_{\text{lb}}$ (s)	$\overline{\#\text{cols}}$	$\bar{t}_{\text{tot}}$ (s)	$\overline{\text{gap}}$ (%)
C1	25	9/9	0.55	914	0.98	0.152
C1	50	9/9	2.89	7	2.96	0.006
C1	100	9/9	13.32	5	13.35	0.000
R1	25	12/12	0.01	51	0.03	0.509
R1	50	12/12	0.33	5 157	1.01	0.783
R1	100	9/12	4.87	2 191 386	173.94	0.438
RC1	25	8/8	0.08	9	0.10	1.198
RC1	50	8/8	0.81	26 435	5.52	5.635
RC1	100	6/8	2.84	2 578 777	425.24	1.798

Our algorithm is less efficient than the one proposed by Lichau (2026), as they solve all 12 instances for the R1 class, and their computational times are shorter. As we have shown, the pricing algorithm of Lichau (2026) is theoretically weaker than ours, but they use solvers developed by Sadykov and Vanderbeck (2021) and Sadykov et al. (2021), which include some mechanisms we are using, like heuristic pricing and enumeration, as well as other generic mechanisms that we do not, e.g., dual price smoothing stabilization, bucket-arc elimination techniques, limited-memory rank-1 cuts, multi-phase strong branching strategy, etc. However, these mechanisms are generic and independent of the dominance criterion, the remaining gap is therefore an implementation effect rather than a consequence of weaker pricing on our side. Ultimately, we note that despite including significantly fewer acceleration techniques, our solution approach is not being completely outperformed.

5.7. Comparison with Michelini et al. (2026)

Now, we compare our enumeration algorithm with the best method (NG-DSSR-L) reported by Michelini et al. (2026). Their computer was slower than ours; we carried out our computational experiments on a single thread on an Apple M4 Pro processor with 24 GB of memory, while theirs were on an Intel i7–3930K processor with 60 GB of memory. Based on PassMark single-thread scores (4,560 vs. 1,738), their 6 h limit corresponds to roughly 2.3 h on our hardware[1]. Since we use a 2 h time limit, our computational setting is no more favorable than theirs. We also stress that, while this translation on the computational performance is approximate, the order-of-magnitude improvements reported below are far larger than any plausible hardware factor. Table 3 reports: n_{opt} is the number of instances solved to proven optimality from the total number of instances in the group. $\bar{t}$ is the shifted geometric mean of computing times (shift $s = 1$, in seconds), computed over the same subset of instances: those that NG-DSSR-L solves optimally.

[1]PassMark single-thread scores retrieved from cpubenchmark.net on 23 April 2026.

Table 3: Comparison of our enumeration algorithm with NG-DSSR-L.

		NG-DSSR-L		This work	
Class	n	n_{opt}	$\bar{t}$ (s)	n_{opt}	$\bar{t}$ (s)
C1	25	7/9	80.86	9/9	0.45
C1	50	6/9	82.25	9/9	0.32
C1	100	5/9	435.00	9/9	1.36
R1	25	12/12	6.02	12/12	0.03
R1	50	9/12	253.35	12/12	0.44
R1	100	3/12	4 901.83	9/12	12.67
RC1	25	8/8	13.21	8/8	0.10
RC1	50	1/8	737.80	8/8	4.17
RC1	100	0/8	—	6/8	—

Our algorithm proves optimality for every instance solved by Michelini et al. (2026). In contrast, the number of instances solved by NG-DSSR-L is much lower than ours: 51 against 82. In addition, NG-DSSR-L cannot solve large instances. In fact, if we look at the computation times ($\bar{t}$) of our algorithm on the instances solved by NG-DSSR-L, we can see that the latter only solves the easiest instances. Moreover, as we show in the next subsection, the dominance rules of Michelini et al. (2026) can incorrectly discard labels that should not be pruned. Therefore, some of their results may be incorrect, or they may find the optimum in a faster way than would normally be the case without the extra pruning.

5.8. Incorrectness of the pricing algorithm of Michelini et al. (2026)

We would like to remark that the dominance rules of Michelini et al. (2026) are incorrect if a bound is imposed on the duration of a route. This is the case in their bidirectional search algorithm for the pricing problem, where they utilize a route duration limit. Therefore, their computational results may be incorrect. They use a bidirectional dynamic programming algorithm to solve pricing problems. The split resource is the duration of the routes. However, the example below shows that their extension rules combined with their dominance rules do not guarantee that feasibility is preserved with respect to an upper bound on duration denoted by μ.

Example 5.1. Consider two forward labels at the same node i, both with cumulative travel times $\theta_i = 60$.

	σ	e	d	h
ℓ_1	-40	100	60	-50
ℓ_2	-5	100	60	-80

One can verify that both labels are consistent with the extension rules: $d = \max\{\theta_i,\, e+h\} = \max\{60,\, 100+h\}$ gives 60 in both cases since $e + h_1 = 50$ and $e + h_2 = 20$.

Dominance check. We verify that ℓ_1 dominates ℓ_2 under the rules in Equations (20)–(22) as follows:

- Rule (20): $e_1 = 100 \leq 100 = e_2$.
- Rule (21): $d_1 = 60 \leq 60 = d_2$.
- Rule (22): $\max\{0,\, h_1 - h_2\} = \max\{0,\, -50 - (-80)\} = 30 \leq 35 = (-5) - (-40) = \sigma_2 - \sigma_1$.

All three conditions are satisfied; therefore, ℓ_2 is pruned as follows:

Extension to node j. Let $t_{ij} = 5$, $a_j = 200$, $b_j = 300$, and $\mu = 130$. The cumulative travel time becomes $\theta_j = \theta_i + t_{ij} = 65$. The earliest service start time is $e' = \max\{e + t_{ij},\, a_j\} = \max\{105,\, 200\} = 200$ for both labels. Applying Rules (18) and (19), we obtain

$$\begin{aligned} h'_1 &= \max\{h_1,\ \theta_j - b_j\} = \max\{-50,\ -235\} = -50, \\ d'_1 &= \max\{d_1 + t_{ij},\ e' + h'_1\} = \max\{65,\ 200 + (-50)\} = 150, \end{aligned}$$

$$\begin{aligned} h'_2 &= \max\{h_2,\ \theta_j - b_j\} = \max\{-80,\ -235\} = -80, \\ d'_2 &= \max\{d_2 + t_{ij},\ e' + h'_2\} = \max\{65,\ 200 + (-80)\} = 120. \end{aligned}$$

The feasibility check $d' \leq \mu = 130$ yields

- $\ell_1 \to j$: $d'_1 = 150 > 130$. Infeasible.
- $\ell_2 \to j$: $d'_2 = 120 \leq 130$. Feasible.

Because ℓ_2 was pruned, the feasible path through ℓ_2 to j was removed.

As they utilize a bidirectional search for the pricing using an upper bound μ on the route duration, this makes their algorithm incorrect. Their dominance rules remain valid when no upper bound on the route duration is enforced, since the feasibility condition $d' \leq \mu$ is then never active and the error demonstrated above cannot arise.

One may verify that ℓ_1 dominates ℓ_2 according to the proposed dominance rules (9), (14), and (15): $e_1 = 100 \leq 100 = e_2$, $d_1 + \sigma_1 = 20 \leq 55 = d_2 + \sigma_2$, and $h_1 - h_2 = 30 \leq 35 = \sigma_2 - \sigma_1$. However, this does not threaten the correctness of our algorithm. The incorrect label pruning illustrated above arises specifically because Michelini et al. (2026) use route duration as the critical resource in their bidirectional pricing, which requires the feasibility test $d' \leq \mu$ to be applied during label extension. Our pricing algorithm (Section 3) does not perform such a split, and our enumeration (Section 4) uses backward labels only to compute completion bounds on reduced cost; forward labels are pruned only when $UB + lb \geq \gamma$. Rules (14) and (15) are preserved after labels extension (Proposition 3.1), so the reduced cost of the dominating label remains at least as good as that of the dominated label along every extension. Hence, whenever a forward label is pruned, every completion of any label it dominates yields a reduced cost at least as large, and therefore also falls outside the gap γ.

6. Conclusion

We introduced an exact algorithm based on column generation for a variant of the VRPTW, where the objective is to minimize the total route duration called the Vehicle Routing Problem with Time Windows and Route Duration Minimization (VRPTWDM). Notably, we defined a dynamic programming labeling algorithm for the pricing problem. This algorithm is based on the work of Tilk and Irnich (2017) for the Minimum Tour Duration Problem. This pricing algorithm uses dominance rules that are theoretically stronger than the rules proposed for the VRPTWDM recently published by Michelini et al. (2026). This theoretical advantage is clearly visible as our pricing algorithm allows the computation of all the lower bounds on Solomon's instances with 100 customers. This is not possible using the dominance rules proposed in Michelini et al. (2026). We also established that our rules form a subset of the rules proposed by Lichau (2026), and as they are necessary and sufficient, the pricing algorithm of Lichau (2026) is less effective than ours in terms of dominance. This was computationally confirmed by experiments by using both sets of rules in the same implementation. Our rules can be used for other variants of the VRPTW dealing with route duration minimization. We have also proposed completion bounds for the pricing problem, allowing us to derive an enumeration algorithm to solve exactly 82 out of the 87 Solomon's instances up to 100 customers within the imposed time limit. These results could be improved by using mechanisms similar to Lichau (2026). However, these mechanisms are generic and not specifically related to the VRPTWDM.

Acknowledgements

This work was supported by project ADELE ANR-22-CE22-0017 of the French National Research Agency (ANR). This work was also partly supported by the France 2030 program «Initiative d'Excellence Lorraine (LUE)», reference ANR-15-IDEX-04-LUE.

Appendix A. Proof of the backward dominance rules (43)–(44)

This appendix provides an explicit derivation of the backward dominance rules (43) and (44), which jointly replace the component-wise rules (38), (39), and (41). The argument parallels Proposition 3.1, but uses the backward extension formulas. So, in this appendix we present a proof to demonstrate that these dominance rules can effectively replace the component-wise dominance rules while ensuring the lowest reduced costs for the labels.

For convenience, we recall that the backward extension of a label ℓ_k^b at node j through arc (i, j) to a label $\ell_k^{b'}$ at node i is given by

$$\begin{aligned} d_k^{b'} &= \min\{d_k^b - t_{ij}, h_k^b - t_{ij} + b_j\}, \\ h_k^{b'} &= \min\{d_k^b - a_j, h_k^b\}, \\ \sigma_k^{b'} &= \sigma_k^b - \pi_j. \end{aligned}$$

Proposition Appendix A.1. *Let ℓ_1^b and ℓ_2^b be backward labels at a common node j satisfying dominance rules* (37)*,* (40)*,* (42)*,* (43)*, and* (44)*. Then the same five rules are satisfied by the extended labels $\ell_1^{b'}$ and $\ell_2^{b'}$ at node i obtained through any arc (i, j).*

Proof. Because the dual update subtracts the same π_j from both labels, $\sigma_1^{b'} - \sigma_2^{b'} = \sigma_1^b - \sigma_2^b$ after extension, so preservation of Rules (43) and (44) at i amounts to

$$\sigma_1^b - \sigma_2^b \leq d_1^{b'} - d_2^{b'}, \tag{A.1}$$

$$\sigma_1^b - \sigma_2^b \leq h_1^{b'} - h_2^{b'}. \tag{A.2}$$

Next, we prove each inequality separately. Note that the Rules (37), (40), and (42) are preserved trivially, since as presented in Section 4.3, the backward extension of e^b is monotone, both labels gain the same load q_j, and the same node i.

Preservation of Rule (43)**.** Let

$$\alpha_k = d_k^b - t_{ij}, \qquad \beta_k = h_k^b - t_{ij} + b_j,$$

so that $d_k^{b'} = \min\{\alpha_k, \beta_k\}$. The additive constants $-t_{ij}$ and $b_j - t_{ij}$ cancel in the respective differences, so $\alpha_1 - \alpha_2 = d_1^b - d_2^b$ and $\beta_1 - \beta_2 = h_1^b - h_2^b$. The two inequalities (A.1) and (A.2) therefore give

$$\sigma_1^b - \sigma_2^b \leq \alpha_1 - \alpha_2, \qquad \sigma_1^b - \sigma_2^b \leq \beta_1 - \beta_2.$$

We establish $\sigma_1^b - \sigma_2^b \leq d_1^{b'} - d_2^{b'}$ by case analysis on which of α_k, β_k attains the minimum for each k.

Case 1: $\alpha_1 \leq \beta_1$ and $\alpha_2 \leq \beta_2$. $d_1^{b'} - d_2^{b'} = \alpha_1 - \alpha_2 \geq \sigma_1^b - \sigma_2^b$.

Case 2: $\alpha_1 \leq \beta_1$ and $\alpha_2 > \beta_2$. $d_1^{b'} - d_2^{b'} = \alpha_1 - \beta_2$. Since $\beta_2 < \alpha_2$, we have $\alpha_1 - \beta_2 > \alpha_1 - \alpha_2 \geq \sigma_1^b - \sigma_2^b$.

Case 3: $\alpha_1 > \beta_1$ and $\alpha_2 \leq \beta_2$. $d_1^{b'} - d_2^{b'} = \beta_1 - \alpha_2$. Since $\alpha_2 \leq \beta_2$, we have $\beta_1 - \alpha_2 \geq \beta_1 - \beta_2 \geq \sigma_1^b - \sigma_2^b$.

Case 4: $\alpha_1 > \beta_1$ and $\alpha_2 > \beta_2$. $d_1^{b'} - d_2^{b'} = \beta_1 - \beta_2 \geq \sigma_1^b - \sigma_2^b$.

In every case, $\sigma_1^b - \sigma_2^b \leq d_1^{b'} - d_2^{b'}$, which is equivalent to $\sigma_1^{b'} - d_1^{b'} \leq \sigma_2^{b'} - d_2^{b'}$. Rule (43) is therefore preserved by the extension.

Preservation of Rule (44)**.** Let

$$\gamma_k = d_k^b - a_j, \qquad \delta_k = h_k^b,$$

so that $h_k^{b'} = \min\{\gamma_k, \delta_k\}$. As before, the constant $-a_j$ cancels in $\gamma_1 - \gamma_2$ and $\delta_k = h_k^b$ directly, so Rules (43) and (44) give

$$\sigma_1^b - \sigma_2^b \leq \gamma_1 - \gamma_2, \qquad \sigma_1^b - \sigma_2^b \leq \delta_1 - \delta_2.$$

The four-case analysis of the preceding paragraph, with (α, β) replaced throughout by (γ, δ), applies identically and establishes $\sigma_1^b - \sigma_2^b \leq h_1^{b'} - h_2^{b'}$, i.e. $\sigma_1^{b'} - h_1^{b'} \leq \sigma_2^{b'} - h_2^{b'}$. Rule (44) is therefore preserved by the extension.

Because these rules also hold when the backward label is merged with a forward label via (48), the reduced cost produced by the dominating label is no larger than the reduced cost produced by the dominated label, so pruning does not sacrifice the routes with the minimum reduced cost. □

References

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

Dabia, S., Ropke, S., van Woensel, T., and de Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, 47(3):380–396.

Desaulniers, G., Madsen, O. B. G., and Ropke, S. (2014). The vehicle routing problem with time windows. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications*, volume 18 of *MOS-SIAM Series on Optimization*, pages 119–159. SIAM, Philadelphia, PA.

Favaretto, D., Moretti, E., and Pellegrini, P. (2006). An ant colony system approach for variants of the traveling salesman problem with time windows. *Journal of Information and Optimization Sciences*, 27(1):35–54.

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

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, 43(1):17–26.

Kara, I., Koc, O. N., Altiparmak, F., and Dengiz, B. (2013). New integer linear programming formulation for the traveling salesman problem with time windows: Minimizing tour duration with waiting times. *Optimization*, 62(10):1309–1319.

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

Lichau, S. (2026). *Solving vehicle routing problems with drones in urban logistics: an approach based on exact and approximate optimization methods*. PhD thesis, Université de Bordeaux.

Michelini, S., Kucukaydin, H., and Arda, Y. (2026). A comparative study of branch-and-price algorithms for vehicle routing with time windows and waiting time costs. *International Transactions in Operational Research*.

Rijal, A., Bijvank, M., and De Koster, R. (2023). Dynamics between warehouse operations and vehicle routing. *Production and Operations Management*, 32(11):3575–3593.

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

Sadykov, R. and Vanderbeck, F. (2021). BaPCod - a generic branch-and-price code. Technical report, INRIA Bordeaux Sud-Ouest.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154.

Schmidt, J. (2024). The Vehicle Routing Problem with Drones and Time Windows: Minimizing Route Duration. Technical Report LM-2024-02, Johannes Gutenberg University Mainz, Mainz, Germany.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–269.

Tilk, C. and Irnich, S. (2017). Dynamic Programming for the Minimum Tour Duration Problem. *Transportation Science*, 51(2):549–565.