A Branch-Cut-and-Price algorithm for the vehicle routing problem with time windows and duration constraints

Sylvain Lichau, Rémy Dupas, Julien François, Artur Pessoa, Marcos Roboredo, Eduardo Uchoa

A Branch-Cut-and-Price algorithm for the vehicle routing problem with time windows and duration constraints

Sylvain Lichau,[a,b,*], Rémy Dupas[a], Julien François[a], Artur Pessoa[c], Marcos Roboredo[c], Eduardo Uchoa[c,d]

[a] *Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, Talence, F-33400, France*
[b] *Univ. Bordeaux, CNRS, INRIA, Bordeaux INP, IMB, UMR 5251, Talence, F-33400, France*
[c] *Departamento de Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Patria 156, Niteroi, 24210-240, Rio de Janeiro, Brazil*
[d] *INRIA International Chair (2022-2026), INRIA Bordeaux Sud-Ouest, Talence, F-33400, France*

Abstract

This article studies a variant of the Vehicle Routing Problem with Time Windows (VRPTW), which integrates two operationally relevant features: flexible departure times from the depot and strict limits on route duration. These features reflect frequent real-world requirements such as driver working hour regulations. The resulting problem, termed VRPTW with Duration Constraints (VRPTWDC), introduces a scheduling dimension that makes the routing computationally challenging and largely unexplored in the literature. We develop an exact branch-cut-and-price algorithm, featuring a bi-bidirectional labeling algorithm as its primary component for solving the complex pricing subproblem. Indeed, as the only previously proposed labeling algorithm for the VRPTWDC has a mistake, we provide rigorous proofs of its correctness. We also show how the related problem of minimizing total route duration (MD-VRPTW) can be treated. Computational experiments on benchmark instances demonstrate the effectiveness of the approach.

Keywords: Vehicle routing problem with time windows, duration constraints, branch-cut-and-price, labeling algorithm

1. Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) is a fundamental model for systems that handle deliveries or provide field services. In its standard form, a homogeneous fleet of capacitated vehicles is dispatched from a central depot to serve a set of customers, each with a predefined time window. Vehicles are allowed to arrive early and wait for a customer's time window to open, but arriving after the window closes is strictly prohibited. The objective is to minimize the total travel time — although in some variants, minimizing the number of vehicles used is also considered. The depot also has a time window: it opens at time zero (by convention) and closes at a specified time. The standard VRPTW assumes that vehicles can depart at time zero and must return before the depot closes. As a result, routes with very long durations, up to 16 or even 24 hours in some classic benchmark instances, are allowed. However, such long routes are often unrealistic due to operational and regulatory constraints. Indeed, driver working hour regulations frequently impose strict limits on route duration. Other situations that require duration constraints include the need to preserve perishable goods in cold chain logistics or passenger comfort considerations in school bus services. Additionally, emerging drone-based services may also impose route duration limits, especially when drones are prohibited from landing en route for security or environmental reasons.

The VRPTW with Duration Constraints (VRPTWDC) addresses those critical issues. In its main variant, the duration of a route is measured from the moment a vehicle leaves the depot until it returns, accounting for travel times, service times, and any waiting times at customer locations. We also consider the close variant where the duration of a route is measured from the departing time until the last customer is served. In both cases, the duration constraints make the new problem significantly more complex: in VRPTW, all routes can be assumed to depart at time zero; *in VRPTWDC, the*

*Corresponding author
Email addresses: sylvain.lichau@math.u-bordeaux.fr (Sylvain Lichau,), remy.dupas@u-bordeaux.fr (Rémy Dupas), julien.francois@u-bordeaux.fr (Julien François), arturpessoa@id.uff.br (Artur Pessoa), mcroboredo@id.uff.br (Marcos Roboredo), eduardo_uchoa@id.uff.br (Eduardo Uchoa)

route departure time becomes a critical decision. In particular, devising an efficient Branch-Cut-and-Price (BCP) for VRPTWDC is a challenge. In these algorithms, the pricing subproblem is typically solved using a labeling dynamic programming algorithm. In VRPTW, early arrivals can be accommodated without incurring any costs through waiting. Therefore, when comparing partial routes from a time perspective, the earlier one is better. This leads to effective dominance in the labeling. However, in VRPTWDC, waiting times interfere with duration, making dominance much more complicated. *The main contribution of this paper is a sophisticated labeling algorithm for pricing VRPTWDC routes, correcting a mistake in a previous attempt by Bettinelli et al. (2011)*.

To assess the effectiveness of the proposed BCP algorithm, we conduct an extensive computational study based on benchmark instances from (Solomon 1987), a well-established dataset for VRPTW. These instances provide a diverse set of problem characteristics by varying the spatial distribution of customers (clustered, random, or mixed) and the tightness of the time windows. We adapt these instances by considering subsets of 50, 75, and 100 customers and by introducing route duration constraints of 6, 8, and 10 hours, which reflect practical limitations often encountered in real-world logistics operations. This experimental setup enables a thorough evaluation of algorithm performance across a range of instance sizes and constraint settings.

The remainder of this paper is organized as follows. Section 2 reviews related work on VRPs with temporal constraints, route duration limits, and flexible departure time. Section 3 formally defines the problem and presents an arc-based formulation. Section 4 presents the set-partitioning formulation. Section 5 describes the branch-cut-and-price algorithm, including column generation, branching, and cutting strategies. Section 6 presents the labeling algorithm used to solve the pricing problem. Section 7 presents computational results on benchmark instances. Section 8 concludes and gives directions for future research.

2. Literature review

Routing with time windows has been studied since the 1970s, but gained prominence after Solomon (1987), which provided the standard VRPTW definition, proposed benchmark instances, and experimented with several heuristic algorithms. Column generation-based methods — Branch-and-Price and Branch-Cut-and-Price (BCP) algorithms — have long been the best exact approaches for VRPTW (Desrochers et al. 1992, Kohl et al. 1999, Kallehauge et al. 2006, Desaulniers et al. 2008, Jepsen et al. 2008, Baldacci et al. 2011a, Pecin et al. 2017a, Sadykov et al. 2021). All classic Solomon instances with up to 100 customers have been solved in Røpke (2012). All Homberger and Gehring instances with 200 customers were solved in Silva et al. (2024). The modern BCP algorithms for VRPTW incorporate advances such as: partial route elementarity by *ng*-paths (Baldacci et al. 2011a), path enumeration (Baldacci et al. 2008a, Contardo and Martinelli 2014), rounded capacity cut separation (Laporte and Nobert 1983a, Lysgaard et al. 2004), rank-1 cuts with limited memory (Jepsen et al. 2008, Pecin et al. 2017b,d), and hierarchical strong branching (Røpke 2012, Pecin et al. 2017b). Moreover, they include highly performing variants of the bi-directional labeling algorithm (Righini and Salani 2006) used for solving the pricing subproblems, the bottleneck of the BCP, and also for fixing by reduced costs.

Savelsbergh (1992) is the first VRPTW work to consider route duration, not as a constraint but in the objective function, which becomes minimizing the sum of durations. That work focused on how local search techniques, specifically edge-exchange methods, can be adapted to handle the complexities introduced by having to recalculate the latest feasible route departure time quickly. The only exact algorithm for that variant appears in Michelini (2018) that addresses what they so-called VRPTW with Waiting Time Costs (VRPTWWTC). However, as they assume that the waiting time costs are identical to the traveling costs, they are actually minimizing the sum of durations. They propose labeling algorithms within a branch-and-price algorithm and solve a few Solomon instances with 100 customers within a 6-hour computation limit. The algorithm for the VRPTWDC proposed in this work can be easily adapted to minimize the sum of durations, as shown in Section 6.4.

We now list other works in the routing literature that address some duration constraints:

- The classic Dial-a-Ride Problem (Cordeau and Laporte 2007) is a pickup and delivery problem with time windows, with the additional constraint that the ride durations (the difference between the delivery and pickup times for each customer) are limited.
- Azi et al. (2010) introduce the Multi-Trip Vehicle Routing Problem with Time Windows and Limited-Duration trips (MTVRPTW-LD). In this problem, vehicles can operate multiple trips from the depot; however, each trip has a duration limit. The authors develop an exact branch-and-price algorithm, able to solve instances with up to 50 customers. Hernandez et al. (2014) propose a different exact algorithm based on column generation for the MTVRPTW-LD, where the subproblem prices trips rather than complete routes.

- A route duration limit is also considered in long-haul transportation due to the hours-of-service regulation. Drivers' working hours in the European Union are regulated with a daily rest period, a maximum driving time per week and per day, and a maximum driving period between breaks. Goel (2009) proposes a large neighborhood search algorithm, utilizing a labeling algorithm to verify the feasibility of routes in solving long-haul transportation VRPTW problems in accordance with European legislation. Goel and Irnich (2017) propose an exact branch-and-price algorithm for the Vehicle Routing and Truck Driver Scheduling Problem (VRTDSP). Their approach is based on a complex labeling algorithm with efficient dominance rules. They solve instances with up to 50 customers. Their work is extended with a bidirectional labeling algorithm in (Tilk and Goel 2020).
- Drivers' schedules can be handled as a post-processing of a VRPTW, as proposed in (Kok et al. 2011), which formulates the Vehicle Departure time Optimization (VDO) for a time-dependent travel time variant of the VRPTW. The optimal route for the VRPTW might be infeasible for the VDO and needs to be adapted. Their case study suggests that a 15% reduction in route duration could be achieved.
- Faiz et al. (2019) address the open vehicle pickup and delivery problem with time windows and vehicle rent period restrictions in the context of humanitarian logistics. The problem includes both route duration and flexible departure time. They propose a Mixed Integer Program (MIP) and a column generation algorithm, solving the pricing subproblem with a MIP solver. They solve instances with up to 50 tasks.

In the next section, we formally define the VRPTWDC.

3. Arc-based formulation

The VRPTW with Duration Constraints (VRPTWDC) is defined by a set of n customers $C = \{1, 2, \ldots, n\}$, m identical vehicles, and a depot, denoted by 0. Let $V = \{0\} \cup C$ be the set of all locations. We define the directed graph $G = (V, A)$ having V as its node-set and $A = \{(i, j) : i, j \in V, i \neq j\}$ as its arc-set. The travel time between a pair of locations $a = (i, j) \in A$ is denoted by c_{ij} or by c_a. All vehicles have the same capacity Q and route duration limit D. Each customer $i \in C$ has a demand d_i, a service time s_i, and a time window $[l_i,\ u_i]$. We extend the notation to the depot with $d_0 = 0$, $s_0 = 0$ and $[l_0 = 0,\ u_0]$, where l_0 is the earliest possible departure time and u_0 is the depot closing time. We also define an arc time $t_a = t_{ij} = c_{ij} + s_i$, $a = (i, j) \in A$, incorporating travel and service time. It is assumed that the triangular inequalities over arc times hold. So, it can also be assumed that $l_i \geq t_{0i}$ and $u_i \leq u_0 - t_{i0}$ for all $i \in C$. These inequations verify that a customer's time window must be coherent with the feasible delivery dates. The objective is to find a set of up to m routes starting and ending at the depot, minimizing the total travel cost and satisfying the following constraints:

- **Visiting Constraints:** Each customer must be visited by exactly one route.
- **Time Window Constraints:** Service at a customer $i \in C$ can only start within its time window $[l_i,\ u_i]$. Arriving at i before l_i and waiting is allowed. The route should have returned to the depot by time u_0.
- **Route Duration Constraints:** The total duration of each route must not exceed D. Route duration is counted from the actual departure time from the depot, which is also a decision variable.
- **Capacity Constraints:** The total load of each route must not exceed Q.

We now present an arc-based formulation for the problem. The arc-set A can be preprocessed to only include an arc $a = (i, j)$ if it can belong to a feasible route, in other words, if:

- j can be reached within its time window:

$$\max\{t_{0i},\ l_i\} + t_{ij} \leq u_j \tag{1}$$

- The route visiting only customers i and j in that order has a feasible duration:

$$t_{0i} + t_{ij} + \max\{0,\ l_j - \max\{t_{0i},\ l_i\} - t_{ij}\} + t_{j0} \leq D \tag{2}$$

  Note that the third term is the waiting time at j.

Define $\delta^+(i)$ as the set of outgoing edges from node $i \in V$ and $\delta^-(i)$ the set of incoming edges to node $i \in V$.

Variables:

- $x_{ij} = x_a$ — a binary variable indicating if $a = (i, j) \in A$ is traversed.
- $r_{ij} = r_a$ — a continuous variable indicating the time at which the vehicle starts to traverse $a = (i, j) \in A$.
- $r'_{ij} = r'_a$ — a continuous variable indicating the duration of the route when the vehicle starts to traverse $a = (i, j) \in A$.
- $q_{ij} = q_a$ — an integer variable indicating the total demand served in the route when the vehicle starts to traverse $a = (i, j) \in A$.

Formulation:

$$\text{Min} \sum_{a \in A} c_a x_a \tag{3a}$$

$$\text{S.t.} \sum_{a \in \delta^-(j)} x_a = 1 \qquad \forall j \in C, \tag{3b}$$

$$\sum_{a \in \delta^+(j)} x_a = 1 \qquad \forall j \in C, \tag{3c}$$

$$\sum_{a \in \delta^+(i)} r'_a - \sum_{a \in \delta^-(i)} r'_a = \sum_{a \in \delta^+(i)} r_a - \sum_{a \in \delta^-(i)} r_a \qquad \forall i \in C, \tag{3d}$$

$$\sum_{a \in \delta^+(i)} r_a \geq \sum_{a \in \delta^-(i)} (r_a + t_a x_a) \qquad \forall i \in C, \tag{3e}$$

$$r_{ij} \geq l_i x_{ij} \qquad \forall (i,j) \in A, \tag{3f}$$

$$r_{ij} \leq u_i x_{ij} \qquad \forall (i,j) \in A, \tag{3g}$$

$$r'_{ij} \leq (D - t_{ij}) x_{ij} \qquad \forall (i,j) \in A, \tag{3h}$$

$$d_i x_{ij} \leq q_{ij} \leq (Q - d_j) x_{ij} \qquad \forall (i,j) \in A, \tag{3i}$$

$$\sum_{a \in \delta^+(i)} q_a = \sum_{a \in \delta^-(i)} (q_a + d_i x_a) \qquad \forall i \in C, \tag{3j}$$

The objective function (3a) minimizes the total travel time. Equations (3b) and (3c) are the flow constraints. Equations (3d) define the route duration variables. Equations (3e) and Equations (3f) ensure that variables r_{ij} correspond to the arrival time at j if arc (i, j) is used and 0 otherwise. Equations (3g) verifies that the arrival time is feasible. Equations (3h) are the duration limit constraints. Equations (3i) and equations (3j) are the capacity constraints. We used this formulation only on small instances to verify the results of our branch-cut-and-price algorithm.

4. Route-based formulation

We now propose a route-based mathematical formulation. Let $\mathcal{R}$ be the set of all routes that respect time windows and duration constraints. Note that routes with a violated capacity constraint can belong to the set $\mathcal{R}$. For a route $r \in \mathcal{R}$, coefficient b^r_a indicates the number of times arc $a \in A$ is used in r. Variables λ_r indicate whether $r \in \mathcal{R}$ is used in the solution. Additional binary arc variables x_a indicate whether some route uses arc $a \in A$. For $S \subseteq C$, let $\delta^-(S) \subset A$ be the set of arcs entering S. The formulation follows:

$$\min \sum_{a \in A} c_a x_a \tag{4a}$$

$$\text{s.t.} \sum_{a \in \delta^-(i)} x_a = 1 \qquad \forall i \in C, \tag{4b}$$

$$\sum_{a \in \delta^-(S)} x_a \geq \left\lceil \frac{\sum_{i \in S} q_i}{Q} \right\rceil \qquad \forall S \subseteq C, \tag{4c}$$

$$\sum_{a \in \delta^-(0)} x_a \leq m, \tag{4d}$$

$$x_a = \sum_{r \in \mathcal{R}} b^r_a \lambda_r \qquad \forall a \in A, \tag{4e}$$

$$\lambda_r \geq 0 \qquad \forall r \in \mathcal{R}, \tag{4f}$$

$$x_a \in \{0, 1\} \qquad \forall a \in A. \tag{4g}$$

The objective function (4a) minimizes the total traveling time. Constraints (4b) ensure that each customer is visited exactly once. Constraints (4c) are the rounded capacity cuts (Laporte and Nobert 1983b), also used to enforce route capacity. As the number of sets S is exponential, these constraints are added to the formulation dynamically. Note that it would be possible to enforce capacity in the route definition, but this is less efficient in practice. Constraints

(4d) limit the number of routes (i.e. number of vehicles). Constraints (4e) define the mapping between arc and route variables, forcing the solution to be a combination of routes in $\mathcal{R}$, and therefore satisfying time windows and duration. Note that only the arc variables need to be defined as binary. This means that it suffices to branch on arc variables to obtain integer solutions.
As the number of routes is exponential over the number of customers, we use a branch-and-price algorithm to solve this formulation.

5. Branch-cut-and-price algorithm

The set $\mathcal{R}$ has an exponential size over the number of customers. It would be inefficient to enumerate it, so we solve this formulation using a column and cut generation approach. We use a similar branch-and-cut-and-price (BCP) as in (Pessoa et al. 2020). The BCP algorithm iteratively generates new variables to improve the linear programming (LP) relaxation of the formulation, keeping the problem size tractable while improving the relaxation bound. Based on the LP solution, branching decisions are made, and cutting planes are added to strengthen the relaxation. Two primal heuristics are used within the BCP to find good upper bounds to prune the branch-and-bound tree. The first one solves the restricted master problem using a MIP solver in a limited time. The second one is called the diving heuristic, where columns with the most significant fractional values are iteratively set to one, and the remaining problem is solved using column generation.

5.1. Column generation

We use a three-stage column generation approach. The first two stages solve the pricing problem using heuristic labeling algorithms, following the methodology of Sadykov et al. (2021), while the third stage employs the exact labeling dynamic programming algorithm described in Pessoa et al. (2020) and Sadykov et al. (2021). The heuristic stages are limited to generating a maximum of 30 columns each, whereas the exact stage can generate up to 150 columns.
To enhance convergence, we apply automatic dual price smoothing stabilization as proposed by Pessoa et al. (2018). Additionally, we use the bucket-arc elimination technique to discard arcs that are guaranteed not to belong to any optimal solution.
We also apply a route enumeration procedure (Baldacci et al. 2008b) that attempts to enumerate all routes that may lead to improving solutions. If the enumeration is successful, subsequent pricing problems are solved by inspecting the set of enumerated routes. Moreover, if the total number of enumerated routes across all subproblems remains below a predefined threshold (set to 5000), all routes are added to the restricted master problem, which is then solved using a MIP solver.

5.2. Cutting

Besides rounded capacity cuts, the BCP separates limited-memory rank-1 cuts with up to five rows, using the multipliers defined in (Pecin et al. 2017d).

5.3. Branching

We consider a collection of branching candidates drawn from the fractional x variables and also for the aggregated variable $z = \sum_{a \in \delta^-(0)} x_a$ corresponding to the number of routes (if it is fractional). To identify the most promising branching candidate, we employ a multi-phase strong branching strategy inspired by Pecin et al. (2017b).

5.4. Iterative Cutoff Scheme

Even with the heuristics, the algorithm struggles to find good primal bounds quickly, which drastically decreases the performance of the algorithm. To mitigate this issue, we use a scheme that iteratively tries to guess primal bounds and uses them as cutoffs for pruning in the BCP algorithm.
The first such cutoff is the dual bound of the root node times 1.01. If the BCP finds a feasible solution using that cutoff, it should be the optimal one. Otherwise, if no solution is found (BCP returns infeasibility), the cutoff is a valid dual bound. In that case, a new cutoff, which is 1.01 larger, is tried. This is repeated until the optimal solution is found. The scheme is efficient because the BCP runs with wrong cutoffs (those that return infeasibility) are usually much faster than the last run, the one that finds the correct optimal solution.

6. The Labeling Algorithm for the Pricing Subproblem

The pricing subproblem, the most critical part of the BCP algorithm, is formulated as a resource-constrained shortest path problem (RCSP). The objective of the RCSP is to determine a shortest path between a source and a sink in a graph where each arc consumes resources and requires that the remaining resources lie between lower and upper limits. Our RCSP is modeled on the graph G, the depot 0 is both the source and the sink. Given a dual solution to the current restricted master problem, the pricing problem seeks to find a route that has the smallest reduced cost and respects the time windows, and route duration constraints. The reduced cost of an arc $a = (i, j)$ with a cost c_{ij} is defined by:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \sum\limits_{\substack{S \subset C \\ |\{i,j\} \cap S| = 1}} \theta_S - \pi_i & i \in C \\ c_{ij} - \sum\limits_{\substack{S \subset C \\ |\{i,j\} \cap S| = 1}} \theta_S - \mu & i = 0 \end{cases} \tag{5}$$

where π_i are the dual variables associated with constraints (4b), μ is the dual variable associated with constraint (4d), θ_S are the dual variables associated with constraint 4c.

The RCSP is solved by a labeling algorithm. The key challenge is ensuring that the route does not violate the route duration constraint or the time window constraints. We realized that the only previous labeling algorithm (Bettinelli et al. 2011) for the problem contains a small error that may lead to missing the optimal solution. This is why we provide detailed formal proofs of the validity of the new corrected labeling algorithm.

Both forward and backward labeling algorithms are defined; together, they define a bi-directional labeling algorithm. We remark that even if a mono-directional algorithm can be used for pricing, the fixing of arcs by reduced costs in Sadykov et al. (2021) requires bi-directional labeling. Note that the actual algorithms used in our BCP also include in their labels information for handling the ng-path relaxation (Baldacci et al. 2011b) and the limited memory rank-1 cuts (Pecin et al. 2017c). However, as this is already standard, this part can be omitted.

6.1. Labeling algorithm

We now present the label definition, the extensions and concatenation functions, and the domination criterion.

6.1.1 Forward Label Definition and Forward Extension

Let $\vec{P} = (v_0 = 0, v_1, \ldots, v_{p-1}, v_p = i)$ be a partial forward path starting at the depot 0 and ending at some node $i \in V$. The intermediate nodes $v_1, \ldots, v_{p-1}$ are necessarily customers. A *schedule* of a partial forward path $\vec{P}$ is a tuple $T = (T_0, \ldots, T_p)$ with values having the following meaning: T_0 is the departure time at the depot; if v_k is a customer node, T_k represents the start of service in v_k; if $v_p = i = 0$ (so $\vec{P}$ is already a complete path), then T_p is the arrival time at the depot. As we do not need to consider schedules with unnecessary wait, it can be assumed that $T_k = \min\{T_{k+1} - t_{v_k v_{k+1}},\ u_{v_k}\}$, $0 \le k \le p-1$, so *T is uniquely determined by T_p*. Let $T(t)$ denote the schedule with $T_p = t$ and $D(t) = T_p - T_0$ be its duration.

Property 1. *If $t < t'$, $T = T(t)$, and $T' = T(t')$, then $0 \le T'_k - T_k \le T'_{k+1} - T_{k+1} \le t' - t$, $0 \le k \le p-1$. (As a consequence, $D(t) \le D(t')$).*

Proof. Proof. First, $0 < T'_p - T_p = t' - t$. Now, consider the decreasing sequence of indices k from $p-1$ to 0. If $T'_k = T'_{k+1} - t_{v_k v_{k+1}}$ then $T_k = T_{k+1} - t_{v_k v_{k+1}}$, so $T'_k - T_k = T'_{k+1} - T_{k+1}$. Otherwise, $T'_k = u_{v_k}$ and $T_k = \min\{T_{k+1} - t_{v_k v_{k+1}},\ u_{v_k}\}$, so $0 \le T'_k - T_k \le T'_{k+1} - T_{k+1}$.

□

A *schedule T is feasible* if the time windows and the route duration constraint are satisfied, i.e., $l_{v_k} \le T_k \le u_{v_k}$, $0 \le k \le p$, and $T_p - T_0 \le D$.

Property 2. *Let $t < t' < t''$. If $T = T(t)$ and $T'' = T(t'')$ are feasible then $T' = T(t')$ is feasible.*

Proof. Proof. By Property 1, $l_{v_k} \le T_k \le T'_k \le T''_k \le u_{v_k}$, $0 \le k \le p$, so time windows are satisfied. Moreover, $D(t') \le D(t'') \le D$.

□

A *path* $\vec{P}$ *is feasible* if there is at least one feasible schedule for it. In general, for a given feasible $\vec{P}$, there is an interval of times t such that $T(t)$ is feasible. Labels represent relevant information about those feasible schedules. The label $\vec{L}_{\vec{P}} = (i, \bar{c}, ET, RD, TB)$ associated with the feasible path $\vec{P}$ is defined by the following attributes:

i - the last node v_p.
$\bar{c}$ - the accumulated reduced cost.
ET - the Earliest t such that $T(t)$ is feasible.
RD - the minimum Route Duration for all feasible schedules, i.e., $D(ET)$.
TB - the Time Buffer, the largest value such that the $D(ET + TB) = RD$.

We prove some additional properties about the schedules for a path $\vec{P}$.

Property 3. *If $t > ET + TB$ then $D(t) = RD + t - (ET + TB)$*

Proof. Proof. Consider the schedule $T' = T(ET + TB)$. There should be a b, $0 \leq b \leq p$ such that $T'_b = u_{v_b}$; otherwise, it would be possible to increase all the times in T' by a positive amount to produce another schedule T'' such that $T''_p - T''_0 = RD$, a contradiction with the definition of TB. The existence of a $T'_b = u_{v_b}$ acts as a block. Let $T = T(t)$. Since $T_b = T'_b$, $T_0 = T'_0$. So, $D(t) = T_p - T'_p + T'_p - T'_0 = RD + t - (ET + TB)$.

□

Property 4. *If $TB > 0$, $RD = \sum_{k=0}^{p-1} t_{v_k v_{k+1}}$. In other words, if $TB > 0$, all schedules $T(t)$, $ET \leq t \leq ET + TB$, are no-waiting.*

Proof. Proof. Consider a schedule $T = T(t)$, $ET \leq t \leq ET + TB$, that is not no-waiting, i.e., $RD = D(t) > \sum_{k=0}^{p-1} t_{v_k v_{k+1}}$. So, there exists a b, $0 \leq b \leq p-1$, such that $T_b = u_{v_b} < T_{b+1} - t_{v_b v_{b+1}}$. Consider the schedule $T' = T(ET + TB)$. Since $T_b = T'_b$, $T_0 = T'_0$. As $D(t) = D(ET + TB)$, then $t = ET + TB$. Consider the schedule $T'' = T(t'')$, where t'' is in the interval $[t - (T_{b+1} - t_{v_b v_{b+1}} - u_{v_b}), t[$, i.e., $t - t''$ is small enough to keep $T''_b = T_b = u_{v_b}$, so that $T_0 = T''_0$. In this case, $D(t'') < RD$ and T'' must be infeasible. Therefore, $t = ET$ and $TB = 0$.

□

The forward labeling algorithm is initialized with the empty forward path $\vec{P} = (0)$ associated with the forward label $(0, 0, 0, 0, u_0)$. The forward extension procedure is presented as Algorithm 1.

Algorithm 1 extendForward($(i, \bar{c}, ET, RD, TB)$, $a = (i, j)$)

```
1:  ET^0 ← ET + t_ij
2:  if ET^0 > u_j then
3:      return infeasible
4:  ET' ← max{ET^0, l_j}
5:  w_j ← ET' − ET^0
6:  TB' ← max{0, min{TB − w_j, u_j − ET'}}
7:  RD' ← RD + t_ij + max{0, w_j − TB}
8:  if RD' > D then                          ▷ Can be improved by using RD' + t_j0 > D
9:      return infeasible
10: c̄' ← c̄ + c̄_ij
11: return (j, c̄', ET', RD', TB')
```

Proposition 1. *The extendForward procedure is correct.*

Proof. Proof. We have to prove that if $L = (i, \bar{c}, ET, RD, TB)$ is the correct label for feasible path $\vec{P} = (v_0 = 0, v_1, \ldots, v_{p-1}, v_p = i)$ then Algorithm 1 either detects that path $\vec{P'} = (v_0 = 0, v_1, \ldots, v_{p-1}, v_p = i, v_{p+1} = j)$ is infeasible or returns the correct label $L' = (j, \bar{c}', ET', RD', TB')$ for $\vec{P'}$. As ET is the earliest arrival time at i, then the time ET' is the correct earliest arrival at j, which is not feasible if larger than u_j. Assuming $ET' \leq u_j$, and noting that $\bar{c}' = \bar{c} + \bar{c}_{ij}$ is trivially correct, it remains to prove that the values of RD' and TB' are also correct. We divide this into two cases, depending on w_j:

1. $w_j \leq TB$. In this case, one can check that Algorithm 2 sets $ET' = \max\{ET + t_{ij},\ l_j\}$, and $ET' + TB' = \min\{ET + TB + t_{ij},\ u_j\}$. Then, for any $ET' \leq t \leq ET' + TB'$, the schedule $T'(t)$ can be obtained by extending the schedule $T(t - t_{ij}) = (T_0, \ldots, T_p)$ for $\vec{P}$ to the schedule $(T_0, \ldots, T_p, T_{p+1} = T_p + t_{ij})$ for $\vec{P'}$, having duration $D'(t) = RD + t_{ij} = RD'$ because $ET \leq t - t_{ij} \leq ET + TB$. Moreover, if $ET' + TB' < t \leq u_j$, $D'(t) = D(t - t_{ij}) + t_{ij} > RD'$ because $t - t_{ij} > ET + TB$.
2. $w_j > TB$. In this case, $ET' = l_j > ET + TB + t_{ij}$, and $TB' = 0$. Then, the schedule $T'(ET')$ can be obtained by extending the schedule $T(ET + TB) = (T_0, \ldots, T_p)$ for $\vec{P}$ to the schedule $(T_0, \ldots, T_p, T_{p+1} = l_j = T_p + t_{ij} + w_j - TB)$ for $\vec{P'}$, having duration $D'(t) = RD + t_{ij} + w_j - TB = RD'$. Moreover, since T_0 cannot be increased in $T(ET + TB)$, $D'(t) > RD'$ for any $ET' < t \leq u_j$.

□

6.1.2 Backward Label Definition and Backward Extension

Let $\overleftarrow{P} = (v_p = i, v_{p-1}, \ldots, v_1, v_0 = 0)$ be a backward partial path starting at a node $i \in V$ and ending at the depot 0. The intermediate nodes $v_{p-1}, \ldots, v_1$ are necessarily customers. A *schedule* of a backward partial path $\overleftarrow{P}$ is a tuple $T = (T_p, \ldots, T_0)$ with values having the following meaning: T_0 is the arrival time at 0; if v_k is a customer node, T_k represents the start of service in v_k; if $v_p = 0$ (so $\overleftarrow{P}$ is already a complete path), then T_p is the departure time from the depot. As we do not need to consider schedules with unnecessary wait, it can be assumed that $T_k = \max\{T_{k+1} + t_{v_{k+1}v_k},\ l_{v_k}\}$, $1 \leq k \leq p$, so T *is uniquely determined by* T_p. Let $T(t)$ denote the schedule with $T_p = t$ and $D(t) = T_0 - T_p$ be its duration.
Proofs of this section can be found in Appendix A due to their similarity with the proofs for the forward labeling algorithm.

Property 5. *If $t < t'$, $T = T(t)$, and $T' = T(t')$, then $0 \leq T'_k - T_k \leq T'_{k+1} - T_{k+1} \leq t' - t$, $0 \leq k \leq p - 1$. (As a consequence, $D(t) \geq D(t')$).*

A *schedule* T is feasible if the time windows and the route duration constraint are satisfied, i.e., $l_{v_k} \leq T_k \leq u_{v_k}$, $0 \leq k \leq p$, and $T_0 - T_p \leq D$.

Property 6. *Let $t < t' < t''$. If $T = T(t)$ and $T'' = T(t'')$ are feasible then $T' = T(t')$ is feasible.*

A *backward path $\overleftarrow{P}$ is feasible* if there is at least one feasible schedule for it. In general, for a given feasible $\overleftarrow{P}$, there is an interval of times t such that $T(t)$ is feasible. Labels represent relevant information about those feasible schedules. The label $\overleftarrow{L}_{\overleftarrow{P}} = (i, \bar{c}, LT, RD, TB)$ associated with the feasible backward path $\overleftarrow{P}$ is defined by the following attributes:

- i - the first node v_p.
- $\bar{c}$ - the accumulated reduced cost.
- LT - the Latest t such that $T(t)$ is feasible.
- RD - the minimum Route Duration for all feasible schedules, i.e., $D(LT)$.
- TB - the Time Buffer, the largest value such that the $D(LT - TB) = RD$.

We prove some additional properties about the schedules for a backward path $\overleftarrow{P}$.

Property 7. *If $t < LT - TB$ then $D(t) = RD + (LT - TB) - t$*

Property 8. *If $TB > 0$, $RD = \sum_{k=0}^{p-1} t_{v_{k+1}v_k}$. In other words, if $TB > 0$, all schedules $T(t)$, $LT - TB \leq t \leq LT$, are no-waiting.*

The backward labeling algorithm is initialized with the empty forward path $\overleftarrow{P} = (0)$ associated with the forward label $(0, 0, u_0, 0, u_0)$. The backward extension procedure is presented as Algorithm 2.

Proposition 2. *The extendBackward procedure is correct.*

6.1.3 Concatenation

A forward label $(i, \bar{c}, ET, RD, TB)$ can be concatenated with a backward label $(i, \bar{c}', LT', RD', TB')$ on vertex i if and only if the following proposition is true. Let $W = \max\{0, (LT' - TB') - (ET + TB)\}$ be the minimum waiting time induced by following the backward path after the forward path.

$$RD + RD' + W \leq D \tag{6a}$$

$$ET \leq LT' \tag{6b}$$

Algorithm 2 extendBackward($(j, \bar{c}, LT, RD, TB)$, $a = (i, j)$)

$LT^0 = LT - t_{ij}$
if $LT^0 < l_i$ **then**
  return infeasible
$LT' \leftarrow \min\{LT^0,\ u_i\}$
$w_i \leftarrow LT^0 - LT'$
$TB' \leftarrow \max\{0, \min\{TB - w_i, LT' - l_i\}\}$
$RD' \leftarrow RD + t_{ij} + \max\{0,\ w_i - TB\}$
if $RD' > D$ **then** ▷ *Can be improved by using* $RD' + t_{0i} > D$
  return infeasible
$\bar{c}' \leftarrow \bar{c} + \bar{c}_{ij}$
return $(i, \bar{c}', LT', RD', TB')$

Equation (6a) verifies that the total duration is feasible. Equation (6b) verifies that the time windows are respected. The reduced cost of the resulting route is equal to $\bar{c} + \bar{c}'$.

Property 9. *A feasible forward label* $L = (i, \bar{c}, ET, RD, TB)$ *representing forward path* $\vec{P}$ *and a feasible backward label* $L' = (i, LT', RD', TB')$ *representing backward path* $\overleftarrow{P}$ *can be concatenated into a feasible route if and only if* (6a) *and* (6b) *are verified.*

Proof. Proof. Let $\vec{T}$ and $\vec{D}$ denote the schedules associated with $\vec{P}$ and their corresponding durations, respectively, and $\overleftarrow{T}$ and $\overleftarrow{D}$ the schedules and durations associated with $\overleftarrow{P}$. We must prove that conditions (6a) and (6b) are both verified if and only if there exists t such that $\vec{T}(t)$ and $\overleftarrow{T}(t)$ are both feasible, and $\vec{D}(t) + \overleftarrow{D}(t) \leq D$. By definition of ET and LT', $ET \leq t \leq LT'$ is a necessary and sufficient condition for the feasibility of $\vec{T}(t)$ and $\overleftarrow{T}(t)$. Hence, the existence of t meeting this criterion is equivalent to satisfying (6b). Given that $ET \leq t \leq LT'$, we have $\vec{D}(t) = RD + \max\{0, t - (ET + TB)\}$ and $\overleftarrow{D}(t) = RD' + \max\{0, (LT' - TB') - t\}$ by Properties 3 and 7, respectively. Thus, the minimum value that can be achieved for $\vec{D}(t) + \overleftarrow{D}(t)$ by varying t is $RD + RD' + W$, which implies that finding t such that $\vec{D}(t) + \overleftarrow{D}(t) \leq D$ is equivalent to satisfying (6a). □

Figure 1 illustrates the different cases happening in concatenation. The horizontal axis represents time, and different labels are represented on the vertical axis. The first one, $\vec{L}$, is a generic forward label, while the others represent backward labels. ET and $ET + TB$ are highlighted with a dashed line. On the right, the necessary and sufficient condition such that the backward label can be concatenated with the forward label is presented. Label $\overleftarrow{L}_1$ minimal start of service that preserves the duration RD_1 is higher than the maximal start of service of $\vec{L}$ that preserves the duration RD. Therefore, the vehicle will have to wait W_1 time units, and the total duration becomes $RD + W_1 + RD_1$. For $\overleftarrow{L}_2, \ldots, \overleftarrow{L}_5$, there exists a time for which additional waiting time is not necessary, and the total duration is the sum of the forward and backward path durations. Finally, $\overleftarrow{L}_6$, can only start service before the earliest start of service of $\vec{L}$, and therefore the concatenation is infeasible.

6.1.4 Forward Dominance Rule

To avoid the enumeration of all feasible paths, provable redundant labels are eliminated through a dominance rule. A dominated label can be discarded as long as a dominating label is kept.
A forward label $(i, \bar{c}, ET, RD, TB)$ dominates another forward label $(i, \bar{c}', ET', RD', TB')$ if:

$$\bar{c} \leq \bar{c}' \tag{7a}$$

$$ET \leq ET' \tag{7b}$$

$$RD \leq RD' - \max\{0, ET' + TB' - ET - TB\} \tag{7c}$$

The intuition behind Equation (7c) is that a smaller value of $ET + TB$ can be compensated by a smaller route duration, but not otherwise, i.e., a larger route duration cannot be compensated by a larger value of $ET + TB$. The following equivalence holds:

$$(7c) \Leftrightarrow \begin{cases} RD \leq RD' \\ RD - ET - TB \leq RD' - ET' - TB' \end{cases} \tag{8}$$

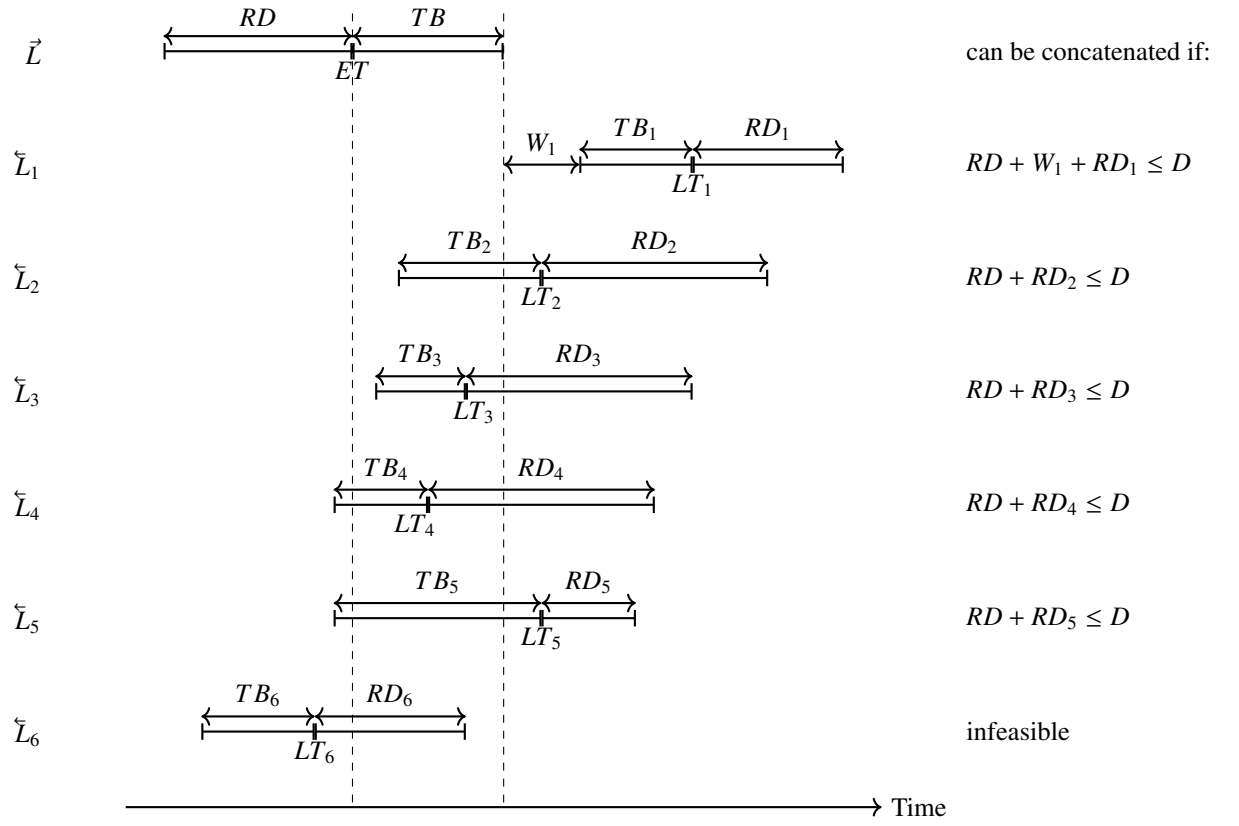


Figure 1: Concatenation cases illustration

Theorem 1. *If* (7a)*,* (7b)*, and* (7c) *are satisfied, then there is an optimal solution to the VRPTWDC that does not use label L'.*

Proof. Proof. Let's assume forward label $L = (i, \bar{c}, ET, RD, TB)$ verifies conditions (7a), (7b), (7c) over forward label $L' = (i, \bar{c}', ET', RD', TB')$. Let's assume $\overleftarrow{L} = (i, \overleftarrow{c}, \overleftarrow{LT}, \overleftarrow{RD}, \overleftarrow{TB})$ a backward label feasible for concatenation with L'. Because (7b) is verified, and condition (6b) is satisfied for labels L' and $\overleftarrow{L}$, we have that $ET \leq ET' \leq \overleftarrow{LT}$, and thus, that (6b) is also satisfied for labels L and $\overleftarrow{L}$.
The total route duration after concatenation with L can be computed as: $RD + \overleftarrow{RD} + \max\{0, \overleftarrow{LT} - \overleftarrow{TB} - ET - TB\} = \overleftarrow{RD} + \max\{RD, \overleftarrow{LT} - \overleftarrow{TB} + RD - ET - TB\}$. Now, we analyze the difference between the route durations of the two complete routes:

$$\begin{aligned}
&\overleftarrow{RD} + \max\{RD, \overleftarrow{LT} - \overleftarrow{TB} + RD - ET - TB\} \\
&\qquad - (\overleftarrow{RD} + \max\{RD', \overleftarrow{LT} - \overleftarrow{TB} + RD' - ET' - TB'\}) \\
&= \max\{RD, \overleftarrow{LT} - \overleftarrow{TB} + RD - ET - TB) \\
&\qquad - \max\{RD', \overleftarrow{LT} - \overleftarrow{TB} + RD' - ET' - TB'\}
\end{aligned}$$

By (8), the previous difference cannot be positive. Thus, since condition (6a) is satisfied for labels L' and $\overleftarrow{L}$, it is also satisfied for labels L and $\overleftarrow{L}$. Therefore, any concatenation feasible for L' is feasible for L. Furthermore, as $\bar{c} \leq \bar{c}'$ and the reduced cost induced by concatenation is $\bar{c} + \overleftarrow{c}$, replacing L' with L in a complete path has a non-increased reduced cost. Therefore, L' can be omitted.

□

6.1.5 Backward Dominance Rule

Similarly, a backward label $(i, \bar{c}, LT, RD, TB)$ dominates another backward label $(i, \bar{c}', LT', RD', TB')$ if:

$$\bar{c} \leq \bar{c}' \tag{9a}$$

$$LT \geq LT' \tag{9b}$$

$$RD \leq RD' - \max\{0, LT - TB - LT' + TB'\} \tag{9c}$$

The intuition behind Equation (9c) is that a larger value of $LT - TB$ can be compensated by a smaller route duration, but not otherwise, i.e., a larger route duration cannot be compensated by a smaller value of $LT - TB$. The following equivalence holds:

$$(9c) \Leftrightarrow \begin{cases} RD \leq RD' \\ RD + LT - TB \leq RD' + LT' - TB' \end{cases} \tag{10}$$

Theorem 2. *If* (9a), (9b), (9c) *are satisfied, then there is an optimal solution to the VRPTWDC that does not use label L'.*

Proof. Proof. Let's assume backward label $L = (i, \bar{c}, LT, RD, TB)$ verifies conditions (9a), (9b), (9c) over backward label $L' = (i, \bar{c}', LT', RD', TB')$. Let's assume $\vec{L} = (i, \vec{c}, \vec{ET}, \vec{RD}, \vec{TB})$ a forward label feasible for concatenation with L'. Because (9b) is verified, and condition (6b) is satisfied for labels $\vec{L}$ and L', we have that $LT \geq LT' \geq \vec{ET}$, and thus, that (6b) is also satisfied for labels $\vec{L}$ and L.
The total route duration after concatenation with L can be computed as: $\vec{RD} + RD + \max\{0, LT - TB - \vec{ET} - \vec{TB}\} = \vec{RD} + \max\{RD, RD + LT - TB - \vec{ET} - \vec{TB}\}$. Now, we analyze the difference between the route durations of the two complete routes:

$$\begin{aligned} &\vec{RD} + \max\{RD, RD + LT - TB - \vec{ET} - \vec{TB}\} \\ &\qquad - (\vec{RD} + \max\{RD', RD' + LT' - TB' - \vec{ET} - \vec{TB}\}) \\ &= \max\{RD, RD + LT - TB - \vec{ET} - \vec{TB}\} \\ &\qquad - \max\{RD', RD' + LT' - TB' - \vec{ET} - \vec{TB}\} \end{aligned}$$

By (10), the previous difference cannot be positive. Thus, since condition (6a) is satisfied for labels $\vec{L}$ and L', it is also satisfied for labels $\vec{L}$ and L. Therefore, any concatenation feasible for L' is feasible for L. Furthermore, as $\bar{c} \leq \bar{c}'$ and the reduced cost induced by concatenation is $\bar{c} + \overleftarrow{c}$, replacing L' with L in a complete path has a non-increased reduced cost. Therefore, L' can be omitted.

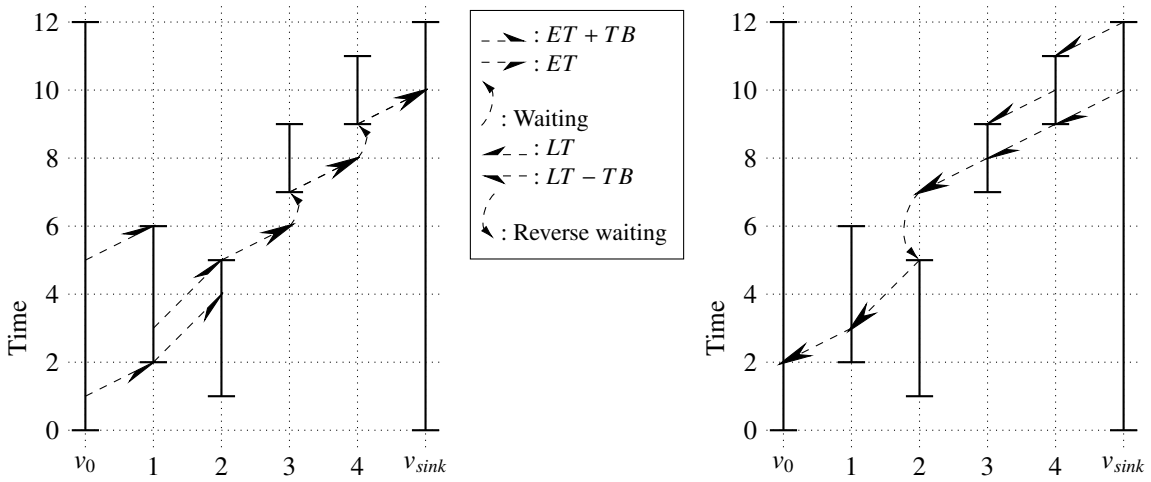
□

6.1.6 Example

Figure 2 presents a small numerical example to illustrate the extension's computation. We use $D = 10$ and a service time of 0 at each customer. The current node and the current accumulated reduced cost are omitted in the labels to improve clarity. Forward and backward labels have the form (ET, RD, TB) and (LT, RD, TB), respectively. The row that starts with "Waiting presents the waiting time induced by the concatenation of the forward label and the backward label at each node, according to Equation (6a).

Forward	$(0,0,12)$	$(2,1,4)$	$(4,3,1)$	$(7,5,0)$	$(9,7,0)$	$(10,8,0)$
	v_0 —1→ 1 —2→ 2 —1→ 3 —1→ 4 —1→ v_{sink}					
TW	$[0,12]$	$[2,6]$	$[1,5]$	$[7,9]$	$[9,11]$	$[0,12]$
Backward	$(2,8,0)$	$(3,7,0)$	$(5,5,0)$	$(9,2,1)$	$(11,1,2)$	$(12,0,12)$
Waiting	(0)	(0)	(0)	(1)	(0)	(0)

Figure 2: Labeling algorithm computation example



(a) Forward extensions

(c) Backward extensions

Figure 3: Example of a forward and the corresponding backward paths

The Figure 3 displays the evolution of the attributes of the label at each node. Arc from node i to node j has always duration t_{ij}. The coordinates of these arcs' heads give the last node and a possible start of service of this node for the corresponding label, as indicated in the legend.

6.1.7 Overall Algorithm

The overall forward labeling algorithm is described in Algorithm 3. The overall backward labeling algorithm is similar and, therefore, not detailed here.

Algorithm 3 Forward Labeling

$\mathcal{L} \leftarrow \{(0, 0, 0, 0, u_0)\}$
while exists non-extended labels in $\mathcal{L}$ **do**
  $L = (i, \bar{c}, ET, RD, TB) \leftarrow$ Choose a non-extended label from $\mathcal{L}$
  Mark L as extended
  for all $a = (i, j) \in \delta^+(i)$ **do**
    $L' \leftarrow$ extendForward($(i, \bar{c}, ET, RD, TB)$, a)
    if L' = infeasible **then**
      Continue to next a
    if L' is not dominated by a label in $\mathcal{L}$ **then**
      Remove from $\mathcal{L}$ the labels dominated by L'
      Insert L' in $\mathcal{L}$
An optimal solution is retrieved from the minimum cost label in $\mathcal{L}$ with end vertex 0

6.2. Counter example for Bettinelli et al. (2011) dominance rule

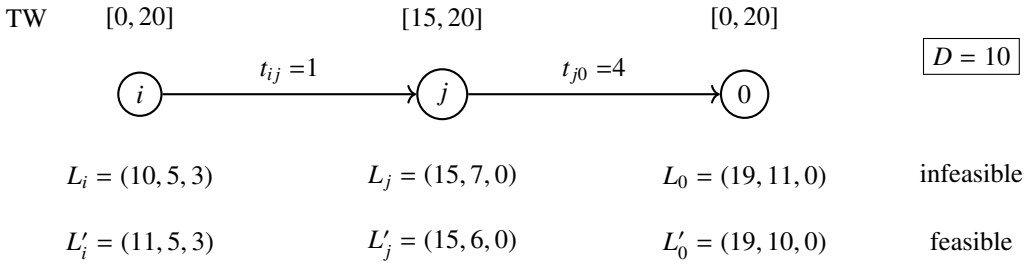


Figure 4: Example: two extensions of labels.

Adapting the labeling algorithm in (Bettinelli et al. 2011) to the label notation used in this paper, its domination criterion can be stated as follows:

$$ET \leq ET' \tag{11}$$
$$RD \leq RD' \tag{12}$$
$$RD - TB \leq RD' - TB' \tag{13}$$

We give a counterexample that shows this dominance criterion to be incorrect. Using the same notation as in Figure 2, Figure 4 presents two labels ending at i, their extensions to vertex j, and then to the depot. It also shows time windows as well as the travel time between successive pairs of nodes. The duration limit is 10. The figure shows that L_i does not dominate L'_i, as there exists a sequence of extensions for which L_i is feasible and L'_i is not.
Table 1 compares the dominance conditions used in (Bettinelli et al. 2011) and in our work and shows that only our conditions return the correct value. We recall that (7c) = $RD \leq RD' - \max(0, ET' + TB' - ET - TB)$.

(Bettinelli et al. 2011)		This work	
Dominance condition	valid for L_i and L'_i ?	Dominance condition	valid for L_i and L'_i ?
$ET \leq ET'$	True	$ET \leq ET'$	True
$RD \leq RD'$	True	(7c)	False
$RD - TB \leq RD' - TB'$	True	-	-
L_i dominates L'_i ?	True	L_i dominates L'_i ?	False

Table 1: Comparaison between our dominance criterion for the time resource and the one from (Bettinelli et al. 2011)

6.3. Duration measured until the last delivery

Some applications require that the duration limit only take into account the travel time until the last customer served, not including the return to the depot. To handle this variant, if the triangular inequality is verified, we do the following

for all $i \in C$:

- set $u_i = \min\{u_i, u_0 - t_{ij}\}$;
- set $t_{i0} = 0$, without modifying the corresponding $\bar{c}_{i0}$.

In this case, the duration resource will only consider the duration limit until the last customer, not until the truck returns to the depot. Furthermore, the depot time window is checked at each customer with the modified deadlines.

6.4. Minimize total duration

We also address a variant of the VRPTW where the objective function is to minimize the total duration. It is referred to as MD-VRPTW. To minimize the total duration, minor adjustments to the algorithm are necessary.

- The master problem's objective function is replaced with the $\sum_{r \in \mathcal{R}} c_r \lambda_r$, where c_r is the duration of route $r \in \mathcal{R}$.
- In both extension functions, lines 8-10 are replaced with
$$\bar{c}' = \bar{c} + \bar{c}_{ij} + \max\{0, w - TB\},$$
with $w = w_j$ in the forward extension and $w = w_i$ in the backward extension.
- In the concatenation conditions, the duration limit condition (6a) is removed, and $W = \max\{0, (LT' - TB') - (ET + TB)\}$ is added to the reduced cost of the complete route.
- In the forward dominance rule, condition (7c) is replaced with $RD \leq RD'$ and condition (7a) is replaced with $\bar{c} \leq \bar{c}' + \max\{0, ET' + TB' - ET - TB\}$
- In the backward dominance rule, condition (9c) is replaced with $RD \leq RD'$ and condition (9a) is replaced with $\bar{c} \leq \bar{c}' + \max\{0, LT - TB - LT' + TB'\}$

7. Computational experiments

This section presents the computational results evaluating the performance of our algorithm. We set a one-hour computation limit for each experiment.

7.1. Configuration

The algorithms are coded in C++. We also used:

- IBM ILOG CPLEX 20.1 as the LP solver in the relaxed restricted master problem and as a MIP solver for enumerated routes.
- The generic branch-cut-and-price code from (Sadykov and Vanderbeck 2021), including a modified version of the labeling algorithm code from (Sadykov et al. 2021).

The experiments presented in this paper were carried out:

- Using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr).
- Running on 2.6 GHz 2x 18-core Cascade Lake Intel Xeon Skylake Gold 6240 CPU using a single thread with 5.3 Go memory.

7.2. Instances

This section presents the instances used for each problem.

- Section 7.3.1 evaluates the performance of the algorithm for the VRPTWDC. We use benchmark instances from (Solomon 1987), the most widely adopted dataset for vehicle routing problems with time windows. We follow Solomon's convention on rounding; in other words, we round distances to one decimal place. To introduce practical constraints, we impose route duration limits of 6, 8, and 10 hours for the VRPTWDC, reflecting common planning horizons such as partial or full-day vehicle shifts. Travel and service times are expressed in consistent time units, as defined in the original instances.
- Section 7.3.2 compares our results with those obtained in (Bettinelli et al. 2011) on instances from (Cordeau et al. 2001), as they are the only instances considered in their article that use a duration limit. They consider the Multi-Depots Heterogeneous Vehicle Routing Problem with Time Windows (MDHVRPTW). In (Bettinelli et al. 2011)'s numerical examples, all instances imposing duration constraints use a homogeneous fleet. Therefore, we do not need to be able to handle a heterogeneous fleet to compare our results with theirs. To address the multi-depot case, we formulated a pricing subproblem for each depot.

- Section 7.4.1 evaluates the performance of the algorithm for the MD-VRPTW, a variant of the VRPTW where the objective function is to minimize the total duration. Experiments on the MD-VRPTW have only been conducted on class 1 (Solomon 1987) instances, as wider time windows would further decrease the difference between the total duration objective and the total travel time objective.
- Section 7.4.2 compares our results with those obtained in (Michelini 2018) for the VRPTWWTC. In this problem, the objective function is to minimize the sum of the durations. Furthermore, time windows on the depot are not considered; instead, the depot's time window span is used as a duration limit. More formally, they set: $D = (u_0 - l_0)$ and $[l_0,\ u_0] = [-\infty, \infty]$. In practice, this exactly states that the first customer $i \in C$ of a route is always visited at l_i, and that the routes do not have a deadline. Furthermore, as the instances considered verify the property $u_i + t_{i0} \leq u_0,\ \forall i \in C$, setting u_0 to ∞ has no impact. To avoid using negative departure date, we add $\max_{i \in C} c_{0,i}$ to all $l_i, u_i\ i \in C$ and we set $u_0 = \max_{c \in C} u_c + t_{c0}$ (to remain generic). To address this problem, we combined the proposed algorithms for the VRPTWDC and for the MD-VRPTW.

Table 2 presents a summary of the instances used in each section. (Solomon 1987) class 1 refers to the set of instances with tight time windows, while class 2 refers to the set of instances with wide time windows. Column "Problem" presents the problem abbreviation. Column "Set" presents the data set used in the computation. Column "Duration constraints" presents whether the problem includes duration constraints or not. Column "Objective" presents the objective function, which is either the total travel time or the total duration. Column "Comparison" presents the articles we compare our results with.

Problem	Set	Duration constraints	Objective	Comparison
VRPTWDC	(Solomon 1987) class 1 and 2	Yes	Travel time	-
MDHVRPTW	(Cordeau et al. 2001)	Yes	Travel time	Bettinelli et al. (2011)
MD-VRPTW	(Solomon 1987) class 1	No	Duration	-
VRPTWWTC	(Solomon 1987) class 1	Yes	Duration	Michelini (2018)

Table 2: Summary of the instance sets used in each section

7.3. Results Duration limit

7.3.1 VRPTWDC

Table 3 summarizes the results obtained for the VRPTWDC. Column "Instances" gives the name of the set of instances as follows <Duration limit in hours>-< Solomon's class >. Column "Nb" gives the number of instances in the set. Column "Opt" displays the number of optimal solutions found. Column "UB" gives the number of instances for which a feasible solution has been found. The "%VRPTW" column displays the average percentage increase of the optimal solution for the VRPTWDC over the optimal solution for the VRPTW. This indicator highlights the significance of time limit constraints and their impact on increasing the objective value. The "Time" column provides the average time in seconds required to solve the instances, including those that remain unsolved within the time limit. The last iteration of the iterative cutoff scheme is the most complex, as it is the only iteration involving feasible solutions. That is why it is detailed here. Column "%rGapL" gives the average root gap in percent at the last iteration, computed as $100 * \frac{\text{PB}-\text{rDB}}{\text{rDB}}$, with PB the primal bound and rDB the best dual bound at the root node. The "NodesL" column provides the average number of nodes explored in the search tree during the last iteration. The "TimeL" column provides the average time spent in the last iteration of the algorithm. The "Iter" column provides the average number of iterations performed by the scheme.
Overall, 129 of the 168 instances are solved to optimality. One can observe that increasing D usually increases the complexity of the instance. Note that all instances solved in R1 and RC1 have the same optimal solutions as the VRPTW, resulting in a value of column %VRPTW of 100. Most of the computation time is spent on the last iteration of the scheme.

7.3.2 Comparison with the MDHVRPTW

Table 4 compares our results with the ones obtained in (Bettinelli et al. 2011) on (Cordeau et al. 2001) instances. Each computational time reported in the literature was multiplied by a factor of 0.645. This normalization factor was

Class	Nb	Opt	UB	%VRPTW	Time	%rGapL	NodesL	TimeL	Iter
6-C1	9	9	9	291	303.4	0.93	92.6	297.0	2.1
8-C1	9	9	9	212	411.1	0.95	27.9	344.6	3.0
10-C1	9	7	8	173	971.1	0.29	8.6	910.7	2.7
6-C2	8	8	8	426	9.5	0.00	1.0	8.0	1.2
8-C2	8	8	8	314	18.5	0.05	1.2	11.8	1.9
10-C2	8	8	8	261	255.8	0.44	12.0	222.7	2.4
6-R1	12	12	12	100	595.9	0.08	2.7	491.3	1.5
8-R1	12	12	12	100	725.5	0.08	2.8	626.3	1.5
10-R1	12	12	12	100	688.7	0.07	2.7	596.1	1.5
6-R2	11	4	4	112	2596.9	0.05	0.7	2223.8	3.2
8-R2	11	3	3	108	2801.8	0.01	0.5	2424.8	3.5
10-R2	11	3	3	102	2872.8	0.15	0.6	2189.8	3.7
6-RC1	8	6	6	100	1171.6	0.17	4.0	957.8	2.5
8-RC1	8	6	6	100	1162.8	0.17	4.0	987.1	2.2
10-RC1	8	6	6	100	1174.4	0.17	5.0	920.9	2.4
6-RC2	8	6	7	114	1246.0	0.02	1.0	940.1	2.4
8-RC2	8	6	6	108	1883.9	0.22	2.2	1436.9	2.5
10-RC2	8	4	4	103	2106.8	0.19	1.0	1382.1	3.1
6	56	45	46	199	1033.3	0.24	16.4	862.1	2.2
8	56	44	44	163	1209.8	0.28	6.2	1013.8	2.4
10	56	40	41	145	1373.2	0.22	4.6	1065.0	2.6

Table 3: VRPTWDC - Summary of our results.

		Our algorithm						(Bettinelli et al. 2011)		
Instance	n	OptVal	Time	Iter	TimeL	NodesL	%rGapL	Time×0.645	LB	UB
Pr01	48	1074.12	2.01	1	1.6	1	0	1.42	1074.21	1074.21*
Pr02	96	1762.21	211.64	2	85.78	3	0.13	2322	1743.53	1851.99
Pr03	144	2373.65	82.14	1	81.03	1	0	-	-	-
Pr04	192	-	3600	3	3600	7	-	-	-	-
Pr05	240	-	3600	3	3600	1	-	-	-	-
Pr06	288	-	3600	3	3600	1	-	-	-	-
Pr07	72	1418.22	19.16	1	18.42	1	0	548.53	1418.22	1418.22
Pr08	144	2096.73	649.84	2	224.51	3	0.1	2322	-	-
Pr09	216	2712.56	3600	3	2155.02	5	0.09	-	-	-
Pr10	288	-	3600	3	2787.8	1	-	-	-	-

Table 4: Detailed results for instances in Bettinelli et al. (2011)

determined by comparing the single-thread performance ratings of the processors used in our experiments and those employed in the literature approach, based on data available at `https://www.cpubenchmark.net/`. Bettinelli et al. (2011) consider a one-hour time limit, resulting in a scaled time limit of 2322 seconds. Column "Instance" gives the instance name. Column "n" gives the number of customers in the instance. Column "OptVal" gives the optimal value of the instance, or "-" when no feasible solution is found within the time limit. Note that our algorithm did not find any non-optimal feasible solution within the time limit. We obtained six optimal solutions out of the ten instances. Column "Time×0.645" gives the scaled computation time used in (Bettinelli et al. 2011), "UB" the best upper bound they find, and "LB" the best lower bound they find. The remaining columns follow the same notation as in Table 3. Note that the optimal value computed for instance Pr01 differs between us and (Bettinelli et al. 2011). It might be a typo, as they have 1074.21 instead of 1074.12, or it might be due to the error in their dominance function explained in Section 6.2.

7.4. Results Minimizing total duration

7.4.1 MD-VRPTW

Table 5 shows the results on the MD-VRPTW on Solomon instances. Notations are the same as for Table 3.

Class	Nb	Opt	Time	Iter	TimeL	NodesL	%rGapL
C1	9	9	49.4	1.2	31.4	1.4	0.00
R1	12	12	472.6	1.7	391.6	3.5	0.07
RC1	8	6	1397.9	2.1	1252.5	10.8	0.10

Table 5: Summary of the results for the MD-VRPTW.

Table 6 presents the difference between the objective function values of the MD-VRPTW and the VRPTW, using the solutions provided at CVRPLIB. We compared the values for both the duration objective, computing the minimal duration for the VRPTW optimal solutions, and the total travel time objective. Given the objective function, column "MD-VRPTW" gives the value of the optimal solution for the MD-VRPTW, or "-" if the optimal solution is unknown. Column "VRPTW" gives the value of the objective for the optimal solution for the VRPTW. Column "Gap (%)" is computed as $100 \times (MDVRPTW/VRPTW - 1)$, where $MDVRPTW$ is the value of the optimal solution for the MD-VRPTW and $VRPTW$ is the value of the optimal solution for the VRPTW. We can see that many optimal values are the same for both problems. This is explained by the fact that the optimal solutions for the VRPTW are no-wait solutions. Computing average gap on instances for which the gap is non-zero, the average duration gap is -3.11%, while the average total travel time gap is 1.87%, showing that considering the duration objective has a less negative impact on the total travel time than vice-versa.

7.4.2 Comparison with the VRPTWWTC

We compare our algorithm with the ones presented in Michelini (2018), solving the VRPTWWTC. Four different branch-cut-and-price settings are proposed: DSSR-L, NG-DSSR-L-E, NGRR and NG-DSSR-L. We refer the reader to Michelini (2018) for more information. In (Michelini 2018), the benchmarking is performed on a computer with an Intel i7–3930K @ 3.20Ghz processor with 6 cores and 60GB of RAM, operating under Windows 10 Pro. According to CPUbenchmark, the processor is similar to what we used. Therefore, the normalization is omitted. They set a computation time limit of 6 hours. Table 7 presents the comparison between the performance of his algorithms and our algorithm. Column "Class" refers to instances from the Solomon instance set, using the following notation: `<Distr><Window>-<n>`. `<Distr>` gives information about the distribution of the customers: "C" for clustered, "R" for random, or "RC" for a mix of the two. `<Window>` specifies the window settings. It is always 1 here. Finally, `<n>` restricts the number of customers: only the `<n>` first customers are considered. Column "TW" shows the average time window durations. Columns "Gap" give the average gap for unsolved instances where a feasible solution was found, computed as $100 * (UB - LB)/LB$, where UB is the best primal bound and LB the best dual bound. Columns "Time*" are different from the column "Time". In the column "Time", the time average includes the unsolved instances with a time limit of one hour, while in columns "Time*", the time average is computed only for the solved instances. The remaining notations are the same as for Table 3.

	Total duration objective			Total travel time objective		
Instance	MD-VRPTW	VRPTW	Gap (%)	MD-VRPTW	VRPTW	Gap (%)
C101	98273	98273	0.00	8273	8273	0.00
C102	98273	98273	0.00	8273	8273	0.00
C103	98273	98700	-0.43	8273	8263	0.12
C104	98263	98553	-0.29	8263	8229	0.41
C105	98273	98273	0.00	8273	8273	0.00
C106	98273	98273	0.00	8273	8273	0.00
C107	98273	98273	0.00	8273	8273	0.00
C108	98273	98273	0.00	8273	8273	0.00
C109	98273	98273	0.00	8273	8273	0.00
R101	28939	31920	-9.34	17298	16377	5.62
R102	26190	28349	-7.62	15701	14666	7.06
R103	22670	24545	-7.64	12476	12087	3.22
R104	19778	20253	-2.35	9760	9715	0.46
R105	23758	24060	-1.26	13631	13553	0.58
R106	22494	23112	-2.67	12486	12346	1.13
R107	20715	20992	-1.32	10715	10646	0.65
R108	19321	19321	0.00	9321	9321	0.00
R109	21469	21469	0.00	11469	11469	0.00
R110	20680	20680	0.00	10680	10680	0.00
R111	20487	20487	0.00	10487	10487	0.00
R112	19486	19486	0.00	9486	9486	0.00
RC101	26505	26858	-1.31	16504	16198	1.89
RC102	24884	25356	-1.86	14884	14574	2.13
RC103	22687	23282	-2.56	12687	12580	0.85
RC104	-	21433	-	-	11323	-
RC105	25427	26627	-4.51	15397	15137	1.72
RC106	23770	23853	-0.35	13770	13727	0.31
RC107	22078	22078	0.00	12078	12078	0.00
RC108	-	21142	-	-	11142	-
Average*	-	-	-3.11	-	-	1.87

Table 6: Comparison of the optimal solutions for the MD-VRPTW and for the VRPTW.

			DSSR-L				NG-DSSR-L-E				NGRR				NG-DSSR-L				This work					
Class	Nb	TW	Opt	UB	Time*	Gap	Opt	UB	Time*	Gap	Opt	UB	Time*	Gap	Opt	UB	Time*	Gap	Opt	UB	Time	Iter	TimeL	NodesL	rGapL
C1–25	9	3100	7	9	3533	0.32	6	9	3054	0.21	6	9	3465	0.27	7	9	5201	1.49	9	9	242.0	1.6	187.2	3.0	0.08
R1–25	12	863	12	12	13.2	0	12	12	23.1	0	12	12	33.3	-	12	12	13.1	0	12	12	0.8	1.1	0.7	1.0	0.00
RC1–25	8	839	8	8	11.2	0	8	8	24.3	0	8	8	1968	-	8	8	96.8	0	8	8	0.6	1.0	0.6	1.0	0.00
C1–50	9	3212	6	8	1701	0.01	6	8	607.2	0.01	6	8	930.9	0	6	9	926.8	0.02	9	9	405.0	1.4	320.5	2.8	0.06
R1–50	12	888	9	12	1528	2.03	10	12	3601	2.06	8	12	1676	2.16	9	12	1898	2.02	12	12	13.6	1.1	12.8	1.0	0.00
RC1–50	8	853	2	8	10478	6.72	1	8	1213	6.39	1	8	8058	7.21	1	8	737.8	6.7	8	8	28.5	1.1	15.2	1.0	0.00
C1–75	9	3169	5	9	4274	0.21	5	8	1787	0.2	5	9	3614	0	5	8	2992	0.2	8	8	686.4	1.3	642.5	1.1	0.00
R1–75	12	869	3	12	54.3	1.35	4	12	4923	1.34	4	12	1986	1.43	4	12	2925	1.47	12	12	88.2	1.5	52.8	2.5	0.05
RC1–75	8	849	0	8	-	3.12	0	8	-	3.03	0	8	-	3.5	1	8	17613	3.45	7	7	682.4	2.1	552.2	4.4	0.21
C1–100	9	3216	5	8	1896	-	5	8	955.4	-	5	8	4172	-	5	9	1702	0	8	8	870.7	1.3	838.3	1.3	0.00
R1–100	12	870	2	9	9121	0.74	3	10	10726	0.8	2	7	7270	0.4	3	9	6492	0.8	8	9	1871.9	2.1	1755.7	2.0	0.08
RC1–100	8	854	0	8	-	1.27	0	8	-	1.72	0	8	-	1.97	0	7	-	2.05	6	7	1671.1	2.1	1533.2	7.8	0.13

Table 7: VRPTWWTC - Summary of the results in Michelini (2018).

All instances with up to 50 customers are solved to optimality. Two instances with 75 customers and seven instances with 100 customers remain unsolved. Results also highlight the difficulty our algorithm faces in finding primal bounds, as only two primal bounds were found for unsolved instances. The instances of class C solving times are significantly higher for the instances with at most 50 customers. It can be explained by the fact that some instances have customers with unconstrained time windows. For example, C104 with 50 customers has 40 customers with unconstrained time windows. Wider time windows lead to more labels being able to extend to this customer, but also to a less effective dominance rule. Indeed, wider time windows lead to less waiting time, and therefore less stress on the departure date from the depot, and overall less dominated labels.

8. Conclusion

In this article, we proposed a novel variant of the VRPTW, where route duration is constrained while allowing flexible departure times from the depot. It enables the modeling of driver shift duration, perishable goods deliveries, or battery-related constraints. Although route duration limits introduce additional computational complexity, they better reflect practical operational constraints faced in logistics planning. This variant improves the realism of routing models and supports more accurate scheduling in last-mile delivery contexts.

To address this problem, we proposed a branch-cut-and-price algorithm incorporating an efficient bi-directional labeling procedure and several state-of-the-art components. We have proven the correctness of the labeling procedure, thereby closing a gap in the literature where a prior attempt was flawed. Results indicate that the method can efficiently solve instances with up to 100 customers. We also proposed modifications of the algorithm to address important variants of the problem, such as duration constraints until the last delivery and minimizing the total duration.

The algorithm was evaluated on classical Solomon benchmark instances with route duration constraints of 6, 8, and 10 hours. To compare the performance of our algorithm with the literature, we adapted the method to address two variants of the problem, namely the multi-depot variant and the duration minimization under duration constraints variant. We also presented the results for the VRPTW with duration minimization, without duration constraints. Numerical experiments demonstrate the effectiveness of our algorithm.

Future work could explore integrating stochastic travel times, dynamic customer requests, or real-time traffic data to further increase the model's applicability in dynamic logistics environments. Another research direction is to adapt this algorithm to more complex hours-of-service related VRPTW variants.

References

Azi, N., Gendreau, M., Potvin, J.Y., 2010. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. European Journal of Operational Research 202, 756–763.

Baldacci, R., Christofides, N., Mingozzi, A., 2008a. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming 115, 351–385.

Baldacci, R., Christofides, N., Mingozzi, A., 2008b. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming 115, 351–385. URL: https://doi.org/10.1007/s10107-007-0178-5, doi:doi:10.1007/s10107-007-0178-5.

Baldacci, R., Mingozzi, A., Roberti, R., 2011a. New route relaxation and pricing strategies for the vehicle routing problem. Operations research 59, 1269–1283.

Baldacci, R., Mingozzi, A., Roberti, R., 2011b. New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem. Operations Research 59, 1269–1283. URL: https://pubsonline.informs.org/doi/10.1287/opre.1110.0975, doi:doi:10.1287/opre.1110.0975.

Bettinelli, A., Ceselli, A., Righini, G., 2011. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. Transportation Research Part C: Emerging Technologies 19, 723–740.

Contardo, C., Martinelli, R., 2014. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discrete Optimization 12, 129–146.

Cordeau, J.F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Annals of operations research 153, 29–46.

Cordeau, J.F., Laporte, G., Mercier, A., 2001. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational research society 52, 928–936.

Desaulniers, G., Lessard, F., Hadjar, A., 2008. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science 42, 387–404. Doi: 10.1287/trsc.1070.0223.

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations research 40, 342–354.

Faiz, T.I., Vogiatzis, C., Noor-E-Alam, M., 2019. A column generation algorithm for vehicle scheduling and routing problems. Computers & Industrial Engineering 130, 222–236.

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

Goel, A., Irnich, S., 2017. An exact method for vehicle routing and truck driver scheduling problems. Transportation Science 51, 737–754.

Hernandez, F., Feillet, D., Giroudeau, R., Naud, O., 2014. A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. 4or 12, 235–259.

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

Kallehauge, B., Larsen, J., Madsen, O.B.G., 2006. Lagrangian duality applied to the vehicle routing problem with time windows. Computers & Operations Research 33, 1464–1487. Doi: 10.1016/j.cor.2004.11.002.

Kohl, N., Desrosiers, J., Madsen, O.B.G., Solomon, M.M., Soumis, F., 1999. 2-path cuts for the vehicle routing problem with time windows. Transportation Science 33, 101–116. Doi: 10.1287/trsc.33.1.101.

Kok, A.L., Hans, E.W., Schutten, J.M.J., 2011. Optimizing departure times in vehicle routes. European Journal of Operational Research 210, 579–587.

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

Laporte, G., Nobert, Y., 1983b. A branch and bound algorithm for the capacitated vehicle routing problem. Operations-Research-Spektrum 5, 77–85. URL: https://doi.org/10.1007/BF01720015, doi:doi:10.1007/BF01720015.

Lysgaard, J., Letchford, A.N., Eglese, R.W., 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical programming 100, 423–445. URL: https://link.springer.com/article/10.1007/s10107-003-0481-8.

Michelini, S., 2018. A Comparative Study of Labeling Algorithms within the Branch-and-Price Framework for Vehicle Routing with Time Windows. Ph.D. thesis. ULiège - Université de Liège.

Pecin, D., Contardo, C., Desaulniers, G., Uchoa, E., 2017a. New enhancements for the exact solution of the vehicle routing problem with time windows. INFORMS Journal on Computing 29, 489–502. Doi: 10.1287/ijoc.2016.0744.

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

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., 2017c. Improved branch-cut-and-price for capacitated vehicle routing. Mathematical Programming Computation 9, 61–100. URL: https://link.springer.com/article/10.1007/s12532-016-0108-8.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., Santos, H., 2017d. Limited memory rank-1 cuts for vehicle routing problems. Operations Research Letters 45, 206–209.

Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2018. Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS Journal on Computing 30, 339–360. URL: https://doi.org/10.1287/ijoc.2017.0784, doi:doi:10.1287/ijoc.2017.0784.

Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2020. A generic exact solver for vehicle routing and related problems. Mathematical Programming 183, 483–523. URL: https://link.springer.com/10.1007/s10107-020-01523-z, doi:doi:10.1007/s10107-020-01523-z.

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

Røpke, S., 2012. Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. Presentation at International Workshop on Column Generation 2012. https://www.gerad.ca/colloques/ColumnGeneration2012/presentations/session7/Ropke.pdf (accessed on August 2024).

Sadykov, R., Uchoa, E., Pessoa, A., 2021. A Bucket Graph–Based Labeling Algorithm with Application to Vehicle Routing. Transportation Science 55, 4–28. URL: http://pubsonline.informs.org/doi/10.1287/trsc.2020.0985, doi:doi:10.1287/trsc.2020.0985.

Sadykov, R., Vanderbeck, F., 2021. BaPCod - a generic branch-and-price code. Technical Report. Inria Bordeaux Sud-Ouest. URL: https://inria.hal.science/hal-03340548.

Savelsbergh, M.W., 1992. The vehicle routing problem with time windows: Minimizing route duration. ORSA journal on computing 4, 146–154.

Silva, J.M.P., Uchoa, E., Subramanian, A., 2024. Cluster branching for vehicle routing problems. Technical Report L-2024-2. Cadernos do LOGIS-UFF. Niterói, Brazil.

Solomon, M.M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research 35, 254–265. URL: https://pubsonline.informs.org/doi/10.1287/opre.35.2.254, doi:doi:10.1287/opre.35.2.254.

Tilk, C., Goel, A., 2020. Bidirectional labeling for solving vehicle routing and truck driver scheduling problems. European Journal of Operational Research 283, 108–124.

Appendices

A. Backward labeling proofs

Proof of Property 5

Proof. Proof. First, $0 < T'_p - T_p = t' - t$. Now, consider the decreasing sequence of indices k from $p-1$ to 0. If $T_k = T_{k+1} + t_{v_{k+1}v_k}$ then $T'_k = T'_{k+1} + t_{v_{k+1}v_k}$, so $T'_k - T_k = T'_{k+1} - T_{k+1}$. Otherwise, $T_k = l_{v_k}$ and $T'_k = \max\{T'_{k+1} + t_{v_{k+1}v_k},\ l_{v_k}\}$, so $0 \leq T'_k - T_k \leq T'_{k+1} - T_{k+1}$.

□

Proof of Property 6

Proof. Proof. By Property 5, $l_{v_k} \leq T_k \leq T'_k \leq T''_k \leq u_{v_k}$, $0 \leq k \leq p$, so time windows are satisfied. Moreover, $D(t') \leq D(t) \leq D$.

□

Proof of Property 7

Proof. Proof. Consider the schedule $T' = T(LT - TB)$. There should be a b, $0 \leq b \leq p$ such that $T'_b = l_{v_b}$; otherwise, it would be possible to decrease all the times in T' by a positive amount to produce another schedule T'' such that $T''_0 - T''_p = RD$, a contradiction with the definition of TB. The existence of a $T'_b = l_{v_b}$ acts as a block. Let $T = T(t)$. Since $T_b = T'_b$, $T_0 = T'_0$. So, $D(t) = T'_0 - T'_p + T'_p - T_p = RD + (LT - TB) - t$.

□

Proof of Property 8

Proof. Proof. Consider a schedule $T = T(t)$, $LT - TB \leq t \leq LT$, that is not no-waiting, i.e., $RD = D(t) > \sum_{k=0}^{p-1} t_{v_{k+1}v_k}$. So, there exists a b, $0 \leq b \leq p-1$, such that $T_b = l_{v_b} > T_{b+1} - t_{v_{b+1}v_b}$. Consider the schedule $T' = T(LT - TB)$. Since $T_b = T'_b$, $T_0 = T'_0$. As $D(t) = D(LT - TB)$, then $t = LT - TB$. Consider the schedule $T'' = T(t'')$, where t'' is in the interval $[t, t + l_{v_b} - T_{b+1} - t_{v_{b+1}v_b}[$, i.e., $t'' - t$ is small enough to keep $T''_b = T_b = l_{v_b}$, so that $T_0 = T''_0$. In this case, $D(t'') < RD$ and T'' must be infeasible. Therefore, $t = LT$ and $TB = 0$.

□

Proof of Proposition 2

Proof. Proof. We have to prove that if $L = (j, \bar{c}, LT, RD, TB)$ is the correct label for feasible backward path $\overleftarrow{P} = (v_p = j, v_{p-1}, \ldots, v_1, v_0 = 0)$ then Algorithm 2 either detects that backward path $\overleftarrow{P}' = (v_{p+1} = i, v_p = j, v_{p-1}, \ldots, v_1, v_0 = 0)$ is infeasible or returns the correct label $L' = (i, \bar{c}', LT', RD', TB')$ for $\overleftarrow{P}'$. As LT is the latest departure time from j, then the time LT' is the correct latest departure from i, which is not feasible if smaller than l_i. Assuming $LT' \geq l_i$, and noting that $\bar{c}' = \bar{c} + \bar{c}_{ij}$ is trivially correct, it remains to prove the values of RD' and TB' are also correct. We divide this into two cases, depending on w_i:

1. $w_i \leq TB$. In this case, one can check that Algorithm 2 sets $LT' = \min\{LT - t_{ij},\ u_i\}$, and $LT' - TB' = \max\{LT - TB - t_{ij},\ l_i\}$. Then, for any $LT' - TB' \leq t \leq LT'$, the schedule $T'(t)$ can be obtained by extending the schedule $T(t + t_{ij}) = (T_p, ..., T_0)$ for $\overleftarrow{P}$ to the schedule $(T_{p+1} = T_p - t_{ij}, T_p, ..., T_0)$ for $\overleftarrow{P}'$, having duration $D'(t) = RD + t_{ij} = RD'$ because $LT - TB \leq t + t_{ij} \leq LT$. Moreover, if $l_i \leq t < LT' - TB'$, $D'(t) = D(t + t_{ij}) + t_{ij} > RD'$ because $t + t_{ij} < LT - TB$.

2. $w_i > TB$. In this case, $LT' = u_i < LT - TB - t_{ij}$, and $TB' = 0$. Then the schedule $T'(LT')$ can be obtained by extending the schedule $T(LT-TB) = (T_p, \ldots, T_0)$ for $\overleftarrow{P}$ to the schedule $(T_{p+1} = u_i = T_p - t_{ij} - w_i + TB, T_p, ..., T_0)$ for $\overleftarrow{P}'$, having duration $D'(t) = RD + t_{ij} + w_i - TB = RD'$. Moreover, since T_0 cannot be decreased in $T(LT - TB)$, $D'(t) > RD'$ for any $l_i \leq t < LT'$.

□

B. Detailed computational experiments

This section presents the detailed computational experiments. Each instance is denoted using the format <Duration>-<Distr><Window
where:

- <Distr> indicates the spatial distribution of customer locations:
  C: clustered,
  R: random,
  RC: a mix of clustered and random.
- <Window> specifies the time window setting:
  1: tight time windows,
  2: relaxed time windows.
- <ID> refers to the identifier of the original Solomon instance.
- <Duration> is the route duration limit in hours (6, 8, or 10).
- <n> is the number of customers considered (50, 75, or 100).

"<Duration>-" and "-<n>" are omitted if not constrained in the experiment. This naming convention facilitates systematic evaluation across different instance characteristics. We also recall optimal values for the VRPTW as a <Distr><Window><ID>. Column "Instance" gives the name of the instance. Column "Value" gives the value of the optimal solution, the dual bound - primal bound interval when the solution is not proved optimal, or "-" when no primal bound is found. Column "VRPTW" gives the increase between the VRPTW and the VRPTWDC solutions in percent. "Time" gives the computing time in seconds, "Iter" the number of iterations of our algorithm, and "TimeL" the computation time of the last iteration of the iterative cutoff scheme. Column "NodesL" gives the number of nodes processed in the search tree in the last iteration. Column "%rGapL" gives the root gap computed as $100(UB - rootLB)/rootLB$ where UB is the best primal bound and $rootLB$ is the dual bound found at the root node in the last iteration. Column "R1C-L" gives the number of rank-1 cuts at the root node in the last iteration, column "CutSepL" provides the time in seconds taken to compute the cut separations in the last iteration of the algorithm.

B.1. VRPTWDC

Table 8: Detailed results for 100 customer instances for the VRPTWDC

Instance	Value	VRPTW	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
6-C101	24271	293%	1.44	1	1.26	1	0	0	0.01
8-C101	18021	217%	8.32	2	2.38	1	0	15	0.53
10-C101	14703	177%	2.74	1	2.63	1	0	12	0.9
C101	8273								
6-C102	24160	292%	31.38	3	22.15	5	1.3	81	13.04
8-C102	17659	213%	157.01	3	132.38	15	1.73	99	62.49
10-C102	14337	173%	119.37	3	51.32	7	0.67	103	31.21
C102	8273								
6-C103	24049	291%	232.71	3	220.94	97	1.8	73	149.6
8-C103	17242	208%	38.14	3	23.45	1	0	145	14.87
10-C103	14022	169%	1080.12	3	871.96	55	0.88	160	320.16
C103	8263								
6-C104	24002	291%	2155.7	3	2135.11	693	1.93	90	1150.26
8-C104	17196	208%	1109.06	4	1020.81	77	2.49	127	814.06
10-C104	-	-	3600	4	3560.47	-	-	-	-
C104	8229								

Table 8: Detailed results for 100 customer instances for the VRPTWDC

Instance	Value	VRPTW	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
6-C105	24209	292%	2.12	1	2.0	1	0	0	0.01
8-C105	17838	215%	17.08	2	7.37	1	0	73	3.32
10-C105	14430	174%	10.43	2	2.68	1	0	23	0.51
C105	8273								
6-C106	24172	292%	2.41	1	2.29	1	0	0	0.01
8-C106	17801	215%	90.35	3	57.4	7	1.17	168	40.4
10-C106	14402	174%	29.3	2	13.11	1	0	161	3.67
C106	8273								
6-C107	24198	292%	46.71	1	46.56	1	0	0	0.01
8-C107	17789	215%	34.53	2	19.51	1	0	144	8.13
10-C107	14366	173%	28.43	2	16.82	1	0	93	5.34
C107	8273								
6-C108	24061	290%	38.97	3	31.54	3	1.55	117	13.48
8-C108	17590	212%	734.41	4	406.55	37	0.88	257	309.02
10-C108	14189	171%	269.07	3	132.24	11	0.5	257	49.1
C108	8273								
6-C109	24033	290%	218.76	3	210.74	31	1.81	95	77.73
8-C109	17254	208%	1510.78	4	1431.42	111	2.25	178	1143.93
10-C109	[13842, 13996]	-	3600	4	3545.1	-	-	-	-
C109	8273								
6-C201	25549	433%	0.86	1	0.79	1	0	0	0.01
8-C201	18835	319%	4.1	1	4.03	1	0	67	3.22
10-C201	15950	270%	7.66	1	7.56	1	0	129	5.04
C201	5891								
6-C202	25000	424%	3.54	1	3.39	1	0	0	0.01
8-C202	18612	315%	14.39	2	6.8	1	0	45	1.82
10-C202	15505	263%	40.55	2	23.73	1	0	105	4.94
C202	5891								
6-C203	24831	421%	13.09	2	7.45	1	0	16	0.65
8-C203	18282	310%	14.53	2	6.01	1	0	65	2.35
10-C203	15177	257%	175.76	3	109.61	11	0.78	162	59.22
C203	5887								
6-C204	24808	421%	51.24	2	45.75	1	0	35	3.44
8-C204	18194	309%	30.0	2	20.82	3	0.37	92	14.03
10-C204	14930	253%	112.48	2	93.64	1	0	246	40.57
C204	5881								
6-C205	25284	431%	2.35	1	2.26	1	0	0	0.01
8-C205	18595	317%	10.11	2	3.76	1	0	30	0.62
10-C205	15539	264%	29.28	2	13.43	1	0	142	2.67
C205	5864								
6-C206	25041	427%	1.59	1	1.49	1	0	0	0.01
8-C206	18458	314%	12.1	2	4.23	1	0	46	1.01
10-C206	15340	261%	166.03	3	137.43	5	0.61	347	49.67
C206	5860								
6-C207	25068	427%	1.72	1	1.61	1	0	0	0.01
8-C207	18461	315%	13.24	2	7.34	1	0	46	1.11
10-C207	15363	262%	410.73	3	339.78	27	1.02	241	153.5
C207	5858								
6-C208	24908	425%	1.41	1	1.29	1	0	0	0.01
8-C208	18443	314%	49.36	2	41.42	1	0	71	2.98
10-C208	15281	260%	1104.02	3	1056.15	49	1.1	363	446.15

Table 8: Detailed results for 100 customer instances for the VRPTWDC

Instance	Value	VRPTW	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C208	5858								
6-R101	16377	100%	8.4	2	2.19	3	0.09	4	0.83
8-R101	16377	100%	7.19	2	3.03	3	0.09	3	1.68
10-R101	16377	100%	6.22	2	2.48	3	0.09	3	0.84
R101	16377								
6-R102	14666	100%	1.92	1	1.85	1	0	0	0.01
8-R102	14666	100%	1.54	1	1.47	1	0	0	0.01
10-R102	14666	100%	1.47	1	1.39	1	0	0	0.02
R102	14666								
6-R103	12087	100%	21.9	1	21.77	1	0	34	0.88
8-R103	12087	100%	20.47	1	20.34	1	0	28	0.68
10-R103	12087	100%	23.61	1	23.48	1	0	43	0.94
R103	12087								
6-R104	9715	100%	1479.5	2	1144.85	3	0.03	274	11.78
8-R104	9715	100%	1392.54	2	1120.45	3	0.05	264	6.17
10-R104	9715	100%	1518.04	2	1166.44	3	0.07	247	3.64
R104	9715								
6-R105	13553	100%	5.25	1	5.12	1	0	31	1.19
8-R105	13553	100%	8.2	1	8.08	1	0	45	3.03
10-R105	13553	100%	5.3	1	5.18	1	0	43	1.14
R105	13553								
6-R106	12346	100%	80.41	1	80.2	1	0	88	4.78
8-R106	12346	100%	57.37	1	57.17	1	0	98	4.6
10-R106	12346	100%	51.22	1	51.03	1	0	84	3.91
R106	12346								
6-R107	10646	100%	273.29	1	273.01	1	0	201	11.31
8-R107	10646	100%	296.41	1	296.1	1	0	208	12.28
10-R107	10646	100%	266.8	1	266.48	1	0	200	13.46
R107	10646								
6-R108	9321	100%	2104.15	2	1831.94	9	0.15	199	1.75
8-R108	9321	100%	2589.82	2	2294.97	9	0.17	202	1.4
10-R108	9321	100%	1631.81	2	1378.02	3	0.05	247	0.27
R108	9321								
6-R109	11469	100%	171.88	2	110.42	3	0.01	176	20.86
8-R109	11469	100%	164.18	2	107.58	3	0.02	166	23.7
10-R109	11469	100%	159.71	2	107.69	3	0.03	156	15.11
R109	11469								
6-R110	10680	100%	132.54	1	132.32	1	0	160	6.54
8-R110	10680	100%	161.28	1	161.05	1	0	188	7.65
10-R110	10680	100%	154.07	1	153.84	1	0	146	8.79
R110	10680								
6-R111	10487	100%	606.79	2	343.09	3	0.33	179	13.24
8-R111	10487	100%	882.79	2	637.7	3	0.3	187	14.76
10-R111	10487	100%	846.58	2	611.56	3	0.3	189	16.82
R111	10487								
6-R112	9486	100%	2265.1	2	1949.07	5	0.37	251	1
8-R112	9486	100%	3124.53	2	2807.09	7	0.36	261	27.28
10-R112	9486	100%	3600	2	3385.79	11	0.33	293	3.63
R112	9486								
6-R201	13207	115%	33.31	1	33.17	1	0	109	4.04
8-R201	12588	110%	213.35	2	158.58	3	0.03	201	25.33

Table 8: Detailed results for 100 customer instances for the VRPTWDC

Instance	Value	VRPTW	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
10-R201	11856	103%	249.2	2	185.72	3	0.02	185	18.5
R201	11432								
6-R202	11828	114%	1347.43	2	919.63	3	0.18	239	26.77
8-R202	11182	108%	626.09	1	625.75	1	0	151	7.11
10-R202	10647	103%	1137.43	2	573.87	1	0	141	1.73
R202	10296								
6-R203	-	-	3600	4	3069.71	-	-	-	-
8-R203	-	-	3600	4	3081.03	-	-	-	-
10-R203	-	-	3600	5	2356.48	-	-	-	-
R203	8708								
6-R204	-	-	3600	4	3373.69	-	-	-	-
8-R204	-	-	3600	4	3271.62	-	-	-	-
10-R204	-	-	3600	5	2245.9	-	-	-	-
R204	7313								
6-R205	10335	108%	421.65	2	137.06	3	0.02	88	2.93
8-R205	10070	106%	1180.91	2	729.86	1	0	280	7.73
10-R205	9667	101%	1414.39	2	934.9	3	0.44	193	4.7
R205	9498								
6-R206	9566	109%	1563.31	2	953.03	1	0	170	0.41
8-R206	-	-	3600	4	2938.53	-	-	-	-
10-R206	-	-	3600	4	3267.87	-	-	-	-
R206	8759								
6-R207	-	-	3600	4	3173.33	-	-	-	-
8-R207	-	-	3600	5	2881.9	-	-	-	-
10-R207	-	-	3600	5	2305.58	-	-	-	-
R207	7940								
6-R208	-	-	3600	4	3294.29	-	-	-	-
8-R208	-	-	3600	4	3589.66	-	-	-	-
10-R208	-	-	3600	4	2893.29	-	-	-	-
R208	7010								
6-R209	-	-	3600	4	3200.6	-	-	-	-
8-R209	-	-	3600	4	3042.48	-	-	-	-
10-R209	-	-	3600	4	3174.06	-	-	-	-
R209	8548								
6-R210	-	-	3600	4	3127.5	-	-	-	-
8-R210	-	-	3600	4	3083.3	-	-	-	-
10-R210	-	-	3600	4	2974.16	-	-	-	-
R210	9005								
6-R211	-	-	3600	4	3180.32	-	-	-	-
8-R211	-	-	3600	4	3269.89	-	-	-	-
10-R211	-	-	3600	4	3175.9	-	-	-	-
R211	7467								
6-RC101	16198	100%	9.55	1	9.4	1	0	54	2.19
8-RC101	16198	100%	10.16	1	10.03	1	0	64	3.21
10-RC101	16198	100%	10.24	1	10.09	1	0	71	2.78
RC101	16198								
6-RC102	14574	100%	132.02	2	62.8	1	0	134	4.61
8-RC102	14574	100%	120.68	2	57.95	1	0	107	4.94
10-RC102	14574	100%	126.49	2	63.55	1	0	113	4.36
RC102	14574								
6-RC103	12580	100%	737.95	2	456.58	11	0.42	112	66.88

Table 8: Detailed results for 100 customer instances for the VRPTWDC

Instance	Value	VRPTW	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
8-RC103	12580	100%	801.42	2	495.08	15	0.42	143	71.75
10-RC103	12580	100%	729.86	2	478.81	15	0.42	140	88.05
RC103	12580								
6-RC104	-	-	3600	5	3172.45	-	-	-	-
8-RC104	-	-	3600	4	3395.66	-	-	-	-
10-RC104	-	-	3600	4	3013.02	-	-	-	-
RC104	11323								
6-RC105	15137	100%	29.57	1	29.27	1	0	56	1.75
8-RC105	15137	100%	20.35	1	20.15	1	0	53	1.17
10-RC105	15137	100%	26.86	1	26.6	1	0	57	1.21
RC105	15137								
6-RC106	13727	100%	589.18	2	410.94	9	0.39	246	145.08
8-RC106	13727	100%	500.81	2	309.37	7	0.39	247	92.43
10-RC106	13727	100%	670.58	2	478.71	15	0.4	251	144.67
RC106	13727								
6-RC107	12078	100%	674.44	2	257.15	9	0.18	80	5.94
8-RC107	12078	100%	649.32	2	213.51	7	0.18	102	13.18
10-RC107	12078	100%	630.91	2	281.52	7	0.17	94	7.83
RC107	12078								
6-RC108	-	-	3600	5	3263.55	-	-	-	-
8-RC108	-	-	3600	4	3394.96	-	-	-	-
10-RC108	-	-	3600	5	3014.73	-	-	-	-
RC108	11142								
6-RC201	14752	116%	46.45	2	19.57	1	0	57	3.38
8-RC201	13853	109%	17.44	1	17.28	1	0	16	0.4
10-RC201	13144	104%	28.07	1	27.96	1	0	30	0.12
RC201	12618								
6-RC202	12772	116%	216.38	1	216.11	1	0	51	5.22
8-RC202	11992	109%	440.31	1	439.96	1	0	113	5.71
10-RC202	11378	104%	1023.83	2	448.64	1	0	62	0.15
RC202	10923								
6-RC203	10887	117%	1275.34	2	670.97	1	0	93	0.13
8-RC203	10235	110%	2696.31	3	1562.68	3	0.31	76	0.1
10-RC203	-	-	3600	6	1029.47	-	-	-	-
RC203	9237								
6-RC204	[8741, 8815]	-	3600	4	2951.81	-	-	-	-
8-RC204	-	-	3600	4	3346.46	-	-	-	-
10-RC204	-	-	3600	4	2759.22	-	-	-	-
RC204	7835								
6-RC205	13613	117%	139.92	1	139.69	1	0	79	6.03
8-RC205	12848	111%	842.98	2	434.28	3	0	166	20.23
10-RC205	11939	103%	924.83	2	504.49	3	0.08	175	22.94
RC205	11540								
6-RC206	11934	113%	369.84	2	138.13	3	0.12	159	12.52
8-RC206	11189	106%	970.98	2	393.11	3	0.16	194	28.5
10-RC206	10529	100%	477.5	2	234.08	3	0.67	28	6.03
RC206	10511								
6-RC207	10277	106%	720.25	2	298.03	1	0	99	0.08
8-RC207	9675	100%	2902.81	3	2021.52	7	0.84	108	0.54
10-RC207	-	-	3600	4	2905.44	-	-	-	-

Table 8: Detailed results for 100 customer instances for the VRPTWDC

Instance	Value	VRPTW	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
RC207	9629								
6-RC208	-	-	3600	5	3086.18	-	-	-	-
8-RC208	-	-	3600	4	3280.01	-	-	-	-
10-RC208	-	-	3600	4	3147.43	-	-	-	-
RC208	7761								

B.2. VRPTWWTC

We present the detailed results for the VRPTWWTC variant from (Michelini 2018) for the 25, 50, 75 and 100 customer instances in Table 9, 10, 11, and 12, respectively.

Table 9: Detailed results for 25 customer instances for the VRPTWWTC

Instance	TWsize	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C101-25	604	24652	0.33	1	0.17	1	0	0	0
C102-25	3594	24642	47.47	1	47.29	1	0	7	0.02
C103-25	6116	24575	175.23	2	33.9	1	0	32	0
C104-25	7815	24544	370.2	2	142.02	1	0	28	0
C105-25	1211	24652	1.86	1	1.63	1	0	30	0.74
C106-25	737	24652	0.36	1	0.21	1	0	0	0
C107-25	1800	24652	19.24	2	11.38	3	0.13	39	2.91
C108-25	2422	24652	49.64	2	31.24	3	0.24	49	7.61
C109-25	3600	24638	1513.41	2	1416.96	15	0.35	22	28.3
R101-25	100	9789	0.27	1	0.15	1	0	0	0
R102-25	634	8752	0.4	1	0.29	1	0	0	0
R103-25	1069	7442	0.36	1	0.29	1	0	0	0
R104-25	1366	6868	0.35	1	0.29	1	0	0	0
R105-25	300	8054	0.16	1	0.07	1	0	0	0
R106-25	778	7445	0.69	1	0.64	1	0	4	0.22
R107-25	1165	6839	0.81	1	0.76	1	0	0	0
R108-25	1430	6534	2.27	1	2.2	1	0	18	0.61
R109-25	584	6913	0.09	1	0.06	1	0	0	0
R110-25	833	6941	1.37	2	0.21	1	0	0	0
R111-25	937	6846	0.95	1	0.9	1	0	7	0.22
R112-25	1164	6430	2.08	1	2.02	1	0	14	0.76
RC101-25	300	7224	0.3	1	0.24	1	0	0	0
RC102-25	754	6018	0.47	1	0.43	1	0	0	0
RC103-25	1139	5851	0.22	1	0.19	1	0	0	0
RC104-25	1402	5724	0.37	1	0.33	1	0	0	0
RC105-25	553	6815	0.63	1	0.5	1	0	0	0
RC106-25	600	5955	0.34	1	0.3	1	0	0	0
RC107-25	860	5483	0.88	1	0.84	1	0	0	0
RC108-25	1105	5445	1.8	1	1.75	1	0	0	0

Table 10: Detailed results for 50 customer instances for the VRPTWWTC

Instance	TWsize	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C101-50	601	49296	0.87	1	0.56	1	0	0	0
C102-50	3366	48972	336.45	2	237.93	3	0.22	1	1.15
C103-50	5904	48694	483.46	2	230.04	3	0	2	0.43

Table 10: Detailed results for 50 customer instances for the VRPTWWTC

Instance	TWsize	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C104-50	9080	48614	2761.02	2	2364.81	11	0.26	0	1.04
C105-50	1204	48941	31.05	2	19.82	3	0.02	29	1.49
C106-50	944	48650	0.77	1	0.72	1	0	0	0
C107-50	1800	48672	1.18	1	1.13	1	0	0	0
C108-50	2408	48624	5.0	1	4.95	1	0	0	0
C109-50	3600	48624	25.06	1	24.99	1	0	0	0.01
R101-50	100	17399	0.6	1	0.56	1	0	2	0.34
R102-50	592	15095	0.46	1	0.42	1	0	0	0
R103-50	1026	13120	13.91	1	13.82	1	0	21	1.62
R104-50	1574	11323	19.31	1	19.17	1	0	26	0.34
R105-50	300	14327	1.42	1	1.36	1	0	12	0.7
R106-50	740	12948	6.14	1	6.07	1	0	4	0.85
R107-50	1126	12185	28.4	1	28.27	1	0	56	2.39
R108-50	1614	11103	18.57	1	18.41	1	0	70	0.1
R109-50	589	12868	12.52	2	3.73	1	0	33	0.41
R110-50	864	12000	6.48	1	6.4	1	0	14	0.01
R111-50	955	12106	18.68	1	18.57	1	0	43	2.06
R112-50	1178	11302	36.51	1	36.36	1	0	140	1.79
RC101-50	300	14641	1.26	1	1.19	1	0	43	0.43
RC102-50	711	13041	5.04	1	4.94	1	0	63	0.45
RC103-50	1088	12201	65.24	1	65.14	1	0	112	1.56
RC104-50	1565	10520	12.39	1	12.3	1	0	0	0
RC105-50	564	13800	2.64	1	2.54	1	0	95	0.43
RC106-50	600	12232	2.33	1	2.24	1	0	74	0.18
RC107-50	881	11403	23.09	1	23.01	1	0	86	1.45
RC108-50	1116	10981	115.91	2	9.89	1	0	91	0.21

Table 11: Detailed results for 75 customer instances for the VRPTWWTC

Instance	TWsize	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C101-75	605	73992	1.46	1	1.28	1	0	0	0.01
C102-75	3010	73992	72.94	1	72.85	1	0	0	0.01
C103-75	5816	73992	2219.46	2	1825.16	3	0.02	0	0.53
C104-75	8783	-	3600	3	3600	-	-	-	-
C105-75	1212	73992	1.93	1	1.86	1	0	0	0.01
C106-75	1270	73973	9.39	1	9.29	1	0	8	0.23
C107-75	1800	73973	2.36	1	2.28	1	0	0	0.01
C108-75	2424	73963	93.3	1	93.08	1	0	14	1.39
C109-75	3600	73892	177.1	1	176.56	1	0	1	1.85
R101-75	100	24351	1.24	1	1.17	1	0	5	0.41
R102-75	529	21691	1.51	1	1.46	1	0	0	0
R103-75	1014	18246	64.2	2	15.94	1	0	56	1.07
R104-75	1519	15593	305.96	2	140.6	5	0.21	141	31.79
R105-75	300	19311	2.77	1	2.68	1	0	2	0.29
R106-75	684	18467	49.24	2	19.38	1	0	53	1.46
R107-75	1115	16804	158.55	2	92.74	5	0.16	173	23.56
R108-75	1564	15324	87.48	1	87.28	1	0	110	3.69
R109-75	596	17716	101.78	2	69.6	9	0.17	110	28.9

Table 11: Detailed results for 75 customer instances for the VRPTWWTC

Instance	TWsize	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
R110-75	885	16757	199.15	2	115.93	3	0.05	165	17.58
R111-75	931	16442	31.73	1	31.6	1	0	40	0.02
R112-75	1190	15650	55.2	1	55.04	1	0	85	1.35
RC101-75	300	21585	7.55	1	7.43	1	0	86	3.43
RC102-75	660	20000	50.1	2	16.63	3	0.11	56	2.06
RC103-75	1094	18194	612.82	2	295.35	3	0.15	171	6.34
RC104-75	1560	16865	744.91	2	545.2	15	0.76	135	63.31
RC105-75	534	20180	30.99	2	11.67	5	0.26	61	1.32
RC106-75	600	18758	62.24	2	17.56	3	0.06	108	1.92
RC107-75	895	17449	350.65	2	125.2	5	0.14	78	12.66
RC108-75	1146	-	3600	4	3398.22	-	-	-	-

Table 12: Detailed results for 100 customer instances for the VRPTWWTC

Instance	TWsize	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C101-100	608	98273	2.89	1	2.65	1	0	0	0.01
C102-100	3257	98273	383.78	1	383.67	1	0	0	0.58
C103-100	5936	97350	3600	2	3309.7	5	0.03	1	0.89
C104-100	8529	-	3600	3	3600	-	-	-	-
C105-100	1216	98273	3.45	1	3.31	1	0	0	0.03
C106-100	1562	98273	8.48	1	8.38	1	0	0	0.01
C107-100	1800	98273	5.71	1	5.53	1	0	0	0.05
C108-100	2433	98273	57.39	1	57.19	1	0	0	0.04
C109-100	3600	98273	174.19	1	173.85	1	0	6	0.49
R101-100	100	28865	2.79	1	2.66	1	0	3	0.4
R102-100	574	25933	12.52	1	12.45	1	0	0	0.01
R103-100	1030	22471	1809.05	2	1612.54	5	0.04	17	5.22
R104-100	1483	[19584, 19644]	3600	3	3600	-	-	-	-
R105-100	300	23758	21.02	1	20.88	1	0	75	5.03
R106-100	724	22440	2796.5	2	2462.38	7	0.28	52	40.97
R107-100	1130	20574	1578.72	2	1218.52	3	0.15	68	0.07
R108-100	1533	-	3600	3	3600	-	-	-	-
R109-100	589	21469	451.73	2	335.03	3	0.07	144	17.2
R110-100	865	20680	1390.91	2	1003.38	3	0.06	91	0.11
R111-100	931	-	3600	3	3600	-	-	-	-
R112-100	1176	-	3600	3	3600	-	-	-	-
RC101-100	300	26407	5.27	1	5.17	1	0	33	0.01
RC102-100	715	24808	314.97	2	191.05	3	0	82	2.88
RC103-100	1125	22632	3600	2	3290.51	29	0.3	255	154.33
RC104-100	1546	[20858, 21231]	3600	3	3600	-	-	-	-
RC105-100	543	25427	53.09	1	52.87	1	0	85	1.61
RC106-100	600	23770	892.01	2	764.98	15	0.38	240	181.92
RC107-100	882	22054	1303.35	2	1024.62	13	0.1	104	18.29
RC108-100	1123	-	3600	4	3336.07	-	-	-	-

B.3. MD-VRPTW

Table 13: Detailed results for 100 customer instances for the MD-VRPTW

Instance	Value	Time	Iter	TimeL	NodesL	%rGapL	R1C-L	CutSepL
C101-100	98273	1.89	1	1.89	1	0	0	0.01
C102-100	98273	24.49	2	12.82	3	0	0	1.1
C103-100	98273	22.09	1	20.37	1	0	3	0.66
C104-100	98263	362.0	2	221.8	3	0	15	4.47
C105-100	98273	4.31	1	2.65	1	0	0	0.03
C106-100	98273	4.5	1	2.87	1	0	0	0.02
C107-100	98273	5.2	1	3.51	1	0	0	0.04
C108-100	98273	6.66	1	4.95	1	0	0	0.05
C109-100	98273	13.45	1	11.72	1	0	9	0.12
R101-100	28939	16.27	2	7.35	3	0.04	14	3.42
R102-100	26082	6.53	1	4.95	1	0	7	0.43
R103-100	22673	51.4	2	38.6	3	0.13	37	5.99
R104-100	19778	1117.08	2	887.14	7	0.14	248	77
R105-100	23758	13.86	1	12.23	1	0	65	5.28
R106-100	22494	107.21	2	70.14	3	0.25	86	7.62
R107-100	20715	49.41	1	47.73	1	0	78	4.11
R108-100	19321	993.99	2	741.81	3	0.03	238	0.55
R109-100	21469	138.18	2	96.03	3	0.04	161	19.4
R110-100	20680	75.69	1	74.07	1	0	157	6.92
R111-100	20487	552.72	2	414.67	3	0.12	172	13.53
R112-100	19486	2549.37	2	2304.03	13	0.14	259	10.47
RC101-100	26505	7.4	1	5.84	1	0	51	0.43
RC102-100	24895	55.03	1	53.34	1	0	146	6.57
RC103-100	22687	1465.75	2	1229.96	11	0.13	244	29.87
RC104-100	-	3600	4	3481.44	25	∞	80	20.6
RC105-100	25427	25.47	1	23.87	1	0	69	0.98
RC106-100	23770	1608.97	2	1470.12	29	0.36	339	300.42
RC107-100	22078	820.35	2	608.19	7	0.1	132	12.19
RC108-100	-	3600	4	3147.62	11	∞	90	0.11