An Enhanced Branch and Price Algorithm for the Time-Dependent Vehicle Routing Problem with Time Window[*]

Gonzalo Lera-Romero [†] Juan José Miranda Bront [‡] Francisco J. Soulignac [§]

gleraromero@dc.uba.ar, jmiranda@utdt.edu, francisco.soulignac@unq.edu.ar

Abstract

In this paper we implement a branch and price (BP) algorithm for a time dependent vehicle routing problem with time windows (TDVRPTW) in which the goal is to minimize the total route duration (DM-TDVRPTW). The travel time between two customers depends on the departure time and, thus, it need not remain fixed along the planning horizon. We propose several improvements to the exact labeling algorithm by Dabia et al. (Branch and price for the time-dependent vehicle routing problem with time windows, *Transportation Science* 47(3):380–396, 2013) for solving the pricing problem, while we provide a tailored implementation for the dominance tests that relies on efficient data structures for storing the enumerated labels. Computational results show that the proposed techniques are effective for accelerating the column generation step. The obtained BP algorithm is able to solve benchmark instances with up to 100 customers, being able to solve all the instances with 25 customers. Furthermore, heuristic adaptations are able to find good quality solutions in reasonable computing times, showing its potential to be applied in practice.

Keywords: vehicle routing problem; time windows; time dependent travel times; branch and price; dynamic programming

1 Introduction

The classical *vehicle routing problem* (VRP) and most of its variants assume that travel times (or distances, or costs) are time independent, meaning that they remain fixed along the planning horizon. The problem is much more complex in practice since the operations are heavily affected by congestion, specially in urban areas. As suggested by Savelsbergh and Van Woensel (2016), incorporating this aspect at a tactical and operational level results in distribution plans that are closer to real-world operations, particularly for last mile logistics problems in large cities.

The *time-dependent vehicle routing problem with time windows* (TDVRPTW) addresses a key aspect regarding the impact of congestion in terms of the feasibility and efficiency of a route, in particular related to the time in which customers must be visited. In this paper we study the following variant proposed by Dabia et al. (2013). Consider a transportation network defined over a digraph $D = (V, A)$ with $V = \{0, \dots, n+1\}$. Vertices in $V_c = \{1, \dots, n\}$ represent *customers*, whereas 0 and $n+1$ represent the *depot* and indicate the beginning and end of every

[*]This research was partially funded by FONCyT grants PICT-2015-2419 (Soulignac) and PICT-2016-2677 (Lera-Romero and Miranda Bront) from the Ministry of Science, Argentina.

[†]Instituto de Investigación en Ciencias de la Computación (ICC) CONICET, Universidad de Buenos Aires, Buenos Aires, Argentina.

[‡]Universidad Torcuato Di Tella, Buenos Aires, Argentina and CONICET.

[§]CONICET and Departamento de Ciencia y Tecnología, Universidad Nacional de Quilmes, Bernal, Argentina.

route, respectively. There is a fixed (continuous) *planning horizon* $[0, T]$ in which vehicles are allowed to move along the network. Each vertex $i \in V$ has an associated *time window* $[a(i), b(i)]$ that represents the time interval in which customer i must be visited, a *service time* $s(i)$, and a *demand* $q(i)$ to be delivered. For simplicity, we assume $q(v) = s(v) = a(v) = 0$ and $b(v) = T$ for $v \in \{0, n+1\}$. Operations are carried out by an unlimited fleet of homogeneous vehicles, each having a limited *capacity* Q. As proposed by Ichoua et al. (2003), the impact of congestion is captured by considering that the time required to travel from vertex i to vertex j, when departing from i at time t, is given by a *travel time* function $\tau_{ij}(t)$ that satisfies the first-in first-out (FIFO) property (see Section 2.1). A vehicle is allowed to delay its departure in order to reduce the duration of its route. Under this setting, the *duration minimizing TDVRPTW* (DM-TDVRPTW) consists in finding a set of feasible routes minimizing the total duration.

The most effective exact algorithms for VRPs (and TDVRPs as well) rely on decomposition techniques and column generation. The pricing problem is a variant of the strongly $\mathcal{NP}$-hard *elementary shortest path problem with resource constraints* (ESPPRC), and is usually tackled with labeling algorithms (Feillet et al. 2004). Several techniques were proposed to speed up ESPPRC algorithms, including bi-directional search (Righini and Salani 2006) and its asymmetric variant (Tilk et al. 2017), the decremental state space relaxation (Righini and Salani 2008), the use of ng-routes (Baldacci et al. 2011, Martinelli et al. 2014), and the k-cycle elimination (Irnich and Villeneuve 2006). In particular, we highlight the approaches in Ioachim et al. (1998) and Irnich and Villeneuve (2006), where implementation decisions to accelerate the dominance tests are discussed. Recently, a state-of-the-art branch-cut and price (BCP) algorithm for the VRPTW including several of these enhancements was proposed in Pecin et al. (2017).

Distribution problems dealing with congestion and traffic information have caught the attention of the research community in the last few years. Gendreau et al. (2015) provide a recent and updated survey on TDVRPs; for other variants we refer to Toth and Vigo (2014). Dabia et al. (2013) propose a BP algorithm for DM-TDVRPTW. To tackle the pricing problem, which stands for a *time dependent ESPPRC* (TDESPPRC), they develop a tailored labeling algorithm that is able to solve 63% of the instances with 25 customers and 15% of the instances with 100 customers. In a follow-up paper, Sun et al. (2018) isolate the pricing problem for a DM-TDVRPTW with pickups and deliveries and consider a time dependent tour problem with capacity, time windows, and precedence constraints. Spliet et al. (2018) consider the time window assignment with time dependent travel times, for which they developed a BCP algorithm that is able to solve instances with up to 25 customers and three different demand scenarios. Another variant by Huang et al. (2017) studies the impact of the so-called *path flexibility*, where more than one physical path can join two customers. The paper shows the benefits of this problem by evaluating instances obtained from the road network in Beijing. Finally, we point out that some algorithms use labels to encode not only the path, but also all the possible starting times. This is the case for the VRP with soft time windows studied by Liberatore et al. (2011) and for the multi-trip VRP problem studied by Hernández et al. (2016).

1.1 Our Contributions

We develop a new BP algorithm for the DM-TDVRPTW that solves the pricing problem with a tailored labeling algorithm. The major contributions of the paper are the following. First, we provide a detailed implementation that reduces the computational effort at several critical parts of the labeling algorithm for the TDESPPRC. These enhancements are complementary to the previous developments on the DM-TDVRPTW and related problems, and thus could lead to further improvements in the corresponding algorithms. Second, we improve the labeling algorithm by: incorporating the asymmetric heuristic by Tilk et al. (2017) to speed up the bidirectional search; extending the definition of dominance given by Dabia et al. (2013) to consider that a label can be *partially* dominated; and a new rule to ignore labels that cannot be part of the optimal solution in the master problem. Third, we re-design the column generation algorithm by including different labeling-based heuristics, as well as by adding multiple columns with negative reduced cost per iteration, following an early stop criterion. Fourth, we evaluate the efficiency of the algorithm in different contexts using the benchmark instances proposed by Dabia et al. (2013). The results show that the approach is highly effective when compared to those existing in the literature, providing new benchmarks for the DM-TDVRPTW. Finally, we release the source code of the algorithm to ease the reproducibility of our results and to provide a baseline for future studies on the DM-TDVRPTW and related problems.

The rest of the paper is organized as follows. The set partitioning model for the DM-TDVRPTW, as well as the remaining notation and definitions used throughout the paper, is given in Section 2. Section 3 reviews the main ideas behind the labeling algorithm proposed by Dabia et al. (2013). Our enhacements to this algorithm appear in Section 4, whereas Section 5 describes the main characteristics of the BP algorithm. The implementation details is covered in Section 6. Section 7 reports extensive computational results and, finally, Section 8 presents the conclusions and future research directions.

2 Mathematical Formulation

This section presents the mathematical model for the DM-TDVRPTW. We follow the notation in Dabia et al. (2013), with some convenient adaptation. For a bounded set I, sometimes we write I^- and I^+ as shortcuts for $\min(I)$ and $\max(I)$, respectively.

2.1 Modelling Time-Dependent Travel Times

As described in Section 1, the time required to travel an arc (i,j), when i is departed from at time t, is given by a *travel time* function $\tau_{ij}(t)$. Formally, τ_{ij} is a continuous function with domain $[0,T]$ that is characterized by a set of *breakpoints* $0 = \beta_0 < \ldots < \beta_M = T$ such that $\tau_{ij} > 0$ is linear within $[\beta_i, \beta_{i+1}]$ for $0 \leq i < M$ (Figure 1). Furthermore, $t + \tau_{ij}(t) < t' + \tau_{ij}(t')$ for every $0 \leq t < t' \leq T$, i.e., τ_{ij} satisfies the *FIFO property*. We remark that the family of travel time functions need not satisfy the triangle inequality (Dabia et al. 2013). Thus, $\tau_{ik}(t)$ could be greater than $\tau_{ij}(t) + \tau_{jk}(\tau_{ij}(t))$.

If v is departed from at time t when traversing a path $p = (v = v_1, \ldots, v_k = w)$, then the earliest time at which the service at w is completed is given by the *ready time* function $\delta_p(t)$; see Figure 1. To recursively compute $\delta_p(t) = \delta_p(t,k)$, let $\delta_p(t,1) = t$ and, for $1 < i \leq k$,

$$\delta_p(t,i) = s(v_i) + \max\{a(v_i), \delta_p(t,i-1) + \tau_{v_{i-1}v_i}(\delta_p(t,i-1))\}. \quad (1)$$

The *duration* of p when leaving v at time t is $\Delta_p^v(t) = \delta_p(t) - t$. Since τ_{ij} is continuous, piecewise linear, and satisfies the FIFO property, it follows that δ_p is continuous, piecewise linear, and non-decreasing. Moreover, there exists $t_0 \in \mathbb{R}$ such that δ_p is constant before t_0 and increasing after t_0, thus Δ_p^v is decreasing before t_0. This implies that a vehicle traversing p should not leave v at time $t < t_0$ if it can be dispatched at time $t' \in (t, t_0]$.

Because of the time window constraints, it is not possible to leave v at any given time. Say that $t \in \mathbb{R}$ is a *feasible (departing) time* for p when $\delta_p(i,t) \leq b(v_i) + s(v_i)$ for every $1 \leq i \leq k$. Since δ_p is non-decreasing, the set of feasible times is a (possibly empty) range $\mathrm{dom}(p) = [a(v) + s(v), t_1]$. Consequently, the *minimum duration* (or *cost*) $c_p = \min\{\Delta_p^v(t) \mid t \in \mathrm{dom}(p)\}$ of p is equal to $\Delta_p^v(t)$ for some $t \in \mathrm{dw}(p)$, where $\mathrm{dw}(p) = \mathrm{dom}(p) \cap [\min(t_0, t_1), \infty]$ is the

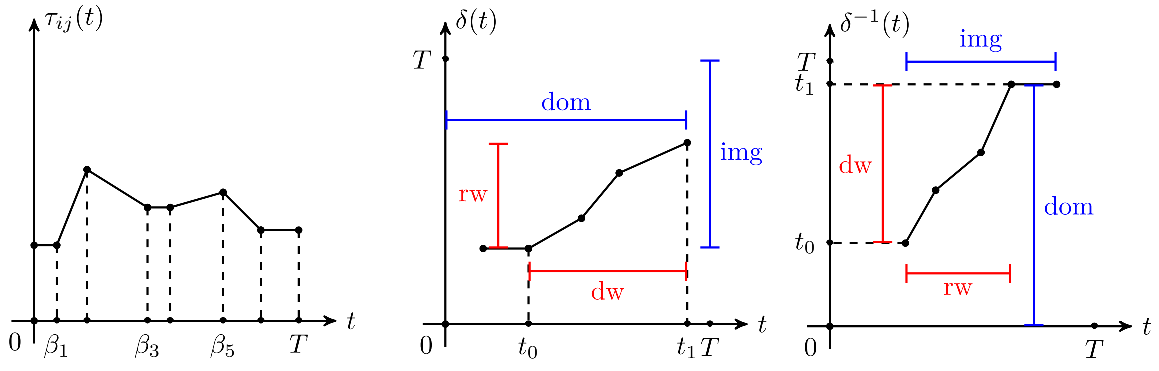

FIGURE 1. *Examples of τ, δ, and δ^{-1} with breakpoints and windows.*

departing window of p. The *ready window* $\text{rw}(p) = \{\delta_p(t) \mid t \in \text{dw}(p)\}$ of p is also a range $[r, d]$ because δ_p is continuous. Furthermore, as it is increasing, δ_p is a bijection from $\text{dw}(p)$ to $\text{rw}(p)$, thus δ_p^{-1} is well defined in $\text{rw}(p)$. It can be immediately noted that $\delta_p^{-1}(t)$ is the latest departing time from v such that the service at w is completed at t. Even though it increases the duration of p, the service of w can be deferred until its deadline. Thus, following Dabia et al. (2013), we write $\text{img}(p) = [\text{rw}^-(p), b(w)+s(w)]$, and we define $\delta_p^{-1}(t) = \delta_p^{-1}(\text{rw}^+(p))$ for $t \in \text{img}(p) \setminus \text{rw}(p)$. Each $t \in \text{img}(p)$ is called a *feasible ready time*, while $\Delta_p^w(t) = t - \delta_p^{-1}(t)$ measures the *duration* of being ready at time t. Figure 1 depicts δ^{-1} and the corresponding windows.

It is useful to generalize the notion of duration to all the vertices of p. Let $\lambda = (v_1, \dots, v_i)$, $\rho = (v_i, \dots, v_k)$, and $z = v_i$. For $t \in \text{img}(\lambda) \cap \text{dom}(\rho)$, the *duration* of p when the service at z is completed at t is $\Delta_p^z(t) = \Delta_\lambda^z(t) + \Delta_\rho^z(t)$. Note that $\Delta_p^z(t) = \Delta_p^v(\delta_\lambda^{-1}(t))$ when $t \in \text{rw}(\lambda)$ or $t \notin \text{dw}(\rho)$. However, if $t' = \text{rw}^+(\lambda) < t$ and $\text{dw}^-(\rho) < t$, then $t' = \delta_\lambda(\delta_\lambda^{-1}(t)) < t = \delta_\rho^{-1}(\delta_\rho(t))$, hence $\delta_\rho(t') < \delta_\rho(t)$ and, consequently, $\Delta_p^v(\delta_\lambda^{-1}(t)) < \Delta_p^z(t)$. Intuitively, departing at t from z forces an unneeded wait. (See function Φ in Dabia et al. 2013 for an alternative discussion.) Similarly, $\Delta_p^z(t) \geq \Delta_p^w(\delta_\rho(t))$, and equality holds if and only if $t \in \text{dw}(\rho)$ or $t \notin \text{rw}(\lambda)$.

As in the previous paragraph, joining two paths with a common vertex is often required. Thus, for any pair of paths $\lambda = (v_1, \dots, v_i)$ and $\rho = (v_i, \dots, v_k)$, we write $\lambda \oplus \rho = (v_1, \dots, v_k)$.

2.2 Set Partitioning Formulation and the Pricing Problem

Say that a path $p = (v_1, \dots, v_k)$ is *feasible* (for DM-TDVRPTW) when p is simple, $q(p) = \sum_{i=1}^k q(v_i) \leq Q$, and $\text{dw}(p) \neq \emptyset$. If $v_1 = 0$ and $v_k = n+1$, then p is a *route*. Let Ω be the set of all the feasible routes and $a_{vr} \in \{0, 1\}$ be an indicator of the number of times vertex v is visited by route $r \in \Omega$. Define binary variables y_r indicating if route $r \in \Omega$ is selected in the optimal solution. Then, the set partitioning formulation for the DM-TDVRPTW is:

$$\min \sum_{r\in\Omega} c_r y_r \tag{2}$$

$$\text{s.t.} \sum_{r\in\Omega} a_{vr} y_r = 1, v \in V_c \tag{3}$$

$$y_r \in \{0,1\}, r \in \Omega \tag{4}$$

The objective function (2) minimizes the total duration of the distribution plan. Constraints (3) impose that each customer is visited by exactly one route. Finally, constraints (4) establish the domain of the decision variables.

Given that Ω has an exponential number of routes, the traditional approach is to solve the LP relaxation using column generation. The algorithm starts with a *restricted master problem*

(RMP), which is the original formulation but containing only a subset of the variables. Then, the RMP is iteratively solved and columns with a negative reduced cost are added until the optimal solution of the LP relaxation is found.

At each iteration of the algorithm, the *pricing problem* aims to identify routes (columns) $r \in \Omega$ with negative reduced cost $\bar{c}_r$, if any exists. Let π_v be the dual variable associated with constraint (3) for vertex $v \in V_c$. Let $\bar{c}_r = c_r - \sum_{v \in r} \pi_v$ be the reduced cost, which balances the route cost with the profits collected at the visited vertices. The pricing problem for the formulation (2)–(4) becomes a TDESPPRC defined by $\min\{\bar{c}_r \mid r \in \Omega\}$.

Labeling algorithms are among the most effective techniques for tackling variants of the ESPPRC, specially under the presence of time windows. The standard strategy in most BP algorithms is to identify only one route at each iteration, in general one with the lowest reduced cost. This approach is followed by Dabia et al. (2013). Instead, we apply two well known techniques to accelerate the column generation algorithm. We return many columns with a negative reduced cost, when possible, and we consider a variation of the *forced early stop* (see, e.g., Desaulniers et al. 2005). The details are discussed in Sections 5 and 6.

3 Labeling Algorithm Revisited

This section reviews the labeling algorithm by Dabia et al. (2013) for the pricing problem. The algorithm has three main steps that implement a bidirectional search (Righini and Salani 2006). In the first step, forward labels are iteratively extended from the depot 0 to its successors while $\mathrm{rw}^-(\cdot) < t_m$ for a given $0 < t_m < T$. Then, backward labels are extended from the depot $n+1$ to its predecessors until $\mathrm{dw}^+(\cdot) \leq t_m$. Finally, forward and backward labels are merged to build complete routes. Except for some key differences, the backward step is mostly ignored through the article as it is analogous to the forward step.

3.1 Forward Labeling

Each *forward label* L is a tuple that represents a path $p(L) = (0 = v_1, \dots, v_k)$, whose components are $v(L) = v_k$, $\pi(L) = \sum_{v \in p(L)} \pi_v$, $q(L) = q(p(L))$, $\delta_L = \delta_{p(L)}$, a pointer $\mathrm{prev}(L)$ to the label representing $(v_1, \dots, v_{k-1})$, and a set $S(L) \supseteq p(L)$ of unreachable vertices (see below). For $\bullet \in \{\mathrm{dom}, \mathrm{img}, \mathrm{dw}, \mathrm{rw}, \Delta\}$, let $\bullet(L) = \bullet(p(L))$, and say that L is *feasible* when $p(L)$ is feasible. As discussed in Section 2, δ_L is restricted to $\mathrm{dw}(L) \to \mathrm{rw}(L)$ because Δ^0_L is decreasing in $\mathrm{dom}(L) \setminus \mathrm{dw}(L)$.

Beginning with a label representing the path (0), the algorithm iteratively processes each label L until no unprocessed labels remain. If L is infeasible, $\mathrm{rw}^-(L) > t_m$, or L fails to pass the dominance tests (see below), then L is discarded. Otherwise, L is *extended* by creating a label M that represents $p(L) + v$, for each arc $(v(L), v)$ such that $v \notin S(L)$.

Dominance tests are performed to reduce the number of labels created by the algorithm. For $t \in \mathrm{img}(L)$, let $E(L, t)$ be the set of paths p from v such that $t \in \mathrm{dom}(p)$ and $p(L) \oplus p$ is a feasible route. Say that L is *globally dominated* by a label M *at time* t when $E(L, t) \subseteq E(M, t)$, and $\Delta^v_{p(L) \oplus p}(t) - \pi(L) \geq \Delta^v_{p(M) \oplus p}(t) - \pi(M)$ for every $p \in E(L, t)$. That is, L is globally dominated by M at the feasible ready time t of L when, for every feasible extension p of $p(L)$, p is also a feasible extension of $p(M)$ and the route $p(M) \oplus p$ has a lower (time-dependent) reduced cost than the route $p(L) \oplus p$ when $v(L)$ is ready at time t. Thus, if $t \in \mathrm{rw}(L)$, then we can safely discard t from $\mathrm{rw}(L)$ and $\delta^{-1}(t)$ from $\mathrm{dw}(L)$.

Unfortunately, testing whether L is globally dominated by M can require the computation of all the extensions of L and M. Instead, efficient dominance tests that do not require the extension of L are performed. In this article, we say that M *dominates* L at $t \in \mathrm{img}(L)$

($M \preceq_t L$) when L and M satisfy Proposition 1 below. Dabia et al. (2013) propose to discard L when $L \preceq_t M$ for every $t \in \mathrm{img}(L)$. We refer to this criteria as *full dominance.*

Proposition 1 (Dabia et al. 2013, Proposition 2)**.** *A label L is globally dominated by a label M at $t \in \mathrm{img}(L)$ if $t \in \mathrm{img}(M)$, $v(M) = v(L)$, $S(M) \subseteq S(L)$, $q(M) \leq q(L)$, and $\Delta_M^{v(M)}(t) - \pi(M) < \Delta_L^{v(L)}(t) - \pi(L)$.*

Dominance tests are strengthened when the set of unreachable vertices $S(L) \supseteq p(L)$ contains as many vertices as possible. This results in a faster algorithm, provided that $S(L)$ can be efficiently computed (Feillet et al. 2004). As travel times do not satisfy the triangle inequality, a vertex v could be reached via a diverted route even when it cannot be reached directly. Yet, as Dabia et al. (2013) noted, v is unreachable from $p(L)$ when $b(v) < t = \min\{t_w \mid w \in V_c \setminus p(L) \text{ and } b(w) \geq t_w\}$ for $t_w = \mathrm{rw}^-(L) + \tau_{v(L)w}(\mathrm{rw}^-(L))$. As this test is quick, Dabia et al. (2013) define $S(L) = \{v \in V_c \mid v \in p(L),\ q(v) + q(L) > Q, \text{ or } b(v) < t\}$.

3.2 Backward Labeling

Each *backward label* L represents a path $p(L) = (v_1, \ldots, v_k = n + 1)$; its components are $v(L) = v_1$, $\pi(L)$, $q(L)$, $\delta_L \colon \mathrm{dw}(L) \to \mathrm{rw}(L)$, a set $S(L) \supseteq p(L)$ of unreachable vertices, and a pointer $\mathrm{next}(L)$ to the label representing $(v_2, \ldots, v_k)$. As in the forward case, L is discarded if it is infeasible or it fails to pass a dominance test. This time, however, L is not discarded when $\mathrm{dw}^+(L) < t_m$; instead, it is simply not extended.

3.3 Bidirectional Labeling

A *route label* L represents a route $p(L)$, and it is described by $\pi(L)$, $q(L)$, δ_L, and $p(L)$ as encoded by the next and prev pointers. To solve the pricing problem, routes labels with negative reduced cost are searched for. Disregarding domination, route labels with $\mathrm{rw}^-(\cdot) \leq t_m$ (resp. $\mathrm{dw}^+(\cdot) > t_m$) correspond to labels generated in the forward (resp. backward) step. The family $\mathcal{R}$ of the remaining route labels is found by joining labels F and B with $v(F) = v(B)$ generated at the forward and backward steps, respectively.

The *merge* of F and B is the route label $L = F \oplus B$ that represents the path $p(L) = p(F) \oplus p(B)$; let $B' = \mathrm{next}(B)$. By definition, $q(L) = q(F) + q(B')$, $\pi(L) = \pi(F) + \pi(B')$, and $\delta_L(t) = \delta_B(\delta_F(t))$, where δ_L is defined over the domain $\mathrm{dom}(L) = \{\delta^{-1}(t) \mid t \in \mathrm{img}(F) \cap \mathrm{dom}(B)\}$. Thus, L is feasible if and only if 1. $q(F) + q(B') \leq Q$ and 2. $\mathrm{img}(F) \cap \mathrm{dom}(B) \neq \emptyset$. In the merge step, every combination of F and B satisfying 1. and 2. is evaluated to compute the feasible label $F \oplus B$. By (Dabia et al. 2013, Proposition 4), every non-dominated label $L \in \mathcal{R}$ is eventually computed. In fact, many combinations of F and B could yield L; Section 6 examines two strategies to avoid generating many copies of L.

4 Enhanced Labeling Algorithm

This section discusses three enhancements to the labeling algorithm.

4.1 The Partial Dominance Criteria

The algorithm by Dabia et al. (2013) follows a full dominance criteria in which a label L is discarded only if it is fully dominated by another label M. If L is not discarded, then δ_L is maintained for all the departing and ready windows. Yet, $t \in \mathrm{rw}(L)$ can be discarded when $M \preceq_t L$. To implement this criteria, it suffices to consider that $\mathrm{dw}(L)$ and $\mathrm{rw}(L)$ are not intervals, but a families of disjoint intervals. Then, when L is processed, only those points in $\mathrm{rw}(L)$ are considered for extension and dominance tests. Ultimately, L is discarded when $\mathrm{rw}(L) = \emptyset$. We refer to this new criteria as *partial dominance*.

Partial dominance is not a new concept, and it was already applied to SPPRC problems with resource-dependent costs (e.g., Ioachim et al. 1998, Luo et al. 2017, Spliet et al. 2018). In fact, many dynamic programming algorithms keep only the best known solution for each state. In our case, this means storing the label of minimum cost among those with demand q and unreachable vertices S that are ready at vertex v at time t. Without a partial dominance strategy, it is inevitable to keep many labels for some state (v, t, q, S).

An issue to take into account when partial dominance is applied is how to merge labels. Recall that a forward label F can be merged with a backward label B, to obtain the route label L, only if $\mathrm{dom}(L) = \{\delta_F^{-1}(t) \mid t \in \mathrm{img}(F) \cap \mathrm{dom}(B)\}$ is nonempty. In this case, the cost of $p(L)$ is $c(L) = \Delta_L^0(t^*)$ for some $t^* \in \mathrm{dom}(L)$. Yet, as some points were discarded, only $\mathrm{rw}(F) \subseteq \mathrm{img}(F)$ and $\mathrm{dw}(B) \subseteq \mathrm{dom}(B)$ are readily available. Consequently, only the upper bound $c = \min\{\Delta_{p(L)}^0(\delta_F^{-1}(t)) \mid t \in (\mathrm{rw}(F) \cup \mathrm{dw}(B)) \cap \mathrm{img}(F) \cap \mathrm{dom}(B)\}$ of $c(L)$ can be efficiently computed (in a time linear with respect to the size of F and B). As finding $c(L)$ is more expensive, we only compute it when we can assure that $c(L) - \pi(L) < 0$. That is, if $c \geq \pi(L)$, then L is discarded; otherwise, $c_{p(L)} = c(L)$ is computed when $p(L)$ is inserted to the master problem. We remark that L is discarded only if it is dominated.

Thus far, we eluded the question of which dominance test to apply. There are two main strategies, regardless of the criteria used for domination. On the one hand, *label setting* algorithms keep the family $\mathcal{L}$ of non-discarded labels. Then, in a *setting* step, each processed label L is tested for domination by comparing it to every label in $\mathcal{L}$, updating $\mathrm{rw}(L)$ as a by-product. If $\mathrm{rw}(L) \neq \emptyset$, then L is inserted to $\mathcal{L}$ and it remains unchanged to the end of the algorithm. On the other hand, *label correcting* algorithms add an extra *correcting* step on which each $M \in \mathcal{L}$ is compared to L to reduce $\mathrm{rw}(M)$. Label setting is well suited when labels are generated in an order that guarantees that L has no effects in M, for $M \in \mathcal{L}$. Nevertheless, a label setting algorithm can be applied even if there are no guarantees about domination. Similarly, label correcting algorithms with partial dominance can be enough to prove that at most one label is stored for every state. Yet, dominance test could be heuristically skipped, breaking the one label per state condition (Section 6).

In this work, we evaluate both dominance criteria within a label setting algorithm in which forward and backward labels are processed according to rw^- and dw^+, respectively. As $\mathcal{L}$ grows exponentially, the *one processing rule* is obeyed: 1. no two labels represent the same path, and 2. each label is processed once. Note that the inclusion of labels into $\mathcal{L}$ need not follow the same pattern as the processing, as rw^- can change. Finally, we also implement a label correcting algorithm with a partial dominance criteria.

4.2 Asymmetric Bidirectional Labeling

The pricing algorithm in Section 3 uses the same time limit t_m for the forward and backward phases. Yet, the algorithm is correct as long as the backward time limit t_b is not greater than the forward time limit t_f. In this work we apply an heuristic by Tilk et al. (2017) to exploit the asymmetry of an instance. The idea is to fix an initial value $t_b \leq t_f$ before running the algorithm; say $t_b = 0$ and $t_f = T$. Then, t_f and $T - t_b$ are decreased while forward and backward labels are alternatively processed, keeping the invariant that $t_b \leq t_f$. Eventually, an iteration at which $t_b \approx t_f$ cannot be further updated is reached. In particular, when t_b and t_f are initialized to t_m, the algorithm is equivalent to that in Section 3.

Recall that a forward label F is processed only if $\mathrm{rw}^-(F) \leq t_f$. However, F should be

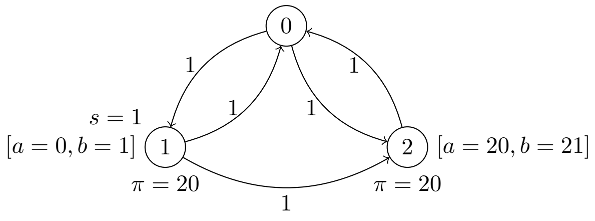

FIGURE 2. *The optimal route* $(0, 1, 2, 0)$ *with cost* -19 *is missed because* $(0, 1)$ *is not extended to* $(0, 1, 2)$. *Spending* 2 *units of time to take the arcs* $(1, 0)$ *and* $(0, 2)$ *is cheaper than spending* 18 *units of time to take* $(1, 2)$.

discarded if $\mathrm{rw}^-(F)$ surpasses the final value $t^*_b \approx t^*_f$. Therefore, after F is processed, we set $t_b = \max(t_b, \mathrm{rw}^-(F)) \leq t_f$. Recall also that, as a by-product of the dominance tests, F is updated into a label F' with $\mathrm{rw}^-(F') \geq \mathrm{rw}^-(F)$. By the one processing rule, it could be the case that $\mathrm{rw}^-(F') > t^*_f$, meaning that time was spent in a label that will be ultimately discarded. Different approaches exists, e.g. Spliet et al. (2018). The update of t_f is similar, i.e., we set $t_f = \min(t_f, \mathrm{dw}^+(B))$ for each backward label B that is extended.

The efficiency of the algorithm depends on the values $t^*_b \approx t^*_f$ finally computed, which are determined by the order in which labels are processed. To take the smallest step at each iteration, forward and backward labels are processed according to rw^- and dw^+, respectively, whereas t_b and t_f are alternatively updated by processing labels of one kind are until one is not fully discarded, before considering labels of the other kind.

4.3 Triangle inequality through the depot

As noted by (Dabia et al. 2013), the input digraph D can be pre-processed by removing those infeasible arcs (i, j) such that either $a(i) + s(i) + \tau_{ij}(a(i) + s(i)) > b(j)$ or $q(i) + q(j) \geq Q$. Similarly, (i, j) can be removed when passing through the depot is cheaper than waiting at j. Formally, let $\tau^0_{ij}(t) = \tau_{i(n+1)}(t) + \tau_{0j}(t + \tau_{i(n+1)}(t))$ be the *travel time* from i to j *through the depot*. If $\tau^0_{ij}(b(i) + s(i)) \leq a(j) - b(i) - s(i)$, then (i, j) is removed from D because $c(r) \geq c(r_1) + c(r_2)$ for any feasible route r containing (i, j), where r_1 and r_2 are the feasible routes obtained from r by replacing (i, j) with the arcs $(i, n + 1)$ and $(0, j)$.

A similar analysis holds for the labels of the pricing algorithm. Suppose L is a forward label with $v(L) = i$ and let M be the label representing $p(L) + j$ for $j \notin S(L)$. If $\tau_{ij}(\mathrm{rw}^+(L)) < a(j)$, then $\mathrm{rw}(M) = \{a(j)\}$ and, hence, waiting is mandatory. Again, it is cheaper to wait at the depot, thus L is not extended to M when $\tau^0_{ij}(\mathrm{rw}^+(L)) \leq a(j)$. This is correct because $\bar{c}_{p(M)\oplus p} > \bar{c}_{p(L)+(n+1)} + \bar{c}_{0+p}$ for any path p.

Incidentally, the optimal solution to the pricing problem could be missed because of the previous rule (Figure 2). This is not important when solving the master problem, as (2)–(4) can be strengthened by removing those routes with high waiting times from Ω.

5 Branch and Price Algorithm

This section describes our BP algorithm by enumerating its differences to the one proposed by Dabia et al. (2013).

5.1 Column Generation Algorithm

To solve the pricing problem, the column generation algorithm considers the exact labeling algorithm described before as well as a heuristic adaptation (see below). To accelerate the convergence of the column generation, multiple columns with a negative reduced cost are added at each iteration. As described in Section 6.2, the pricing algorithm generates these columns in an iterative fashion by attempting to merge labels before t_b and t_f meet. To keep the size of the RMP moderate, the pricing algorithm is terminated when $L_{\max}$ columns have been identified. Aligned with our preliminary experiments, these ideas have shown to be effective in other contexts (Desaulniers et al. 2005, Liberatore et al. 2011).

5.2 Labeling-Based Heuristics

A common approach for the pricing heuristic is to limit the number of labels enumerated by a labeling algorithm. Instead, we obtain multiple heuristics by removing, one at a time, the dominance constraints in Proposition 1. As a result, some labels that are not dominated in the original algorithm become dominated and, therefore, the overall number of labels to be considered is reduced. By design, these heuristics are slower than those obtained by limiting the number of labels enumerated. Yet, they yield better solutions and, as experimentally observed, require substantially less computing time than the exact algorithm.

5.3 Branching and Node Selection Strategy

A robust branching scheme based on the classical flow variables $x_{uv} = \sum_{r\in\Omega,(u,v)\in r} \bar{y}_r$, where $\bar{y}$ is the (fractional) optimal solution of the LP relaxation of (2)–(4), is adopted. The branching variable x_{uv} is selected using *strong branching*: (u, v) is the arc that maximizes the lowest of the estimates of the value of the LP relaxation of its descendants, among those k most fractional variables. The estimation consists in solving the LP from the parent by adjusting the bounds on the variables that become infeasible due to the branching. Finally, the next node to be processed is selected using the *best bound* rule.

6 Implementation Details

This section discusses those decisions of our implementation that have a significant impact on the efficiency of the pricing algorithm.

6.1 Data Structures and Hierarchical Storage of Labels

According to Section 4, each forward label L is updated by comparing it to every label M in the family $\mathcal{L}$ of non-discarded labels. If either $v(M) \neq v(L)$ or $q(M) > q(L)$, then M has no effects on L and it can be safely ignored. To implement this idea, $\mathcal{L}$ is kept sorted in such a way that no ignored label is ever traversed. The next two paragraphs discuss an additional filter, depending on whether a full or partial dominance criteria is applied.

Full dominance criteria. In this case, M can be ignored if $\Delta^v_M(\bar{t}) - \pi(M) > \Delta^v_L(\bar{t}) - \pi(L)$ for $\bar{t} \in \mathrm{rw}(L) \cap \mathrm{img}(M)$; here $v = v(L) = v(M)$. We take $\bar{t} = \mathrm{img}^+(L) = \mathrm{img}^+(M)$ even if $\bar{t} \notin \mathrm{rw}(L)$, because the order in $\mathcal{L}$ is fixed before L is known. Thus, if we define $\bar{c}_{\text{last}}(F) = -\,\mathrm{dw}^+(F) - \pi(F)$ for every label F, then we skip M when $\bar{c}_{\text{last}}(M) > \bar{c}_{\text{last}}(L)$ because

$$\begin{aligned}\bar{c}_{\text{last}}(M) = -\,\mathrm{dw}^+(M) - \pi(M) = -\delta^{-1}_M(\bar{t}) - \pi(M) = \Delta^v_M(\bar{t}) - \bar{t} - \pi(M) >\\ \bar{c}_{\text{last}}(L) = -\,\mathrm{dw}^+(L) - \pi(L) = -\delta^{-1}_L(\bar{t}) - \pi(L) = \Delta^v_L(\bar{t}) - \bar{t} - \pi(L).\end{aligned}$$

Incidentally, note that if $M \preceq_t L$ at $t = \mathrm{rw}^+(L)$, then $\bar{c}_{\text{last}}(M) \leq \bar{c}_{\text{last}}(L)$ as well.

Partial dominance criteria. Let $\bar{c}_{\max}(L) = \max\left\{\Delta_L^{v(L)}(t) \mid t \in \operatorname{rw}(L)\right\} - \pi(L)$ be the *worst reduced cost* of L and $\bar{c}_{\min}(M) = \min\left\{\Delta_M^{v(M)}(t) \mid t \in \operatorname{rw}(M)\right\} - \pi(M)$ be the *best reduced cost* of M. Clearly, M affects L only if $\bar{c}_{\max}(L) \geq \bar{c}_{\min}(M)$, thus M can be ignored otherwise.

Dominance criteria vs. ignorance rule. Call *ignore by last* and *ignore by min* to the rules defined in the previous paragraphs, respectively. While designing these rules, the main concern was that no label affecting L is ignored. This feature is not mandatory: a partial dominance criteria with an ignore by last rule is possible. Although more labels are potentially stored in $\mathcal{L}$, less of them are traversed when processing L. In fact, unreported preliminary experiments show that ignore by last outperforms ignore by min in some particular instances with few customers.

The data structure for $\mathcal{L}$. Let $\bar{c}(M) = \lceil \bar{c}_{\text{last}}(M) \rceil$ or $\bar{c}(M) = \lceil \bar{c}_{\max}(M) \rceil$ according to whether ignore by last or ignore by min is applied, respectively. To represent $\mathcal{L}$, a set $\mathcal{L}_{v,q,\bar{c}} = \{M \in \mathcal{L} \mid v(M) = v, q(M) = q \text{ and } \bar{c}(M) = \bar{c}\}$ is stored for each $v \in V$, each demand q, and each value of $\bar{c}$. These sets are stored in levels that, from the higher to the lower level, are accessed through the keys v, q, and $\bar{c}$. In turn, each level is ordered by its corresponding key, thus only an interval of each level is traversed when L is processed.

6.2 Merging Labels in Bidirectional Algorithms

Two problems arise in the merge step. On the one hand, those labels F and B such that $p(F) \oplus p(B)$ is a feasible route must be found, perhaps traversing pairs that do not yield feasible routes. Since the number of labels grows exponentially, the cost of traversing these pairs cannot be neglected. On the other hand, different combinations of F and B can yield L when merged. To avoid the cost of merging duplicates, the *half-way* strategy by Righini and Salani (2006) can be applied. That is, F and B are merged only if $|\operatorname{rw}^-(F) - \operatorname{dw}^+(B)|$ is minimized. Yet, traversing these combinations could require a time linear with respect to $|p(L)|$. This cost can also be avoided using the *last-edge* strategy described below.

The last-edge strategy. Let L be a feasible and non-dominated route label. If L is not generated by the forward step, then $L = F \oplus B$ for some discarded label F such that $\operatorname{rw}^-(F') < t_f$ for $F' = \operatorname{prev}(F)$. Since L is feasible, F was discarded because $t_b \leq t_f < \operatorname{rw}^-(F)$. Thus, as $\operatorname{rw}^-(F) \leq \operatorname{dw}^+(B)$, it follows that B is processed in the backward step and, hence, the backward label B' representing $v(F') + p(B)$ is generated.

In general, a forward label F and a backward label B are *mergeable* when $v(F) = v(B)$, $q(F) + q(B') \leq Q$, and the forward label F' representing $p(F) + v(B')$ is discarded, for $B' = \operatorname{next}(B)$. By definition, at most one mergeable pair yielding L exists for every route label L and, by the discussion above, one mergeable pair exists when L is feasible and not dominated. In the *last-edge* strategy only mergeable pairs are considered for merging.

To implement the last-edge strategy, the forward step keeps, for each arc (i, j) and demand q, the set $\mathcal{M}_{ij,q}$ of labels F such that $v(F) = i$, $q(F) = q$, and the label representing $p(F) + j$ is discarded. The merge step is implemented within the backward step: each label B with $q(B) + q \leq Q$ whose first arc is (i, j) is merged with each label in $\mathcal{M}_{ij,q}$.

Iterative merging. The forward step must be completed before labels are merged when the half-way or last-edge strategies are applied. This is against our approach of adding $L_{\max}$ columns to the restricted master problem as early as possible without running the pricing algorithm to completion. Yet, we still apply the above strategy when the pricing algorithm is expected to end before finding $L_{\max}$ columns.

The node of the BP tree being solved is either in an *opening* or *closing* state. The state is opening if and only if all the previous calls to the pricing algorithm found $L_{\max}$ columns without enumerating all the labels. When the node is closing, it is assumed that successive calls to the pricing algorithm will run to completion, thus the last-edge strategy is applied. When the node is opening, instead, every pair of forward and backward labels is merged. For the implementation, the set $\mathcal{M}_{ij,q}$ of forward labels with demand q that end with the arc (i, j) is kept. Then, each generated backward label B with $q(B) + q \leq Q$ and $v(B) = j$ is merged with every $F \in \mathcal{M}_{ij,q}$. A similar set containing the backward labels is maintained to combine each forward label that is processed.

Analogously to Desaulniers et al. (2008) for the DSSR, the high cost of iterative merging is mitigated by heuristically deferring the earliest moment in which labels are merged. At the i-th call to the pricing algorithm, the number k_i of forward labels generated is computed. Then, at the $(i + 1)$-th call with an opening state, a merge is attempted only if more than $(k_i + k_{i-1})/2$ forward labels were generated.

Ignoring labels to be merged. Let $\bar{c}(L) = c(p(L)) - \pi(L)$ for any label L. Clearly, $\bar{c}(F) + \bar{c}(B) + \pi_{v(B)} \leq \bar{c}_{p(F)\oplus p(B)}$ for every forward label F and backward label B with $v(F) = v(B)$. Hence F and B need not be merged when $\bar{c}(F) \geq -\bar{c}(B) - \pi_{v(F)}$, because $\bar{c}(p(F) \oplus p(B)) \geq 0$. Regardless of the state of the current BP node, it suffices to keep $\mathcal{M}_{ij,q}$ sorted by $\bar{c}$ to ignore all such pairs F and B.

7 Computational Results

This section reports the experiments conducted to assess the effectiveness of the different components of the BP algorithm. The code is implemented in `C++` using CPLEX Studio 12.8 as the LP and ILP solver. The experiments were executed on an Intel(R) Core(TM) i5-4570 CPU@3.20GHz workstation with 16GB of RAM using a time limit of 2 hours. The results are reported in an aggregated fashion; see Appendix for detailed information.

Experimentation is carried out on the set of instances proposed by Dabia et al. (2013), that incorporate time dependency to the Solomon benchmark instances for the VRPTW. This set contains 56 instances for each $n \in \{25, 50, 100\}$, with different settings regarding the location of the customers and the length of the time windows. The improvements provided by some of the components of the labeling algorithm are hard to make evident when incorporated within the BP algorithm. For this reason, we evaluate different variants of the labeling algorithm over 54 isolated pricing problems that arise during the execution of the BP algorithm on 18 of the aforementioned instances. To obtain a meaningful insight of the behavior of the algorithm, the pricing instances selected are of an intermediate difficulty and can be solved within the imposed time limit.

The Solomon instances are identified by its class (R, C, or RC), group (1 or 2), and number within the group. When required, n is indicated following an underscore. The pricing instances follow the same pattern, with the inclusion of an extra identifier (a, b, or c) to distinguish the pricing subproblem within the BP instance.

7.1 Impact of asymmetric labeling and partial dominance in the TDESPPRC

Table 1 compares four settings for the (exact) pricing algorithm. Each cell reports the average time required for solving the pricing instances, grouped by the number of customers n. The first

	Full		Partial	
n	Sym	Asym	Partial-LS	Partial-LC
25	144.19	18.32	**9.11**	15.27
50	552.16	204.68	**37.71**	63.41
100	327.64	258.53	**45.08**	71.74

Table 1. *Average times (in seconds) of symmetric and asymmetric exact labeling algorithms, on pricing instances.*

two columns correspond to a label setting algorithm with a full dominance criteria (Full) and an ignore by last strategy that applies a bidirectional search that is either symmetric (Sym) or asymetric (Asym). The other columns (Partial) correspond to either a label setting (Partial-LS) or a label-correcting (Partial-LC) algorithm with a partial dominance criteria and an ignore by min strategy that applies an asymmetric bidirectional search. In all cases, the algorithm is executed to completion applying the improvements of Section 4.3. Recall that, consequently, some optimal solutions may be (safely) discarded.

The first two columns show a significant reduction in the computing times when an asymmetric search is performed. This reduction is explained by the fact that the number of non-dominated labels enumerated in the forward and backward phases tend to be more balanced in the asymmetric version. A further speed-up is obtained when a partial dominance criteria is applied, because the number of non-dominated labels enumerated is greatly reduced. It is interesting to note that even though Partial-LC enumerates less non-dominated labels than Partial-LS, the additional cost of the correcting step hinders this improvement. The detailed results can be retrieved from Table 4 in the Appendix.

7.2 The BP algorithm

Table 2 summarizes the results obtained by the BP algorithm for all instances, grouped by n. According to the results in the previous section, an asymetric label setting algorithm with a partial dominance criteria and ignore by min strategy is applied to solve the pricing problem. Also, labeling-based heuristics with the same configuration are executed before the exact algorithm. Recall that the pricing algorithms are adapted to stop immediately after $L_{\max}$ columns with negative reduced cost are identified. Preliminary experiments were carried out to tune this parameter, considering $L_{\max} \in \{100, 300, 1000, 3000\}$ over a reduced set of instances (Table 11 of Appendix). Based on these results, we set $L_{\max} = 3000$.

The columns of Table 2 indicate the class of the instances (Class), the number of instances solved to optimality (Opt), and the average time consumed by BP algorithm (Time), restricted to those instances that were solved to optimality. As expected, the instances become harder as n increases. The algorithm is able to solve all instances with $n = 25$, 46 out of 56 with $n = 50$, and 20 for $n = 100$ within the time limit imposed.

We focus on $n = 50$ for a more detailed analysis (see Table 6). The common behavior is that the bound at the LP relaxation is in general tight, with gaps below 1%. The hardest instances appear to be those having a larger gap in the root node. We note that the LP relaxation cannot be solved in some instances and, in particular, the heuristics are still generating columns when reaching the time limit. We acknowledge that our labeling-based heuristics are computationally more demanding than those that limit the number of labels enumerated. We believe this aspect and the re-design of the column generation algorithm contribute significantly to the reductions in computing time of the algorithm. Finally, we remark that for an additional set of 15 unsolved instances with $n = 100$, the BP algorithm is able to solve the LP relaxation and, in many cases, it enumerates a large number of nodes within the branch and bound tree. These results

n		25		50		100	
Class	#Inst	Opt	Time	Opt	Time	Opt	Time
C1	9	9	46.79	9	481.92	8	403.68
R1	12	12	1.83	12	242.09	6	2098.60
RC1	8	8	3.19	6	1972.96	0	—
C2	8	8	22.69	6	928.98	4	1692.33
R2	11	11	16.88	8	1724.69	1	7004.37
RC2	8	8	221.29	5	1080.60	1	5808.87
Total:	56	56		46		20	

TABLE 2. *Summarized results of the BP algorithm for the Dabia et al. instances.*

open the possibility of increasing the number of instances solved by considering complementary techniques to reduce the number of enumerated nodes, as the inclusion of valid inequalities to the master problem.

7.3 Heuristic adaptations

Regarding the potential utilization of the framework in practice, we report the results of two heuristics called BPH-CG and BPH-Heur. The former solves the LP relaxation of (2)–(4) with column generation, and then it calls a general purpose ILP solver on (2)–(4), restricted to the columns thus generated. The latter is similar, but it stops the column generation procedure immediately after the pricing heuristics fail to identify columns with negative reduced cost, thus preventing the execution of the exact labeling algorithm. In this fashion, we can measure the tradeoff between the reduction in computing times with the impact in terms of the quality of the solution obtained. Table 3 presents the results for the BP algorithm and the two heuristics, aggregated by n and the type of instance. For comparison purposes, we restrict the analysis to the instances for which the LP relaxation can be solved within the time limit. Thus, for each combination, the table depicts the number of instances considered (#Inst), the percentage final gap of the BP algorithm (%fG $= u/b - 1$, where b and u are the best lower bound and the best solution known for the instance, respectively), and the optimality gap of the solution found by the heuristic (%gap $= h/b - 1$, where h represents the solution found by the heuristic).

Reasonable computing times are observed for BPH-CG in all cases. On the most difficult instances, solutions with gaps around 1–2% are obtained in less than an hour (on average). Most of the effort is devoted to the solution of the LP relaxation. In general, the computing times as well as the %gap are considerably smaller, the latter around 0.5%. Second, we note that the quality of the solutions obtained by BPH-Heur does not degrade significantly compared to BPH-CG, and the computing times are considerably reduced approximately by half. From an algorithmic standpoint, the inclusion of 3000 columns per iteration also contribute in this context by populating the RMP with columns that have potential to generate good quality solutions while keeping the resulting ILP formulation manageable.

8 Conclusions

In this paper we developed a BP algorithm for the DM-TDVRPTW that includes state-of-the-art methods. We proposed a stronger domination rule where a label can partially dominate another, which showed to be very effective in reducing the computing times when tackling the pricing problem. In addition, we provided a detailed implementation of the labeling algorithm, including the data structures involved in the dominance tests, as well as the characteristics of the new column generation procedure. All these features were evaluated experimentally on a

				BP		BPH-CG		BPH-Heur	
n	Class	#Inst	Opt	Time	%fG	Time	%gap	Time	%gap
25	C1	9	9	46.79	0.00%	1.79	0.08%	1.14	0.09%
	R1	12	12	1.83	0.00%	0.56	0.09%	0.45	0.28%
	RC1	8	8	3.19	0.00%	2.45	0.16%	1.03	1.03%
	C2	8	8	22.69	0.00%	22.69	0.00%	3.92	0.12%
	R2	11	11	16.88	0.00%	8.72	0.04%	2.13	0.49%
	RC2	8	8	221.29	0.00%	208.19	0.05%	11.44	1.44%
50	C1	9	9	481.92	0.00%	456.26	0.12%	13.75	0.19%
	R1	12	12	242.09	0.00%	8.45	0.27%	4.14	0.30%
	RC1	8	6	1972.96	0.99%	80.64	1.29%	13.69	1.47%
	C2	6	6	928.98	0.00%	143.37	0.03%	36.80	0.08%
	R2	8	8	1724.69	0.00%	373.92	0.47%	30.92	0.83%
	RC2	5	5	1080.60	0.00%	578.68	0.03%	29.41	1.09%
100	C1	8	8	403.68	0.00%	283.83	0.09%	52.34	0.12%
	R1	12	6	2098.60	0.48%	225.23	0.60%	59.24	0.70%
	RC1	8	0	-	1.87%	205.61	1.87%	63.14	1.99%
	C2	4	4	1692.33	0.00%	1309.78	0.00%	273.51	0.11%
	R2	1	1	7004.37	0.00%	930.63	0.95%	102.21	0.95%
	RC2	2	1	5808.87	0.31%	2640.97	0.57%	916.41	2.28%

TABLE 3. *Aggregated results for the BP and the heuristic adaptations for the Dabia et al. instances.*

set of benchmark instances proposed in Dabia et al. (2013). The resulting BP algorithm is able to solve to optimality instances with up to 100 customers and, in particular, it closes all the instances with 25 customers. A possible extension is to incorporate our developments into a BPC algorithm to improve the overall performance, specially for large instances. Additionally, well known relaxation procedures could be considered for the pricing problem, aiming to find a good tradeoff between the computing times and the quality of the lower bound obtained when solving the LP relaxation.

References

R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.*, 59(5):1269–1283, 2011. doi:10.1287/opre.1110.0975.

S. Dabia, S. Ropke, T. Van Woensel, and T. G. de Kok. Branch and price for the time-dependent vehicle routing problem with time windows. *Transp. Sci.*, 47(3):380–396, 2013. doi:10.1287/trsc.1120.0445.

G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column generation.* Springer Science & Business Media, New York, NY, USA, 2005. doi:10.1007/b135457.

G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transp. Sci.*, 42(3):387–404, 2008. doi:10.1287/trsc.1070.0223.

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

M. Gendreau, G. Ghiani, and E. Guerriero. Time-dependent routing problems: a review. *Comput. Oper. Res.*, 64:189–197, 2015. doi:10.1016/j.cor.2015.06.001.

F. Hernández, D. Feillet, R. Giroudeau, and O. Naud. Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *Eur. J. Oper. Res.*, 249(2):551–559, 2016. doi:10.1016/j.ejor.2015.08.040.

Y. Huang, L. Zhao, T. Van Woensel, and J.-P. Gross. Time-dependent vehicle routing problem with path flexibility. *Transp. Res. B*, 95:169–195, 2017. doi:10.1016/j.trb.2016.10.013.

S. Ichoua, M. Gendreau, and J. Potvin. Vehicle dispatching with time-dependent travel times. *Eur. J. Oper. Res.*, 144(2):379–396, 2003. doi:10.1016/S0377-2217(02)00147-9.

I. Ioachim, S. Gélinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3):193–204, 1998. doi:10.1002/(SICI)1097-0037(199805)31:3<193::AID-NET6>3.0.CO;2-A.

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

F. Liberatore, G. Righini, and M. Salani. A column generation algorithm for the vehicle routing problem with soft time windows. *4OR-Q J. Oper. Res.*, 9(1):49–82, 2011. doi:10.1007/s10288-010-0136-6.

Z. Luo, H. Qin, W. Zhu, and A. Lim. Branch and price and cut for the split-delivery vehicle routing problem with time windows and linear weight-related cost. *Transp. Sci.*, 51(2):668–687, 2017. doi:10.1287/trsc.2015.0666.

R. Martinelli, D. Pecin, and M. Poggi. Efficient elementary and restricted non-elementary route pricing. *Eur. J. Oper. Res.*, 239(1):102–111, 2014. doi:10.1016/j.ejor.2014.05.005.

D. Pecin, C. Contardo, G. Desaulniers, and E. Uchoa. New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS J. Comput.*, 29(3):489–502, 2017. doi:10.1287/ijoc.2016.0744.

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

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

M. W. P. Savelsbergh and T. Van Woensel. 50th anniversary invited article—city logistics: Challenges and opportunities. *Transp. Sci.*, 50(2):579–590, 2016. doi:10.1287/trsc.2016.0675.

R. Spliet, S. Dabia, and T. Van Woensel. The time window assignment vehicle routing problem with time-dependent travel times. *Transp. Sci.*, 52(2):261–276, 2018. doi:10.1287/trsc.2016.0705.

P. Sun, L. P. Veelenturf, S. Dabia, and T. Van Woensel. The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *Eur. J. Oper. Res.*, 264(3):1058–1073, 2018. doi:10.1016/j.ejor.2017.07.004.

C. Tilk, A.-K. Rothenbächer, T. Gschwind, and S. Irnich. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *Eur. J. Oper. Res.*, 261(2):530–539, 2017. doi:10.1016/j.ejor.2017.03.017.

P. Toth and D. Vigo, editors. *Vehicle Routing: Problems, Methods, and Applications.* MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2014. doi:10.1137/1.9781611973594.

Appendix — Detailed experimentation

Table 4–11 contain detailed results of the different experiments. The key for the tables are: the number of instances solved to optimality (Opt); the total time (in seconds) for the BP algorithm (Time); the final lower bound obtained by the algorithm (fLB); the optimality gap of the best solution found (%gap), computed as (UB- bestLB)/bestLB, where bestLB represents the best lower bound known for the instance; the percentage final gap of the BP algorithm (%fG $= u/b - 1$, where b and u are the best lower bound known for the instance and the best solution obtained by the algorithm, respectively), and the optimality gap of the solution found by the heuristic (%gap $= h/b - 1$, where h represents the solution found by the heuristic). For the isolated pricing instances, let #nd-F and #nd-B stand for the number of non-dominated forward and backward labels, respectively, and #nd for the total number of non-dominated labels. Finally, instances that reach the time limit of 2 hours are indicated with (—).

Instance	Symmetric			Asymmetric			Partial-LS		Partial-LC	
	Time	#nd-F	#nd-B	Time	#nd-F	#nd-B	Time	#nd	Time	#nd
C203_25_a	2.44	7942	38646	0.89	9165	10857	**0.52**	13842	0.54	13642
C203_25_b	9.53	43574	97850	4.93	49310	45740	**2.40**	48617	2.90	48246
C203_25_c	2.45	9665	37901	1.18	12057	15563	**0.72**	16550	0.75	16318
C204_25_a	40.56	123588	225072	21.77	142380	91636	**3.48**	63129	4.01	60814
C204_25_b	17.16	59806	102996	21.05	130741	87219	**7.84**	107294	9.73	104915
C204_25_c	13.92	11615	122190	7.08	44985	50466	**3.60**	55739	3.89	54881
R207_25_a	1.85	16371	21846	1.91	19049	20940	**0.95**	25527	1.10	25369
R207_25_b	2.88	22295	34365	3.01	26968	32225	**1.50**	36726	1.79	36397
R207_25_c	2.25	23505	20114	2.05	18626	20864	**0.85**	22838	0.97	22742
R208_25_a	11.02	115426	47485	7.71	63361	49517	**2.71**	52344	3.61	52209
R208_25_b	13.48	125070	73009	10.23	75604	78352	**3.99**	68628	5.61	68431
R208_25_c	18.01	122952	120079	17.54	108380	129456	**5.40**	81450	8.09	81301
RC203_25_a	5.47	70067	41021	4.69	54656	43516	**2.66**	53415	3.52	51517
RC203_25_b	9.22	91253	81914	9.21	92264	81997	**7.97**	107766	11.05	104469
RC203_25_c	10.73	137577	48327	6.27	80522	53688	**4.20**	83091	6.35	78994
RC204_25_a	1053.90	2703652	107017	50.02	305141	139272	**33.02**	253972	66.11	248908
RC204_25_b	815.79	2415155	262172	101.61	449941	267331	**59.20**	306269	103.18	303452
RC204_25_c	945.46	2720795	212048	77.50	412458	217661	**41.24**	258420	72.91	255719
RC208_25_a	9.18	111485	38114	8.56	63577	79913	**2.44**	73755	3.81	71729
RC208_25_b	22.45	225324	33145	15.06	91033	111833	**3.60**	97705	5.95	95132
RC208_25_c	20.35	216143	25828	12.50	80998	97028	**3.05**	86327	4.91	83895
C104_50_a	3526.58	2939453	1003771	997.95	1085013	1219832	**160.44**	780440	297.19	778770
C104_50_b	2928.57	2693891	888159	865.73	1001258	1138634	**149.29**	739018	273.22	737151
C104_50_c	2541.24	2508943	888812	759.51	953472	1049526	**137.07**	718357	258.41	716582
R104_50_a	5.57	62450	33190	5.51	43259	50265	**2.26**	48344	2.79	47742
R104_50_b	5.23	58289	31174	5.08	40145	43710	**2.15**	45402	2.62	44787
R104_50_c	5.27	60095	30698	5.17	41387	45561	**2.23**	47054	2.72	46419
C203_50_a	36.19	188766	154738	37.16	205692	152485	**26.76**	206387	44.00	204706
C203_50_b	26.65	146611	115470	28.05	181409	109422	**11.55**	137843	17.67	136210
C203_50_c	27.71	109336	142827	30.28	166652	131672	**17.48**	155638	24.34	153825
C207_50_a	137.32	392608	363794	163.16	359811	555738	**14.14**	126274	17.63	118896
C207_50_b	140.14	404980	351663	165.77	366698	554965	**17.46**	140849	22.33	135784
C207_50_c	127.06	446085	273306	154.32	354098	549492	**15.69**	130988	18.68	123207
RC202_50_a	37.96	366158	59825	23.81	162202	127377	**22.69**	181237	26.41	172055
RC202_50_b	80.05	672950	52633	**23.49**	210669	112088	27.83	225597	40.97	220474
RC202_50_c	37.55	392806	50431	17.86	136685	100275	**15.66**	155270	19.87	151684
RC207_50_a	235.16	1228496	109413	99.97	375842	317483	**20.57**	183044	26.28	177134
RC207_50_b	461.30	1743438	110048	202.45	605296	512433	**19.89**	178003	26.14	173922
RC207_50_c	111.11	637469	116304	98.90	344532	345354	**15.64**	153188	20.14	149401
C103_100_a	116.09	423542	224046	78.92	229942	341261	**13.56**	140815	15.21	134850
C103_100_b	135.86	435329	261605	96.33	264600	388985	**16.48**	167529	19.77	162591
C103_100_c	67.70	273774	153320	51.33	155298	236039	**9.90**	99054	10.74	95206
R104_100_a	599.69	1345611	475064	449.42	754898	828050	**80.29**	546062	133.98	533461
R104_100_b	580.05	1299967	481216	454.35	745720	841690	**81.98**	558108	145.21	544130
R104_100_c	574.98	1313129	458424	431.16	725397	803553	**74.98**	530005	125.02	516868
R108_100_a	784.59	1628510	626322	655.94	963372	1096957	**82.80**	571769	137.90	560765
R108_100_b	784.25	1605999	638697	659.79	969386	1105120	**84.84**	584983	142.92	574474
R108_100_c	780.21	1604456	638454	660.87	965927	1104998	**84.85**	582962	141.64	571986
RC108_100_a	110.38	523384	158501	79.57	362685	270517	**41.10**	386288	60.18	381934
RC108_100_b	110.15	523675	157083	79.58	360645	272060	**40.74**	383333	59.59	378154
RC108_100_c	109.21	520748	154304	78.09	357606	269462	**40.10**	381272	58.40	375978
C206_100_a	74.64	55533	422082	29.39	81320	112934	**7.12**	71264	7.72	69150
C206_100_b	55.08	112734	248966	46.68	126986	164333	**10.66**	93277	11.08	90032
C206_100_c	31.76	65018	165946	26.50	77438	97048	**6.77**	63988	6.82	62130

TABLE 4. *Results for the exact labeling algorithms on the pricing instances.*

Instance	UB	rLB	%rG	rTime	Time	#Nodes	%fG
C101_25	24709.18	24709.18	0.00%	0.06	0.06	1	0.00%
C102_25	24529.89	24480.64	0.20%	1.31	7.56	11	0.00%
C103_25	24395.92	24373.36	0.09%	2.63	8.16	5	0.00%
C104_25	24248.99	24230.33	0.08%	6.84	60.78	27	0.00%
C105_25	24697.51	24669.82	0.11%	0.24	0.53	5	0.00%
C106_25	24709.18	24709.18	0.00%	0.08	0.08	1	0.00%
C107_25	24524.79	24451.23	0.30%	0.54	3.30	13	0.00%
C108_25	24417.63	24350.81	0.27%	0.94	32.18	67	0.00%
C109_25	24337.61	24243.01	0.39%	3.47	308.46	251	0.00%
R101_25	9079.42	9079.42	0.00%	0.04	0.04	1	0.00%
R102_25	7497.76	7497.76	0.00%	0.14	0.14	1	0.00%
R103_25	6551.89	6551.89	0.00%	0.52	0.52	1	0.00%
R104_25	5935.65	5908.74	0.45%	1.03	3.34	9	0.00%
R105_25	7215.46	7215.46	0.00%	0.06	0.06	1	0.00%
R106_25	6537.45	6537.45	0.00%	0.35	0.35	1	0.00%
R107_25	5988.26	5988.26	0.00%	0.83	0.83	1	0.00%
R108_25	5676.84	5666.75	0.18%	1.72	3.65	5	0.00%
R109_25	6130.57	6130.57	0.00%	0.29	0.29	1	0.00%
R110_25	5934.46	5934.46	0.00%	0.54	0.54	1	0.00%
R111_25	6053.87	6025.75	0.46%	0.40	2.61	13	0.00%
R112_25	5627.46	5566.56	1.08%	0.85	9.59	31	0.00%
RC101_25	7038.61	6752.80	4.06%	0.10	1.23	33	0.00%
RC102_25	6187.16	5951.35	3.81%	1.26	5.01	13	0.00%
RC103_25	5447.37	5447.37	0.00%	5.03	5.03	1	0.00%
RC104_25	5199.95	5199.95	0.00%	8.05	8.05	1	0.00%
RC105_25	6835.74	6806.19	0.43%	0.35	1.38	5	0.00%
RC106_25	5615.50	5615.50	0.00%	0.60	0.60	1	0.00%
RC107_25	5030.41	5030.41	0.00%	1.83	1.83	1	0.00%
RC108_25	4918.00	4918.00	0.00%	2.42	2.42	1	0.00%
C201_25	25606.02	25606.02	0.00%	0.13	0.13	1	0.00%
C202_25	24753.47	24753.47	0.00%	4.12	4.12	1	0.00%
C203_25	24509.26	24509.26	0.00%	19.52	19.52	1	0.00%
C204_25	24149.21	24149.21	0.00%	146.90	146.90	1	0.00%
C205_25	25081.26	25081.26	0.00%	0.67	0.67	1	0.00%
C206_25	24953.47	24953.47	0.00%	0.99	0.99	1	0.00%
C207_25	24896.43	24896.43	0.00%	6.61	6.61	1	0.00%
C208_25	24772.62	24772.62	0.00%	2.62	2.62	1	0.00%
R201_25	7932.17	7906.13	0.33%	0.17	0.37	5	0.00%
R202_25	7116.69	7116.69	0.00%	0.98	0.98	1	0.00%
R203_25	6694.06	6643.58	0.75%	2.83	29.65	27	0.00%
R204_25	5792.28	5780.65	0.20%	8.85	26.64	3	0.00%
R205_25	6435.53	6435.53	0.00%	2.18	2.18	1	0.00%
R206_25	6121.82	6121.82	0.00%	3.11	3.11	1	0.00%
R207_25	5701.45	5701.45	0.00%	26.56	26.56	1	0.00%
R208_25	5342.08	5335.30	0.13%	38.21	60.79	3	0.00%
R209_25	5556.38	5530.69	0.46%	1.93	7.11	7	0.00%
R210_25	6501.50	6494.57	0.11%	3.01	6.57	3	0.00%
R211_25	4915.90	4880.65	0.72%	8.07	21.71	5	0.00%
RC201_25	8375.47	8316.78	0.70%	0.81	1.35	3	0.00%
RC202_25	7434.00	7434.00	0.00%	16.34	16.34	1	0.00%
RC203_25	6190.77	6179.99	0.17%	129.01	231.62	3	0.00%
RC204_25	5706.00	5706.00	0.00%	1435.36	1435.37	1	0.00%
RC205_25	7627.38	7627.38	0.00%	2.36	2.36	1	0.00%
RC206_25	6601.87	6587.17	0.22%	3.34	4.97	3	0.00%
RC207_25	5354.50	5354.50	0.00%	29.83	29.83	1	0.00%
RC208_25	4352.80	4352.80	0.00%	48.47	48.47	1	0.00%

TABLE 5. *Results for the BP algorithm, instances proposed by Dabia et al. with $n = 25$.*

Instance	UB	rLB	%rG	rTime	Time	#Nodes	%fG
C101_50	49242.92	49197.45	0.09%	1.30	4.28	5	0.00%
C102_50	48545.68	48528.62	0.04%	15.20	21.70	3	0.00%
C103_50	48251.30	48251.30	0.00%	97.72	97.73	1	0.00%
C104_50	47982.62	47982.62	0.00%	3956.13	3956.13	1	0.00%
C105_50	48809.08	48734.52	0.15%	2.38	6.02	5	0.00%
C106_50	48679.43	48621.64	0.12%	1.65	6.61	7	0.00%
C107_50	48482.48	48348.60	0.28%	4.43	19.37	13	0.00%
C108_50	48219.24	48166.03	0.11%	12.39	29.29	9	0.00%
C109_50	48173.50	48118.34	0.11%	15.10	196.17	55	0.00%
R101_50	16109.19	16109.19	0.00%	0.26	0.26	1	0.00%
R102_50	13450.01	13450.01	0.00%	1.32	1.33	1	0.00%
R103_50	11689.33	11666.72	0.19%	6.76	15.12	5	0.00%
R104_50	10274.33	10209.94	0.63%	23.30	773.09	71	0.00%
R105_50	12805.53	12799.33	0.05%	1.11	1.79	3	0.00%
R106_50	11785.86	11765.39	0.17%	1.97	7.05	7	0.00%
R107_50	11022.48	10976.93	0.41%	5.28	49.07	23	0.00%
R108_50	10006.10	9913.22	0.93%	36.48	1097.80	115	0.00%
R109_50	11302.71	11296.17	0.06%	2.73	3.69	3	0.00%
R110_50	10753.56	10667.62	0.80%	3.78	78.59	79	0.00%
R111_50	10805.47	10741.36	0.59%	6.35	48.39	23	0.00%
R112_50	10167.97	10080.61	0.86%	12.03	828.94	187	0.00%
RC101_50	13996.91	13346.30	4.65%	0.66	1047.85	3069	0.00%
RC102_50	12439.52	11937.56	4.04%	3.74	452.84	551	0.00%
RC103_50	11329.65	10892.09	3.86%	39.89	6654.94	2085	0.00%
RC104_50	9842.81	9842.81	0.00%	477.95	477.95	1	0.00%
RC105_50	13023.63	12615.39	3.13%	3.02	204.31	387	0.00%
RC106_50	12099.94	11310.08	6.53%	2.70	—	2890	4.65%
RC107_50	10614.01	10303.87	2.92%	19.18	2999.89	1541	0.00%
RC108_50	10282.35	9769.20	4.99%	97.98	—	491	3.23%
C201_50	49683.85	49683.85	0.00%	5.05	5.05	1	0.00%
C202_50	48727.41	48727.41	0.00%	374.29	374.30	1	0.00%
C203_50	48482.25	—	—	—	—	0	—
C204_50	51453.12	—	—	—	—	0	—
C205_50	49130.72	49130.72	0.00%	24.09	24.09	1	0.00%
C206_50	49026.62	49026.62	0.00%	67.52	67.52	1	0.00%
C207_50	49055.52	49009.75	0.09%	281.18	4994.79	37	0.00%
C208_50	48799.02	48799.02	0.00%	108.11	108.11	1	0.00%
R201_50	14051.50	14051.50	0.00%	10.16	10.16	1	0.00%
R202_50	12578.28	12552.45	0.21%	42.94	112.65	5	0.00%
R203_50	11271.61	11271.61	0.00%	1256.58	1256.59	1	0.00%
R204_50	9625.19	—	—	—	—	0	—
R205_50	11771.67	11771.67	0.00%	68.14	68.14	1	0.00%
R206_50	10853.84	10820.69	0.31%	634.96	3035.39	7	0.00%
R207_50	13679.03	—	—	—	—	0	—
R208_50	9152.10	—	—	—	—	0	—
R209_50	10111.84	10065.62	0.46%	106.49	1593.45	27	0.00%
R210_50	11091.09	11050.64	0.36%	735.03	4995.90	7	0.00%
R211_50	8910.20	8862.85	0.53%	137.10	2725.22	47	0.00%
RC201_50	14121.19	14103.90	0.12%	109.79	156.99	5	0.00%
RC202_50	13276.71	13264.20	0.09%	302.33	1189.62	3	0.00%
RC203_50	12066.20	—	—	—	—	0	—
RC204_50	9836.32	—	—	—	—	0	—
RC205_50	13978.62	13978.16	0.00%	61.13	376.61	3	0.00%
RC206_50	12032.84	11975.97	0.47%	233.97	1493.59	17	0.00%
RC207_50	9726.40	9726.40	0.00%	2186.19	2186.19	1	0.00%
RC208_50	8348.75	—	—	—	—	0	—

Table 6. *Results for the BP algorithm, instances proposed by Dabia et al. with* $n = 50$.

Instance	UB	rLB	%rG	rTime	Time	#Nodes	%fG
C101_100	97901.11	97901.11	0.00%	21.20	21.20	1	0.00%
C102_100	97662.60	97662.60	0.00%	127.06	127.07	1	0.00%
C103_100	97373.35	97364.43	0.01%	1828.34	2406.29	3	0.00%
C104_100	97290.37	—	—	—	—	0	—
C105_100	97829.25	97689.71	0.14%	30.35	84.66	11	0.00%
C106_100	97692.51	97579.60	0.12%	30.90	242.45	33	0.00%
C107_100	97469.08	97441.53	0.03%	27.21	60.72	5	0.00%
C108_100	97213.78	97177.28	0.04%	82.01	163.49	5	0.00%
C109_100	97004.60	97004.60	0.00%	123.55	123.55	1	0.00%
R101_100	26857.53	26857.53	0.00%	4.58	4.58	1	0.00%
R102_100	23356.16	23313.71	0.18%	14.66	54.45	9	0.00%
R103_100	20630.09	20459.15	0.83%	44.54	—	228	0.38%
R104_100	18076.22	17848.61	1.26%	733.83	—	8	1.10%
R105_100	21802.18	21670.23	0.61%	7.62	463.89	169	0.00%
R106_100	20361.31	20223.07	0.68%	34.03	3197.32	369	0.00%
R107_100	18757.79	18581.06	0.94%	132.73	—	93	0.53%
R108_100	17590.80	17315.03	1.57%	781.10	—	10	1.39%
R109_100	19235.57	19095.36	0.73%	25.43	5674.00	925	0.00%
R110_100	18765.27	18440.62	1.73%	58.13	—	278	1.17%
R111_100	18386.62	18309.10	0.42%	114.86	3197.39	95	0.00%
R112_100	17747.06	17504.70	1.37%	751.30	—	10	1.14%
RC101_100	24889.42	24339.82	2.21%	4.81	—	1782	1.02%
RC102_100	22574.59	21973.08	2.66%	24.74	—	619	1.54%
RC103_100	20721.32	19960.69	3.67%	112.34	—	142	2.91%
RC104_100	19306.81	18796.12	2.65%	1039.40	—	5	2.52%
RC105_100	23564.94	23022.83	2.30%	16.75	—	909	0.98%
RC106_100	21492.78	21025.92	2.17%	18.87	—	799	1.37%
RC107_100	20214.51	19695.18	2.57%	67.40	—	227	2.04%
RC108_100	19462.11	18915.42	2.81%	360.59	—	28	2.59%
C201_100	96115.82	96019.74	0.10%	76.24	1606.38	25	0.00%
C202_100	95960.14	—	—	—	—	0	—
C203_100	—	—	—	—	—	0	—
C204_100	—	—	—	—	—	0	—
C205_100	95763.67	95763.67	0.00%	681.20	681.22	1	0.00%
C206_100	95616.17	95616.17	0.00%	2963.32	2963.35	1	0.00%
C207_100	95613.11	—	—	—	—	0	—
C208_100	95551.43	95551.43	0.00%	1518.36	1518.38	1	0.00%
R201_100	22389.30	22245.40	0.64%	930.63	7004.37	87	0.00%
R202_100	20911.17	—	—	—	—	0	—
R203_100	—	—	—	—	—	0	—
R204_100	17714.77	—	—	—	—	0	—
R205_100	20163.08	—	—	—	—	0	—
R206_100	20327.73	—	—	—	—	0	—
R207_100	—	—	—	—	—	0	—
R208_100	—	—	—	—	—	0	—
R209_100	17961.90	—	—	—	—	0	—
R210_100	—	—	—	—	—	0	—
R211_100	16247.75	—	—	—	—	0	—
RC201_100	24357.18	24237.40	0.49%	228.02	5808.87	45	0.00%
RC202_100	22120.62	21962.90	0.71%	5053.92	—	1	0.63%
RC203_100	—	—	—	—	—	0	—
RC204_100	—	—	—	—	—	0	—
RC205_100	23844.72	—	—	—	—	0	—
RC206_100	21258.98	—	—	—	—	0	—
RC207_100	—	—	—	—	—	0	—
RC208_100	—	—	—	—	—	0	—

TABLE 7. *Results for the BP algorithm, instances proposed by Dabia et al. with* $n = 100$.

	BP		BPH-CG		BPH-Heur	
Instance	%fG	Time	%gap	Time	%gap	Time
C101_25	0.00%	0.06	0.00%	0.06	0.00%	0.06
C102_25	0.00%	7.56	0.01%	1.31	0.01%	1.24
C103_25	0.00%	8.16	0.00%	2.63	0.00%	2.37
C104_25	0.00%	60.78	0.02%	6.84	0.12%	2.61
C105_25	0.00%	0.53	0.00%	0.24	0.00%	0.23
C106_25	0.00%	0.08	0.00%	0.08	0.00%	0.08
C107_25	0.00%	3.30	0.14%	0.54	0.14%	0.53
C108_25	0.00%	32.18	0.23%	0.94	0.23%	0.77
C109_25	0.00%	308.46	0.36%	3.47	0.36%	2.43
R101_25	0.00%	0.04	0.00%	0.04	0.00%	0.03
R102_25	0.00%	0.14	0.00%	0.14	0.00%	0.13
R103_25	0.00%	0.52	0.00%	0.52	0.00%	0.51
R104_25	0.00%	3.34	0.00%	1.03	0.00%	0.81
R105_25	0.00%	0.06	0.00%	0.06	0.00%	0.06
R106_25	0.00%	0.35	0.00%	0.35	0.00%	0.33
R107_25	0.00%	0.83	0.00%	0.83	0.00%	0.65
R108_25	0.00%	3.65	0.11%	1.72	1.90%	1.17
R109_25	0.00%	0.29	0.00%	0.29	0.00%	0.20
R110_25	0.00%	0.54	0.00%	0.54	0.45%	0.45
R111_25	0.00%	2.61	0.27%	0.40	0.27%	0.38
R112_25	0.00%	9.59	0.70%	0.85	0.70%	0.63
RC101_25	0.00%	1.23	0.92%	0.10	0.92%	0.09
RC102_25	0.00%	5.01	0.39%	1.26	0.39%	0.54
RC103_25	0.00%	5.03	0.00%	5.03	0.00%	2.13
RC104_25	0.00%	8.05	0.00%	8.05	6.38%	2.36
RC105_25	0.00%	1.38	0.00%	0.35	0.00%	0.32
RC106_25	0.00%	0.60	0.00%	0.60	0.00%	0.57
RC107_25	0.00%	1.83	0.00%	1.83	0.41%	0.79
RC108_25	0.00%	2.42	0.00%	2.42	0.11%	1.44
C201_25	0.00%	0.13	0.00%	0.13	0.00%	0.12
C202_25	0.00%	4.12	0.00%	4.12	0.13%	1.75
C203_25	0.00%	19.52	0.00%	19.52	0.44%	10.11
C204_25	0.00%	146.90	0.00%	146.90	0.31%	14.37
C205_25	0.00%	0.67	0.00%	0.67	0.00%	0.58
C206_25	0.00%	0.99	0.00%	0.99	0.04%	0.53
C207_25	0.00%	6.61	0.00%	6.61	0.00%	2.33
C208_25	0.00%	2.62	0.00%	2.62	0.05%	1.53
R201_25	0.00%	0.37	0.13%	0.17	0.13%	0.16
R202_25	0.00%	0.98	0.00%	0.98	1.00%	0.85
R203_25	0.00%	29.65	0.00%	2.83	0.97%	2.77
R204_25	0.00%	26.64	0.00%	8.85	0.00%	3.43
R205_25	0.00%	2.18	0.00%	2.18	0.00%	1.23
R206_25	0.00%	3.11	0.00%	3.11	0.00%	1.63
R207_25	0.00%	26.56	0.00%	26.56	1.93%	2.19
R208_25	0.00%	60.79	0.00%	38.21	1.01%	4.77
R209_25	0.00%	7.11	0.36%	1.93	0.36%	1.35
R210_25	0.00%	6.57	0.00%	3.01	0.00%	1.82
R211_25	0.00%	21.71	0.00%	8.07	0.00%	3.23
RC201_25	0.00%	1.35	0.27%	0.81	0.27%	0.59
RC202_25	0.00%	16.34	0.00%	16.34	0.00%	3.88
RC203_25	0.00%	231.62	0.11%	129.01	0.96%	9.57
RC204_25	0.00%	1435.37	0.00%	1435.36	1.37%	62.59
RC205_25	0.00%	2.36	0.00%	2.36	3.27%	1.22
RC206_25	0.00%	4.97	0.00%	3.34	2.68%	1.36
RC207_25	0.00%	29.83	0.00%	29.83	0.17%	7.01
RC208_25	0.00%	48.47	0.00%	48.47	2.79%	5.31

TABLE 8. *Results for the BP and heuristic adaptations, instances proposed by Dabia et al. with* $n = 25$.

	BP		BPH-CG		BPH-Heur	
Instance	%fG	Time	%gap	Time	%gap	Time
C101_50	0.00%	4.28	0.00%	1.30	0.00%	1.24
C102_50	0.00%	21.70	0.03%	15.20	0.08%	7.51
C103_50	0.00%	97.73	0.00%	97.72	0.62%	19.05
C104_50	0.00%	3956.13	0.00%	3956.13	0.02%	70.48
C105_50	0.00%	6.02	0.00%	2.38	0.00%	2.33
C106_50	0.00%	6.61	0.00%	1.65	0.00%	1.62
C107_50	0.00%	19.37	0.38%	4.43	0.38%	4.35
C108_50	0.00%	29.29	0.30%	12.39	0.30%	7.61
C109_50	0.00%	196.17	0.35%	15.10	0.35%	9.59
R101_50	0.00%	0.26	0.00%	0.26	0.00%	0.16
R102_50	0.00%	1.33	0.00%	1.32	0.00%	1.23
R103_50	0.00%	15.12	0.00%	6.76	0.00%	4.48
R104_50	0.00%	773.09	0.71%	23.30	0.71%	11.41
R105_50	0.00%	1.79	0.00%	1.11	0.00%	1.07
R106_50	0.00%	7.05	0.00%	1.97	0.00%	1.53
R107_50	0.00%	49.07	0.21%	5.28	0.42%	3.68
R108_50	0.00%	1097.80	1.04%	36.48	1.26%	10.75
R109_50	0.00%	3.69	0.00%	2.73	0.00%	2.04
R110_50	0.00%	78.59	0.09%	3.78	0.09%	2.18
R111_50	0.00%	48.39	0.78%	6.35	0.78%	4.58
R112_50	0.00%	828.94	0.35%	12.03	0.35%	6.55
RC101_50	0.00%	1047.85	0.67%	0.66	0.99%	0.56
RC102_50	0.00%	452.84	0.52%	3.74	0.52%	2.41
RC103_50	0.00%	6654.94	1.10%	39.89	1.25%	13.32
RC104_50	0.00%	477.95	0.00%	477.95	0.09%	63.33
RC105_50	0.00%	204.31	0.01%	3.02	0.50%	1.84
RC106_50	4.65%	—	4.65%	2.70	4.65%	2.27
RC107_50	0.00%	2999.89	0.14%	19.18	0.53%	6.90
RC108_50	3.23%	—	3.23%	97.98	3.23%	18.89
C201_50	0.00%	5.05	0.00%	5.05	0.00%	5.25
C202_50	0.00%	374.30	0.00%	374.29	0.00%	70.57
C203_50	—	—	—	—	—	—
C204_50	—	—	—	—	—	—
C205_50	0.00%	24.09	0.00%	24.09	0.00%	16.17
C206_50	0.00%	67.52	0.00%	67.52	0.08%	12.88
C207_50	0.00%	4994.79	0.18%	281.18	0.37%	68.72
C208_50	0.00%	108.11	0.00%	108.11	0.05%	47.21
R201_50	0.00%	10.16	0.00%	10.16	0.00%	5.51
R202_50	0.00%	112.65	0.01%	42.94	0.97%	25.46
R203_50	0.00%	1256.59	0.00%	1256.58	0.59%	47.11
R204_50	—	—	—	—	—	—
R205_50	0.00%	68.14	0.00%	68.14	0.99%	6.62
R206_50	0.00%	3035.39	0.33%	634.96	0.65%	28.62
R207_50	—	—	—	—	—	—
R208_50	—	—	—	—	—	—
R209_50	0.00%	1593.45	0.00%	106.49	0.00%	19.32
R210_50	0.00%	4995.90	0.78%	735.03	0.78%	61.81
R211_50	0.00%	2725.22	2.63%	137.10	2.69%	52.92
RC201_50	0.00%	156.99	0.00%	109.79	0.21%	15.20
RC202_50	0.00%	1189.62	0.00%	302.33	0.54%	19.57
RC203_50	—	—	—	—	—	—
RC204_50	—	—	—	—	—	—
RC205_50	0.00%	376.61	0.00%	61.13	0.10%	13.98
RC206_50	0.00%	1493.59	0.17%	233.97	0.99%	18.84
RC207_50	0.00%	2186.19	0.00%	2186.19	3.63%	79.47
RC208_50	—	—	—	—	—	—

TABLE 9. *Results for the BP and heuristic adaptations, instances proposed by Dabia et al. with* $n = 50$.

	BP		BPH-CG		BPH-Heur	
Instance	%fG	Time	%gap	Time	%gap	Time
C101_100	0.00%	21.20	0.00%	21.20	0.00%	20.79
C102_100	0.00%	127.07	0.00%	127.06	0.00%	61.36
C103_100	0.00%	2406.29	0.00%	1828.34	0.22%	139.64
C104_100	—	—	—	—	—	—
C105_100	0.00%	84.66	0.19%	30.35	0.19%	29.84
C106_100	0.00%	242.45	0.52%	30.90	0.52%	30.27
C107_100	0.00%	60.72	0.00%	27.21	0.00%	26.58
C108_100	0.00%	163.49	0.02%	82.01	0.02%	55.50
C109_100	0.00%	123.55	0.00%	123.55	0.01%	54.69
R101_100	0.00%	4.58	0.00%	4.58	0.00%	4.42
R102_100	0.00%	54.45	0.06%	14.66	0.06%	13.89
R103_100	0.38%	—	0.38%	44.54	0.38%	39.75
R104_100	1.10%	—	1.10%	733.83	1.10%	130.68
R105_100	0.00%	463.89	0.27%	7.62	0.27%	7.22
R106_100	0.00%	3197.32	0.23%	34.03	0.23%	26.07
R107_100	0.53%	—	0.53%	132.73	1.27%	82.25
R108_100	1.39%	—	1.39%	781.10	1.39%	144.31
R109_100	0.00%	5674.00	0.32%	25.43	0.46%	15.26
R110_100	1.17%	—	1.17%	58.13	1.17%	36.13
R111_100	0.00%	3197.39	0.57%	114.86	0.57%	46.87
R112_100	1.14%	—	1.14%	751.30	1.53%	164.09
RC101_100	1.02%	—	1.02%	4.81	1.02%	4.50
RC102_100	1.54%	—	1.54%	24.74	1.54%	19.89
RC103_100	2.91%	—	2.91%	112.34	3.45%	179.57
RC104_100	2.52%	—	2.52%	1039.40	2.52%	151.05
RC105_100	0.98%	—	0.98%	16.75	1.24%	10.53
RC106_100	1.37%	—	1.37%	18.87	1.37%	13.84
RC107_100	2.04%	—	2.04%	67.40	2.04%	35.55
RC108_100	2.59%	—	2.59%	360.59	2.72%	90.19
C201_100	0.00%	1606.38	0.00%	76.24	0.00%	76.29
C202_100	—	—	—	—	—	—
C203_100	—	—	—	—	—	—
C204_100	—	—	—	—	—	—
C205_100	0.00%	681.22	0.00%	681.20	0.01%	440.76
C206_100	0.00%	2963.35	0.00%	2963.32	—	300.56
C207_100	—	—	—	—	—	—
C208_100	0.00%	1518.38	0.00%	1518.36	0.31%	276.44
R201_100	0.00%	7004.37	0.95%	930.63	0.95%	102.21
R202_100	—	—	—	—	—	—
R203_100	—	—	—	—	—	—
R204_100	—	—	—	—	—	—
R205_100	—	—	—	—	—	—
R206_100	—	—	—	—	—	—
R207_100	—	—	—	—	—	—
R208_100	—	—	—	—	—	—
R209_100	—	—	—	—	—	—
R210_100	—	—	—	—	—	—
R211_100	—	—	—	—	—	—
RC201_100	0.00%	5808.87	0.51%	228.02	3.23%	1212.58
RC202_100	0.63%	—	0.63%	5053.92	1.34%	620.23
RC203_100	—	—	—	—	—	—
RC204_100	—	—	—	—	—	—
RC205_100	—	—	—	—	—	—
RC206_100	—	—	—	—	—	—
RC207_100	—	—	—	—	—	—
RC208_100	—	—	—	—	—	—

TABLE 10. *Results for the BP and heuristic adaptations, instances proposed by Dabia et al. with $n = 100$.*

	$L_{\max} = 100$		$L_{\max} = 300$		$L_{\max} = 1000$		$L_{\max} = 3000$	
Instance	Time	#Nodes	Time	#Nodes	Time	#Nodes	Time	#Nodes
C104_25	57.65	27	54.91	25	**43.53**	21	61.68	27
C109_25	228.34	247	**136.35**	173	198.73	199	310.71	251
R104_25	**2.23**	9	3.26	9	2.90	9	3.32	9
R112_25	7.72	47	**6.16**	37	11.91	57	9.58	31
RC101_25	**0.69**	19	1.28	39	1.23	33	1.23	33
RC108_25	**1.83**	1	2.04	1	3.38	1	2.40	1
C204_25	160.33	1	193.01	1	**52.16**	1	148.20	1
C207_25	7.22	1	**5.70**	1	6.24	1	6.54	1
R204_25	21.67	3	**18.24**	3	27.04	3	26.58	3
R208_25	**38.67**	3	48.74	3	338.13	3	60.19	3
RC203_25	203.81	3	224.63	3	**174.66**	7	231.36	3
RC204_25	1826.14	1	**290.63**	1	1469.31	1	1415.11	1
C103_50	62.58	1	**46.93**	1	49.41	1	95.76	1
C109_50	**140.99**	39	168.86	57	219.46	71	195.37	55
R104_50	**663.49**	65	668.51	59	697.98	59	772.32	71
R107_50	47.34	23	40.57	21	**40.49**	19	49.22	23
RC102_50	612.33	807	1025.04	1131	515.15	673	**451.92**	551
RC108_50	—	463	—	478	—	498	—	490
C203_50	—	0	—	0	—	0	—	0
C208_50	128.18	1	131.57	1	182.54	1	**107.21**	1
R203_50	2382.96	1	**845.83**	1	1157.65	1	1244.25	1
R209_50	5284.93	25	2055.15	23	**1587.33**	21	1607.32	27
RC202_50	2014.77	3	2898.49	3	**680.12**	3	1207.41	3
RC206_50	3060.39	17	1685.76	15	**1396.84**	17	1507.21	17

TABLE 11. *Performance of the BP when adding at most L_{max} columns per iteration, on a reduced set of instances.*