Delay-Resistant Robust Vehicle Routing with Heterogeneous Time Windows

Lukas Metz Petra Mutzel Tim Niemann Lukas Schürmann
Sebastian Stiller Andreas M. Tillmann

July 21, 2023

Abstract

We consider a robust variant of the vehicle routing problem with heterogeneous time windows (RVRP-HTW) with a focus on delay-resistant solutions. Here, customers have different availability time windows for every vehicle and must be provided with a preferably tight appointment window for the planned service. Different vehicles are a possibility to model different days on which one physical vehicle can serve a customer. This is the main reason why different time windows for different vehicles are of high practical relevance. To ensure that the appointment windows are adhered to as much as possible, we introduce a new objective function that penalizes delays. Our novel approach allows us to find solutions that are robust with respect to uncertainties in travel and service times limited by a budget polytope. We present a set-partitioning model, the solution of which is based on column generation and employs a labeling algorithm that integrates robustness into the calculations and is adapted to our problem-specific constraints. In a Monte-Carlo simulation on real-life data, we evaluate this method in terms of runtime and solution quality. Our solutions show very good performance, even if the data is more uncertain than assumed for the optimization, incurring only marginal extra travel time compared to a naive deterministic planning scheme.

1 Introduction

In this paper, we deal with a timely delivery problem arising in practice. In today's competitive and fast moving environment, customer satisfaction is becoming more and more important and hence receives increased attention. For personal delivery services, a particular issue is that customers often are only available for short periods of time, and to keep them satisfied and retain their business, it is important to reliably meet the stipulated delivery dates. However, uncertain travel times and service durations complicate optimal planning. Thus far, a common method to deal with this is to specify large appointment windows, which can be unsatisfactory for both sides: The customer has long waiting times and the supplier faces a higher probability to miss the customer (who may not be able to stay available for the whole appointment time window).

We propose to tackle these issues by focusing on providing short appointment time windows and explicitly taking delivery delays into account during the optimization. More precisely, we consider a non-deterministic *vehicle routing problem* (VRP) in which the overall task is to schedule a multi-day work period for one vehicle. Each customer has multiple availability time windows, which may be distributed arbitrarily over the planning horizon. The goal is to provide each customer with a reliable and tight appointment window that lies within one of their respective availability time slots; in other words, the

delivery time windows communicated to the customers have to be much smaller than typical in current practice. The punctuality is modeled by penalizing tardiness.

1.1 Related Work

The basic vehicle routing problem is a generalization of the well-known traveling salesman problem (TSP) [6]: Instead of looking for a shortest tour that visits every node of a graph exactly once, it is possible to use multiple tours. Various variants and extensions of the VRP have been investigated over the years, and a comprehensive summary of the rich literature in this field is beyond the scope of the present paper. Thus, we briefly discuss the works most relevant to ours in the following, and refer to [5, 17, 18, 33] for broad and extensive surveys on the VRP and its variants.

In the well-established capacitated vehicle routing problem, which was already used in [12], each customer has a demand that must be fulfilled, and the sum of the demands on a (delivery) tour must not exceed the capacity of the vehicle. Another typical VRP extension introduced by [31] incorporates time windows, in the sense that a customer can only be visited within a given interval. Such time windows can be either hard, restricting the set of feasible solutions, or soft, incurring additional costs in the objective function if they are exceeded, cf., e.g., [28].

Besides VRPs with deterministic data, there are variants in which the costs, driving and/or service times are subject to uncertainties. These uncertainties can be modeled by probability distributions, often requiring strong assumptions about the distribution of the data, see, e.g., the early work [13]. Alternatively, approaches from robust optimization can be chosen, whereby only assumptions about maximum possible deviations are made and, e.g., the number of delay occurrences is limited. The concept of robustness used in our paper is based on the seminal work in [4]. We extend the application of this robustness to vehicle routing problems as found in [23].

The idea of assigning time windows to the customers has been introduced in [32]. In the VRP-TWA (time window assignment), it is assumed that there is a set of demand scenarios that can occur. The goal is then to find the best time window assignment over these scenarios. In the robust VRP-TWA, see [14], this concept is extended to the uncertainty of travel times. Furthermore, in this RVRP-TWA, an additional objective is to minimize the risk of violating one of these endogenous time windows.

In homogeneous VRPs, the assumption is that all tours or vehicles operate under identical constraints. However, in heterogeneous VRPs, the constraints assigned to each vehicle vary, leading to variations in capacities and other factors as in [16]. This can be used, for example, to model a fleet in which not all vehicles are the same size, cannot carry the same weight, or have different maintenance costs.

Furthermore, in the site-dependent VRP, the availability of certain customers may be limited to specific vehicles due to various factors such as location restrictions or specialized equipment requirements. This implies that not all vehicles are capable of serving every customer, introducing an additional layer of heterogeneity to the problem, cf., e.g., [25].

In addition, our model allows for the heterogeneity of the time windows as described in our preliminary work [24]. However, in contrast to the present paper, the problem considered in [24] was not robust against uncertainties and the focus was not on real-world problem instances.

1.2 Contributions and Paper Outline

We present a generalization of the vehicle routing problem with time windows in which each vehicle corresponds to one day and the customers can have different time windows for different vehicles (days). This corresponds to a VRP with heterogeneous time windows which, to the best of our knowledge (see

also [16]), did not receive any attention in the scientific literature yet except in our own preliminary work [24].
Moreover, to deal with the uncertainty in travel and service times, we make use of the concept of *robust optimization*. In addition to the commonly used robust constraints, cf. [4], we introduce a new *delay-resistance term* in the objective function to improve reliability and at the same time allow for decreasing the length of the appointment windows that are to be provided to the customers.
We give a set-partitioning formulation for our resulting *robust vehicle routing problem with heterogeneous time windows* (RVRP-HTW) and propose a branch-and-price algorithm. We develop a dominance check for the labeling algorithm (used for solving the pricing problem) that accounts for our new objective function. Moreover, we exploit the heterogeneous problem structure to create dedicated primal heuristics and to generate smaller pricing subproblems. These pricing subproblems allow us to take advantage of parallel computing.
For our computational evaluation, we use instances based on real-life data from a project partner in logistics. The travel time data is based on historical measurements, the uncertainties are determined using statistics on these measurements and the service times are empirical values. Our exact branch-and-price approach can solve instances with up to 60 customers and a planning horizon of ten days to optimality. In order to tackle larger instances, we employ problem-specific heuristics that yield promising results. Moreover, our experiments show that combining our new objective function with a standard distance/route-length minimization yields the best of two worlds: The tours do not become significantly longer, but the possible delays can be drastically reduced.Finally, Monte-Carlo simulations [22] allow to assess the quality of the generated solutions when, for example, the sum of delays across all customers is used as a criterion.
The remainder of this paper is structured as follows: In Section 2, we will introduce our model for the robust vehicle routing problem with heterogenous time windows. After that, in Section 3, we will explain our customized branch-and-price method including the pricing, labeling algorithm and our heuristics to speed up the solving process. The computational results on the aforementioned data are evaluated in Section 4, concluding with a Monte-Carlo simulation [22] of the calculated tours with different amounts of delay. We close the paper in Section 5 with some concluding remarks and an outlook of future research.

2 Modeling Reliable Delivery

In this section, we will derive and explain our VRP variant of interest along with precursory approaches to tackle certain difficulties occurring in the application and some foundations of the modeling tools we utilize. To that end, we first formally decribe the well-known *vehicle routing problem with (hard) time windows* (VRP-TW) which our problem is based on (Section 2.1), and then establish heterogeneous time windows and appraise their influence on the model (Section 2.2). Subsequently, in Section 2.3, we briefly recall the concept of robust optimization and show how it can be used in the context of delivery reliability. Finally, the full model for our problem is presented in Section 2.4.

2.1 Vehicle Routing with Appointment Windows

The VRP-TW [26, 27] can be defined as follows: Starting from a central depot 0, the set of customers $C \coloneqq \{1, \dots, n\}$ should be supplied by any one of a fleet of vehicles K. Every customer $i \in C$ has an associated individual service time $s_i \geq 0$ for, e.g., unloading, customer service or the way from

the vehicle to the front door. Moreover, each customer i is assumed to have specified a time window $T_i \coloneqq [l_i, u_i] \neq \emptyset$ during which he or she is available for receiving deliveries or customer service appointments. Also, there is a (not necessarily symmetrical) travel time $t_{ij} \geq 0$ between each pair of customers i and j.

Each customer must be visited by exactly one vehicle and, crucially, the service at each customer $i \in C$ must begin within the time window $T_i = [l_i, u_i]$. Thus, a feasible tour of one vehicle starts and ends in the depot, visiting customers one after the other adhering to their respective time windows in between, where a given total working time limit H of the driver must not be exceeded. Typical optimization goals are the total travel distance or time of all vehicles/tours that are employed to meet the requirement of servicing every customer.

The above definitions induce a complete directed graph with node set $V \coloneqq C \cup \{0\}$, wherein a valid vehicle routing can be seen as a set of $|K|$ tours (i.e., ordered sequences of nodes with identical start and end node 0), some of which may be empty. (Henceforth, we will sometimes implicitly refer to this graph representation and use the terms nodes and customers, or arcs and driving route between customers interchangeably.) Such a routing implies appointment times L_i, $i \in C$, at which the service of the respective customer i starts. These are not necessarily the times of arrival at the customer, since a driver might wait (idly by choice, or necessarily if $L_i < l_i$). Nevertheless, for reasons of practicality, unnecessary waiting at any customers' location is prohibited, i.e., service always starts as soon as possible. Note that the appointment times L_i that result from VRP optimization are communicated to the customers by the delivery or service company. In fact, typically, the customers are not provided with a concrete point in time like the actual L_i, but are given time windows that often span several hours, or sometimes even just the date when they are scheduled to be serviced. Depending on the specific nature of the service and the employed VRP approach, this may be due to, e.g., planning in multiple stages or to achieve a minimum degree of reliability—broad appointment time windows are by design relatively insensitive to disruptions of the tour execution in practice, i.e., delays with respect to announced time windows are unlikely. However, it is conceivable that customers generally would much prefer more concise appointment windows as well as more flexibility regarding the availability time windows T_i they are asked to specify. Both of these aspects are addressed by our work, as laid out in the following subsections.

2.2 Heterogeneous Time Windows

Heterogeneity in vehicle routing problems usually pertains to different capacities of the vehicles [2, 16]. We introduce a new concept of heterogeneous availability time windows motivated by the following real-world situation (encountered, e.g., by our project partners): We are given a set of customers that need to be served within a planning period, the latter being represented as a set of days K. For each day, every customer $i \in C$ has a time window $T_i^k = [l_i^k, u_i^k]$, where generally, $T_i^k \neq T_i^{k'}$ for $k \neq k'$, and $T_i^k = \emptyset$ is also possible. Obviously, this problem generalizes its homogeneous variant, in which each customer has the same time window for each vehicle/day. We call this generalization the *vehicle routing problem with heterogeneous time windows* (VRP-HTW), where we additionally allow the driver's working time limit H_k to vary over the days.

The heterogeneity of the time windows leads to a more intricate problem, but in contrast to the heterogeneous capacity variant, we can make use of the additional structure induced by the individual time windows: Instead of only one graph, we get a graph G_k for each day $k \in K$. Furthermore, since in general, not every customer is necessarily available on every day, $V_k \subseteq V$ needs not be tight. In the remainder of the paper, we will repeatedly circle back to the novel availability time window concept and how to make beneficial use of its structural implications.

2.3 Robustness and Delay-Resistance

In practice, service and travel times are not precisely known a priori and therefore need to be estimated. Moreover, there remains uncertainty due to traffic disruptions and the realizations of actual service times at the customers are not known in advance either. Hence, it is common practice to make certain assumptions with respect to the uncertainty distributions to address these issues in a more realistic model. Here, as a first step, we assume that for each arc (i, j) in the underlying graph, a maximum travel time delay $\bar{t}_{ij} \geq 0$ was determined or is known (e.g., from historical data), and analogously, for each customer, a maximum service time delay $\bar{s}_i \geq 0$.

The practical uncertainty issues also relate to our goal to give each customer an appointment window $[L_i, U_i]$ with a maximum length of $W \geq U_i - L_i$ that is much shorter than usual (instead of a just point in time or a very long time span). As shorter appointment time windows increase the chances of uncertain disruptions leading to unkept appointments, i.e., delays, we are therefore looking for a vehicle routing that generates minimal (worst-case) delays with respect to the resulting appointment windows provided to the customers.

In order to hedge against uncertainties and with our aforementioned goal in mind, we will utilize techniques from *robust optimization*. In the following, we briefly recapitulate the relevant robust optimization fundamentals, and describe how robustification in the present context affects constraints and objective function to ultimately give rise to our *robust vehicle routing problem with heterogeneous time windows* (RVRP-HTW).

The idea of robust optimization is to "strengthen" solutions with the help of robustness constraints that yield assured (if not all-encompassing) immunity to data uncertainty. In the present context, such constraints should enforce that solutions stay feasible (meet the time windows) even if a certain amount of delays occur. On the one hand, it is unrealistic that every trip can be carried out without problems, but on the other hand, it is overly pessimistic to assume that every possible delay actually occurs. Indeed, in most realistic scenarios, there are only a few customers/paths that cause a delay. In order to protect our solution against disruptions, we therefore assume that at most $\Gamma \geq 0$ delays per day will occur; service and travel time delays are counted together. In practice, Γ may be defined as the expected value or historical average of the number of delays in a typical tour.

Formally, this means we consider the so-called budgeted uncertainty set, since we only consider certain scenarios induced by the maximum amount of delays. With the values $\xi_i^s \in \{0, 1\}$ and $\xi_{ij}^t \in \{0, 1\}$ indicating whether of not in a certain scenario, there is a service time delay at customer/node i or a travel time delay at arc (i, j), respectively, we can define the uncertainty set

$$\Omega_\Gamma \coloneqq \left\{ \xi = (\xi^s, \xi^t) \in \{0,1\}^C \times \{0,1\}^{V \times V} : \sum_{i \in C} \xi_i^s + \sum_{(i,j) \in E} \xi_{ij}^t \leq \Gamma \right\}. \tag{2.1}$$

Subsequently, we are searching for a solution that is valid for each scenario $\xi \in \Omega_\Gamma$.

Definition 2.1. A tour is said to be Γ-*robust* if it remains feasible for up to Γ delays. If clear from the context, we just speak of *robust* tours.

With the help of the parameter Γ, we can now decide how to weigh security and uncertainty aspects: A small Γ means more willingness to risk missing appointments, but also allows denser tours with smaller intervals between two appointments and therefore shorter (or fewer) tours if there are hardly any delays. A larger Γ produces permissible solutions even in case of large disruptions, but likely results in pessimistic planning in most cases.

The case $\Gamma = 0$ yields a deterministic model that does not take delays into account at all; we will compare our robust model with this variant later. In this setting, the starting times of the service at each customer could easily be calculated for any sequence of customers directly. Unfortunately, dealing with robust solutions ($\Gamma > 0$) leads to a more complex definition, since then, the arrival times are scenario-dependent: Similarly to [23], let begin-of-service variables $y_{i\gamma}$, for $i \in C$ and $\gamma \in \{0, \dots, \Gamma\}$, represent the respective latest point in time at which service will start at node i if there are at most γ delays on the route to that node. Given any path $\{v_0, v_1, \dots, v_j\}$ from the depot ($v_0 = 0$) to node v_j on day $k \in K$, and writing $\alpha_{i\gamma} \coloneqq y_{v_{i-1}\gamma} + s_{v_{i-1}} + t_{v_{i-1}v_i}$ with $s_0 = \bar{s}_0 = 0$, we now can recursively calculate our service variables for any $i \in \{0, \dots, j\}$ via

$$y_{v_i\gamma} = \begin{cases} 0 & \text{if } i = 0, \\ \max\{l_i^k, \alpha_{i\gamma}\} & \text{if } \gamma = 0, i \neq 0, \\ \max\{l_i^k, \alpha_{i\gamma}, \alpha_{i\gamma-1} + \bar{s}_{v_{i-1}}, \alpha_{i\gamma-1} + \bar{t}_{v_{i-1}v_i}\} & \text{if } \gamma = 1, i \neq 0, \\ \max\{l_i^k, \alpha_{i\gamma}, \alpha_{i\gamma-1} + \bar{s}_{v_{i-1}}, & \\ \qquad \alpha_{i\gamma-1} + \bar{t}_{v_{i-1}v_i}, \alpha_{i\gamma-2} + \bar{s}_{v_{i-1}} + \bar{t}_{v_{i-1}v_i}\} & \text{otherwise.} \end{cases} \tag{2.2}$$

Using these variables, our robust constraints can now be defined as

$$y_{i\Gamma} \leq u_i^k, \quad \forall i \in C, k \in K. \tag{2.3}$$

It is important to differentiate between the given, hard customer availability time windows $T_i^k = [l_i^k, u_i^k]$ and the soft appointment windows $[L_i, U_i]$. In particular, $y_{i\Gamma} > U_i$ is allowed but constitutes a delay. Note that setting $L_i = y_{i0}$ and $U_i = L_i + W$ will always be optimal when the goal is to minimize delays $y_{i\Gamma} - U_i$ with respect to the appointment windows since $y_{i\gamma} \leq y_{i\gamma+1}$. In other words, it is optimal to start the appointment window at the earliest possible service start time.
We will combine a so-called *delay-resistance* term with the classical total travel time to a weighted-sum objective function. The delay-resistance is comprised of summands for each tour p that is part of the VRP solution, and each such summand is defined as the sum of the terms

$$(y_{i\Gamma} - y_{i0} - W)_+ \coloneqq \max\{y_{i\Gamma} - y_{i0} - W, 0\} \tag{2.4}$$

over all customers i in the respective tour p. In words, (2.4) is the worst-case delay value with respect to $U_i = y_{i0} + W$ at customer i (on tour p) when assuming there are at most Γ delays.
Note that if there are, in fact, no more than Γ delays per route and $\hat{y}_i$ is the true arrival time at customer i for a scenario in Ω_Γ, it will hold that $y_{i0} \leq \hat{y}_i \leq y_{i\Gamma}$ (and thus, by virtue of (2.3), that $\hat{y}_i \leq u_i^k$ for the day k on which customer i is serviced). Moreover, if all input data are integers, then we also get $y_{i\gamma} \in \mathbb{Z}$ for all i and γ.

2.4 Formulation of the RVRP-HTW

Similarly to standard vehicle routing problems, our RVRP-HTW can be formulated straightforwardly as a mixed-integer program. However, even for simpler VRP variants, these programs quickly get very big and hard to solve. In our case, incorporating robustness as well as heterogeneous time windows requires even more variables and constraints, and renders the resulting program entirely impractical. Therefore, we omit stating the RVRP-HTW mixed-integer program formulation, and instead directly move on to an approach that has emerged as the most efficient for various different VRPs over the past decades: column generation for a set-partitioning model [1, 5, 9, 33].

Let $\mathcal{P}$ be the set of all feasible tours. The definition of feasibility depends on the concrete VRP variant under consideration and will become important only later when we turn to the solution algorithm (see Section 3); for now, we only assume $\mathcal{P}$ is given. We use a binary variable x_p to decide whether a tour $p \in \mathcal{P}$ is used for the overall solution or not. Similarly, parameters $a_{ip} \in \{0,1\}$ indicate whether customer i is visited on tour p or not; note that the a_{ip} are *known* for all tours. In order to enforce that our solutions are comprised of $|K|$ tours—one for each day of the planning horizon—we introduce parameters $b_{kp} \in \{0,1\}$ that indicate whether tour p is carried out on day k. With a nonnegative cost of C_p associated with each tour p, the so-called *set-partitioning formulation* (SPF) is then given by the following binary integer program:

$$z_{\text{SPF}} \coloneqq \min_{x} \quad \sum_{p\in\mathcal{P}} C_p x_p \tag{2.5}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}} a_{ip} x_p = 1 \qquad \forall i \in C \tag{2.6}$$

$$\sum_{p\in\mathcal{P}} b_{kp} x_p = 1 \qquad \forall k \in K \tag{2.7}$$

$$x_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{2.8}$$

Solving this program results in the customers being divided into subsets given by the employed tours, which corresponds to a *partition* of the customer set, cf., e.g., [23]. We emphasize that by choosing a day and the customer order, a tour $p \in \mathcal{P}$ is uniquely defined in our setting. Also, two tours with the same customer order can have different costs on different days.

The definition of costs for each tour depend on the considered VRP variant; here, it will therefore be the combined delay and travel time function alluded to earlier. Note that, in theory, the begin-of-service times could be calculated with the recursive formula (2.2) in advance, since with each tour, the corresponding day and customer sequence is known. Thus, the objective cost coefficient C_p of a tour p is given as the weighted sum over the (delay-induced) costs of all visited customers for this tour and its travel time. Denoting the customer sequence of tour p as $(v_0^p, v_1^p, \ldots, v_{n_p}^p, v_0^p)$, where $v_0^p = 0$ for every p, we define

$$C_p^i \coloneqq \begin{cases} \alpha^d c_i (y_{i\Gamma} - y_{i0} - W)_+ + \alpha^t t_{v_{j-1}^p i} & \text{if } i = v_j^p,\ j > 1, \\ 0 & \text{else.} \end{cases} \tag{2.9}$$

$$\text{and} \quad C_p \coloneqq \sum_{i\in V} a_{ip} C_p^i. \tag{2.10}$$

Here, the parameters α^d and α^t are used to balance the delay-resistance (cf. (2.4)) and travel time minimization. Moreover, the constants c_i ($c_0 = 0$) allow to allot different importance to customers; this aspect was not mentioned yet, but is, in fact, a useful feature in practice. (Although we include it here as it accommodates a request by our project partner, we will work with $c_i = 1$ for all $i \in C$ throughout all later experiments.)

Since each tour only takes place on one day, there is only one b_{kp} for each tour p that is not zero. Consequently, the set of all possible tours $\mathcal{P}$ breaks down into the disjoint subsets $\mathcal{P}_k$, one for each day,

and we can reformulate the SPF with the following Dantzig-Wolfe decomposition:

$$z_{\text{SPM}} := \min_{x} \quad \sum_{k \in K} \sum_{p \in \mathcal{P}_k} C_p x_p \tag{2.11}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{p \in \mathcal{P}_k} a_{ip} x_p = 1 \qquad \forall i \in C \tag{2.12}$$

$$\sum_{p \in \mathcal{P}_k} x_p = 1 \qquad \forall k \in K \tag{2.13}$$

$$x_p \in \{0, 1\} \qquad \forall p \in \mathcal{P}_k \tag{2.14}$$

The complexity of the above formulations is hidden in the set $\mathcal{P}$: Obviously, we cannot efficiently solve or even formulate such integer programs, as there is an exponential amount of tours and, hence, variables or *columns*. Therefore, we will turn to the well-known *column generation* technique to solve our RVRP-HTW, which will be discussed in the next chapter. In that regard, note also that in the above (theoretical) models, the C_p-values are already determined in advance. In the practical column generation approach, most tours are neglected at the beginning, and we will only calculate the actual costs of tours during the assessment of a previously ignored route for potential inclusion in the updated model.

3 Branch-and-Price Method

In this section, we outline the state-of-the-art solution approach for VRPs in the set-partitioning form and introduce the necessary modifications to adapt it to our needs as well as further tweaks that improved our implementation. The most promising method that emerged over the past decades is the *branch-and-price* technique, which we shall also employ to solve our set-partitioning model for the RVRP-HTW. As the name suggests, there are two main parts to focus on when implementing such an algorithm: branching and pricing, i.e., division into subproblems (typically by fixing variables) and identification of promising candidate tours, respectively. While the former is often not analyzed in depth in the vehicle routing literature, the latter is usually crucial for the practical performance, amenable to problem-specific alterations and generally a challenging task itself, and has hence received far more attention.

3.1 Foundations

Although the set-partitioning approach yields an elegant and simple-looking formulation for a class of complex problems such as our RVRP-HTW, solving such an integer linear program (ILP) is, unfortunately, not simple at all—as mentioned before, most of the problem-specific restrictions are "hidden" in the set $\mathcal{P}$, i.e., provided implicitly. In particular, the size of this set, i.e., the number of variables in the set-partitioning ILP, can be exponential in the number of customers.
In general, ILPs are nowadays usually solved with branch-and-bound algorithms (cf., e.g., [19]) whose backbone is to combine branching with bounds derived from the linear programming (LP) relaxation. To that end, one essentially needs to solve the LP relaxation of the current ILP at every subproblem created by branching. We assume the reader is generally familiar with these techniques and refer to standard textbooks such as, e.g., [29, 34] for formal details. In our context, the relaxation of the original

set-partitioning ILP is called the *master problem* (MP) and given as follows:

$$
\begin{aligned}
z_{\text{MP}} \coloneqq \min_{x} \quad & \sum_{k\in K}\sum_{p\in\mathcal{P}_k} C_p x_p & & \\
\text{s.t.} \quad & \sum_{k\in K}\sum_{p\in\mathcal{P}_k} a_{ip} x_p = 1 & & \forall i \in C \\
& \sum_{p\in\mathcal{P}_k} x_p = 1 & & \forall k \in K \\
& x_p \geq 0 & & \forall p \in \mathcal{P}_k
\end{aligned}
$$

The fact that the size of $\mathcal{P} = \bigcup_{k\in K} \mathcal{P}_k$ does not allow solving this MP directly in cases of practical interest, along with the key observation that the above MP always has an optimal solution with at most $|C| + |K|$ non-zero entries, motivates the use of *column generation* techniques, see, e.g., [8, 20] for details. To that end, consider the *Restricted Master Problem* (RMP) in which only a known subset of variables indexed by $\mathcal{P}' \subseteq \mathcal{P}$ is present, or more precisely, by subsets $\mathcal{P}'_k \subseteq \mathcal{P}_k$ for each $k \in K$, some of which may be empty:

$$
\begin{aligned}
z_{\text{RMP}} \coloneqq \min_{x} \quad & \sum_{k\in K}\sum_{p\in\mathcal{P}'_k} C_p x_p & & \\
\text{s.t.} \quad & \sum_{k\in K}\sum_{p\in\mathcal{P}'_k} a_{ip} x_p = 1 & & \forall i \in C \\
& \sum_{p\in\mathcal{P}'_k} x_p = 1 & & \forall k \in K \\
& x_p \geq 0 & & \forall p \in \mathcal{P}' = \bigcup_{k\in K} \mathcal{P}'_k
\end{aligned}
$$

The general idea is to start with few variables (or even none) and then iteratively add *promising* new variables to $\mathcal{P}'$ and reoptimize the RMP until a provably optimal solution to the MP has been found. This way, one hopes to reach such a solution after adding only a comparatively small number of variables, thereby avoiding the explicit inclusion of all or very many variables (most of which are zero anyway). If this final MP solution is integral, the original problem has been solved; otherwise, the problem is split into subproblems by branching on, e.g., one of the currently present variables.
To formally define the term *promising*, we need to take a look at the dual LP of the RMP:

$$
\begin{aligned}
z_{\text{RMP}} = \max_{\pi,\tau} \quad & \sum_{i\in C} \pi_i + \sum_{k\in K} \tau_k & & \\
\text{s.t.} \quad & \sum_{i\in C} a_{ip}\pi_i + \tau_k \leq C_p & & \forall p \in \mathcal{P}'_k,\ k \in K
\end{aligned}
$$

Let (π^*, τ^*) be an optimal solution to this LP. The goal is now to find a tour $p \in \mathcal{P} \setminus \mathcal{P}'$ that violates dual feasibility or to prove that no such tour exists. Aiming at finding tours that allow to improve the

primal problem objective, this gives rise to the so-called *reduced-cost pricing problem*

$$r(\pi^*, \tau^*) \coloneqq \min_{k \in K, p \in \mathcal{P}_k} \left\{ -\tau_k^* + \sum_{i \in C} a_{ip} \left(C_p^i - \pi_i^* \right) \right\}$$
$$= \min_{k \in K} \left\{ -\tau_k^* + \min_{p \in \mathcal{P}_k} \sum_{i \in C} a_{ip} \left(C_p^i - \pi_i^* \right) \right\}.$$

Note that this problem naturally decomposes into separate (reduced-cost) *pricing subproblems*

$$r_k(\pi^*, \tau^*) \coloneqq -\tau_k^* + \min_{p \in \mathcal{P}_k} \left\{ \sum_{i \in C} a_{ip} \left(C_p^i - \pi_i^* \right) \right\}$$

for each $k \in K$, owing to the specific structure in our RVRP-HTW. Since (π^*, τ^*) is a dual-optimal solution, for every $k \in K$, the reduced cost associated with any tour $p \in \mathcal{P}'_k$ is nonnegative, so if $r_k(\pi^*, \tau^*) < 0$, the minimum in its definition is attained at a previously unknown tour in $\mathcal{P}_k \setminus \mathcal{P}'_k$. Thus, we can now simultaneously and independently search for new tours with negative reduced costs *for each day*, rather than just one such tour (on one arbitrary day).

If no new tour with negative reduced cost exists at all, i.e., $r(\pi^*, \tau^*) \geq 0$ or equivalently, $r_k(\pi^*, \tau^*) \geq 0$ for all $k \in K$, then the current optimal RMP solution is also optimal for the MP. In case no valid RMP solution exists to begin with, we can make use of *Farkas pricing* (cf. [20]) to either obtain feasibility of the RMP or prove that the MP is infeasible. Note that even if RMP feasibility can be obtained initially, it may be lost as a consequence of branching decisions; thus, for a functional implementation, Farkas pricing is a mandatory component.

Both the reduced-cost pricing and the Farkas pricing amount to solving an *elementary shortest path problem with resource constraints* (ESPPRC). This NP-hard problem [7, 10] can be solved with a so-called *labeling algorithm*, see Section 3.2 below and [3, 11, 15, 27] for more literature.

Once the master problem is solved, if its optimal solution is integral, it is also optimal to the original problem. Otherwise, we need to take a branching step. Unfortunately, branching on a tour variable x_p yields a different pricing problem, namely the ESPPRC with forbidden paths [15], that introduces a new kind of dual variables which cannot be easily integrated into our labeling algorithm.

Furthermore, the number of potential variables to branch on would result in a large and unbalanced branching tree: Every branch $x_p = 1$ excludes all other possible tours for this day, but the branch $x_p = 0$ hardly changes the set of feasible solutions (at least in lower levels of the search tree).

To avoid this, [9] introduced the arc flow variable branching, which is based on the fact that a solution is integral if and only if for each arc, the sum of the variables of tours that use this arc is integral. This approach works well with the ESPPRC: By making a decision on an edge, we exclude many more paths at once than by branching on only one tour variable $x_p = 0$. In addition, these effects accumulate further down the branching tree which gives a slight speed-up to solve the subproblem, as each branching decision reduces the size of the pricing problem instances.

Another option is vehicle assignment branching, where binary auxiliary variables are introduced to indicate to which vehicle or to which day a customer was assigned.

$$d_i^k \coloneqq \sum_{p \in \mathcal{P}_k} a_{ip} x_p \in [0, 1], \quad \forall\, i \in C, k \in K\,. \tag{3.1}$$

In [24] both branching rules are compared and the vehicle assignment branching almost always gives better results for heterogeneous time windows. But, as also stated there, the more similar the time

Algorithm 1: LABELINGALGORITHM

Input: G_k, (τ_k, π)
Output: $\mathcal{L}$
/* Initialization */
$\mathcal{L} = \emptyset$;
$L_0 = (-\tau_k, \emptyset, 0, 0, \ldots, 0)$;
for *each* $i \in C$ **do**
  $B_i = \emptyset$;
$B_0 = \{(-\tau_k, \emptyset, 0, 0, \ldots, 0)\}$;
while $\bigcup_{i \in V} B_i \neq \emptyset$ **do** /* While there are unpropagated labels ... */
  Choose $i \in V$ with $B_i \neq \emptyset$;
  Choose $L \in B_i$;
  $B_i = B_i \setminus \{L\}$;
  for *each neighbor* $j > 0$ *of* i *in* G_k *with* $j \notin V(L)$ **do**
    $L' = \text{PROPAGATELABEL}(L, j, \pi_j)$;
    if ISFEASIBLE(L') **and** HASPOTENTIAL(L', π) **then**
      $insertLabel$ = True;
      for $L_j \in B_j$ **do**
        if L' *dominates* L_j **then**
          delete L_j;
        else if L_j *dominates* L' **then**
          $insertLabel$ = False;
      if $insertLabel$ **then**
        $B_j = B_j \cup \{L'\}$;
  $L^* = \text{PROPAGATELABEL}(L, 0, 0)$;
  if ISFEASIBLE(L^*) ***and*** $R(L^*) < 0$ **then** /* Check for negative reduced costs */
    $\mathcal{L} = \mathcal{L} \cup \{L^*\}$;

windows are for different days, the smaller the difference becomes. However, since in this work we have been provided with exactly such instances by our industry partner, we will use vehicle assignment branching only for one of our heuristics, see section 3.3.1.

3.2 Labeling Algorithm

An exact branch-and-price method calls for an algorithm that solves the pricing problem. The corresponding tool commonly used for the ESPPRC is the labeling algorithm, which is particularly appealing in the context of vehicle routing problems due to its high adaptivity to different constraints as encountered in the variety of VRP settings [3, 11, 15, 27].
In the following, we describe the labeling algorithm (see Algorithm 1) for a fixed $k \in K$. The basic idea is essentially a smart enumeration of all possible paths similarly to Dijkstra's algorithm to find the shortest path between two nodes. However, in our problem there is not only a shortest path, but also, for example, a most resource-efficient path and combinations of both. As a result, we have to remember

several ways to "best" get to a node. To that end, we represent each subpath from the depot to a customer i by a label

$$L \coloneqq (R, V, T, Y_0, \dots, Y_\Gamma),$$

that gets stored in a bucket B_i where $R(L) = R$ is the reduced cost of the path, $V(L) = V$ the set of already visited customers on this subpath, $T(L) = T$ the starting time at the depot and $Y_\gamma(L) = Y_\gamma$ the arrival time at the current customer with at most $\gamma = 0, \dots, \Gamma$ delays on the route. The labels at each customer are divided into two subsets: The already processed labels are called *propagated* and the newly created ones are called *non-propagated*. Starting with an initial empty label $L_0 = (-\tau_k, \emptyset, 0, 0, \dots, 0)$ at the depot, we extend each non-propagated label $L_i \in B_i$ at a given customer i to each unvisited neighbor $j \notin V(L_i)$ and mark the label L_i as propagated (subroutine PROPAGATELABEL in Algorithm 1). For each neighbor j, this yields a new label L_j with

$$\begin{aligned} &V(L_j) = V(L_i) \cup j, \\ &Y_\gamma(L_j) \text{ as in (2.2)}, \\ &R(L_j) = R(L_i) - \pi_j + C_p^j, \\ \text{and} \quad &T_j = \begin{cases} Y_0(L_j) - t_{0j}, & \text{if } i = 0; \\ T(L_i), & \text{otherwise.} \end{cases} \end{aligned}$$

If L_j is feasible, i.e., $j \notin V(L_i)$, $Y_\Gamma(L_j) \le u_j^k$ and $Y_\Gamma(L_j) - T(L_j) \le H_k$, we add it as a non-propagated label to the bucket B_j of customer j. In Algorithm 1, this check is referred to as ISFEASIBLE. Obviously, the number of generated labels grows drastically the bigger the instance. Thus, we include two features that keep this number comparatively low, which is crucial for achieving acceptable running times.
First, by looking at the current reduced cost of a label as well as the dual values and a lower bound of the costs at the customers that are still reachable, we can estimate the potential of this label (see HASPOTENTIAL in Algorithm 1). In the literature, these estimations a referred to as completion bounds, see, e.g., [21]. Doing so, we can avoid pursuing dead labels, i.e., labels that cannot be propagated to labels with negative reduced costs, as early as possible. The second and even more important aspect is the use of a dominance check, which allows to determine if one label induces redundancy for another one. The sequence of ℓ propagations of a label L_i is called an extension $e = (e_1 \dots, e_\ell)$, where we first propagate to node e_1, then to e_2 and so on. If all labels created along the way are feasible, then e is a valid extension of L_i, which we denote as $e \in E(L_i)$.
Now, comparing two labels L_i^1 and L_i^2 with respect to their sets of valid extensions, we can define the following useful concept.

Definition 3.1. Given two labels L_i^1 and L_i^2 at customer i, we say that the label L_i^1 *dominates* L_i^2, if $E(L_i^2) \subseteq E(L_i^1)$ and for every pair of labels L_j^1, L_j^2 resulting from an extension $e \in E(L_i^2)$, it follows that $R(L_j^1) \le R(L_j^2)$.

Thus, if we find two labels L_i^1, L_i^2 with the above property, we can safely delete L_i^2 from the label list of customer i, as it can never lead to a better solution than L_i^1. Unfortunately, it is generally intractable to check all possible extensions in order to detect such label pairs. Nevertheless, the following result presents a sufficient condition for dominance that can be checked efficiently and is helpful in practice.

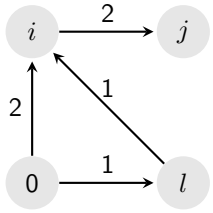

Figure 3.1: Example for failure of condition (v').

Theorem 3.2. A label L_i^1 dominates another label L_i^2 in the same node i if

$$R(L_i^1) \leq R(L_i^2), \tag{i}$$
$$V(L_i^1) \subseteq V(L_i^2), \tag{ii}$$
$$T(L_i^1) \geq T(L_i^2), \tag{iii}$$
$$Y_\gamma(L_i^1) \leq Y_\gamma(L_i^2) \qquad \forall \gamma = 0, \ldots, \Gamma, \tag{iv}$$
$$\text{and} \quad Y_\gamma(L_i^1) - Y_{\gamma-1}(L_i^1) \leq Y_\gamma(L_i^2) - Y_{\gamma-1}(L_i^2) \qquad \forall \gamma = 1, \ldots, \Gamma. \tag{v}$$

In order to prove this result, we need to show that each feasible extension $e_1, \ldots, e_\ell$ (with $e_\ell = j$) of L_i^2 is also a feasible extension of L_i^1 such that for the respective resulting labels L_j^1 and L_j^2, it holds that $R(L_j^1) \leq R(L_j^2)$.
If our objective function consisted only of a standard travel time minimization term, conditions (i)–(iv) would already be sufficient, see Munari et al. [23, Proposition 1]. However, the delay-resistance terms (2.4) may lead to $R(L_j^1) > R(L_j^2)$ for some extensions. A first intuitive idea to avoid this could be to extend (i)–(iv) by additionally imposing the condition

$$Y_\Gamma(L_i^1) - Y_0(L_i^1) \leq Y_\Gamma(L_i^2) - Y_0(L_i^2), \tag{v'}$$

as these terms appear in the objective function. Indeed, for label L_j^1 to have less or euqal delay at customer j than label L_j^2, it is sufficient that (v') holds. However, to ensure that this property continues to hold after the next propagation, we require the stronger condition (v) of the label pair L_j^1 and L_j^2. In the following, we sketch why (v') is, unfortunately, not sufficient, and refer the reader to A for the detailed proof of Theorem 3.2. Consider the example network in Figure 3.1, where we let each arc have travel time 1 and travel delay values $\bar{t}$ as given at each respective arc. Furthermore, let node 0 denote the depot, $\Gamma = 2$, let all service times as well as their delays be 0, and let customers i and l have the same availability time window $[0, 10]$. Consider the labels L_i^1 of the path $(0, i)$ and L_i^2 of the path $(0, l, i)$. Obviously, $V(L_i^1) \subseteq V(L_i^2)$, $T(L_i^1) = T(L_i^2)$, and we have $Y(L_i^1) = (1, 3, 3)$ and $Y(L_i^2) = (2, 3, 4)$. Hence, $Y_\gamma(L_i^1) \leq Y_\gamma(L_i^2)$ holds for all $\gamma \in \{0, 1, 2\}$ and condition (v') is fulfilled as well. Furthermore, there could be dual values such that $R(L_i^1) = R(L_i^2)$ and, consequently, label L_i^1 would dominate label L_i^2. However, propagating both labels to customer j, we get $Y(L_j^1) = (2, 4, 6)$ and $Y(L_j^2) = (3, 5, 6)$, which yields

$$Y_\Gamma(L_j^1) - Y_0(L_j^1) = 4 > 3 = Y_\Gamma(L_j^2) - Y_0(L_j^2) \quad \Rightarrow \quad R(L_j^1) > R(L_j^2),$$

i.e., the property that L_j^1 has smaller or equal delay than L_j^2 is lost. Condition (v) provides a remedy for this issue, cf. A for the details.

It should be noted that with the current definition of a propagation step in Algorithm 1 (PropagateLabel, see the earlier description), Theorem 3.2 is stated only for the case of there being at most one time window per customer per day. Nevertheless, if we extend the propagation of a label to a new customer such that we create one label for each time window of that customer on the given day, the theorem holds analogously for the general case with multiple time windows per customer. To do so, we simply have to adjust the l_i^k term in (2.2); we omit the quite straightforward (if somewhat technical) details.

3.3 Algorithm Engineering

The labeling algorithm alone solves the pricing problem and, therefore, is sufficient for the column generation method to work correctly. However, its potentially large run time naturally evokes a high interest in heuristic options. In this section, we briefly discuss two types of heuristics, namely, *primal heuristics* and *pricing heuristics*. While the latter help to solve the pricing problem, the former are used to find good feasible solutions for the RVRP-HTW. In other words, a pricing heuristic helps to approach a useful valid lower bound for the MP, while a primal heuristic is used to improve the upper bound. Furthermore, we use some of the primal heuristics to warm-start our branch-and-price method, or more precisely, to define the initial $\mathcal{P}'$.

3.3.1 Primal Heuristics

In order to start the reduced-cost pricing, solvability of the restricted master problem (RMP) is mandatory. One way to guarantee this solvability—if the master problem (MP) is solvable at all—is to use *Farkas Pricing* [20]. Although we implemented this method, it is primarily used as a fall-back. Finding a feasible solution is NP-hard in general [7, 10], but for our test instances (cf. Section 4), the following primal heuristics were almost always successful.

First, we try to find at least one feasible solution of the RVRP-HTW with a *greedy nearest neighbor* heuristic and add it to the initial $\mathcal{P}'$. We iterate over the arbitrarily sorted days. For each day we want to find a tour that only consists of customers that have not already been visited on a previous day. Starting at the depot, we set all $\Gamma + 1$ arrival times to 0. While the duration does not exceed W, we search for the *next reachable* customer in the current neighborhood and add it to the current tour. Then we update the arrival times. In the classical VRP or TSP *next reachable* would just refer to the nearest neighbor, but we have to consider time windows as well as robustness. Therefore, we extend the tour by the customer that has the lowest latest arrival time under consideration of his time windows. The rate of getting feasible solutions by this algorithm strongly correlates with the flexibility of the customers.

On instances with a lower amount of (or small) time windows per customer this algorithm does likely not find a feasible solution due to its very limited foresight. Subsequently, we implemented another greedy algorithm which does not concentrate its greedy manner on the time efficiency of a simple tour but rather on the flexibility of the customers. We call it *greedy dispatching*:

Instead of creating tours iteratively for each day, this algorithm does it simultaneously. The idea is to sort all customers by their flexibility (sum of all time window lengths) to first add the least flexible customers to a tour. Afterwards, initialize an empty tour for each day and create a dynamic ranking of the days that sorts by the following value (of day k):

$$\begin{aligned}\mathcal{F}_k = &\text{ [Number of available customers on day } k \text{ (static)]} \\ &- \text{[Number of available customers already assigned to a different day (dynamic)]}\end{aligned}$$

Put in words, this value describes the (remaining) potential of the length of a tour on day k. In the iteration step of customer v first try to assign v to the tour of the day with the least remaining potential over all days that he is available on. If this is not possible, check another possible day that stands lower in the ranking. The assigning step on a certain day adds v to the spot in the current tour that leads to the cheapest feasible result. When v has been assigned to a day k, decrement the remaining potential of all days that v is available on except for day k. If he could not be assigned to any of his possible days, decrement the remaining potential of all of them.
For instances with 'inflexible' customers, greedy dispatching yields goods results. On the other hand, if the tours need to be time efficient to combine into a feasible solution then this algorithm has some weakness. Hence, we run greedy dispatching as well as greedy nearest neighbor to get a good set of starting columns.
Furthermore, it is desirable to find feasible solutions with a low objective value to not only ensure solvability of the RMP but also to initialize a decent upper bound. Because feasibility as well as solution quality are both hard goals to achieve on their own, we split our initialization into two parts. In the first part, we primarily focus on finding feasible solutions while only secondarily caring for their quality. Afterwards, we improve these solutions with the help of local search algorithms that just swap customers around between the tours.
A way to improve the upper bound during the branch-and-price algorithm is to obtain a feasible integral solution based on the fractional solution of the current reduced master problem. We implemented an approach that comes down to rounding fractional variables similar to [30].
In particular, we do not round the tour variables x, but the vehicle assignment variables (3.1) that represent which customer gets served by which vehicle in a solution x, see [24] for more details. If x is fractional, these variables can also be fractional. We start by sorting the vehicle assignment variables with a non-zero value in descending order and use this order to create new tours similar to the greedy dispatching heuristic. In each iteration, we are given a variable, e.g., of customer i and day k. If customer i has not yet been assigned to a tour, we greedily try to add it to the tour of day k and continue with the next variable. In the end, we try to add unassigned customers and again improve the solution with local search algorithms.
Since the outcome of this heuristic strongly depends on the current RMP solution, we call it multiple times in the branch-and-price algorithm.

3.3.2 Pricing

Although we need an exact pricing algorithm to be able to prove or disprove optimality by finding a tour with minimum reduced costs, it is advisable to first search for any tours with negative reduced costs with the help of more efficient heuristic approaches. A natural idea that we implemented is to simplify the labeling algorithm, and thus trade-off solution quality for efficiency. To that end, we focused on two components: heuristic preprocessing and a heuristic dominance check.
For the former, it is common to focus on limited neighborhoods, i.e., delete/ignore arcs that appear unlikely to be part of optimal solutions. In addition to very long arcs, our time window structure motivates us to delete arcs that would lead to long waiting times.
As for designing a heuristic dominance check, there are multiple possibilities. By simply relaxing the conditions that identify whether a label dominates another, we can drastically improve the efficiency of the algorithm, at the cost of correctness: We move from the exact "Label B cannot be extended into a better tour than label A" to the heuristic "Label B will likely not be extended into a better tour than label A". The basic observation is that condition (ii) is only fulfilled by a small fraction of label

pairs. Thus, by leaving out this condition and only requiring (i) and (iii) –(v) to be satisfied for a label to "dominate" another, we can expect many more labels to be purged and the overall running time to improve significantly. Moreover, in practice, it appears that indeed not many important tours are overlooked as a consequence of this heuristic restriction.
Another common heuristic pricing approach is local search. The idea is to locally modify tours that have been added to the model earlier to obtain tours with lower reduced costs. We employ four different simple modifications that we try to apply to a given tour p:

1. Delete a customer from p.
2. Add a customer to p.
3. Switch positions of two customers in p.
4. Exchange a customer from p by one that was not included in p.

It is worth pointing out that there are tours in $\mathcal{P}'$ which seem to be more amenable for this procedure than others. In particular, due to the optimality of $(x^*, (\pi^*, \tau^*))$ for the RMP and the complementary slackness, we know that for each $p \in \mathcal{P}'_{>0} := \{p \in \mathcal{P}' : x^*_p > 0\} \subseteq \mathcal{P}'$, it holds that

$$C_p - \sum_{i \in C} \pi^*_i a_{ip} - \tau^*_k = 0,$$

i.e., even the slightest decrease of the tour costs by one of the mentioned modifications induces negative reduced costs and would thereby immediately result in a new column that can be added to the RMP.
Due to the simplicity of the above modifications, the local search pricing is very efficient compared to the labeling algorithm. Moreover, especially in the early pricing iterations (when $\mathcal{P}'$ is small), the success rate of this method is high. A major disadvantage of local search schemes is that such an algorithm can get stuck in local minima. In our application, we can usually overcome this issue by alternating local search with other heuristics or the exact labeling algorithm.

4 Experimental Evaluation

In this section, we will discuss computational experiments with our branch-and-price solver for the RVRP-HTW set partitioning formulation. In particular, we focus on the following questions:

(Q1) What can be said about the overall performance of our algorithm?

(Q2) What are the practical implications of our novel delay-resistant objective function?

After describing the test instances we worked with (cf. Section 4.1), the subsequent discussion will be divided into two parts. First, we address question (Q1) and assess the performance of our branch-and-price implementation, in terms of running time and solution quality; see Section 4.2. In the second part, we will take a closer look at the influence of the proposed delay-resistance objective function term on the solutions and its implications for practical application; see Section 4.3. To that end, we conduct Monte Carlo simulations to evaluate multiple realizations of the uncertainties for our computed tours, and will also assess scenarios in which the amount of uncertainty exceeds that which was assumed for the optimization.

All experiments were conducted on a system running Ubuntu 20.04 with an AMD Ryzen Threadripper 3960X 24-core processor and 128 GB of memory. We implemented our method using SCIP 8.0.1 with the SoPlex 6.0.1 LP-solver and set a time limit of one hour (3600 seconds) for all instances. The code to our project and our instances can be found on GitHub[1].

4.1 Data Set

Our instances are based on real-world data provided by our project partner from the logistics industry. This data originates from actual customer geolocations (real addresses) and delivery regions (each serviced by a single depot) of a German company, combined with historical statistical data and empirical parameter values.

For each base instance, a random set of 500 customers was chosen. In each such set, all customers are located in the same geographic area which loosely corresponds to a specific delivery area. The driving times between any two customers were calculated based on the existing road network. To account for different driving times in the course of a day, historical traffic data was applied to the routing graph for different times of the day. This results in different driving times, reflecting, e.g., rush hour traffic in mornings or afternoons. We use the difference between the largest and the smallest value of every edge as our travel time delays. The service times and associated delays are integers chosen uniformly at random from $[1, 5]$ minutes and $[1, 10]$ minutes, respectively.

Moreover, since road network topologies in urban areas typically differ from those in rural areas, we generated different instances for urban, suburban and rural areas to cover the various delivery environments encountered in real life (by our project partner). However, since we did not detect any systematic difference with respect to our evaluation criteria, we calculated the average values over these different areas in the following evaluations.

In every instance, every customer has one availability time window per day, and each instance consist of three weeks, i.e., 15 successive working days. These time windows were also generated uniformly at random so that their length varies between one and three hours. The random availability window starting times were all selected between 8 a.m. and 4 p.m. on the hour, following a uniform distribution. Finally, from the above-described base instances, we generated further smaller instances by reducing the number of days, customers, and/or availability time windows. For each instance, we will henceforth denote its size parameters by r (number of customers), d (number of days) and w (number of time windows per customer).

4.2 Computational Results: Overall Performance and Scalability

In the following, we focus on the subset of the instances with parameters $r \in \{20, 40, 60\}$, $d \in \{5, 10\}$ and $w \in \{0.2d, 0.4d, 0.6d, 0.8d, d\}$. For each parameter set (r, d, w), there are 8 instances. The model objective function was defined using $\alpha^d = 0.2$, $\alpha^t = 0.8$, $W = 0$ and $\Gamma = 7$, and we impose a work time limit of eleven hours per day—in minutes, $H = 660$. Variations of the model parameters will be considered in the next subsection.

Table 4.1 summarizes our experiments with the above parameter setup. The columns provide the instance size parameters r, d and w, average primal bounds (PB), number of solved instances to optimality (# Solved) and the average time in seconds taken for these solved instances (Time). If some of the 8 instances are not solved to optimality in the time limit, we denote the number of unsolved instances where we at least solved the root master problem (# SolvedRootMP) and the average remaining optimality

[1] https://github.com/TimNiemann/RVRP-HTW

Table 4.1: *Computational results for branch-and-price solver on different instance sizes.*

r	d	w	PB	# Solved	Time	# SolvedRootMP	Gap
20	5	5	10580.50	8	60.01	–	–
20	5	4	12479.05	7	215.43	1	2.20
20	5	3	14309.80	8	25.60	–	–
20	5	2	17955.65	8	3.12	–	–
20	5	1	25770.98	8	<0.01	–	–
20	10	10	10546.48	8	58.15	–	–
20	10	8	11904.22	8	509.56	–	–
20	10	6	13584.85	8	21.38	–	–
20	10	4	16858.70	8	4.20	–	–
20	10	2	25191.38	8	<0.01	–	–
40	5	5	17519.28	1	3600.00	3	3.10
40	5	4	19525.25	1	804.60	5	4.56
40	5	3	21541.75	3	1170.10	5	7.96
40	5	2	25799.70	3	889.50	5	2.22
40	5	1	33325.65	8	0.07	–	–
40	10	10	17501.22	2	2817.95	2	4.60
40	10	8	19081.20	2	758.15	2	5.95
40	10	6	21078.35	2	163.05	5	8.06
40	10	4	25012.20	5	982.56	3	7.37
40	10	2	32122.00	8	0.10	–	–
60	5	5	24399.48	0	–	2	2.95
60	5	4	26304.32	0	–	2	4.50
60	5	3	28172.38	0	–	6	4.68
60	5	2	32840.52	3	1874.07	5	4.70
60	5	1	41744.57	8	1.15	–	–
60	10	10	24331.02	0	–	2	2.85
60	10	8	26000.50	0	–	2	4.65
60	10	6	29071.25	1	1847.00	4	7.75
60	10	4	31668.05	0	–	7	3.99
60	10	2	40877.77	8	3.91	0	–

gap of only these unsolved instances (Gap) in percent. If either all or no instances of a parameter set could be solved to optimality, we note a "–", instead of the averages, for all values that can therefore not be calculated.

We make several observations from Table 4.1 concerning question (Q1), the overall performance of the algorithm: Firstly, it is noted that instances with fewer time windows are comparatively easier to solve. However, these instances also exhibit a worse primal bound, indicating that the quality of the solution may be compromised. Secondly, the impact of additional days on the running time does not demonstrate an exponential relationship which we would otherwise expect with a larger instance size. Moving on

Table 4.2: Monte Carlo evaluation for instance B-100 and uncertainty factor $\delta = 1$

Γ	W	α^d	α^t	duration	% serv.	% trav.	delay	# delays	fail time	# fails
6	0	0	1	154329	21.87	36.69	17538	64.42	0	0
6	0	0.1	0.9	163232	20.73	34.83	15924	61.93	0	0
6	0	0.2	0.8	195938	17.24	30.00	10563	51.49	0	0
6	0	0.4	0.6	209967	16.13	28.96	8697	47.44	0	0
6	0	0.6	0.4	225321	15.03	29.19	6176	41.07	0	0
6	0	0.8	0.2	246757	13.68	30.08	4534	39.11	0	0
6	0	1	0	247913	13.67	38.71	4168	35.95	0	0
0	0	0.2	0.8	140095	24.21	39.29	31436	73.71	2617	5.97
2	0	0.2	0.8	192056	17.57	30.43	12194	53.54	0	0
4	0	0.2	0.8	198636	17.03	29.75	10029	50.52	0	0
6	0	0.2	0.8	195938	17.24	30.00	10563	51.49	0	0
8	0	0.2	0.8	198636	17.03	29.80	9853	50.51	0	0
10	0	0.2	0.8	198636	17.03	29.80	9853	50.51	0	0
6	0	0.2	0.8	198636	17.03	29.80	9853	50.51	0	0
6	600	0.2	0.8	158202	21.40	36.20	311	2.40	0	0
6	1200	0.2	0.8	150714	22.51	37.45	1	0.05	0	0
6	1800	0.2	0.8	150667	22.53	37.45	0	0	0	0
6	2400	0.2	0.8	154336	22.00	36.62	0	0	0	0
6	3000	0.2	0.8	154336	22.00	36.62	0	0	0	0
6	3600	0.2	0.8	150667	22.53	37.45	0	0	0	0

to the specific problem being addressed, the RVRP-HTW is already challenging to solve for smaller instances. This indicates that the difficulty of the problem does not only correlate with the size of the instance. Furthermore, when considering instances with fully flexible time windows (i.e., instances where every customer has a time window for every single day), they are sometimes easier to solve compared to instances with fewer time windows per customer. However, as the number of time windows per customer increases, the runtime generally exhibits an exponential growth, implying that the computational effort required escalates rapidly.

4.3 Influence of the Objective Function Parameters

In order to analyze the dependence of our RVRP-HTW solutions on specific objective function parameters and to assess the quality of these solutions from a practical viewpoint—i.e., to address question (Q2)—we will evaluate the results of Monte-Carlo simulations in the following. More specifically, we simulate multiple realizations of the uncertainties for our calculated tours and evaluate the resulting performance of these tours. To that end, in each scenario, the realized service and travel times are drawn uniformly from the interval $[s_i, \delta\, \tilde{s}_i]$ for every customer and from the interval $[t_{ij}, \delta\, \tilde{t}_{ij}]$ for every arc, respectively. The uncertainty factor δ determines how large the ratio of actually realized travel and service time is in

Table 4.3: Monte Carlo evaluation for instance B-200 and uncertainty factor $\delta = 1$

Γ	W	α^d	α^t	duration	% serv.	% trav.	delay	# delays	fail time	# fails
6	0	0	1	262872	25.48	36.00	60500	147.43	1	0.02
6	0	0.1	0.9	256878	26.04	38.45	39196	136.95	0	0
6	0	0.2	0.8	256842	26.07	35.88	34938	136.75	0	0
6	0	0.4	0.6	267817	25.00	36.91	28957	132.00	0	0
6	0	0.6	0.4	267961	24.92	40.04	24753	128.55	0	0
6	0	0.8	0.2	270409	24.72	43.37	22981	128.30	0	0
6	0	1	0	275492	24.30	51.70	21898	127.94	0	0
0	0	0.2	0.8	255703	26.18	35.38	71015	151.17	3878	9.60
2	0	0.2	0.8	267762	25.04	35.60	40675	137.83	6	0.07
4	0	0.2	0.8	268910	24.90	36.27	32734	132.08	0	0
6	0	0.2	0.8	256842	26.07	35.88	34938	136.75	0	0
8	0	0.2	0.8	267110	25.04	36.54	32555	133.07	0	0
10	0	0.2	0.8	262835	25.36	35.11	35698	137.14	0	0
6	0	0.2	0.8	256842	26.07	35.88	34938	136.75	0	0
6	600	0.2	0.8	258269	25.92	36.02	1149	8.65	0	0
6	1200	0.2	0.8	254927	26.21	34.47	5	0.08	0	0
6	1800	0.2	0.8	260171	25.70	34.46	0	0	0	0
6	2400	0.2	0.8	262874	25.48	36.00	0	0	1	0.02
6	3000	0.2	0.8	253495	26.40	36.66	0	0	0	0
6	3600	0.2	0.8	253495	26.40	36.66	0	0	0	0

the Monte-Calo simulation compared to the values assumed during optimization. Thereby, we will also assess routing performance in scenarios that exceed the degree of uncertainty our robust solutions are protected against.

We compute initial solutions using the heuristic methods described in Section 3.3.1 to our vehicle routing problem for a number of given parameter sets on two example base instances we will be referring to as B-100 and B-200, which correspond to delivery regions in Bavaria with sizes given by $(r, d, w) = (100, 10, 2)$ and $(r, d, w) = (200, 10, 2)$, respectively. In the Tables 4.2, 4.3, 4.4 and 4.5, each row presents the average results over 100 randomly drawn scenarios for a given parameter setting, separately for each of the two instances.

Generally, we assume $\alpha^d + \alpha^t = 1$ and

$$\alpha^d = 0.2, \quad \alpha^t = 0.8, \quad W = 0, \quad \Gamma = 6, \tag{4.1}$$

but in each different setting, we change one specific parameter while leaving all others at these default values.

In each scenario, we get the actual arrival time $\tilde{y}_i$ per customer i and compare this with the appointment window $[L_i, U_i]$ that we have given the customer. Similarly to our objective function, we calculate the average delay $\sum_{i \in C} (\tilde{y}_i - U_i)_+$ over all evaluated scenarios. Situations in which a vehicle arrives outside

Table 4.4: Monte Carlo evaluation for instance B-100 and uncertainty factor $\delta = 1.5$

Γ	W	α^d	α^t	duration	% serv.	% trav.	delay	# delays	fail time	# fails
6	0	0	1	152915	27.18	37.16	31362	69.04	19	0.13
6	0	0.1	0.9	175174	23.72	33.34	21199	58.38	0	0
6	0	0.2	0.8	200291	20.67	29.72	14997	50.79	0	0
6	0	0.4	0.6	211594	19.60	28.91	13255	47.77	0	0
6	0	0.6	0.4	237584	17.38	28.26	9212	40.55	5	0.06
6	0	0.8	0.2	248273	16.64	30.05	6922	37.61	0	0
6	0	1	0	249570	16.63	38.61	6360	36.42	0	0
0	0	0.2	0.8	142838	29.08	38.75	50139	75.71	5438	7.70
2	0	0.2	0.8	174672	23.77	33.28	22515	58.91	0	0.01
4	0	0.2	0.8	200291	20.68	29.68	15316	50.96	0	0
6	0	0.2	0.8	201112	20.68	29.49	16319	52.96	1	0.04
8	0	0.2	0.8	200291	20.67	29.72	14997	50.79	0	0
10	0	0.2	0.8	200291	20.67	29.72	14997	50.79	0	0
6	0	0.2	0.8	200944	20.63	29.36	16796	53.81	0	0
6	600	0.2	0.8	174874	23.72	33.32	1747	9.86	0	0
6	1200	0.2	0.8	152915	27.18	37.16	304	1.74	19	0.13
6	1800	0.2	0.8	155678	26.74	36.61	10	0.12	27	0.24
6	2400	0.2	0.8	155678	26.74	36.61	0	0	27	0.24
6	3000	0.2	0.8	152115	27.34	37.43	0	0	19	0.13
6	3600	0.2	0.8	152545	27.24	37.31	0	0	19	0.13

of a feasible time window are denoted as "fails", since we consider these failed deliveries. We also note the fail time, which is the average total difference between the arrival time and the end of the time window over all customers.

The tables contain averaged values obtained from the Monte-Carlo simulations. These simulations were conducted to analyze various aspects of the system under consideration. The parameter Γ represents the maximum number of delays per day that the system plans for, while W denotes the length of the appointment windows allocated to customers. For all average values, the unit of time is seconds. Following, the tables present the total tour duration and the percentage of that duration accounted for by service and travel time. The remaining time is spent waiting for a customer's time window to begin. Additionally, the tables provide information on the total delay time and the average number of delays per scenario. The average service time, however, is not listed in the tables. This omission is justified because the average service time remains constant regardless of the specific parameter values. This is because each customer must be visited once, and over multiple simulations, the uncertainty in service times balances out, resulting in consistent average values.

In almost all scenarios with $\delta = 1$ (cf. Tables 4.2 and 4.3), there are no failed deliveries, i.e., no arrival times are outside of a time window specified by the customer. Indeed, a small robustification with $\Gamma = 2$ is already sufficient to almost completely avoid such failures.

Table 4.5: Monte Carlo evaluation for instance B-200 and uncertainty factor $\delta = 1.5$

Γ	W	α^d	α^t	duration	% serv.	% trav.	delay	# delays	fail time	# fails
6	0	0	1	266983	30.68	35.59	100218	150.19	563	1.82
6	0	0.1	0.9	261228	31.33	34.68	58831	138.71	26	0.25
6	0	0.2	0.8	259596	31.50	35.63	54501	138.53	22	0.17
6	0	0.4	0.6	269964	30.29	36.97	43560	132.26	7	0.11
6	0	0.6	0.4	270076	30.20	39.68	37835	129.89	12	0.13
6	0	0.8	0.2	273334	29.95	43.38	34966	129.63	0	0
6	0	1	0	278403	29.47	51.47	34272	130.06	19	0.29
0	0	0.2	0.8	259298	31.57	35.03	115009	154.72	11185	16.68
2	0	0.2	0.8	261429	31.38	35.81	65976	145.37	464	1.60
4	0	0.2	0.8	270408	30.28	36.14	53483	135.79	7	0.15
6	0	0.2	0.8	259596	31.50	35.63	54501	138.53	22	0.17
8	0	0.2	0.8	269822	30.39	36.36	49486	134.22	0	0
10	0	0.2	0.8	266050	30.67	35.20	50584	136.86	13	0.08
6	0	0.2	0.8	259596	31.50	35.63	54501	138.53	22	0.17
6	600	0.2	0.8	256828	31.81	37.29	7698	34.04	0	0.02
6	1200	0.2	0.8	257958	31.70	34.14	1440	7.42	52	0.44
6	1800	0.2	0.8	264627	30.90	33.93	566	2.08	365	1.42
6	2400	0.2	0.8	264670	30.91	34.49	60	0.34	185	0.93
6	3000	0.2	0.8	266986	30.68	35.59	11	0.07	563	1.82
6	3600	0.2	0.8	266986	30.68	35.59	0	0.01	563	1.82

With $\delta = 1.5$ (cf. Tables 4.4 and 4.5) we see similar effects, but failures can no longer be completely avoided. However, as soon as delays are penalized in the objective function, i.e., $\alpha^d > 0$, $\Gamma > 0$ and an appropriately small appointment window length W, failures become negligibly rare. Thus, we can conclude that minimizing delays also implicitly reduces fail time.

Next, consider Figures 4.1 and 4.2, which plot the travel times and delay times for B-100 and B-200 calculated from Tables, respectively, over different values for α^d on the horizontal axis. For both instance sizes, we can see that a slight inclusion of the delay-resistance term in the minimization objective (i.e., with $\alpha^d \ll \alpha^t$) does not significantly influence the travel time, and vice versa. A mixture of the two objective function parts should therefore be selected for a balanced planning. This is why we decided to give a larger weight α^t to minimizing travel time in (4.1), as this is the classic objective function from the literature and because small weights α^d for the delay-term already provide a significant decrease of delays.

Moreover, when $\alpha^d \neq 0$, the accumulated delay generally decreases when increasing the robustness parameter Γ. However, as depicted in Figure 4.3, there apparently is a value for Γ beyond which a further increase only has a marginal influence on reducing the delay.

By construction, the delay also decreases when increasing the value for the appointment window length W, since this allows to tolerate more deviation for the arrival times. In this case, and with

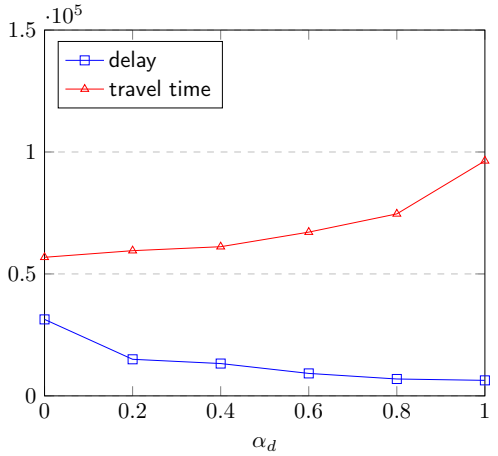

Figure 4.1: Travel time and delay for B-100 and $\Gamma = 6$, $W = 0$.

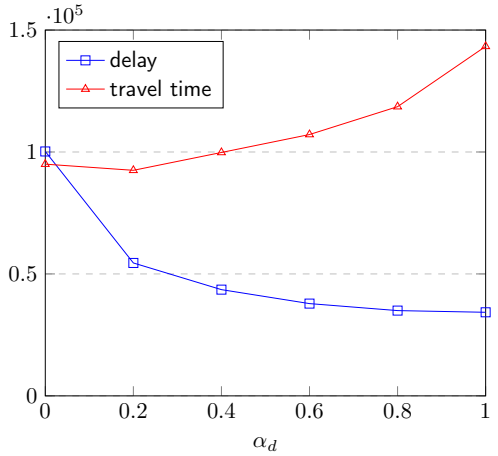

Figure 4.2: Travel time and delay for B-200 and $\Gamma = 6$, $W = 0$.

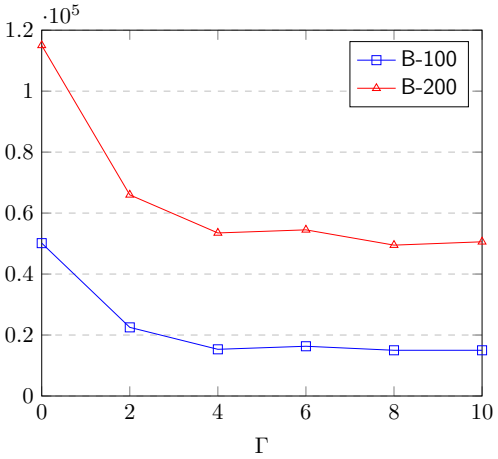

Figure 4.3: Delay for B-100 and B-200 with $\alpha^d = 0.2$, $W = 0$ for different values of Γ.

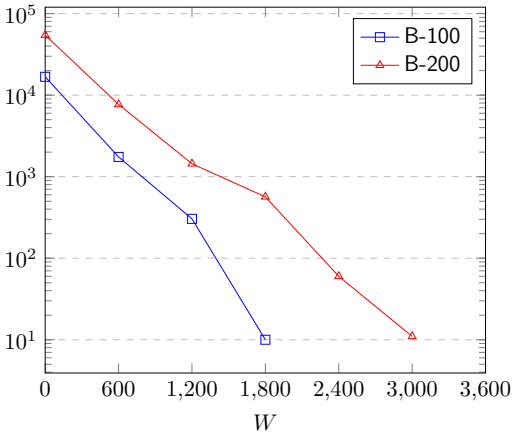

Figure 4.4: Delay for B-100 and B-200 with $\alpha^d = 0.2$, $\Gamma = 6$ for different values of W.

a robustness of $\Gamma = 6$, we are able to reduce the accumulated delays to nearly zero, see Figure 4.4. It is worth noting that Figure 4.4 also clearly shows that the delay falls roughly exponentially when linearly increasing the appointment window length; note the logarithmic scale on the vertical axis.

5 Concluding Remarks

We proposed a branch-and-price algorithm for a novel delay-resistant variant of the robust VRP with newly-introduced heterogeneous time windows. Unlike the typical routing a fleet of vehicles for one day, our model was motivated by the need to schedule one vehicle for a period of working days. While the

heterogeneity of customer-specified availability time windows added some level of complexity, it also enabled us to divide the pricing problem into smaller subproblems.
Our algorithm had to deal with two issues that were also motivated from practice. Firstly, all customers should be served within their hard availability time windows, and secondly, we aimed to minimize the delay regarding the appointment time windows derived from the RVRP-HTW solution and communicated to the customers. The first aspect was dealt with utilizing standard robust optimization, and adding the delay-resistance terms (2.4) to the common travel time minimization objective allowed us to generate solutions with (near-)minimal delays at the customers. The latter required an adjustment of the dominance check for the labeling algorithm used to solve the pricing problems. Our experimental evaluation demonstrated the practical benefit of including both objective terms; in particular, we can achieve significant reduction of delays with only a small trade-off increase in travel times, or vice versa. Moreover, the Monte-Carlo simulations showed that our solutions are robust even if the actual level of uncertainty w.r.t. travel and service times is higher than expected, or optimized for. Thus, even when requiring small appointment window lengths (as typically desirable), our approach can be expected to allow serving the customers without delay in most cases.

Acknowledgments We would like to acknowledge the funding by the German Federal Ministry for Education and Research, grant numbers 05M20MBA-LeoPlan, 05M20PDA-LeoPlan and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2163/1 - Sustainable and Energy Efficient Aviation – Project-ID 390881007.

References

[1] Yogesh Agarwal, Kamlesh Mathur, and Harvey M Salkin. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19(7):731–749, 1989.

[2] Artur Alves Pessoa, Michael Poss, Ruslan Sadykov, and François Vanderbeck. Branch-and-cut-and-price for the robust capacitated vehicle routing problem with knapsack uncertainty. HAL-ID: hal-01958184, 2018.

[3] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

[4] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1):49–71, 2003.

[5] Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact Branch-Price-and-Cut Algorithms for Vehicle Routing. *Transportation Science*, 53(4):946–985, 2019.

[6] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

[7] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[8] Jacques Desrosiers and Marco E. Lübbecke. A primer in column generation. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 1–32. Springer, New York, NY, 2005.

[9] Jacques Desrosiers, François Soumis, and Martin Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

[10] Moshe Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978, 1994.

[11] Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks: An International Journal*, 44(3):216–229, 2004.

[12] Bruce L Golden, Thomas L Magnanti, and Hien Q Nguyen. Implementing vehicle routing algorithms. *Networks*, 7(2):113–148, 1977.

[13] Bruce L Golden and WR Stewart. Vehicle routing with probabilistic demands. In *Computer Science and Statistics: Tenth Annual Symposium on the Interface, NBS Special Publication*, volume 503, pages 252–259, 1978.

[14] Maaike Hoogeboom, Yossiri Adulyasak, Wout Dullaert, and Patrick Jaillet. The robust vehicle routing problem with time window assignments. *Transportation Science*, 55(2):395–413, 2021.

[15] Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In *Column generation*, pages 33–65. Springer, 2005.

[16] Çağrı Koç, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, 249(1):1–21, 2016.

[17] Suresh Nanda Kumar and Ramasamy Panneerselvam. A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, 4(3):66–74, 2012.

[18] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358, 1992.

[19] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[20] Marco E. Lübbecke. Column Generation. In James J. Cochran, Louis A. Cox Jr., Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science (EORMS)*. John Wiley & Sons, Inc., New York, NY, 2010.

[21] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[22] Christopher Z Mooney. *Monte carlo simulation*. Number 116 in Quantitative Applications in the Social Sciences. Sage, 1997.

[23] Pedro Munari, Alfredo Moreno, Jonathan De La Vega, Douglas Alem, Jacek Gondzio, and Reinaldo Morabito. The Robust Vehicle Routing Problem with Time Windows: Compact Formulation and Branch-Price-and-Cut Method. *Transportation Science*, 53(4), 2019.

[24] Petra Mutzel, Tim Niemann, Lukas Schürmann, Sebastian Stiller, and Andreas M. Tillmann. Vehicle Routing with Heterogeneous Time Windows. To appear in *Proc. OR 2022*.

[25] Barin N. Nag, Bruce L. Golden, and Arjang A. Assad. Vehicle routing with site dependencies. In *Vehicle Routing: Methods and Studies, Studies in Management Science and Systems*, volume 16, 1988.

[26] Daniel A. Nera, Maichel M. Aguayo, Rodrigo De la Fuente, and Mathias A. Klapp. New compact integer programming formulations for the multi-trip vehicle routing problem with time windows. *Computers & Industrial Engineering*, 144, 2020. Art. No. 106399.

[27] Diego Pecin, Claudio Contardo, Guy Desaulniers, and Eduardo Uchoa. New Enhancements for the Exact Solution of the Vehicle Routing Problem with Time Windows. *INFORMS Journal on Computing*, 29(3):489–502, 2017.

[28] Ali Gul Qureshi, Eiichi Taniguchi, and Tadashi Yamada. An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):960–977, 2009.

[29] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[30] Marc Sol. *Column generation techniques for pickup and delivery problems*. PhD thesis, Technische Universiteit Eindhoven Netherlands, 1994.

[31] Marius Mihai Solomon. *Vehicle routing and scheduling with time window constraints: models and algorithms*. PhD thesis, University of Pennsylvania, 1984.

[32] Remy Spliet and Adriana F Gabor. The time window assignment vehicle routing problem. *Transportation Science*, 49(4):721–731, 2015.

[33] Thibaut Vidal, Gilbert Laporte, and Piotr Matl. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, (accepted), 2019.

[34] Laurence Wolsey. *Integer Programming*. John Wiley & Sons, 2nd edition, 2021.

A Proof of Dominance Theorem 3.2

Let L_i^1 and L_i^2 be labels at some node i that satisfy (i)–(v). We need to show that each extension $e \in E(L_i^2)$ also lies in $E(L_i^1)$ and that for the resulting labels L_j^1 and L_j^2, respectively, it holds that $R(L_j^1) \leq R(L_j^2)$. By induction, it suffices to show that for each extension $e = (j) \in E(L_i^2)$ of length 1, for any $j \in C$, it holds that $e \in E(L_i^1)$ and the respective resulting labels L_j^1 and L_j^2 obey conditions (i)–(v).

To that end, let $e = (j) \in E(L_i^2)$ for some $j \in C$. Since extensions only pertain to previously unvisited nodes, we may and do assume w.l.o.g. that $j \notin V(L_i^2)$, and as $V(L_i^2) \supseteq V(L_i^1)$ by (ii), we then also have $j \notin V(L_i^1)$. Furthermore, from (iv) (i.e., $Y_\gamma(L_i^1) \leq Y_\gamma(L_i^2)$ for all $\gamma \in \{0, \dots, \Gamma\}$) it follows that $Y_\gamma(L_j^1) \leq Y_\gamma(L_j^2) \leq u_i^k$, and together with (iii), we obtain $Y_\Gamma(L_j^1) - T(L_j^1) \leq Y_\Gamma(L_j^2) - T(L_j^2) \leq H_k$. Consequently, L_i^1 is also a feasible label, i.e., $e \in E(L_j^1)$.

It remains to prove that L_j^1 and L_j^2 again obey (i)–(v). Obviously,

$$V(L_j^1) = V(L_i^1) \cup \{j\} \subseteq V(L_i^2) \cup \{j\} = V(L_j^2), \quad T(L_j^1) = T(L_i^1) \geq T(L_i^2) = T(L_j^2),$$

and by the recursive formula (2.2), $Y_\gamma(L_j^1) \leq Y_\gamma(L_j^2)$ for all $\gamma \in \{0, \dots, \Gamma\}$, i.e., (ii)–(iv) are indeed satisfied. Moreover, note that if (v) is now also satisfied, (i) holds as well, because under (ii)–(iv), $Y_\gamma(L_j^1) - Y_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^2) - Y_{\gamma-1}(L_j^2)$ for all $\gamma \in \{1, \dots, \Gamma\}$ implies (cf. (2.9)) that

$$\begin{aligned} R(L_j^1) &= R(L_i^1) - \pi_j^* + \alpha^d c_j (Y_\Gamma(L_j^1) - Y_0(L_j^1) - W)_+ + \alpha^t t_{ij} \\ &\leq R(L_i^2) - \pi_j^* + \alpha^d c_j (Y_\Gamma(L_j^2) - Y_0(L_j^2) - W)_+ + \alpha^t t_{ij} = R(L_j^2). \end{aligned}$$

The inequality holds because the parameters α^d and c_j are non-negative and for any $x \geq x'$, we have $\max\{0, x\} =: (x)_+ \geq (x')_+$. Thus, it now suffices to show that L_j^1 and L_j^2 obey (v), i.e., that

$$Y_\gamma(L_j^1) - Y_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^2) - Y_{\gamma-1}(L_j^2) \quad \forall \gamma = 1, \dots, \Gamma.$$

To that end, we will distinguish five cases in the following, each pertaining to a different assumption regarding which of the four variables $Y_\gamma(L_j^1)$, $Y_{\gamma-1}(L_j^1)$, $Y_\gamma(L_j^2)$ and $Y_{\gamma-1}(L_j^2)$ attains its lower bound l_j^k (the starting point of the availability time window of customer j, cf. (2.2)), and show that (v) holds in all of them:

1: $Y_\gamma(L_j^1) = l_j^k$ (and $Y_{\gamma-1}(L_j^1),\ Y_\gamma(L_j^2),\ Y_{\gamma-1}(L_j^2) \geq l_j^k$)

2: $Y_\gamma(L_j^2) = l_j^k$ (and $Y_\gamma(L_j^1),\ Y_{\gamma-1}(L_j^1),\ Y_{\gamma-1}(L_j^2) \geq l_j^k$)

3: $Y_{\gamma-1}(L_j^2) = l_j^k$ (and $Y_\gamma(L_j^1),\ Y_{\gamma-1}(L_j^1),\ Y_\gamma(L_j^2) \geq l_j^k$)

4: $Y_{\gamma-1}(L_j^1) = l_j^k$ (and $Y_\gamma(L_j^1),\ Y_\gamma(L_j^2),\ Y_{\gamma-1}(L_j^2) > l_j^k$)

5: $Y_\gamma(L_j^1),\ Y_{\gamma-1}(L_j^1),\ Y_\gamma(L_j^2),\ Y_{\gamma-1}(L_j^2) > l_j^k$

Note that in cases 1–4, we consider propagation with waiting time, whereas in case 5, there is no extra waiting time before the beginning of the availability time window.

As already shown, we know that (in all cases) $Y_\gamma(L_j^1) \leq Y_\gamma(L_j^2)$ and $Y_{\gamma-1}(L_j^1) \leq Y_{\gamma-1}(L_j^2)$, and by definition of the arrival times, we recognize that $Y_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^1)$ and $Y_{\gamma-1}(L_j^2) \leq Y_\gamma(L_j^2)$. Consequently, we get the following implications:

$$\underbrace{Y_\gamma(L_j^2) = l_j^k}_{\text{case 2}} \quad \Rightarrow \quad \underbrace{Y_\gamma(L_j^1) = l_j^k}_{\text{case 1}} \quad \Rightarrow \quad Y_{\gamma-1}(L_j^1) = l_j^k$$

and

$$\underbrace{Y_\gamma(L_j^2) = l_j^k}_{\text{case 2}} \quad \Rightarrow \quad \underbrace{Y_{\gamma-1}(L_j^2) = l_j^k}_{\text{case 3}} \quad \Rightarrow \quad Y_{\gamma-1}(L_j^1) = l_j^k.$$

Cases 1 and 2 induce $Y_\gamma(L_j^1) - Y_{\gamma-1}(L_j^1) = 0 \leq Y_\gamma(L_j^2) - Y_{\gamma-1}(L_j^2)$, and for case 3 we have

$$\begin{aligned} Y_\gamma(L_j^1) \leq Y_\gamma(L_j^2) \quad &\Leftrightarrow \quad Y_\gamma(L_j^1) - l_j^k \leq Y_\gamma(L_j^2) - l_j^k \\ &\Leftrightarrow \quad Y_\gamma(L_j^1) - Y_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^2) - Y_{\gamma-1}(L_j^2). \end{aligned}$$

Thus, in the first three cases, L_j^1 and L_j^2 indeed obey (v). Moreover, note that the above implications already prove the same if $Y_{\gamma-1}(L_j^1) = l_j^k$ (as in case 4) and any one of the other three variables attains the value l_j^k as well; therefore, in case 4, we can require $Y_{\gamma-1}(L_j^1)$ to be the only variable having value l_j^k.

For technical reasons, we leave case 4 for later, and first consider case 5, i.e., we now suppose that none of $Y_\gamma(L_j^1)$, $Y_{\gamma-1}(L_j^1)$, $Y_\gamma(L_j^2)$, $Y_{\gamma-1}(L_j^2)$ has the value l_j^k. Then, the l_j^k-terms in (2.2) are inactive, i.e., $Y_\gamma(L_j^1)$, $Y_{\gamma-1}(L_j^1)$, $Y_\gamma(L_j^2)$, $Y_{\gamma-1}(L_j^2) > l_j^k$. To simplify the remainder of the proof for case 5, we split the recursive formula (2.2) into two steps: For $d \in \{1, 2\}$, define

$$X_\gamma(L_j^d) \coloneqq s_i + t_{ij} + \begin{cases} Y_\gamma(L_i^d), & \text{if } \gamma = 0, \\ \max\{Y_\gamma(L_i^d), Y_{\gamma-1}(L_i^d) + \bar{s}_i\}, & \text{otherwise } (1 \leq \gamma \leq \Gamma), \end{cases} \tag{A.1}$$

from which it then follows that

$$Y_\gamma(L_j^d) \coloneqq \begin{cases} X_\gamma(L_j^d), & \text{if } \gamma = 0, \\ \max\{X_\gamma(L_j^d), X_{\gamma-1}(L_j^d) + \bar{t}_{ij}\}, & \text{otherwise } (1 \leq \gamma \leq \Gamma). \end{cases} \tag{A.2}$$

We can rewrite $X_\gamma(L_j^d)$ as

$$X_\gamma(L_j^d) = s_i + t_{ij} + Y_\gamma(L_i^d) + \hat{X}_\gamma(L_j^d),$$

where $\hat{X}_\gamma(L_j^d) \coloneqq 0$ if $\gamma = 0$, and $\hat{X}_\gamma(L_j^d) \coloneqq \max\{0, Y_{\gamma-1}(L_i^d) - Y_\gamma(L_i^d) + \bar{s}_i\}$ otherwise. We first proceed to prove that $X_\gamma(L_j^1) - X_{\gamma-1}(L_j^1) \leq X_\gamma(L_j^2) - X_{\gamma-1}(L_j^2)$. To that end, we rephrase

$$\begin{aligned} &X_\gamma(L_j^1) - X_{\gamma-1}(L_j^1) - (X_\gamma(L_j^2) - X_{\gamma-1}(L_j^2)) \\ &= Y_\gamma(L_i^1) + \hat{X}_\gamma(L_j^1) - (Y_{\gamma-1}(L_i^1) + \hat{X}_{\gamma-1}(L_j^1)) \\ &\quad - (Y_\gamma(L_i^2) + \hat{X}_\gamma(L_j^2) - (Y_{\gamma-1}(L_i^2) + \hat{X}_{\gamma-1}(L_j^2))) \\ &= \underbrace{Y_\gamma(L_i^1) - Y_{\gamma-1}(L_i^1) - (Y_\gamma(L_i^2) - Y_{\gamma-1}(L_i^2))}_{=:a} \\ &\quad + \underbrace{\hat{X}_\gamma(L_j^1) - \hat{X}_\gamma(L_j^2)}_{=:b} + \underbrace{\hat{X}_{\gamma-1}(L_j^2) - \hat{X}_{\gamma-1}(L_j^1)}_{=:c}. \end{aligned}$$

Since by assumption, the labels L_i^1 and L_i^2 satisfy condition (v), we immediately get that $a \leq 0$ and

$Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) + \bar{s}_i \geq Y_{\gamma-1}(L_i^2) - Y_\gamma(L_i^2) + \bar{s}_i$. Therefore,

$$\begin{aligned}
b &= \underbrace{\max\{0, Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) + \bar{s}_i\}}_{=:\max\{0,f^1\}} - \underbrace{\max\{0, Y_{\gamma-1}(L_i^2) - Y_\gamma(L_i^2) + \bar{s}_i\}}_{=:\max\{0,f^2\}} \\
&\leq \max\{0-0, 0-f^2, f^1-0, f^1-f^2\} \\
&= \max\{0, Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) + \bar{s}_i, Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) - (Y_{\gamma-1}(L_i^2) - Y_\gamma(L_i^2))\} \\
&\leq \max\{0, Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) + Y_\gamma(L_i^2) - Y_{\gamma-1}(L_i^2), \\
&\qquad Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) - (Y_{\gamma-1}(L_i^2) - Y_\gamma(L_i^2))\} \\
&= \max\{0, Y_{\gamma-1}(L_i^1) - Y_\gamma(L_i^1) - (Y_{\gamma-1}(L_i^2) - Y_\gamma(L_i^2))\} = \max\{0, -a\} \leq -a,
\end{aligned}$$

where for the second equality, we note that the term $0 - f^2$ can be dropped (since $\max\{0, f^2\} = f^2$ implies that $-f^2 \leq 0$), and for the subsequent inequality, we used that $\max\{0, f^2\} = 0$ implies $\bar{s}_i \leq Y_\gamma(L_i^2) - Y_{\gamma-1}(L_i^2)$. Consequently, $a + b \leq 0$. Moreover, for $\gamma = 1$, obviously $c = 0$, and otherwise (for $2 \leq \gamma \leq \Gamma$), we obtain

$$c = \max\{0, Y_{\gamma-2}(L_i^2) - Y_{\gamma-1}(L_i^2) + \bar{s}_i\} - \max\{0, \underbrace{Y_{\gamma-2}(L_i^1) - Y_{\gamma-1}(L_i^1)}_{\geq Y_{\gamma-2}(L_i^2) - Y_{\gamma-1}(L_i^2) \text{ due to (v)}} + \bar{s}_i\} \leq 0.$$

Hence, $a + b + c \leq 0$, which shows that indeed $X_\gamma(L_j^1) - X_{\gamma-1}(L_j^1) \leq X_\gamma(L_j^2) - X_{\gamma-1}(L_j^2)$. Using a completely analogous line of argumentation, it is now easy to show that

$$Y_\gamma(L_j^1) - Y_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^2) - Y_{\gamma-1}(L_j^2)$$

holds as well; we omit the straightforward details that complete the proof for case 5.

It remains to consider case 4. In the following, we potentially underestimate $Y_{\gamma-1}(L_j^1) = l_j^k$ by leaving out one term from the maximum in (2.2), and thus actually show a stronger statement than (v), from which (v) follows directly. We retain the definitions (A.1) and (A.2) from before, and analogously define

$$X'_\gamma(L_j^1) \coloneqq s_i + t_{ij} + \begin{cases} Y'_\gamma(L_i^1), & \text{if } \gamma = 0, \\ \max\{Y'_\gamma(L_i^1), Y'_{\gamma-1}(L_i^1) + \bar{s}_i\}, & \text{otherwise } (1 \leq \gamma \leq \Gamma - 1) \end{cases}$$

and

$$Y'_\gamma(L_j^1) \coloneqq \begin{cases} X'_\gamma(L_j^1), & \text{if } \gamma = 0, \\ \max\{X'_\gamma(L_j^1), X'_{\gamma-1}(L_j^1) + \bar{t}_{ij}\}, & \text{otherwise } (1 \leq \gamma \leq \Gamma - 1). \end{cases}$$

In contrast to the previously discussed case 5, we only know $Y_{\gamma-1}(L_j^1) \geq Y'_{\gamma-1}(L_j^1)$, since now, the first term in the maximum in (2.2) is active for $Y_{\gamma-1}(L_j^1) = l_j^k$. However, importantly, the other three variables $Y_\gamma(L_j^1)$, $Y_\gamma(L_j^2)$ and $Y_{\gamma-1}(L_j^2)$ *cannot* correspond to the first term in the respective maximum, because they each are strictly larger than l_j^k by assumption of the present case 4. By these two arguments, we can drop the first (l_k^k-)term in the maximum for all four variables and proceed analogously to case 4 to prove that $X_\gamma(L_j^1) - X'_{\gamma-1}(L_j^1) \leq X_\gamma(L_j^2) - X_{\gamma-1}(L_j^2)$ and, again with analogous argumentation and using $-Y_{\gamma-1}(L_j^1) \leq -Y'_{\gamma-1}(L_j^1)$, that

$$Y_\gamma(L_j^1) - Y_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^1) - Y'_{\gamma-1}(L_j^1) \leq Y_\gamma(L_j^2) - Y_{\gamma-1}(L_j^2);$$

which concludes the proof in case 4.

Thus, we have now shown that the propagated labels L_j^1 and L_j^2 always satisfy (v), and as detailed at the beginning, this conpletes the proof. □