Gutenberg School of Management and Economics

& Research Unit “Interdisciplinary Public Policy”

Discussion Paper Series

Exact Solution of the Vehicle Routing Problem With Drones

Jeanette Schmidt, Christian Tilk, Stefan Irnich

May, 2023

Discussion paper number 2311

Johannes Gutenberg University Mainz
Gutenberg School of Management and Economics
Jakob-Welder-Weg 9
55128 Mainz
Germany
https://wiwi.uni-mainz.de/

Contact details

Jeanette Schmidt
Chair of Logistics Management
Johannes Gutenberg University
Jakob-Welder-Weg 9
55128 Mainz
Germany
sjeanett@uni-mainz.de

Christian Tilk
Department of Business Decisions and Analytics
University of Vienna
Kolingasse 14–16
AT-1090 Vienna
Austria
christian.tilk@univie.ac.at

Stefan Irnich
Chair of Logistics Management
Johannes Gutenberg University
Jakob-Welder-Weg 9
55128 Mainz
Germany
irnich@uni-mainz.de

Exact Solution of the Vehicle Routing Problem With Drones

Jeanette Schmidt[a,*], Christian Tilk[b], Stefan Irnich[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*
[b] *Department of Business Decisions and Analytics, University of Vienna, Kolingasse 14–16, AT-1090 Vienna, Austria.*

Abstract

The vehicle routing problem with drones (VRP-D) is an extension of the capacitated vehicle routing problem, in which the fleet consists of trucks equipped with one drone each. A truck and its drone can either move together or separately. To operate alone, a truck can release its drone at the depot or at a customer location and likewise pick it up at a later location visited by the same truck. In this way, both trucks and drones deliver goods to customers working together as synchronized working units. A feasible route has to satisfy the capacity constraints of both the truck and the drone. The VRP-D consists of finding a minimum-cost set of feasible routes such that each customer is served exactly once by either a truck or a drone. We develop a branch-price-and-cut (BPC) algorithm to solve the VRP-D exactly for both standard objectives considered in the literature, i.e., the minimization of the total routing cost and the sum of the routes' durations. To solve the column-generation subproblems, we present a new forward and implicit bidirectional labeling algorithm defined over an artificial network. The new bidirectional labeling algorithm substantially accelerates the solution process compared its monodirectional counterpart. In several computational experiments, we analyze algorithmic components of the BPC algorithm, compare the cost and duration objectives, and highlight the impact of the drones' speed on the structure of VRP-D solutions. The final version of the BPC algorithm is able to solve VRP-D instances with up to 50 vertices to proven optimality within one hour of computation time.

Keywords: routing, drone delivery, synchronization, branch-price-and-cut, bidirectional labeling

1. Introduction

A *drone*, a.k.a. unmanned aerial vehicle, is an aircraft that does not have a pilot onboard and is controlled remotely (ITA, 2023). Compared to ground vehicles such as trucks and ships, modern drones are typically faster, can move in three dimensions to surmount obstacles on the ground, can take off and land autonomously, have a lower cost per kilometer to operate, and emit less CO_2 (Goodchild and Toy, 2018; D'Andrea, 2014). In this sense, the use of drones represents a greener and more versatile alternative to conventional delivery modes. However, they have the drawback of a limited payload and flight range (even if modern parcel delivery drones can fly up to several kilometers with a payload of about 5kg, more than 5 percent of parcels weigh between 15 and 30kg (Joerss *et al.*, 2016)). We consider combined *delivery operations* of trucks and drones, which can lead to a significant increase of efficiency and effectiveness in performing the delivery operations (Wang *et al.*, 2017; Carlsson and Song, 2018). According to (Otto *et al.*, 2018, Sect. 5), such combined truck and drone operations are of four types: (1) trucks support the delivery operations performed solely by drones, (2) drones support the delivery operations performed solely by trucks, (3) trucks and drones operate independently and perform separate delivery routes, and (4) trucks and drones

*Corresponding author.
Email addresses: sjeanett@uni-mainz.de (Jeanette Schmidt), christian.tilk@univie.ac.at (Christian Tilk), irnich@uni-mainz.de (Stefan Irnich)

work together as synchronized working units. For the numerous existing and potential applications we refer the reader to the surveys (Otto *et al.*, 2018; Chung *et al.*, 2020; Macrina *et al.*, 2020; Moshref-Javadi and Winkenbach, 2021; Madani and Ndiaye, 2022). We focus on the synchronized routing problem of type (4).

In the *vehicle routing problem with drones* (VRP-D), a fleet of trucks is equipped with one drone each. A truck and its drone can either move together or separately. When truck and drone move together, the drone is either inside or on top of the truck. The drone can leave the truck at the depot or any customer location and perform a single delivery alone (this is realistic given the limited payload of the drone and for operational/safety reasons). Meanwhile, the truck continues its journey alone serving one or several other customers. In any case, the drone must return to the same truck at a customer or depot location. The VRP-D consists of finding feasible combined truck and drone routes to serve a given set of customers. A route is feasible if it starts and ends at a given depot and the capacity of the truck and of the drone is respected on all drone subtours. Common objectives of the VRP-D are minimizing the total routing cost and the sum of the durations of all routes.

The partially dependent movements of trucks and drones make the VRP-D more challenging compared to the classical *capacitated vehicle routing problem* (CVRP, Irnich *et al.*, 2014), because additional decisions about which type of vehicle serves which subset of customers must be made (called *task synchronization* in Drexl, 2012). The synchronization of a truck and its drone in space (*movement synchronization*) and, for duration minimization, synchronization in time (*operation synchronization*) must be considered. Although it is known that these two objectives are in conflict, the literature does not discuss this conflict for synchronized routing with drones. Hence, we will analyze the different structure of solutions obtained with each objective.

We develop a VRP-D-tailored *branch-price-and-cut* (BPC) algorithm that allows the exact solution for both objectives, i.e., the minimization of the total routing cost and the sum of the durations. The most important algorithmic component is a bidirectional labeling algorithm to solve the column-generation subproblems. This subproblem is defined over an artificial network originally introduced by Roberti and Ruthmair (2021) for the single-vehicle version of this problem, i.e., for the *traveling salesman problem with drone* (TSP-D). While Roberti and Ruthmair use a non-trivial monodirectional labeling algorithm, we use a bidirectional labeling algorithm, which is responsible for the convincing performance of the overall BPC algorithm: We are able to tackle larger instances than solved for the TSP-D, which is opposed to the often observed relationship between the practical difficulty of single- and multiple-vehicle routing problems.

The artificial network is a multi-digraph, in which vertices represent possible combinations of locations where a truck and its associated drone perform their service. Arcs of the artificial network model possible truck and drone movements. A major complication in designing an effective bidirectional labeling algorithm is that the same route, once considered in forward and once in backward direction, passes through different vertices of the artificial network whenever the drone operates without the truck. In particular, the merge procedure is unclear in such a setting. Moreover, the subproblem is asymmetric even though the VRP-D is defined as a symmetric routing problem. Unexpectedly, the inherent symmetry of the VRP-D can be exploited: Instead of explicitly generating forward and backward labels, the same set of labels can be considered for both directions. This is what we name *implicit* bidirectional labeling.

Accordingly, the main contributions of the paper at hand are as follows:

- We develop a new exact algorithm based on BPC to solve the VRP-D exactly for both routing-cost and duration objectives.
- We show that an implicit bidirectional labeling algorithm is essential for solving the pricing subproblems effectively. A particular novelty is that we apply two different merge procedures in the bidirectional labeling algorithm depending on the current truck-and-drone locations.
- We present several computational analyses: First, we evaluate the new algorithmic components suggested to accelerate the merge procedure that constitutes the bottleneck operation of the overall BPC algorithm. Second, monodirectional labeling is compared with the implicit bidirectional labeling. Moreover, we compare the two objectives and the impact of the drone's and truck's speeds on structure and quality of solutions.

The remainder of this work is structured as follows: We review the pertinent literature in Section 2. Section 3 formally introduces the VRP-D. In Section 4, we present the new BPC algorithm to solve the

VRP-D with a particular focus on the implicit bidirectional labeling algorithm. Computational results are discussed in Section 5, and final conclusions are drawn in Section 6.

2. Literature Review

The integration of drones in vehicle routing is receiving increasing attention in the research literature (Otto *et al.*, 2018; Chung *et al.*, 2020; Macrina *et al.*, 2020; Moshref-Javadi and Winkenbach, 2021; Madani and Ndiaye, 2022). These surveys show the vast amount of scientific contributions that have been published in recent years, where heuristic solution approaches are predominant. For the sake of brevity, we limit the scope of our literature review to exact optimization approaches in the context of vehicle routing in which more than one truck is considered.

The first article dealing with the VRP-D was published by Wang *et al.* (2017). They define the VRP-D as a generalization of the TSP-D by using multiple trucks each equipped with multiple drones. Wang *et al.* assume that each drone is assigned to a specific truck and that a drone can only be released and picked up at either a customer location or the depot. The objective is to minimize the sum of the durations of the routes. Besides the introduction of the VRP-D, they conduct comprehensive analyses of worst-case scenarios on the benefits of using trucks and drones as synchronized work units instead of truck-only solutions. The same authors later extend their analyses in a follow-up paper (Poikonen *et al.*, 2017).

Bakir and Tiniç (2020) consider the VRP-D with flexible drones, i.e., a drone does not necessarily have to return to the truck from where it was released. They formulate the problem as a *mixed integer linear program* (MILP) on a time-space network and propose a dynamic discretization discovery approach that solves instances with up to 25 customers within five hours. Tamke and Buscher (2021) use a branch-and-cut algorithm to solve a version of the VRP-D in which each truck can be equipped with more than one drone. For larger instances, the number of drones on each truck is limited to two drones. Instances with up to 30 customers are solved within twelve hours. Zhen *et al.* (2023) developed a BPC algorithm to solve the VRP-D with the objective to minimize the total routing cost. They can solve instances with up to 14 customers to proven optimality within a 3-hours time limit. In the publication by Zhou *et al.* (2022), each truck is equipped with more than one drone that is uniquely assigned to this truck. The truck has to wait at a customer location until all launched drones return before serving the next customer. For this setting with a simple synchronization, the authors develop a branch-and-price algorithm that uses a bidirectional labeling algorithm that solves some instances with up to 35 customers within three hours. Li and Wang (2022) extend the VRP-D by considering additional customer time windows. To the best of our knowledge, they propose the first exact solution approach which is a BPC algorithm for this VRP-D variant. Instances with up to 50 customers are solved to optimality within 20 hours. To solve larger instances, they combine the BPC algorithm with an adaptive large neighborhood search.

Table 1 gives an overview of the slightly different VRP-D definitions. The column entries have the following meaning: *Objective* indicates whether routing cost or the sum of the durations is minimized. The *Trucks* columns indicate whether they have a limited capacity (*Capacity*) and whether they are allowed to revisit a customer (*Revisits*). The *Drones* columns indicate the number of drones carried by each truck (*#Drones per truck*), whether each drone is assigned to a specific truck (*Assignment*), whether the drones have limited flight range (*Flight range*), whether the drones serve a specific number of customers consecutively (*#customer deliveries*), and whether a setup time has to be considered for each drone flight (*setup time*). The *Sync* columns describe the synchronization of trucks and drones, i.e, whether trucks may have to wait while the drones are serving customers (*Drone loops*) and the potential release and pickup sites (*Release & Pickup site*). Drones can be released at a customer location (c) and/or at the depot (d). The *Customers* columns indicate whether customers need to be served during predefined time windows (*Time windows*) and whether some of them must be served by a truck (*Truck-only*).

3. The Vehicle Routing Problem with Drones

We define the VRP-D on an undirected complete graph (V, E) with vertex set V and edge set E. The vertex set V comprises $\{0, 0'\} \cup N$, where 0 and $0'$ represent the *depot* (two copies for start point and end

		Trucks		Drones					Sync.		Customers	
Publication	Objective	Capacity	Revisits	#Drones per truck	Assignment	Flight range	#Customer deliveries	setup time	Drone loops	Release & pickup site	Time windows	Truck-only
Wang *et al.* (2017)	duration	✓	✗	≥ 1	✓	✓	1	✗	✗	c/d	✗	✗
Bakir and Tiniç (2020)	duration	✗	✓	> 1	✗	✓	1	✗	✗	c/d	✗	✗
Tamke and Buscher (2021)	duration	✗	✗	> 1	✓	✓	1	✓	✗	c	✗	✗
Li and Wang (2022)	cost	✓	✗	> 1	✓	✓	> 1	✗	✗	c/d	✓	✓
Zhen *et al.* (2023)	cost	✓	✗	$= 1$	✓	✓	1	✗	✗	c/d	✗	✗
Zhou *et al.* (2022)	duration	✓	✗	$> 1^*$	✓	✓	1	✓	✓	c	✗	✗
Our paper	cost/duration	✓	✗	$= 1$	✓	✗	1	✗	✗	c/d	✗	✓

Table 1: Overview of different problem setting in VRP-D publications.
*: The number of drones per truck is a decision variable.

point) and $N = \{1, 2, \ldots, m\}$ denotes the set of *customers*. Each customer $i \in N$ has a positive integer *demand* q_i which has to be served by either a truck or a drone. If the demand q_i exceeds a given threshold $\bar{q}$, i is called a *truck-only customer* which cannot be served by drone.

A fleet of K homogeneous trucks is stationed at the depot. Each truck has a *capacity* Q and is equipped with a single drone (the assignment of the drone to its truck is assumed fixed). The truck and its drone can either move together or separately. When truck and drone move together, only the truck can *serve* customers. The drone can leave the truck at the depot 0 or at any customer location to operate alone. In the meantime, the truck continues its journey and can serve other customers on its own. The drone must return to the same truck at the truck's current location, which is either a customer location or the depot $0'$. Note that the drone is not allowed to leave and return to the truck at the same customer. Moreover, each drone has a limited capacity in the sense that it can serve at most one customer before returning back to its truck. Accordingly, a *route* in the VRP-D is a pair $r = (P, D)$ consisting of the *truck path* P and a possibly empty sequence D of *drone subpaths*. The truck path $P = (i_0, i_1, \ldots, i_\ell, i_{\ell+1})$ starts at the depot $i_0 = 0$, serves the customers $i_p \in N$ for all $p \in \{1, \ldots, \ell\}$ in this sequence, and ends at the depot $i_{\ell+1} = 0'$. The drone subpaths $D = (D^1, \ldots, D^\ell)$ specify the drone flights along the truck path P. For each index $s \in \{1, \ldots, \ell\}$, each subpath $D^s = \langle j^s, k^s, l^s \rangle$ contains exactly three vertices: at $j^s \in \{0\} \cup N$ the drone leaves the truck, $k^s \in N$ specifies the customer that is served by the drone, and $l^s \in N \cup \{0'\}$ indicates where the drone lands on the truck again. In a feasible route $r = (P, D)$ there exist, for each index $s \in \{1, 2, \ldots, \ell\}$, two indices $g, h \in \{0, 1, \ldots, \ell, \ell + 1\}$ with $g < h$ such that $j^s = i_g$ and $l^s = i_h$ holds. For convenience, we define $I(s) = \{g, g + 1, \ldots, h - 1\}$ as the positions (indices) of P where either the drone leaves the truck or the truck serves a customer alone (intentionally, the position where the drone lands on the truck is not included). If $\ell > 1$, the sets $I(s)$ and $I(s')$ for $s \neq s'$ must have empty intersection (the segments where the truck is alone do not overlap). Finally, the route $r = (P, D)$ is *feasible* if $\sum_{j=1}^{\ell} q_{i_j} + \sum_{s=1}^{\ell} q_{k^s} \leq Q$ and $q_{k^s} \leq \bar{q}$ for all $s \in \{1, \ldots, \ell\}$. The route $r = (P, D)$ is *elementary* if $i_1, \ldots, i_\ell$ and $k^1, \ldots, k^\ell$ are all different.

To distinguish between routing-cost and duration minimization, each edge $\{i, j\} \in E$ is associated with a non-negative routing cost c_{ij} and a non-negative travel time t_{ij} for a truck. Likewise, the routing cost of a drone is denoted by c^{dr}_{ij} and the travel time by t^{dr}_{ij}. The routing cost c_r of a route $r = (P, D)$ is defined as the sum of the routing costs of the edges traversed by both types of vehicles, i.e., $c_r = \sum_{p=0}^{\ell} c_{i_p, i_{p+1}} + \sum_{s=1}^{\ell} (c^{\mathrm{dr}}_{j^s, k^s} + c^{\mathrm{dr}}_{k^s, l^s})$. The duration t_r of a route $r = (P, D)$ is more intricate to describe, because the truck (drone) may have to wait for the drone (truck) depending on which vehicle is faster on a particular segment.

Let $I = \bigcup_{s=1}^{\ell} I(s)$. Then, the duration of route r is

$$t_r = \sum_{p \in \{0,1,\ldots,\ell\} \setminus I} t_{i_p, i_{p+1}} + \sum_{s=1}^{\ell} \max \left\{ \sum_{p \in I(s)} t_{i_p, i_{p+1}}, t^{\mathrm{dr}}_{j^s, k^s} + t^{\mathrm{dr}}_{k^s, l^s} \right\}. \tag{1}$$

The left-hand term describes travel times when truck and drone travel together, while the right-hand term sums over the ℓ drone subpaths taking the maximum of the travel time of either the truck or the drone.

The task of the VRP-D is to determine a set R of at most K feasible routes such that each customer is served exactly once and either $\sum_{r \in R} c_r$ or $\sum_{r \in R} t_r$ is minimized.

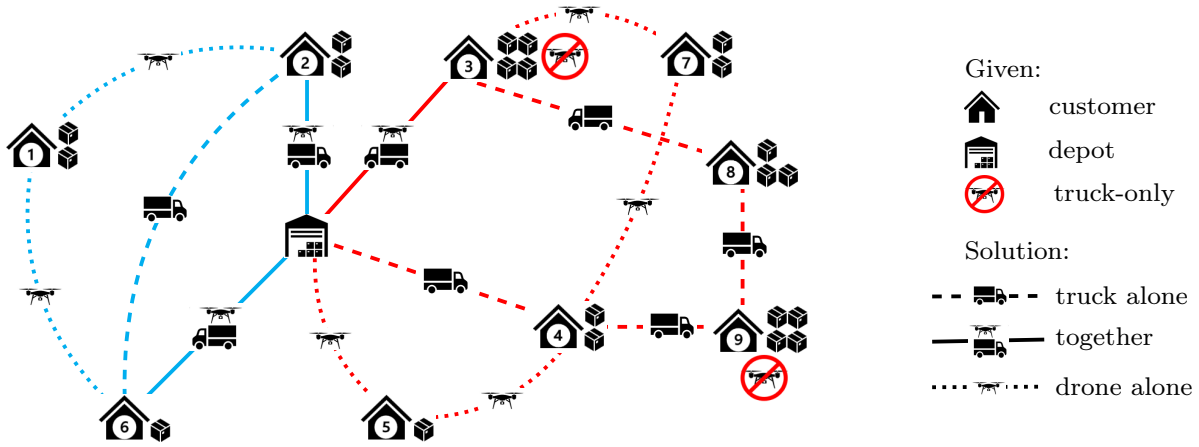


Figure 1: VRP-D instance with a feasible solution.

Example 1. *Figure 1 shows a VRP-D instance with customers* $N = \{1, 2, \ldots, 9\}$ *as well as a feasible solution with two routes. On the first route (depicted in red), the truck releases its drone directly at the depot (we assume a counterclockwise travel direction for this route). While the truck travels to customer 4 and serves it, the drone moves to customer 5, serves this customer, and returns back to the truck at customer 4. Immediately at customer 4, truck and drone separate again: while the truck serves the customers 9, 8, and 3 alone, the drone serves customer 7 and lands on the truck again at the location of customer 3. Afterwards, truck and drone together travel back to the depot. Hence, the route serves customers* $\{3, 4, 5, 7, 8, 9\}$. *The corresponding truck path is* $P = (i_0, i_1, \ldots, i_6) = (0, 4, 9, 8, 3, 0')$ *and the sequence of drone subpaths is* $D = (\langle 0, 5, 4\rangle, \langle 4, 7, 3\rangle)$. *The second route (depicted in blue) serves another three customers* $\{1, 2, 6\}$ *with* $P = (0, 6, 2, 0')$ *and* $D = (\langle 6, 1, 2\rangle)$. *Assuming that all travel costs and travel times of all edges* $\{i, j\} \in E$ *are* $c_{ij} = t_{ij} = c^{dr}_{ij} = t^{dr}_{ij} = 1$, *the truck has to wait for the drone at location 4, and vice versa, the drone has to wait for the truck at location 3 in the first (red) route. Hence, the duration of the first route is* $t_r = 6$ *while its cost is* $c_r = 9$. *Under these assumptions, the truck has to wait for the drone at location 2 in the second (blue) route, which has a duration of* $t_r = 4$ *while its cost is* $c_r = 5$. □

4. Branch-Price-and-Cut Algorithm

We employ a BPC algorithm to solve the VRP-D. BPC algorithms are 'the leading exact algorithms for solving many classes of vehicle routing problems' (see Costa *et al.*, 2019). They rely on an extensive, route-based formulation (a.k.a. master problem (MP)), i.e., a model with variables for all feasible routes. A BPC algorithm is a branch-and-bound algorithm in which, at each node, the linear relaxation of the MP is solved. At some nodes, valid inequalities are added to strengthen the linear relaxation. Due to the huge number of variables in the MP, relaxations are not solved directly but with a column-generation procedure:

Iteratively, *restricted master programs* (RMPs) defined over a subset of routes are solved, missing routes are identified by solving subproblems (a.k.a. pricing problems), and these are added to the RMPs (Desaulniers *et al.*, 2005).

4.1. Route-based Formulation

Let Ω denote the set of all feasible VRP-D routes. The route-based formulation presented below uses binary variables λ_r to indicate whether a route $r \in \Omega$ is part of the optimal solution. In addition, the non-negative variable f describes the number of trucks employed. For each vertex $i \in V$ and each route $r \in \Omega$, let the integer coefficient b_{ir} indicates how often route r serves customer i.

$$\min \quad \sum_{r\in\Omega} c_r\lambda_r \qquad \text{or} \qquad \min \quad \sum_{r\in\Omega} t_r\lambda_r \tag{2a}$$

$$\text{subject to} \quad \sum_{r\in\Omega} b_{ir}\lambda_r = 1 \qquad \forall i \in N \tag{2b}$$

$$\sum_{r\in\Omega} \lambda_r - f = 0 \tag{2c}$$

$$K_N \le f \le K \tag{2d}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in \Omega \tag{2e}$$

The objective (2a) is to minimize the total routing cost or the sum of the durations of all routes. Constraints (2b) ensure that each customer is served exactly once. The number of trucks f in use is determined via (2c) and restricted to the interval between a lower bound K_N and the given fleet size K via (2d). The domain of the route variables is specified in (2e).

A valid lower bound K_N for the number of necessary trucks can be calculated by solving a bin-packing problem as follows. The bin size is set equal to the truck's capacity Q. For each customer $i \in N$, an item with a weight equal to demand q_i must be introduced. We solve the bin-packing problem with the arc-flow formulation of Valério de Carvalho (1999) using a MIP solver. Note that the optimal number K_N of bins is an exact lower bound that helps to not solve some infeasible branch-and-bound nodes, which may otherwise result from branching on the number of vehicles (see Section 4.4). Solving these infeasible nodes is often very time-consuming.

4.2. Column Generation

Let $(\pi_i)_{i\in N}$ be the dual prices of the set-partitioning constraints (2b) and μ be the dual price of constraint (2c). The reduced cost of a route $r \in \Omega$ is

$$\tilde{c}_r = c_r - \sum_{i\in N} b_{ir}\pi_i - \mu \quad \text{or} \quad \tilde{c}_r = t_r - \sum_{i\in N} b_{ir}\pi_i - \mu$$

depending on whether the objective is cost or duration minimization.

This pricing problem can be modeled as an *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005) and solved by means of a dynamic-programming labeling algorithm on an artificial network.

4.2.1. Artificial Network

Compared to the classical CVRP, the major additional complication in the VRP-D is that truck and drone can move independently, at least on subpaths of their itinerary. Already the decision which type of vehicle will serve which subset of the customers adds a substantial degree of freedom. All these decisions impact a route's cost and duration, which can no longer be computed as the length of a single path. We have shown in Section 3 that route feasibility constraints are much more involved compared to the CVRP, because a route in the given network (V, E) consists of a truck path and a sequence of drone subpaths.

For solving the single-vehicle version of the problem, the TSP-D, Roberti and Ruthmair (2021) model a route/tour with the help of an *artificial network* $\mathscr{N} = (W, A)$. Its vertices

$$W = \{(0,0)\} \cup (N \cup \{0'\}) \times (\{0\} \cup N) \cup \{(0',0')\} \subset V \times V$$

represent possible truck-and-drone positions, i.e., the vertex $(i^{\text{tr}}, i^{\text{dr}}) \in W$ describes that the truck is at position i^{tr} and the drone at i^{dr}. In particular, the *origin* vertex $(0,0)$ represents the initial situation at the depot when truck and drone are together ready to start a route. Likewise, the *destination* vertex $(0',0')$ represents the final situation when truck and drone are back together at the depot after completing a route.

Arcs of the artificial network represent movements and a possible additional service of a customer by the drone. For the latter service aspect, let $N^{dr} = \{i \in N : q_i \leq \bar{q}\} \cup \{\perp\}$ denote such a service by drone with $\perp$ representing that no service is performed. The arcs fall into the following three categories: First, the *alone arcs*

$$A^{alone} = \left\{[(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), \perp] \in W \times W \times \{\perp\} : i^{\text{dr}} = j^{\text{dr}} \text{ and } i^{\text{tr}} \neq j^{\text{tr}} \text{ and } i^{\text{dr}} \neq j^{\text{tr}}\right\} \tag{3a}$$

represent that only the truck is moving from position i^{tr} to position j^{tr} while the drone still resides at position $i^{\text{dr}} = j^{\text{dr}}$. There is no additional service possible. Second, the *together arcs*

$$A^{tgthr} = \left\{[(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), \perp] \in W \times W \times \{\perp\} : i^{\text{tr}} = i^{\text{dr}} \neq j^{\text{tr}} = j^{\text{dr}}\right\} \tag{3b}$$

represent that truck and drone are moving together from position $i^{\text{tr}} = i^{\text{dr}}$ to a new position $j^{\text{tr}} = j^{\text{dr}}$. Also here no additional service is possible. Third, the *drone arcs*

$$A^{drone} = \left\{[(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), k] \in W \times W \times N^{dr} : i^{\text{tr}} = j^{\text{tr}} = j^{\text{dr}} \neq i^{\text{dr}} \text{ and } k \notin \{i^{\text{dr}}, i^{\text{tr}}, \perp\}\right\} \tag{3c}$$

represent that the drone flies from the position i^{dr} where it has separated from the truck to customer k, serves customer k, and subsequently flies to the position $i^{\text{tr}} = j^{\text{tr}} = j^{\text{dr}}$ where the truck is waiting. Note that the arc sets defined via (3a)–(3c) are disjoint. Hence, each arc in $A = A^{alone} \cup A^{tgthr} \cup A^{drone}$ belongs to exactly one category (alone, together, or drone arc). Moreover, the arcs in $A^{alone} \cup A^{tgthr}$ are referred to as *truck arcs*, since they model serving customer j^{tr} by truck.

We stress that the artificial network $\mathscr{N}$ is directed and has parallel arcs, i.e., it is a *multi-digraph*. Parallel arcs are always drone arcs $[(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), k] \in A^{drone}$ and they differ in the additional service that is performed at customer k.

The fundamental idea of Roberti and Ruthmair was to artificially separate truck movement decisions from drone movement decisions. When truck and drone separate, it is not necessary to know in advance which customer the drone will serve and where and when it will return to the truck. Hence, these decisions can be postponed. Instead, the bookkeeping reduces to record the position where the drone has departed from the truck. The truck is then routed independently serving an arbitrary number of customers. The postponed decision about the drone's subpath is made when truck and drone meet again. This enables a forward labeling algorithm with an efficient dominance procedure as shown in Section 4.2.2.

Example 2. *Figure 2 visualizes the artificial network for a VRP-D instance with three customers* $N = \{1,2,3\}$. *The path* $((0,0),(1,1),(2,2),(3,3),(0',0'))$ *in* $\mathscr{N}$ *represents a route where truck and drone travel together from the depot* 0 *to the three customers 1, 2, and 3 (in this sequence), and back to the depot* $0'$. *The path* $((0,0),(2,0),(2,2),(3,3),(0',0'))$ *in* $\mathscr{N}$ *has an associated unique truck path* $(0,2,3,0')$. *However, there are two parallel arcs connecting* $(2,0)$ *with* $(2,2)$ *leaving the choice of using one of them. Either customer 1 could be served leading to the associated drone subpath* $\langle 0,1,2\rangle$, *or customer 3 could be served leading to the associated drone subpath* $\langle 0,3,2\rangle$. *In the first case, the route is elementary, serving customers 1, 2, and 3 exactly once, while in the second case, the route is non-elementary, serving customer 3 twice and customer 2 once.* □

More generally, every $(0,0)$-$(0',0')$-path represents a joint truck-and-drone route. Since the artificial network is a multi-digraph, only the sequence of its arcs describes it uniquely. Note that the corresponding route is not necessarily elementary or feasible. The labeling algorithm presented in Section 4.2.2 checks elementary and feasibility.

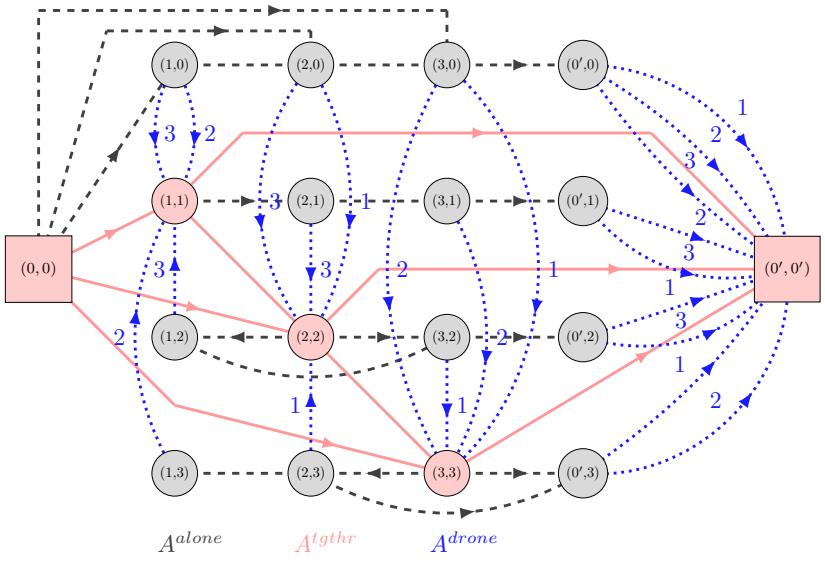


Figure 2: Artificial network $\mathcal{N} = (W, A)$ with customer set $N = \{1, 2, 3\}$. For the sake of simplicity, not all truck arcs are made explicit. Antiparallel arcs between vertices are depicted as undirected links. For example, in the first row, antiparallel arcs exist between the vertices $(1, 0)$, $(2, 0)$, and $(3, 0)$.

Each $(0, 0)$-$(0', 0')$-path can be resolved into a corresponding truck path and drone subpaths using the information on each arc $[(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), k]$. If the arc is a truck arc, i.e., $k = \perp$, the trucks destination j^{tr} has to be appended to the truck path. If the arc is a drone arc, i.e., $k \in N$, the drone subpath $\langle j, k, l \rangle$ contains the following three pieces of information: the release location $j = i^{\text{dr}}$ of the drone, the drone customer k, and the drone's destination location $l = j^{\text{dr}}$.

Example 3. *We consider a VRP-D instance with* $N = \{1, 2, 3, 4, 5\}$. *The* $(0, 0)$-$(0', 0')$*-path*

$$([(0,0),(1,1),\perp],[(1,1),(2,1),\perp],[(2,1),(2,2),3],[(2,2),(4,2),\perp],[(4,2),(0',2),\perp],[(0',2),(0',0'),5])$$

has $P = (0, 1, 2, 4, 0')$ *and* $D = (\langle 1, 3, 2\rangle, \langle 2, 5, 0'\rangle)$. *In the following, we also use the more intuitive representation*

$$(0,0) = (1,1) - (2,1) \overset{3}{-} (2,2) - (4,2) - (0',2) \overset{5}{-} (0',0')$$

as a sequence of vertices where the connections are either $=$, $-$, *or* $\overset{k}{-}$. *The symbols* $-$ $(=)$ *indicate a connection with an alone (together) arc, while the symbol* $\overset{k}{-}$ *indicates a drone arc of type* $[(\cdot,\cdot),(\cdot,\cdot),k]$. □

4.2.2. Forward Labeling

In this section, we present a dynamic-programming labeling algorithm to solve the SPPRC pricing problem. We begin with a basic forward labeling approach and then discuss refinements including implicit bidirectional labeling and a non-trivial merge procedure.

Forward labeling starts with an initial label at the origin depot $(0, 0)$ and propagates forward labels over all type of arcs $A = A^{alone} \cup A^{tgthr} \cup A^{drone}$ towards the destination depot $(0', 0')$. Each label refers to a partial path $P_i = ((0, 0), \ldots, (i^{\text{tr}}, i^{\text{dr}}))$ that starts at $(0, 0)$ and ends at some vertex $i = (i^{\text{tr}}, i^{\text{dr}}) \in W$. The label stores the corresponding resource consumption up to vertex i. The set of attributes (=resources) depends on the objective function used in the VRP-D.

Cost Objective. For the cost objective and a partial path P_i, the associated label $\mathcal{L}_i$ comprises the resources $(R_i^{rdc}, R_i^{load}, (R_i^{cust,n})_{n\in N})$ defined as follows:

R_i^{rdc}: the reduced cost of path P_i;
R_i^{load}: the total demand served by path P_i excluding the demand of i; and
$R_i^{cust,n}$: the number of times that customer $n \in N$ is served along the path P_i.

The initial label is given by $\mathcal{L}_0 = (0, 0, \mathbf{0})$. Propagating a label $\mathcal{L}_i$ over an arbitrary arc $a = [(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), k] \in A$ results in a new label $\mathcal{L}_j$ at vertex $j = (j^{\text{tr}}, j^{\text{dr}})$. We assume that dual prices $\pi = (\pi_i)_{i \in N}$ of the constraints (2b) and the dual price μ of the constraint (2c) are given. For convenience, we define $\pi_{0'} = \mu$ for the destination $0'$. Moreover, we define $q_0 = q_{0'} = 0$ for the depot. The associated resource values for the partial path $P_j = (P_i, j)$ are computed with the help of the following *resource extension functions* (REFs, Irnich, 2008):

$$R_j^{rdc} = \begin{cases} R_i^{rdc} + c_{i^{\text{tr}}, j^{\text{tr}}} - \pi_{j^{\text{tr}}}, & \text{if } a \in A^{alone} \cup A^{tgthr} \\ R_i^{rdc} + c^{\text{dr}}_{i^{\text{dr}}, k} + c^{\text{dr}}_{k, j^{\text{dr}}} - \pi_k, & \text{if } a \in A^{drone} \end{cases} \tag{4a}$$

$$R_j^{load} = \begin{cases} R_i^{load} + q_{i^{\text{tr}}}, & \text{if } a \in A^{alone} \cup A^{tgthr} \\ R_i^{load} + q_k, & \text{if } a \in A^{drone} \end{cases} \tag{4b}$$

$$R_j^{cust,n} = \begin{cases} R_i^{cust,n} + 1, & \text{if } n = j^{\text{tr}} \text{ and } a \in A^{alone} \cup A^{tgthr} \\ R_i^{cust,n} + 1, & \text{if } n = k \text{ and } a \in A^{drone} \\ R_i^{cust,n}, & \text{otherwise} \end{cases} \tag{4c}$$

Note that in (4b) the update of the load resource is postponed regarding the customer j^{tr} to the point after the truck leaves customer j^{tr}. We will see later that this is convenient for the symmetric bidirectional labeling algorithm.

The new label $\mathcal{L}_j$ is feasible, if the capacity and the elementary constraints are fulfilled, i.e.,

$$R_j^{load} \le \begin{cases} Q - q_{j^{\text{tr}}}, & \text{if } a \in A^{alone} \cup A^{tgthr} \\ Q, & \text{if } a \in A^{drone} \end{cases} \tag{5a}$$

$$R_j^{cust,n} \le 1, \qquad \forall n \in N. \tag{5b}$$

Infeasible labels are discarded immediately. To avoid the enumeration of all feasible paths, redundant labels are eliminated with the help of dominance tests. Since all forward REFs are non-decreasing, we can use the standard dominance rules (Irnich and Desaulniers, 2005):

Dominance Rule 1. *A label $\mathcal{L}_1 = (R_1^{rdc}, R_1^{load}, (R_1^{cust,n})_{n \in N})$ dominates another label $\mathcal{L}_2 = (R_2^{rdc}, R_2^{load}, (R_2^{cust,n})_{n \in N})$ if $R_1^{rdc} \le R_2^{rdc}$, $R_1^{load} \le R_2^{load}$, and $R_1^{cust,n} \le R_2^{cust,n}$ holds for all $n \in N$.*

Duration Objective. When considering the duration objective, additional information is required. We follow the ideas presented in Roberti and Ruthmair (2021). Hence, a label now comprises the resources $(R_i^{rdc}, R_i^{dur}, R_i^{load}, (R_i^{cust,n})_{n \in N})$ defined as:
R_i^{rdc}: the reduced cost based on the duration of path P_i;
R_i^{dur}: the duration for which the truck travels alone on the last open subtour of path P_i;
and R_i^{load} and $R_i^{cust,n}$ as defined in (4b) and (4c), respectively.

The initial label is now given by $\mathcal{L}_0 = (0, 0, 0, \mathbf{0})$. The REFs for the resources *rdc* and *dur* and arc $a = [(i^{\text{tr}}, i^{\text{dr}}), (j^{\text{tr}}, j^{\text{dr}}), k] \in A$ are given by:

$$R_j^{rdc} = \begin{cases} R_i^{rdc} - \pi_{j^{\text{tr}}}, & \text{if } a \in A^{alone} \\ R_i^{rdc} + t_{i^{\text{tr}}, j^{\text{tr}}} - \pi_{j^{\text{tr}}}, & \text{if } a \in A^{tgthr} \\ R_i^{rdc} + \max\{t^{\text{dr}}_{i^{\text{dr}}, k} + t^{\text{dr}}_{k, j^{\text{dr}}}, R_i^{dur}\} - \pi_k, & \text{if } a \in A^{drone} \end{cases} \tag{6a}$$

$$R_j^{dur} = \begin{cases} R_i^{dur} + t_{i^{\text{tr}}, j^{\text{tr}}}, & \text{if } a \in A^{alone} \\ 0, & \text{if } a \in A^{drone} \cup A^{tgthr} \end{cases} \tag{6b}$$

Compared to the cost objective, there are no further feasibility requirements in addition to (5).

Dominance Rule 2. *We consider two labels at a the same vertex* (i^{tr}, i^{dr}) *of the artificial network. A label* $\mathcal{L}_1 = (R_1^{rdc}, R_1^{dur}, R_1^{load}, (R_1^{cust,n})_{n\in N})$ *dominates another label* $\mathcal{L}_2 = (R_2^{rdc}, R_2^{dur}, R_2^{load}, (R_2^{cust,n})_{n\in N})$ *if* $R_1^{load} \le R_2^{load}$, $R_1^{cust,n} \le R_2^{cust,n}$ *for all* $n \in N$, *and*
(1) $i^{tr} = i^{dr}$ *and* $R_1^{rdc} \le R_2^{rdc}$,
(2) *or:* $i^{tr} \neq i^{dr}$ *and* $R_1^{rdc} \le R_2^{rdc}$, $R_1^{rdc} + R_1^{dur} \le R_2^{rdc} + R_2^{dur}$,
(3) *or:* $i^{tr} \neq i^{dr}$ *and* $R_1^{rdc} + R_1^{dur} \le R_2^{rdc} + R_2^{dur}$ *and* $R_1^{rdc} + \max_{k,l}\{t^{dr}_{i^{dr},k} + t^{dr}_{k,l}\} \le R_2^{rdc} + R_2^{dur}$, *where the indices* k *and* l *in the maximum term run over* $k \in N$ *and* $l \in N \cup \{0'\}$ *with* $k \neq l$.

Note that in part (1) the precondition $i^{\mathrm{tr}} = i^{\mathrm{dr}}$ implies $R_1^{dur} = R_2^{dur} = 0$ because of (6b), which proves correctness of the dominance due to non-decreasing REFs (4b), (4c), (6a), and (6b). In part (2), the condition $R_1^{rdc} + R_1^{dur} \le R_2^{rdc} + R_2^{dur}$ is fulfilled whenever $R_1^{rdc} \le R_2^{rdc}$ and $R_1^{dur} \le R_2^{dur}$, which shows that this dominance is stronger than a simpler pairwise comparison with $\le$. The two conditions in part (3) guarantee that the reduced cost of the first path is not larger than that of the second when both are extended to a vertex $(j^{\mathrm{tr}}, j^{\mathrm{dr}})$ with $j^{\mathrm{tr}} = j^{\mathrm{dr}}$, either directly or with additional intermediate vertices. Such a vertex $(j^{\mathrm{tr}}, j^{\mathrm{dr}})$ with $j^{\mathrm{tr}} = j^{\mathrm{dr}}$ is certainly reached when arriving at $(0', 0')$. The validity of this Dominance Rule 2 is formally shown in Roberti and Ruthmair (2021).

4.2.3. Symmetry Considerations

By definition, the VRP-D is a symmetric routing problem which was thus defined using an undirected graph. In this graph (V, E), the two depot vertices 0 and $0'$ can be swapped without changing the VRP-D instance. By reversing a route's truck path and drone subpaths (if any) and by swapping 0 and $0'$, the resulting route is equivalent to the one considered originally. In particular, both routes serve the same customers and have identical routing cost and duration. In contrast, the artificial network $\mathcal{N} = (W, A)$ is intentionally defined as a directed multi-graph. This is consistent with the idea that the second component i^{dr} of a vertex $(i^{\mathrm{tr}}, i^{\mathrm{dr}}) \in W$ indicates the location where the truck has released the drone in case that $i^{\mathrm{dr}} \neq i^{\mathrm{tr}}$ holds. This piece of information is asymmetric because the choice of the drone's later landing location is unspecified.

Even more, a given route r and its reversed counterpart r', who both represent the same thing in the symmetric VRP-D, are represented, in general, by paths that visit different vertices in the artificial network $\mathcal{N}$ as shown in the following example.

Example 4. (continued from Example 3) *The truck path* $P = (0, 1, 2, 4, 0')$ *and associated drone subpaths* $D = (\langle 1, 3, 2\rangle, \langle 2, 5, 0'\rangle)$ *have reverse counterparts* $P' = (0, 4, 2, 1, 0')$ *and* $D' = (\langle 0, 5, 2\rangle, \langle 2, 3, 1\rangle)$, *which are perfectly symmetric in the original network* (V, E). *In the artificial network, however, the corresponding* $(0, 0)$-$(0', 0')$*-paths*

$$(0,0) = (1,1) - (2,1) \overset{3}{-} (2,2) - (4,2) - (0',2) \overset{5}{-} (0',0') \text{ and } (0,0) - (4,0) - (2,0) \overset{5}{-} (2,2) - (1,2) \overset{3}{-} (1,1) = (0',0')$$

differ in the vertices $(1, 2)$, $(2, 1)$, $(4, 2)$, $(4, 0)$, $(0', 2)$, *and* $(2, 0)$. □

Having pointed out the asymmetry in forward and backward paths, it becomes clear that the reversal of paths in the asymmetric network is somewhat confusing and not straightforward. It seems rather impossible to develop a bidirectional labeling algorithm over the artificial network. These observations may explain why Roberti and Ruthmair did not present a bidirectional approach for the TSP-D although the general methodology for an exact solution algorithm for TSP variants is known (Baldacci *et al.*, 2012; Tilk and Irnich, 2017; Lera-Romero *et al.*, 2022).

4.2.4. Implicit Backward and Bidirectional Labeling

The general motivation for applying a bounded bidirectional labeling procedure is the following: For many SPPRCs, the number of labels generated by a monodirectional labeling algorithm increases rapidly when feasible partial paths grow longer. This undesirable situation is known as *combinatorial explosion*, which often renders the solution practically impossible. To mitigate the combinatorial explosion, Righini and Salani (2006) introduced a bounded bidirectional labeling procedure for SPPRCs. In a bidirectional

labeling, not only forward labels but also backward labels are created by propagating a label against the orientation of the arcs starting from the sink of the network. Both forward and backward labels are only extended up to a so-called *halfway point* (HWP). When forward and backward propagation is completed, suitable labels are merged to obtain complete feasible origin-destination-paths.

The selection of a monotone resource (often the load or time attribute) and the concrete choice of the HWP not exactly 'in the middle' can be important for the labeling algorithm's performance for asymmetric SPPRC instances (Tilk *et al.*, 2017). If the underlying network is however perfectly symmetric, an *implicit* bidirectional technique can be applied to further accelerate the labeling. Since forward and backward labeling create exactly the same partial paths, it suffices to propagate labels only in forward direction. These labels are combined in the merge procedure. Implicit bidirectional labeling has already been applied successfully in several works (e.g., Bode and Irnich, 2012; Goeke *et al.*, 2019; Heßler and Irnich, 2023).

We now show that implicit bidirectional labeling is possible for the artificial network of the VRP-D although it is asymmetric. Since the only constrained resource is the capacity of the trucks, we use the load resource R^{load} as the critical resource and fix the HWP to $Q/2$, i.e., only labels with accumulated load $R^{load} < Q/2$ are extended.

The following observation can be made from Example 4: The two $(0,0)$-$(0',0')$-paths that represent reversed routes differ only in vertices $(i^{\mathrm{tr}}, i^{\mathrm{dr}}) \in W$ with $i^{\mathrm{tr}} \neq i^{\mathrm{dr}}$. These vertices occur in segments of the routes where truck and drone move independently. In contrast, when truck and drone are at the same location $i^{\mathrm{tr}} = i^{\mathrm{dr}}$, segments are identical. In Example 4, the sequence $(0,0)$, $(1,1)$, $(2,2)$, $(0',0')$ is traversed in either this or the reverse order.

As a consequence, we apply two different merge procedures depending on whether $i^{\mathrm{tr}} = i^{\mathrm{dr}}$ or $i^{\mathrm{tr}} \neq i^{\mathrm{dr}}$ holds at the vertex of a label that could potentially be merged. The procedure for $i^{\mathrm{tr}} = i^{\mathrm{dr}}$ is rather standard, while the one for $i^{\mathrm{tr}} \neq i^{\mathrm{dr}}$ is more sophisticated.

Merge at Vertices where Truck and Drone are together ($i^{tr} = i^{dr}$). In this case, we consider two labels $\mathcal{L} = (R)$ and $\mathcal{L}' = (R')$ of partial paths that both end at the same vertex $(i,i) = (i^{\mathrm{tr}}, i^{\mathrm{dr}}) \in W$. These vertices exist for all $i \in V$ in the artificial network. The merge condition is the one known from the CVRP, i.e., the resulting path must respect the truck's capacity and be elementary:

$$R^{load} + R'^{load} + q_{i^{\mathrm{tr}}} \leq Q \tag{7a}$$

$$R^{cust,n} + R'^{cust,n} \leq 1 \qquad \forall i \in N \setminus \{i^{\mathrm{tr}}\} \tag{7b}$$

The reduced cost of the route r resulting from the merge is

$$\tilde{c}_r = R^{rdc} + R'^{rdc} + \pi_i \tag{8a}$$

independent of the objective function. The route itself is $r = (P, \mathrm{reverse}(P'))$ where P and P' are the partial paths associated with $\mathcal{L}$ and $\mathcal{L}'$, respectively, and the function $\mathrm{reverse}(P')$ denotes the reversed path to P'.

Even if we do not need to know the reversed path for feasibility testing via (7a) and (7b) and the reduced cost check $\tilde{c}_r < 0$ based on (8a), we must formally describe the reversal of the second partial path P' associated with $\mathcal{L}'$. For the reversal of an arbitrary path, it suffices to describe the reversal of a subpath between two subsequent vertices (i,i) and (j,j). In general, such a subpath (=segment) in the artificial network is either trivial $(i,i) = (j,j)$, i.e., it consists of a single together arc, or of the form

$$(i,i) - (i_1,i) - (i_2,i) - \cdots - (i_l,i) - (j,i) \overset{k}{-} (j,j)$$

with associated drone subpath $\langle i,k,j \rangle$ (we explicitly include the special case that $l = 0$ holds). In this subpath, all arcs are alone arcs except for the last arc which must be a drone arc $[(j,i),(j,j),k]$. The reverse subpath to the given subpath is

$$(j,j) - (i_l,j) - \cdots - (i_2,j) - (i_1,j) - (i,j) \overset{k}{-} (i,i),$$

i.e., except for the first and last vertex, the second entry i for the drone position must be replaced by j, while the sequence $i_1, i_2, \ldots, i_l$ of truck positions is reversed in the usual way. The second last vertex (j,i)

becomes the second last vertex (i, j) in the reversed path. Note also that in both subpaths, the customer k is served by the drone, i.e., the last arc in the reversed path is a drone arc serving customer k.

Finally, several reversed (i, i)-(j, j)-subpaths can also be concatenated in the usual way.

Example 5. (continued from Example 4) *We assume that the path* $(0,0) = (1,1) - (2,1) \overset{3}{-} (2,2) - (4,2) - (0',2) \overset{5}{-} (0',0')$ *considered in Example 4 is feasible, i.e.,* $\sum_{l=1}^{5} q_l \leq Q$ *is fulfilled.*

Let vertex $(2,2)$ *be the merge vertex for the path, hence, it must hold* $q_1 < Q/2$ *and* $q_1 + q_3 \geq Q/2$*. In this case, the merge of the two partial paths*

$$(0,0) = (1,1) - (2,1) \overset{3}{-} (2,2) \quad \textit{and} \quad (0,0) - (4,0) - (2,0) \overset{5}{-} (2,2)$$

produces the given path. Note that these are exactly the partial paths (up to vertex $(2,2)$*) of the two partial paths shown in Example 4.*

Their labels $\mathcal{L} = (R)$ *and* $\mathcal{L}' = (R')$ *qualify for the merge, because* $R^{load} = q_1 + q_3 \geq Q/2$ *and* $R'^{load} = q_4 + q_5$*. The merge conditions are also fulfilled, what can be seen as follows: First, the load-related feasibility test (7a) adds together* $R^{load} = q_1 + q_3$*,* $R'^{load} = q_4 + q_5$*, and* q_2 *for the merge vertex* $(2,2)$*, which is exactly the demand delivered by the resulting route. The assumption that the given route is feasible implies that (7a) is fulfilled.*

Second, regarding the served customers, $R^{cust,n} = 1$ *if and only if* $n \in \{1,2,3\}$*, and* $R'^{cust,n} = 1$ *if and only if* $n \in \{2,4,5\}$*. The elementarity check (7b) excludes* $n = 2$ *from the test* $R^{cust,n} + R'^{cust,n} \leq 1$*. Hence, labels* $\mathcal{L}$ *and* $\mathcal{L}'$ *passes all merge conditions.* □

Example 5 also provides a good example of why we must delay the addition of the customer demand in the REF (4b). In an alternative model, the demand would always be immediately added, i.e., $q_{j^{\mathrm{tr}}}$ were added to R^{load} on all arcs $[(i^{\mathrm{tr}}, i^{\mathrm{dr}}), (j^{\mathrm{tr}}, j^{\mathrm{dr}}), k]$ with $i^{\mathrm{tr}} \neq j^{\mathrm{tr}}$. Then, the partial paths of both labels R and R' could sooner pass the HWP in which case they would not be extended to the merge point.

Example 6. (continued from Example 5) *We consider an instance of the VRP-D with demands* $q_1 = q_2 = q_4 = 10$ *and* $q_3 = q_5 = 1$ *and a vehicle capacity of* $Q = 38$ *as well as the same paths as in Example 5. Moreover, we assume a modified REF for the load resource that directly incorporates the demand of a new customer visited by the truck.*

The first path $(0,0) = (1,1) - (2,1) \overset{3}{-} (2,2)$ *would pass the HWP* $Q/2 = 19$ *already at vertex* $(2,1)$ *with an accumulated demand of* $q_1 + q_2 = 20 > Q/2$*. Similarly, the second path* $(0,0) - (4,0) - (2,0) \overset{5}{-} (2,2)$ *would pass the HWP already at vertex* $(2,0)$ *with an accumulated demand of* $q_4 + q_2 = 20 > Q/2$*. As a result, the merge procedure would not generate the route that has the two partial paths as forward and (reversed) backward partial path.* □

Merge at Pairs of Vertices where Truck and Drone are at Different Locations ($i^{tr} \neq i^{dr}$)*.* This second case is more intricate: A label belonging to a vertex $(i^{\mathrm{tr}}, i^{\mathrm{dr}})$ with $i^{\mathrm{tr}} \neq i^{\mathrm{dr}}$ *must not* be merged with labels belonging to the same vertex. Instead, the second vertex $(i^{\mathrm{tr}}, j^{\mathrm{dr}})$ must have identical truck position i^{tr} but a different drone position $j^{\mathrm{dr}} \neq i^{\mathrm{dr}}$. As a consequence, the merge must be performed 'over an arc' so that the total length of the forward and backward partial paths is by one arc smaller than the length of the resulting merged route.

Example 7. (continued from Example 5) *We consider the same route as in Example 5, but assume that the merge is now 'shifted one position to the right' so that the forward and backward partial paths are*

$$(0,0) = (1,1) - (2,1) \overset{3}{-} (2,2) - (4,2) \quad \textit{and} \quad (0,0) - (4,0),$$

i.e., the first (forward) partial path with label $\mathcal{L}$ *ends at vertex* $(i^{tr}, i^{dr}) = (4,2)$*, and the second (backward) partial path with label* $\mathcal{L}'$ *ends at vertex* $(i^{tr}, j^{dr}) = (4,0)$*. Note that, compared to Example 5, the forward path is one arc/vertex longer but the backward path is two arcs/vertices shorter.*

Clearly, the merged truck path (in forward direction) is $(0, 1, 2, 4, 0')$. *The first partial path has two drone subpath* $\langle 1, 3, 2\rangle$ *and* $\langle 2, \cdot, \cdot\rangle$, *where the latter is incomplete. Likewise, the second partial path has an incomplete subpath* $\langle 0, \cdot, \cdot\rangle$. *The two incomplete subpaths can be combined into one subpath* $\langle 2, \cdot, 0'\rangle$ *(by reversing* $\langle 0, \cdot, \cdot\rangle$*). For the drone subpath to be valid, a drone customer* k^* *needs to be chosen from the set* N *so that it does not conflict with the elementarity constraints.* □

The example shows that an appropriate drone customer k must be chosen when merging two partial paths with an incomplete (last) drone subpath. Let label $\mathcal{L} = (R)$ represent a partial path ending at $(i^{\mathrm{tr}}, i^{\mathrm{dr}}) \in W$ with $i^{\mathrm{tr}} \neq i^{\mathrm{dr}}$ and label $\mathcal{L}' = (R')$ be a partial paths ending at $(j^{\mathrm{tr}}, j^{\mathrm{dr}}) \in W$ with $j^{\mathrm{dr}} \neq i^{\mathrm{dr}}$. When merging $\mathcal{L}$ and $\mathcal{L}'$, the completed drone subpath is $\langle i^{\mathrm{dr}}, k, j^{\mathrm{dr}}\rangle$ for a customer $k \in N$ served on the additional drone arc. The reduced cost of the resulting route r is

$$\tilde{c}_r = R^{rdc} + R'^{rdc} + \pi_{i^{\mathrm{tr}}} + c^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + c^{\mathrm{dr}}_{k,j^{\mathrm{dr}}} - \pi_k \quad \text{or} \tag{8b}$$

$$\tilde{c}_r = R^{rdc} + R'^{rdc} + \pi_{i^{\mathrm{tr}}} + \max\left\{R^{dur} + R'^{dur}, t^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + t^{\mathrm{dr}}_{k,j^{\mathrm{dr}}}\right\} - \pi_k \tag{8c}$$

depending on the objective, i.e., either cost or duration minimization. The route is feasible if

$$R^{load} + R'^{load} + q_{i^{\mathrm{tr}}} + q_k \leq Q \tag{9a}$$

$$R^{cust,n} + R'^{cust,n} \leq \begin{cases} 1, & n \in N \setminus \{i^{\mathrm{tr}}, k\} \\ 0, & n = k \end{cases} \tag{9b}$$

Accordingly, to obtain a minimum reduced-cost route, the drone customer k^* is chosen as

$$k^* = \operatorname*{arg\,min}_{k \in N(R,R',i^{\mathrm{tr}})} \left\{c^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + c^{\mathrm{dr}}_{k,j^{\mathrm{dr}}} - \pi_k\right\} \quad \text{or} \quad k^* = \operatorname*{arg\,min}_{k \in N(R,R',i^{\mathrm{tr}})} \left\{\max\{R^{dur} + R'^{dur}, t^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + t^{\mathrm{dr}}_{k,j^{\mathrm{dr}}}\} - \pi_k\right\} \tag{10}$$

where the subset of the customers to choose from is defined as $N(R, R', i^{\mathrm{tr}}) = \{k \in N : (9) \text{ is fulfilled}\}$. For convenience, we assume that the minimum is ∞ for $N(R, R', i^{\mathrm{tr}}) = \varnothing$. All routes with $\tilde{c}_r \geq 0$ are rejected in the merge procedure.

Finally, to avoid the generation of the same route multiple times, we restrict the merge to the cases when either $R^{load} > Q/2$ or $R'^{load} > Q/2$ or $i^{\mathrm{tr}} = 0'$ holds true.

4.2.5. ng-Path Relaxation

The elementarity constraints are those constraints that make many SPPRCs not only theoretically but also practically difficult to solve. The use of an SPPRC relaxation offers an easier solution of the pricing problems but comes at the cost of a generally weaker linear relaxation of the corresponding MP. For the CVRP and the *VRP with time windows* (VRPTW), Baldacci *et al.* (2011) introduced the *ng*-path relaxations to gradually control the trade-off between the strength of the LP-relaxation and the practical difficulty of the SPPRC pricing problems. The concrete *ng*-route relaxation is defined by neighborhoods $N_i \subseteq N$, one for each vertex i. A cycle $(i, i_1, \ldots, i_\ell, i)$ over vertex i (with $\ell \geq 1$) is feasible in the relaxation, if at least one vertex i_p (for some $p \in \{1, 2, \ldots, \ell\}$) fulfills $i \notin N_{i_p}$.

The *ng*-path relaxation is not directly applicable in the artificial network of the VRP-D. We adapt it as follows:

- Each vertex $(i^{\mathrm{tr}}, i^{\mathrm{dr}})$ of the artificial network is equipped with its own neighborhood.
- However, all neighborhoods are subsets $N_{(i^{\mathrm{tr}},i^{\mathrm{dr}})} \subseteq N$ of the customers, i.e., the service tasks of the original network.
- For simplicity, we define $N_{(i^{\mathrm{tr}},i^{\mathrm{dr}})}$ based on the n_{ng} customers closest to i^{tr} (parameter n_{ng} controls the size of the neighborhoods and, herewith, the strength and difficulty of the relaxation).
- In particular, for $i^{\mathrm{tr}} \in N$, the neighborhood $N_{(i^{\mathrm{tr}},i^{\mathrm{dr}})}$ contains customer i^{tr}.

The REF (4c) is altered by setting $R^{cust,n}_j = 0$ if $n \notin N_j$. The BPC implementation used in the computational experiments presented in Section 5 uses $n_{\mathrm{ng}} = 8$. Note that the merge conditions $R^{cust,n} + R'^{cust,n} \leq 1$ and (9b) remain valid.

Monodirectional (forward) labeling and bidirectional labeling have slightly different *ng*-path relaxations even if the neighborhoods are chosen identically: The monodirectional relaxation can be weaker. The effect is observed for routes with two drone subpaths serving the same drone customer k. Such a non-elementary route can be feasible in the *ng*-path relaxation when labeling is performed completely in forward direction, but the same route can be excluded in the merge.

Example 8. *We consider a route with truck path* $(0, 1, 2, 3, 0')$ *and two drone subpaths* $\langle 0, 4, 2\rangle$ *and* $\langle 2, 4, 0'\rangle$*, i.e., the drone customer* $k = 4$ *is served two times. In the artificial network* $\mathcal{N} = (W, A)$*, the associated route is*

$$(0,0) - (1,0) - (2,0) \overset{4}{-} (2,2) - (3,2) - (0',2) \overset{4}{-} (0',0').$$

If the neighborhood of vertex 3 *or* $0'$ *does not include* $k = 4$*, this route is feasible in the forward labeling algorithm.*

For the bidirectional case, we assume that the merge happens at vertex $(i^{tr}, i^{dr}) = (2, 2)$ *and that vertex 2 has* $k = 4$ *included in its neighborhood. Then, the two partial paths are*

$$(0,0) - (1,0) - (2,0) \overset{4}{-} (2,2) \quad \textit{and} \quad (0,0) - (3,0) - (2,0) \overset{4}{-} (2,2)$$

are both ng-feasible with resource values $R^{cust,4} = R'^{cust,4} = 1$ *at the merge vertex* $(2, 2)$*. However, the merge condition for the case* $i^{tr} = i^{dr}$ *requires* (7b)*, i.e.,* $R^{cust,4} + R'^{cust,4} \leq 1$*, which is clearly violated. Hence, the considered route is excluded in the implicit bidirectional labeling.* □

Finally, note that inconsistent results would be observed if the half-way point would be chosen dynamically (Tilk *et al.*, 2017). However, implicit bidirectional labeling requires a fixed half-way point exactly in the middle at $Q/2$.

4.2.6. Acceleration Techniques

We now describe exact and heuristic acceleration techniques for the column-generation process.

Acceleration of the Merge Procedure. Pre-tests have shown that the merge procedure is often the bottleneck of the column-generation process. In comparison to classical VRPs (like CVRP, VRPTW, etc.), the artificial network contains a quadratic number $\mathcal{O}(m^2)$ of vertices $(i^{\text{tr}}, i^{\text{dr}})$, and for those with $i^{\text{tr}} \neq i^{\text{dr}}$ the merge must be performed 'over arcs'. All these arcs are drone arcs $[(i^{\text{tr}}, i^{\text{dr}}), (i^{\text{tr}}, j^{\text{dr}}), k]$ for some $j^{\text{dr}} \neq i^{\text{dr}}$ and $k \in N$. As a result, the number of arcs to consider in the merge is $\mathcal{O}(m^4)$. This explains the empirical observation that the merge is very time consuming.

To reduce the computational effort of the merge procedure, we apply two exact acceleration techniques: First, when the forward labeling process is finished, all labels R belonging to the same vertex $(i^{\text{tr}}, i^{\text{dr}})$ are sorted in non-decreasing order by their load resource R^{load}. If truck and drone are at the same position, i.e., both forward and backward label belong to a vertex $(i^{\text{tr}}, i^{\text{dr}})$ with $i^{\text{tr}} = i^{\text{dr}}$, the capacity constraint $R^{load} + R'^{load} + q_{i^{\text{tr}}} \leq Q$ can be exploited as follows. In two nested loops running over candidate labels R and R' with $R^{load} \leq R'^{load}$, the inner loop for R' can be discontinued as soon as $R^{load} + R'^{load} + q_{i^{\text{tr}}} > Q$. Moreover, the outer loop for R can be discontinued as soon as $2R^{load} + q_{i^{\text{tr}}} > Q$. If truck and drone are at different positions, then we know that the backward label must belong to another vertex $(i^{\text{tr}}, j^{\text{dr}})$ with $j^{\text{dr}} \neq i^{\text{dr}}$. In this case, the capacity constraint can be further exploited: The load onboard is not smaller than $R^{load} + R'^{load} + q_{i^{\text{tr}}} + \min_{k \in N}\{q_k\}$. Hence, when considering the two nested loops, the outer over labels R at vertex $(i^{\text{tr}}, i^{\text{dr}})$ and the inner over labels R' at vertex $(i^{\text{tr}}, j^{\text{dr}})$, the inner loop can be discontinued if the above value exceeds Q.

Second, the determination of a best drone customer in (10) is another time-consuming task that has to be done numerous times in the merge procedure. To reduce te computational burden, we pre-compute the reduced cost for each possible drone subpath $\langle i^{\text{dr}}, k, j^{\text{dr}}\rangle$ before the actual procedure starts. For each combination $(i^{\text{dr}}, j^{\text{dr}}) \in V \times V$, the reduced cost of the drone subpath (i.e., $c^{\text{dr}}_{i^{\text{dr}},k} + c^{\text{dr}}_{k,j^{\text{dr}}} - \pi_k$ or $t^{\text{dr}}_{i^{\text{dr}},k} + t^{\text{dr}}_{k,j^{\text{dr}}} - \pi_k$) is stored in a list together with the respective drone customer k. The list is then sorted in non-decreasing order by reduced cost. When using the cost objective, a drone customer $k \in N(R, R', i^{\text{tr}}) \subset N$ for a merge

with minimum reduced cost is found by inspecting the sorted list until a feasible merge is found (one that fulfills (9)). When using the duration objective, however, the reduced cost of the merge also depends on the sum $R^{dur} + R'^{dur}$ and we cannot determine a merge with minimum reduced cost as the first one in the sorted list that is feasible. However, inspecting potential drone customers in sorted ordering enables breaking the loop over $k \in N(R, R', i^{\text{tr}})$ as soon as reduced cost can be estimated being non-negative, which accelerates the pricing also in this case.

Partial Pricing with Reduced Networks. We use a hierarchy of heuristics to quickly identify negative reduced-cost routes. The use of one or more heuristics is also known as *partial pricing* (Gamache *et al.*, 1999). In the context of dynamic programming-based labeling, partial pricing results from using the bidirectional labeling algorithm applied to an appropriate subgraph of the artificial network $\mathcal{N} = (W, A)$. For the VRP-D, we use two reduced networks. The first has up to four ingoing and four outgoing truck arcs ($A^{alone} \cup A^{tgthr}$) as well as at most two drone arcs (A^{drone}) per artificial vertex. The second reduced network doubles the number of arcs per vertex (eight and four). To build the networks, the arcs with the smallest reduced cost in the current pricing iteration are selected. Pricing always starts with the first network, and the pricing problem is solved exactly only when both heuristics fail to provide any routes with negative reduced cost.

4.3. Valid Inequalities

We strengthen the linear relaxation of formulation (2) with the help of *subset-row inequalities* (SRIs, Jepsen *et al.*, 2008) and non-robust *capacity cuts* (CCs, Baldacci *et al.*, 2007).

A SRI is defined for a subset $S \subset N$ of customers and non-negative weights for each $i \in S$. We restrict ourselves to subsets of size three and weight $1/2$, because these can be easily separated by straightforward enumeration and handled with a single binary attribute per active SRI in the labeling. The valid inequality reads:

$$\sum_{r \in \Omega} \left\lfloor \sum_{i \in S} \frac{b_{ir}}{2} \right\rfloor \lambda_r \leq 1 \tag{11}$$

A CC is also defined for a subset $C \subset N$ of customers. Let K_C be a lower bound on the number of routes needed to serve C. The valid inequality for (C, K_C) is

$$\sum_{r \in \Omega} \max_{i \in C} \{\min\{1, b_{ir}\}\} \lambda_r \geq K_C, \tag{12}$$

where we use the easy-to-compute lower bound $K_C = \lceil \sum_{i \in C} q_i / Q \rceil$ (the exact solution of a bin-packing problem like in Section 4.1 would sometimes create stronger inequalities at the cost of a much more time consuming separation procedure).

Let $\mathcal{S}$ and $\mathcal{C}$ denote the sets of active SRIs and CCs, respectively. Further, let $\sigma_S < 0$ be the dual price of the SRI defined by $S \in \mathcal{S}$ and let $\tau_C > 0$ be the dual price of the CC defined by $(C, K_C) \in \mathcal{C}$. We define the additional binary resources $(R^{sri,S})_{S \in \mathcal{S}}$ and $(R^{cc,C})_{(C,K_C) \in \mathcal{C}}$. Initially, they are set to $(R^{sri,S}) = \mathbf{0}$ and $(R^{cc,C}) = \mathbf{0}$. For every second time a partial path serves a customer in S, the dual price σ_S has to be subtracted from the reduced cost. Similarly, the first time when a partial path serves a customer in C, the dual price τ_C has to be subtracted from the reduced cost. As for the load resource, it is important for a correct bidirectional labeling to delay the manipulation of the resource values when propagating over an

arc $a = [(i^{\mathrm{tr}}, i^{\mathrm{dr}}), (j^{\mathrm{tr}}, j^{\mathrm{dr}}), k] \in A$. This is modeled in the following way:

$$R_j^{rdc} = \begin{cases} R_i^{rdc} + c_{i^{\mathrm{tr}},j^{\mathrm{tr}}} - \pi_{j^{\mathrm{tr}}} - \sum\limits_{\substack{S\in\mathcal{S}: R_i^{sri,S}=1,\\ i^{\mathrm{tr}}\in S}} \sigma_S - \sum\limits_{\substack{C\in\mathcal{C}: R_i^{cc,C}=0,\\ i^{\mathrm{tr}}\in C}} \tau_C, & \text{if } a \in A^{alone} \cup A^{tgthr} \\ R_i^{rdc} + c^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + c^{\mathrm{dr}}_{k,j^{\mathrm{dr}}} - \pi_k - \sum\limits_{\substack{S\in\mathcal{S}: R_i^{sri,S}=1,\\ k\in S}} \sigma_S - \sum\limits_{\substack{C\in\mathcal{C}: R_i^{cc,C}=0,\\ k\in C}} \tau_C, & \text{if } a \in A^{drone} \end{cases} \tag{4a'}$$

$$R_j^{sri,S} = \begin{cases} 1 - R_i^{sri,S}, & \text{if } (a \in A^{alone} \cup A^{tgthr} \text{ and } i^{\mathrm{tr}} \in S) \text{ or } (a \in A^{drone} \text{ and } k \in S) \\ R_i^{sri,S}, & \text{otherwise} \end{cases}$$

$$R_j^{cc,C} = \begin{cases} 1, & \text{if } (a \in A^{alone} \cup A^{tgthr} \text{ and } i^{\mathrm{tr}} \in C) \text{ or } (a \in A^{drone} \text{ and } k \in C) \\ R_i^{cc,C}, & \text{otherwise} \end{cases}$$

The new formula (4a') for the update of the reduced cost replaces the original REF (4a) for the cost-minimization objective. Likewise,

$$R_j^{rdc} = \begin{cases} R_i^{rdc} - \pi_{j^{\mathrm{tr}}} - \sum\limits_{\substack{S\in\mathcal{S}: R_i^{sri,S}=1,\\ i^{\mathrm{tr}}\in S}} \sigma_S - \sum\limits_{\substack{C\in\mathcal{C}: R_i^{cc,C}=0,\\ i^{\mathrm{tr}}\in C}} \tau_C, & \text{if } a \in A^{alone} \\ R_i^{rdc} + t_{i^{\mathrm{tr}},j^{\mathrm{tr}}} - \pi_{j^{\mathrm{tr}}} - \sum\limits_{\substack{S\in\mathcal{S}: R_i^{sri,S}=1,\\ i^{\mathrm{tr}}\in S}} \sigma_S - \sum\limits_{\substack{C\in\mathcal{C}: R_i^{cc,C}=0,\\ i^{\mathrm{tr}}\in C}} \tau_C, & \text{if } a \in A^{tgthr} \\ R_i^{rdc} + \max\{t^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + t^{\mathrm{dr}}_{k,j^{\mathrm{dr}}}, R_i^{dur}\} - \pi_k - \sum\limits_{\substack{S\in\mathcal{S}: R_i^{sri,S}=1,\\ k\in S}} \sigma_S - \sum\limits_{\substack{C\in\mathcal{C}: R_i^{cc,C}=0,\\ k\in C}} \tau_C, & \text{if } a \in A^{drone} \end{cases} \tag{6a'}$$

replaces the original REF (6a) for the duration objective. There are no feasibility constraints associated with new resources. The dominance Rules 1 and 2 have to be modified in the standard way as explained in (Jepsen *et al.*, 2008) and (Baldacci *et al.*, 2007). Moreover, in the merge procedure, reduced cost computations (8a) and (8b) have to be altered. For the merge of two labels R and R' at the same vertex $(i, i) = (i^{\mathrm{tr}}, i^{\mathrm{dr}})$, the reduced cost becomes

$$\tilde{c}_r = R^{rdc} + R'^{rdc} + \pi_i + \underbrace{\left(\sum_{\substack{C\in\mathcal{C}:\\ R^{cc,C}=R'^{cc,C}=1}} \tau_C - \sum_{\substack{S\in\mathcal{S}:\\ R^{sri,S}=R'^{sri,S}=1}} \sigma_S - \sum_{\substack{S\in\mathcal{S}: i\in S\\ R^{sri,S}+R'^{sri,S}=1}} \sigma_S - \sum_{\substack{C\in\mathcal{C}: i\in C\\ R^{cc,C}=R'^{cc,C}=0}} \tau_C \right)}_{(*)}$$

(the first sum corrects dual prices of CCs whose customers have been visited in both partial paths, the second sum incorporates the dual prices of SRIs with an odd number of visits in both partial paths, the third incorporates dual prices of SRIs with $i \in S$, and the last sum CCs where only customer i is visited).

For the merge of a label R at vertex $(i^{\mathrm{tr}}, i^{\mathrm{dr}})$ with $i^{\mathrm{tr}} \neq i^{\mathrm{dr}}$ and a label R' at vertex $(i^{\mathrm{tr}}, i^{\mathrm{dr}})$ with $i = i^{\mathrm{tr}}$ and $j^{\mathrm{dr}} \neq i$ serving customer $k \in N$ on the drone arc, the reduced cost becomes

$$\begin{aligned} \tilde{c}_r \;=\; & R^{rdc} + R'^{rdc} + \pi_{i^{\mathrm{tr}}} + c^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + c^{\mathrm{dr}}_{k,j^{\mathrm{dr}}} - \pi_k \\ & - \sum_{\substack{S\in\mathcal{S}: k\in S, i^{\mathrm{tr}}\notin S\\ R^{sri,S}+R'^{sri,S}=1}} \sigma_S - \sum_{\substack{S\in\mathcal{S}: k\in S, i^{\mathrm{tr}}\in S\\ R^{sri,S}=R'^{sri,S}=0}} \sigma_S - \sum_{\substack{C\in\mathcal{C}: k\in C, i^{\mathrm{tr}}\notin S\\ R^{cc,C}=R'^{cc,C}=0}} \tau_C \;+\; (*) \end{aligned}$$

in case of the cost-based objective and

$$\begin{aligned} \tilde{c}_r \;=\; & R^{rdc} + R'^{rdc} + \max\left\{ R^{dur} + R'^{dur}, t^{\mathrm{dr}}_{i^{\mathrm{dr}},k} + t^{\mathrm{dr}}_{k,j^{\mathrm{dr}}} \right\} + \pi_{i^{\mathrm{tr}}} - \pi_k \\ & - \sum_{\substack{S\in\mathcal{S}: k\in S, i^{\mathrm{tr}}\notin S\\ R^{sri,S}+R'^{sri,S}=1}} \sigma_S - \sum_{\substack{S\in\mathcal{S}: k\in S, i^{\mathrm{tr}}\in S\\ R^{sri,S}=R'^{sri,S}=0}} \sigma_S - \sum_{\substack{C\in\mathcal{C}: k\in C, i^{\mathrm{tr}}\notin S\\ R^{cc,C}=R'^{cc,C}=0}} \tau_C \;+\; (*) \end{aligned}$$

in case of the duration objective.

Violated SRIs can be identified with a straightforward approach that enumerates all subsets S with $|S| = 3$ and then determines the left-hand side of (11), checks whether it is greater than 1, and stores S together with the degree of violation. Separating violated CCs requires more computational effort. For the capacitated arc routing problem, Martinelli *et al.* (2013) introduced a MIP-based CC separation algorithm that simultaneously computes C and K_C (for the above lower bound). The algorithm is directly applicable here.

Pretests have shown that adding many SRIs makes the labeling algorithm considerably more difficult. In contrast, the labeling algorithm can cope much better with CCs. Therefore, we always start with solving the separation problem for CCs and search for violated SRIs only if no CCs have been added. Moreover, we limit the total number of violated SRIs to be added to a maximum of 10 inequalities in total. Among the most violated inequalities, we allow no more than five per round and limit the number of SRIs per customer to not more than three. Violated inequalities are added only at the root node and we allow up to 100 CCs in total.

4.4. Branching

Let (λ_r^*, f^*) be a solution of the RMP. Whenever some component of (λ_r^*, f^*) is fractional, branching is necessary to obtain an integer solution. We apply the following four-level hierarchical branching scheme: (1) total number of routes, (2) number of drone visits at each customer, (3) flow of trucks on customer-to-customer arcs, and (4) flow on drone subpaths. At each branch-and-bound node, all routes that are incompatible with the current branching decision are temporarily removed from the RMP. The branch-and-bound tree is explored using a best-first search. Note that all constraints added to the RMP enforce additional dual prices in the labeling algorithm that all can be added on the corresponding arcs. We describe the four levels in detail now.

At the first level, when f^* is fractional, we branch on the total number of routes by creating two branches defined by $f \leq \lfloor f^* \rfloor$ and $f \geq \lceil f^* \rceil$, which is implemented by changing the bounds in (2d).

At the second level, we branch on the number of drone services performed for each customer (as proposed by Roberti and Ruthmair (2021) as a first-level branching decision in the algorithms for the TSP-D). To this end, let y_{kr} be the number of drone services performed by route r at customer k, which is the number of drone arcs of type $[(\cdot,\cdot),(\cdot,\cdot),k]$ used in the artificial network. If $y_k^* = \sum_{r\in\Omega} y_{kr}\lambda_r^*$ is fractional for some customers $k \in N$, we choose a customer k^* with value $y_{k^*}^* - \lfloor y_{k^*}^* \rfloor$ closest to 0.5 and branch on $y_{k^*} = 0$ and $y_{k^*} = 1$. Instead of adding a constraint to the RMP, these decisions can be enforced directly on the artificial network: The zero-branch results from eliminating all drone arcs of type $[(\cdot,\cdot),(\cdot,\cdot),k]$ serving customer k^*. Moreover, k^* must be removed from the sets $N(R, R', i^{\text{tr}})$ that are used in the merge procedure. The one-branch results from eliminating all vertices representing that customer k^* is served by a truck, i.e., all vertices $(k^*,\cdot) \in W$ and $(\cdot,k^*) \in W$.

At the third level, we consider pairs of customers $i, j \in N, i \neq j$ and the flow induced by truck movements between the two customers. Formally, we consider the following subset of together arcs and alone arcs

$$A_{ij}^{\text{tr}} = \{[(i,i),(j,j),\perp],[(j,j),(i,i),\perp]\} \cup \left\{[(i,j^{\text{dr}}),(j,j^{\text{dr}}),\perp],[(j,j^{\text{dr}}),(i,j^{\text{dr}}),\perp] \in A^{alone} : j^{\text{dr}} \in V\right\}$$

and determine the flow values $e_{ij}^* = \sum_{r\in\Omega}\sum_{a\in A_{ij}^{\text{tr}}} e_{ar}\lambda_r^*$ for each customer pair i and j, where e_{ar} is the (standard) flow value of route r on arc a. Note that the symmetry in the definition of A_{ij}^{tr} implies $e_{ij}^* = e_{ji}^*$ for all $i, j \in N, i \neq j$. If at least one of these values is fractional, we choose a customer pair i^* and j^* with value $e_{i^*j^*}^* - \lfloor e_{i^*j^*}^* \rfloor$ closest to 0.5 and create the two branches $\sum_{r\in\Omega}\sum_{a\in A_{ij}^{\text{tr}}} e_{ar}\lambda_r = 0$ and $\sum_{r\in\Omega}\sum_{a\in A_{ij}^{\text{tr}}} e_{ar}\lambda_r = 1$. As in the second level, the zero-branch can be directly enforced by removing all arcs A_{ij}^{tr} from the artificial network. The one-branch results from adding the above constraint to the RMP. In addition, it is feasible and advantageous to reduce the artificial network by removing further arcs that are incompatible with the branching constraint. In particular, the enforced truck customers i^* and j^* must never be served by a drone so that the arcs of type $[(\cdot,\cdot),(\cdot,\cdot),i^*] \in A^{drone}$ and $[(\cdot,\cdot),(\cdot,\cdot),j^*] \in A^{drone}$ can be eliminated. Likewise, i^* and j^* are disregarded as drone customers in the merge procedure. Branching decisions on the third level finally ensure unique truck paths. However, drone subpaths can still be fractional.

Hence, at the fourth level, we consider the flow on drone subpaths for branching. Recall that any drone subpath $\langle j,k,l\rangle$ has a symmetric counterpart $\langle l,k,j\rangle$, which are uniquely determined by the two drone arcs $[(l,j),(l,l),k]$ and $[(j,l),(j,j),k]$. We define

$$A^{\mathrm{dr}}_{\langle j,k,l\rangle} = \{[(l,j),(l,l),k],[(j,l),(j,j),k]\}$$

and determine the flow values $e^*_{\langle j,k,l\rangle} = \sum_{r\in\Omega}\sum_{a\in A^{\mathrm{dr}}_{\langle j,k,l\rangle}} e_{ar}\lambda^*_r$ for each drone subpath $\langle j,k,l\rangle$ with $j,l \in V, k \in N$, $j \neq l \neq k$, and $j \neq k$. Note that the symmetry in the definition of $A^{\mathrm{dr}}_{\langle j,k,l\rangle}$ implies $e^*_{\langle j,k,l\rangle} = e^*_{\langle l,k,j\rangle}$. We apply a similar strategy as at the third level: For the triplet (j^*,k^*,l^*) with value $e^{\mathrm{dr}}_{\langle j^*,k^*,l^*\rangle} - \lfloor e^{\mathrm{dr}}_{\langle j^*,k^*,l^*\rangle}\rfloor$ closest to 0.5, the two branches $\sum_{r\in\Omega}\sum_{a\in A^{\mathrm{dr}}_{\langle j,k,l\rangle}} e_{ar}\lambda_r = 0$ and $\sum_{r\in\Omega}\sum_{a\in A^{\mathrm{dr}}_{\langle j,k,l\rangle}} e_{ar}\lambda_r = 1$ are created. Again, the zero-branch results from removing the arcs $A^{\mathrm{dr}}_{\langle j^*,k^*,l^*\rangle}$ from the artificial network. The one-branch is enforced by adding the corresponding constraint to the RMP. As before, the artificial network can be reduced by eliminating arcs that are inconsistent with this branching decision: The vertices (k^*,i) and $(i,k^*) \in W$ for all $i \in N$ can be removed because the customers i are not visited by a truck alone. Further, all drone arcs of type $[(\cdot,\cdot),(\cdot,\cdot),j^*]$ and $[(\cdot,\cdot),(\cdot,\cdot),l^*]$ can be eliminated because the drone never serves the truck customers j^* and l^*. Finally, j^* and l^* are disregarded as drone customers in the merge procedure.

The branching decisions at the third and fourth level imply unique truck paths and drone subpaths. Since a solution is fully determined by both types of decisions, this four-level branching scheme is complete.

5. Computational Results

The following experiments were conducted on a standard PC equipped with 64 GB RAM running MS Windows 10 on an Intel® Core™ i7-5930K CPU clocked at 3.5 GHz. The BPC algorithm is coded in `C++` and compiled with MS Visual Studio Community 2022 in release mode into a 64-bit single-thread executable. The callable library of CPLEX 20.1.0 is used for the (re)optimization of the RMPs, for solving a primal MIP-based heuristic, and for separating CCs. CPLEX's default parameters are kept, except for setting the number of threads to one. The MIP-based heuristic solves a restricted integer MP using all columns generated up to this point. It is applied at the first and second level of the branch-and-bound-tree as well as if the BPC algorithm terminates without finding a feasible solution. The BPC computation time is limited to 3600 seconds per instance and additional 60 seconds are granted for the MIP-based heuristic at the end.

5.1. VRP-D Instances

Up to date, no suitable VRP-D benchmark set has been made available. Therefore, we created new VRP-D instances based on the *Capacitated Vehicle Routing Problem Library* (CVRPLIB) publicly available at `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/` to properly evaluate the performance of our BPC algorithm. We use CVRPLIB instances from *set A* and *set B* (proposed by Augerat *et al.*, 1995) and consider the first 20, 30, 40, or 50 vertices, respectively. The first vertex always represents the depot. As a result, instances have $m = 19$, 29, 39, or 49 customers. For each size m, we consider 20 instances so that the VRP-D instance set comprises 80 instances.

VRP-D-specific data is determined in the following way: All customers with a demand greater than $\bar{q} = Q/5$ are truck-only customers. The share of truck-only customers varies between 10 and 33 percent with an average of 20 percent over all instances. As a sidenote, if the value is kept in that interval, we could not see predictable differences in the difficulty of the instances. This statement can be proven with the instance-specific results presented in Tables A.6–A.13 in the appendix. The literature stresses that routing costs and travel times are lower for drones than for trucks. Hence, we compute c_{ij} and t_{ij} for trucks as Manhattan distances. In contrast, c^{dr}_{ij} and t^{dr}_{ij} for drones are computed as Euclidean distances divided by the given drone speed factor β. We set $\beta = 3$ as the default value and analyze the impact of the drones' speed in Section 5.5 using alternative factors $\beta = 1$ and $\beta = 5$.

5.2. Acceleration of the Merge Procedure

When using implicit bidirectional labeling, pretest have shown that the merge procedure is the bottleneck operation of the entire BPC algorithm. Therefore, we first investigate the effects of the acceleration techniques presented in Section 4.2.6 on the time spent in the merge procedure. The following four algorithmic setups are considered: (*None*) no acceleration technique is used for the merge, (*Pre*) the reduced cost of each possible drone subpath is pre-computed, (*Sort*) all labels belonging to the same vertex are sorted by their load resource, and (*Both*) both acceleration techniques are combined. We restrict the analysis to the cost objective function.

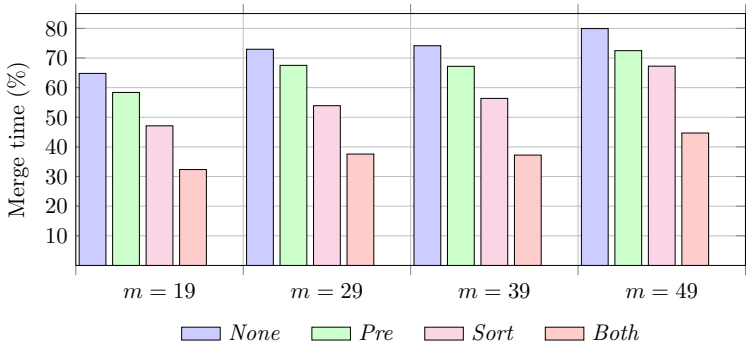


Figure 3: Average share (in percent) of the total computation time spent in the merge procedure for different acceleration techniques.

Figure 3 shows the share of the total BPC computation time spent in the merge procedure for the above acceleration techniques. Results are aggregated over instance of the same size. For all setups, the share of the merge time increases with the size m of the instances. For setup *None*, the merge accounts for 65 to 80 percent of the total BPC runtime. *Pre*-computation of reduced cost of drone subpaths helps to decrease the merge time by approximately seven percentage points on average. The effect of sorting (*Sort*) labels by their load resource is also advantegeous, since it reduces the share by approximately 16.5 percentage points compared to setup *None*. However, even if absolute savings are larger for *Sort* than for *Pre*, the respective savings in relative computation time decrease with the size m of the instance. Combining both techniques (*Both*) reduces the merge times considerably: compared to setup *None*, they are almost halved. Interestingly, the sum of the reductions resulting from *Pre* and *Sort* (compared to *None*) is smaller than the reduction resulting from *Both*.

For all experiments conducted in the following, we integrate the pre-computation of the reduced cost of drone subpaths and the sorting of the labels, i.e., setup *Both* becomes the default.

5.3. Comparison of Forward and Bidirectional Labeling

We conduct two types of computational experiments to compare the performance of the forward and the implicit bidirectional labeling strategy in the BPC algorithm. In both experiments, we again use the objective that minimizes routing cost.

In the first analysis, we consider the root node, i.e., the linear relaxation of the MP including the strengthening with SRIs and CCs. In each pricing iteration, the SPPRC subproblem is solved using both labeling strategies (forward and implicit bidirectional) applying the labeling algorithms consecutively. Since the dual prices and reduced costs used in both labeling algorithms are identical, a perfectly fair comparison is ensured. We add the negative reduced cost routes generated by the bidirectional algorithm to the RMP (this choice is arbitrary). As performance measures, we use the ratio of (1) the number of generated labels, (2) the number of processed labels, and (3) the total pricing time in the implicit bidirectional and forward labeling. The ratios are aggregated by taking the geometric mean over all pricing iterations of a VRP-D instance. Whenever the linear relaxation is not completely solved within the time limit, we omit the last iteration for both strategies. To increase the accuracy, we also omit those pricing iterations that consume

less than 100 milliseconds of computation time. Table 2 reports the minimum (*min*), geometric mean (*mean*) or arithmetic mean (*avg*), and maximum (*max*) of each measure, aggregated over VRP-D instances with an identical number of customers. Additionally, the table lists the number of iterations needed to solve the LP-relaxation of the MP.

	Ratio of											
	generated labels			processed labels			pricing time			Num. iterations		
m	min	mean	max	min	mean	max	min	mean	max	min	avg	max
19	0.22	0.26	0.30	0.16	0.19	0.25	0.20	0.29	0.40	1	23.85	58
29	0.21	0.23	0.27	0.15	0.17	0.22	0.22	0.28	0.54	19	69.25	109
39	0.20	0.22	0.26	0.14	0.17	0.20	0.02	0.23	0.59	41	88.25	136
49	0.18	0.21	0.27	0.14	0.17	0.21	0.02	0.21	0.42	64	98.20	149
Total		0.23			0.17			0.25			69.89	

Table 2: Comparison of forward and implicit bidirectional labeling using identical dual prices.

All ratios presented in the table are less than one, which shows the implicit bidirectional labeling strategy is superior to its monodirectional counterpart. The number of iterations is sufficiently large to provide statistically significant results. The number of generated (extended) labels reduces to 23% (17%) on average, i.e., by more than a factor of four (five). These reductions do not fully translate to the time consumed by the labeling algorithms, which is reduced to 25% on average (exactly factor four). Even in the worst-case (a VRP-D instance with $m = 39$ customers), the pricing time is still reduced to 59%. More importantly, all geometric means decrease with the number m of customers, which highlights that the advantage of using the implicit bidirectional labeling algorithm increases with the instance size.

In the second analysis, we compare two fully-fledged BPC algorithms described in Section 4, where one uses the forward labeling and the other the implicit bidirectional labeling algorithm. Detailed results for each instance can be found in Tables A.6–A.9 in the Appendix. Table 3 aggregates these results obtained with both BPC algorithms, again grouped by VRP-D instances with an identical number m of customers. The column entries have the following meaning: (#Inst) number of instances per group, (#Opt) number of instances solved to proven optimality within the time limit, (#UB) number of instances for which an upper bound was found, (Gap) average relative difference between the upper bound (UB) and lower bound (LB) in percent at termination, i.e., $100(UB - LB)/LB$ (computed only when an upper and lower bound was found), (Time) average solution time in seconds, and (#BB) average number of branch-and-bound nodes solved per VRP-D instance.

		Forward labeling					Implicit bidirectional labeling				
m	#Inst	#Opt	#UB	Gap	Time	#BB	#Opt	#UB	Gap	Time	#BB
19	20	19	20	<0.01	295.2	19.0	20	20	—	41.9	20.5
29	20	14	20	0.36	2,160.4	34.2	17	20	0.07	1,046.8	72.3
39	20	6	20	1.55	3,233.0	27.8	11	20	0.22	2,237.8	64.3
49	20	2	20	8.45	3,400.0	8.6	3	20	2.70	3,215.6	21.0
Total	80	41	80	2.51	2,272.1	22.4	51	80	0.75	1,635.5	44.5

Table 3: Comparison of two BPC algorithms equipped with a forward or implicit bidirectional labeling algorithm.

The results of the second analysis are very consistent over different instance sizes. The implicit bidirectional labeling algorithm outperforms the forward labeling algorithm, i.e., more instances are solved to optimality (51 versus 41), average gaps and computation times are smaller, and more branch-and-bound

nodes are processed (in shorter time). In total, the BPC algorithm with implicit bidirectional labeling is on average around 30% faster than the BPC with a forward labeling. The speedup for the smallest instances with $m = 19$ customers is more than by a factor of 7. The results for the instances with more customers are biased due to the one-hour time limit. Hence, the implicit bidirectional labeling is even more valuable than the 30% speedup suggests: For instances with $m = 29$ (39) customers, the number of proven optimal solutions increases from 14 to 17 (6 to 11), and for the largest instances with $m = 49$ customers, approximately 2.5 times more branch-and-bound nodes can be processed reducing the average gap from 8.5 to 2.7 percent.

All experiments conducted in the following use the implicit bidirectional labeling algorithm to solve the pricing subproblems.

5.4. Comparison of Cost and Duration Objectives

Next, we highlight the differences between the cost and duration objectives. Recall that, up to now, only results for the minimization of the routing costs have been shown. We evaluate the impact of the parameter $\beta \in \{1, 3, 5\}$, which is the discount factor of the routing costs of the drones in the cost-objective case and the speed factor of the drones (relative to the trucks' speed) in the duration-objective case. Table 4 shows results regarding the difficulty of the VRP-D instances for the BPC algorithm. The column entries have the same meaning as those in Table 3, now grouped by objective, the number of customers m, and the value of β. Note that the entries for $\beta = 3$ and cost minimization are identical to the ones shown in Table 3.

		Cost objective					Duration objective				
m	β	#Opt	#UB	Gap	Time	#BB	#Opt	#UB	Gap	Time	#BB
19	1	20	20	—	31.5	28.5	16	20	2.69	1,281.1	61.5
	3	20	20	—	41.9	20.5	19	20	0.02	525.9	170.8
	5	19	20	0.01	235.2	85.5	19	20	0.04	722.7	144.1
Subtotal		59	60	<0.01	102.9	44.8	54	60	0.92	843.2	125.4
29	1	20	20	—	183.3	20.4	2	20	3.11	3,446.8	18.2
	3	17	20	0.07	1,046.8	72.3	5	20	1.25	2,988.7	113.5
	5	18	20	0.05	965.8	91.8	4	20	2.30	2,994.3	80.3
Subtotal		55	60	0.04	732.0	61.5	11	60	2.22	3,143.3	70.6
39	1	9	20	0.80	2,424.1	91.0	0	20	5.47	3,660.0	3.6
	3	11	20	0.22	2,237.8	64.3	0	19	1.83	3,660.0	23.8
	5	7	20	0.65	2,641.8	54.3	1	19	1.84	3,484.0	26.7
Subtotal		27	60	0.56	2,434.5	69.8	1	58	3.04	3,601.3	18.0
49	1	6	20	2.07	2,927.6	23.7	0	20	4.44	3,660.0	0.3
	3	3	20	2.70	3,215.6	21.0	0	19	4.10	3,660.0	6.3
	5	2	20	3.38	3,392.1	20.9	0	17	3.45	3,660.0	9.6
Subtotal		11	60	2.72	3,178.4	21.8	0	56	4.00	3,660.0	5.4
Total		152	240	0.83	1,612.0	49.5	66	234	2.54	2,828.7	54.9

Table 4: Comparison of BPC results for objectives cost and duration minimization.

On average, VRP-D instances with duration objective are more difficult to solve for the BPC algorithm than the respective instance with cost objective. For the smallest instances with $m = 19$ customers, only 54 out of 60 instances are solved to proven optimality with an average gap of 0.92 percent for duration minimization, while almost all instances (59 out of 60) are solved approximately eight times faster leaving an average gap smaller than 0.01 percent in the cost-minimization case. For instances with $m = 29$ ($m = 39$) customers, the number of provably optimal solutions drops to 11 (1) compared to 55 (27) out of 60. For $m = 49$, no optima were computed for the duration-minimization objective. In these cases, also the MIP-based heuristic is less effective as it does not provide a solution and upper bound for six instances. The

increased difficulty of the duration objective can, e.g., be explained by the need to introduce the additional resource R^{dur} in the labeling algorithm (see Section 4.2.2).

Comparing different β-values, there are opposite tendencies visible for the two objectives. For cost minimization, the smaller instances (with $m = 19$ and 29) show that computation times increase with larger values of β. On the contrary, for $m = 19$ and 29, increasing β reduces the average computation time when duration minimization is the goal. For instances with more customers, average computation times are dominated by the number of unsolved instances (taken into account with 3660 seconds).

5.5. Effect of the Drones' Routing Cost and Speed

The routing cost and speed of drones compared to trucks have a strong impact on the structure of optimal and nearly optimal solutions. This last section details the impact in four analyses.

Conflicting Objectives. First, we show that the two objectives are conflicting. To this end, we re-evaluate optimal and best-known solutions for one objective using the other objective. These values are then compared with best-known solutions for the other objective. Figure 4 shows the resulting average percentage increase of the objective value for different values of β and m.

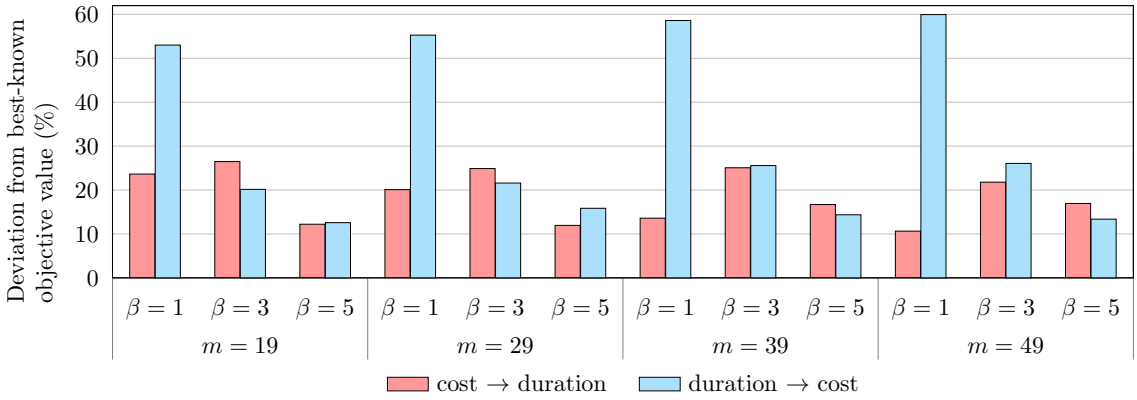


Figure 4: Increase of the objective value.

Considering the evaluation of solutions optimized with respect to routing cost using the duration objective (indicated as 'cost → duration'), we find that the deviations are highest (26.5%) for $\beta = 3$ and lower for $\beta = 1$ and 5 (the smallest deviation is 10.6%). Results for $m = 49$ should be considered with care, since the number of optimal solutions is small so that best-known upper bounds are used instead.

For the reverse evaluation (indicated as 'duration → cost'), the cost increase for $\beta = 1$ stands out. Solutions optimized with regard to duration are (on average) rather bad solutions for routing-cost minimization. For $\beta = 3$ and $\beta = 5$, the deviations resulting from the two re-evaluations are in a rather similar range for all instance sizes.

Share of drone customers. We now analyze the structure of optimal and best-known solutions. As shown in Table 5, the share of drone customers varies considerably depending on the objective, the size of the instance, and the β-value. On average, the share increases with β and when replacing cost by duration minimization. In turn, the share decreases slightly when instances become larger. The impact of the parameter β is as expected, since cheaper and faster drones foster their use.

The most surprising and striking result is that drones are hardly ever used when routing cost is minimized and trucks and drones have identical cost/speed characteristics ($\beta = 1$). The following example explains the above outcome.

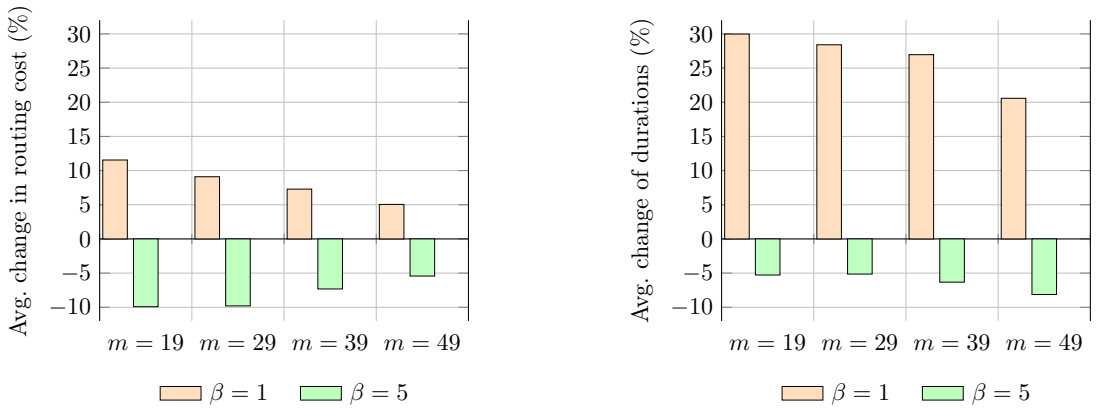
	Cost objective			Duration objective		
m	$\beta = 1$	$\beta = 3$	$\beta = 5$	$\beta = 1$	$\beta = 3$	$\beta = 5$
19	0	27	42	28	51	56
29	0	27	40	27	51	54
39	0	23	37	27	49	52
49	1	22	34	26	47	49

Table 5: Share (in percent) of drone customers in optimal/best-known solutions.

Example 9. *Consider a tiny VRP-D instance with three customers $N = \{1, 2, 3\}$. The complete undirected graph is then $G = (V, E)$ with $V = \{0, 1, 2, 3, 0'\}$. Recall from Section 5.1 that routing costs and travel times are computed from Manhattan distances for trucks and Euclidean distances for drones. Accordingly, we define $t_{ij} = c_{ij} = 2$ and $t^{dr}_{ij} = c^{dr}_{ij} = 1.5/\beta$ for all $\{i, j\} \in E$. (Similar distances of 2 and $1.5 \approx \sqrt{2}$ occur if coordinates differ by 1 in x- and y-direction.)*

When all three customers are served by the same truck, the corresponding route r has an objective value of $c_r = 8$, regardless of the order in which the customers are visited. This applies for both objectives. Alternatively, we assume that one of the customers is served by the drone and the two others by the same truck to which the drone is assigned. The corresponding route is denoted by r'. Its drone subpath has routing cost of $2 \cdot 1.5/\beta$, and the duration increases by $\max\{2, 2 \cdot 1.5/\beta\}$. Hence, routing costs accumulate to $c_{r'} = 6 + 3/\beta$, and the duration is $t_{r'} = 4 + \max\{2, 3/\beta\}$.

For $\beta < 1.5$, it follows $c_r < c'_r$ for routing costs so that the use of the drone is not beneficial (in particular for $\beta = 1$). This relation reverses for $\beta > 1.5$ so that the use of the drone is cost-effective (in particular for $\beta \in \{3, 5\}$). This reasoning is empirically proven by the results shown on the left-hand-side of Table 5. For the duration objective, the first travel time $t_r = 8$ is never smaller than the second travel time $t_{r'} = 4 + \max\{2, 3/\beta\}$ (assuming that the drone speed is never lower than the truck speed, i.e., $\beta \geq 1$). Hence, using the drone is always advantageous as can be seen from the results on the right-hand-side of Table 5. □



(a) Routing cost.

(b) Duration.

Figure 5: Average percentage change in routing cost and duration for different β-values.

Comparison of different β-Values. Next, we compare optimal and best-known solutions for $\beta = 3$ with optimal and best-known solutions for the values $\beta = 1$ and $\beta = 5$, respectively. Figure 5 visualizes the average percentage chances relative to the baseline solutions for $\beta = 3$. First and foremost, percentage changes are smaller for the cost-minimization (Figure 5a) and larger for the duration minimization (Figure 5b). For

routing costs, increasing the drones' routing costs by a factor of 3 (from $\beta = 3$ to $\beta = 1$) results in an overall cost increase between 5 and 12 percent, depending on the instance size. Reducing the dones' routing costs (from $\beta = 3$ to $\beta = 5$) generates cost savings in roughly the same amount. For duration minimization, the respective changes amount to between 20 and 30 percent and between 5 and 9 percent, respectively. The latter result means that drones should be faster than trucks, but further increasing their speed beyound $\beta = 5$ will not generate substantial additional savings.

Comparison with Approximation Results. The above results are in line with those of Carlsson and Song (2018), namely, that the efficacy of the combined truck and drone delivery increases with the ratio of the drone's and truck's speeds. Carlsson and Song's most important finding is that 'the improvement in efficiency [resulting from the use of a drone] is proportional to the square root of the ratio of the speeds of the truck and the unmanned aerial vehicle' [drone]. Carlsson and Song do not use MIP-based optimization techniques but a continuous approximation model (see Daganzo, 2005). They measure efficiency as the duration of a tour (a *max*-term like in the duration objective (1)). However, in their model, only the drone delivers to customers and the truck serves as a supportive vehicle (like in case (1) in Section 1). When reducing the drone speed from $\beta = 3$ to $\beta = 1$, Carlsson and Song's model predicts an increase of durations by 73 percent ($\sqrt{3} = 1.73$). Our results show relatively smaller duration increases of on average 26 percent, which can be partly explained by the fact that, in our setting, only approximately half of the customers can be served by drone (see Table 5). When increasing the drone speed from $\beta = 3$ to $\beta = 5$, they predict a reduction of durations by 23 percent ($\sqrt{3/5} = 0.77$). Our results are then rather consistent with the above numbers showing average duration reductions of only 6.2 percent.

6. Conclusions

Drones are an emerging technology. This explains why planning problems related to drones have attracted researchers from various disciplines, e.g., engineering, computer science, and control theory (Otto *et al.*, 2018). The VRP-D considered in this work is the prototypical routing problem for combined operations of trucks and drones when they are used as synchronized working units. Combined delivery operations make particularly sense when conventional vehicles cannot easily and quickly reach some delivery points, e.g., for parcel delivery to remote households, medical supply delivery (vaccines, blood, drugs) and collection (samples of blood, urine, etc.), delivery of spare parts, etc.

From an Operations Research perspective, the planning of combined delivery operations with trucks and drones collaborating in a synchronized way is very challenging. A a consequence, heuristic approaches are predominant. Complementing these approaches, we presented a new exact solution algorithm that is based on the BPC principle. Its most important algorithmic component is an innovative implicit bidirectional labeling algorithm for solving pricing subproblems. For this purpose, we adapted an artificial network originally introduced by Roberti and Ruthmair (2021) for modeling truck and drone synchronization in the TSP-D. In this artificial network, the same route, once considered in forward and once in backward direction, passes through different vertices. Our new implicit bidirectional labeling algorithm exploits the symmetry inherent in the VRP-D as it however does not distinguish forward and backward directions. Labeling performance is further improved by problem-tailored preprocessing and acceleration techniques of the merge procedure, which is otherwise the bottleneck of the overall solution algorithm. For the performance of the BPC algorithm, several known techniques had to be adapted to the VRP-D case: delayed propagation of some resource values is helpful to handle the non-robust cutting planes (subset-row inequalities and capacity cuts) that strengthen the linear relaxation of the column-generation MP. At the end, the BPC algorithm is able to exactly minimize the routing cost for instances of the VRP-D with up to 49 customers. The minimization of the duration is more difficult, but feasible solutions with an average gap of less than 2.2 (3.0) percent are found for instances with 29 (39) customers. These gaps tend to be smaller for realistic settings in which drones are significantly faster than trucks. Compared to other exact algorithms for variants of the VRP-D (see Section 2) that allow solution times of up to 3, 5, 10, and even 20 hours for instances with not more than 35 customers, our new BPC algorithm obtains the reported results within one hour. Roberti and Ruthmair (2021) have shown how to modify the artificial network to cope with TSP-D variants that

consider, e.g., drone loops in addition to done subpaths, a (possibly weight-dependent) drone flying range, and a maximum number of customers per truck segment. Using the same ideas, our approach is adaptable to the corresponding VRP-D variants.

Acknowledgement

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

References

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical report, Institut National Polytechnique, 38 - Grenoble (France).

Bakir, I. and Tiniç, G. Ö. (2020). Optimizing drone-assisted last-mile deliveries: The vehicle routing problem with flexible drones. Optimization online. https://optimization-online.org/?p=16382.

Baldacci, R., Christofides, N., and Mingozzi, A. (2007). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, **115**(2), 351–385.

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

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, **24**(3), 356–371.

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, **60**(5), 1167–1182.

Carlsson, J. G. and Song, S. (2018). Coordinated logistics with a truck and a drone. *Management Science*, **64**(9), 4052–4069.

Chung, S. H., Sah, B., and Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, **123**, 105004.

Costa, L., Contardo, C., and Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, **53**, 946–985.

Daganzo, C. (2005). *Logistics Systems Analysis*. Springer, Berlin, Germany, 4th edition.

D'Andrea, R. (2014). Guest editorial Can drones deliver? *IEEE Transactions on Automation Science and Engineering*, **11**(3), 647–648.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, **46**(3), 297–316.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations Research*, **47**(2), 247–263.

Goeke, D., Gschwind, T., and Schneider, M. (2019). Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier. *Discrete Applied Mathematics*, **264**, 43–61.

Goodchild, A. and Toy, J. (2018). Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO_2 emissions in the delivery service industry. *Transportation Research Part D: Transport and Environment*, **61**, 58–67.

Heßler, K. and Irnich, S. (2023). Partial dominance in branch-price-and-cut for the basic multicompartment vehicle-routing problem. *INFORMS Journal on Computing*, **35**(1), 50–65.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer.

Irnich, S., Toth, P., and Vigo, D. (2014). The family of vehicle routing problems. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, chapter 1, pages 1–33. Society for Industrial & Applied Mathematics (SIAM).

ITA (2023). Unmanned aircraft systems (UAS). International Trade Administration (ITA) https://www.trade.gov/unmanned-aircraft-systems, Last accessed 2023-05-03.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Joerss, M., Schröder, J., Neuhaus, F., Klink, C., and Mann, F. (2016). Parcel delivery: The future of last mile. McKinsey&Company, Brochure.

Lera-Romero, G., Miranda Bront, J. J., and Soulignac, F. J. (2022). Dynamic programming for the time-dependent traveling salesman problem with time windows. *INFORMS Journal on Computing*, **34**(6), 3292–3308.

Li, H. and Wang, F. (2022). Branch-price-and-cut for the truck–drone routing problem with time windows. *Naval Research Logistics (NRL)*, **70**(2), 184–204.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., and Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, **120**, 102762.

Madani, B. and Ndiaye, M. (2022). Hybrid truck-drone delivery systems: A systematic literature review. *IEEE Access*, **10**, 92854–92878.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Moshref-Javadi, M. and Winkenbach, M. (2021). Applications and research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, **177**, 114854.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, **72**(4), 411–458.

Poikonen, S., Wang, X., and Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, **70**(1), 34–43.

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

Roberti, R. and Ruthmair, M. (2021). Exact methods for the traveling salesman problem with drone. *Transportation Science*, **55**(2), 315–335.

Tamke, F. and Buscher, U. (2021). A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological*, **144**, 174–203.

Tilk, C. and Irnich, S. (2017). Dynamic programming for the minimum tour duration problem. *Transportation Science*, **51**(2), 549–565.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, **11**(4), 679–697.

Zhen, L., Gao, J., Tan, Z., Wang, S., and Baldacci, R. (2023). Branch-price-and-cut for trucks and drones cooperative delivery. *IISE Transactions*, **55**(3), 271–287.

Zhou, H., Qin, H., Cheng, C., and Rousseau, L.-M. (2022). A branch-and-price algorithm for the vehicle routing problem with drones. Optimization online. https://optimization-online.org/?s=8801.

Appendix A. Detailed Results per Instance

Tables A.6–A.13 present detailed computational results for the complete VRP-D instance set. Tables A.6–A.9 report the results for routing-cost minimization and Tables A.10–A.13 for the duration minimization.

All tables are structured according to the following scheme: The first two columns state the instance name (*Instance*) and the number of truck-only customers (#To) considered in the instances. Next, blocks with three columns each show the following values for $\beta = 1, 3$, and 5: The computed upper bound (UB), lower bound (UB), and the solution time in seconds (Time). Cells filled with 'TL' indicate that the time limit of 3,600 seconds for the BPC algorithm was reached. Cells filled with '—' indicate that the root node could not be solved within the time limit so that no value for LB is known.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n32-k5-20	4	6,700	6,700	1.1	6,618	6,618	2.5	6,398	6,398	1.4
A-n33-k5-20	3	5,900	5,900	0.7	5,597	5,597	0.5	5,266	5,266	3.8
A-n33-k6-20	5	5,980	5,980	34.0	5,292	5,292	1.3	4,816	4,816	55.2
A-n34-k5-20	5	6,200	6,200	8.0	5,596	5,596	10.0	5,185	5,185	47.8
A-n36-k5-20	3	6,560	6,560	12.5	5,937	5,937	355.1	4,852	4,852	247.6
A-n37-k5-20	4	6,120	6,120	82.6	4,947	4,947	2.9	4,576	4,576	6.1
A-n37-k6-20	3	6,557	6,557	34.5	5,808	5,808	52.0	5,075	5,075	32.7
A-n38-k5-20	2	5,880	5,880	2.2	5,266	5,266	8.7	4,813	4,813	131.3
A-n39-k5-20	5	5,040	5,040	24.4	4,336	4,336	13.0	3,983	3,983	43.3
A-n39-k6-20	6	7,040	7,040	5.6	6,169	6,169	14.5	5,580	5,580	24.4
A-n44-k7-20	2	6,000	6,000	0.8	5,565	5,565	3.8	4,796	4,796	2.2
A-n45-k6-20	3	6,780	6,779	168.4	5,937	5,937	57.3	5,220	5,220	25.0
A-n45-k7-20	3	6,840	6,839	132.5	5,769	5,769	4.4	5,154	5,154	44.1
A-n46-k7-20	3	6,280	6,280	2.6	5,326	5,326	137.4	4,850	4,845	*TL*
A-n48-k7-20	6	7,400	7,399	70.9	6,760	6,760	7.3	6,162	6,162	19.4
A-n53-k7-20	5	6,500	6,500	1.9	6,285	6,285	28.4	5,762	5,761	203.6
A-n54-k7-20	3	6,280	6,280	12.7	5,770	5,769	115.7	5,168	5,168	109.5
A-n55-k9-20	4	5,940	5,940	2.8	5,186	5,186	4.9	4,479	4,479	2.4
A-n60-k9-20	3	5,840	5,840	3.3	5,522	5,522	17.3	4,863	4,862	100.4
A-n61-k9-20	4	5,040	5,040	29.3	4,264	4,264	1.7	3,847	3,847	1.6

Table A.6: Detailed results for instances with $m = 19$ customers for cost minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n32-k5-30	6	9,540	9,540	226.7	9,124	9,124	930.0	8,343	8,343	37.6
A-n33-k5-30	4	7,680	7,680	245.4	6,961	6,960	651.9	6,377	6,377	472.1
A-n33-k6-30	10	8,880	8,880	68.2	8,423	8,423	355.6	7,690	7,689	1,661.9
A-n34-k5-30	7	8,480	8,480	41.7	7,908	7,908	146.9	7,479	7,479	556.7
A-n36-k5-30	3	8,438	8,438	385.6	7,310	7,310	677.0	6,295	6,295	795.9
A-n37-k5-30	5	7,116	7,116	373.3	6,350	6,350	662.7	5,656	5,656	500.9
A-n37-k6-30	3	9,220	9,220	158.7	8,141	8,141	207.4	7,287	7,286	985.7
A-n38-k5-30	4	7,920	7,920	67.7	7,140	7,139	524.6	6,295	6,295	260.0
A-n39-k5-30	7	7,920	7,920	16.8	7,236	7,235	2,147.7	6,335	6,335	185.9
A-n39-k6-30	8	8,560	8,560	46.2	7,615	7,615	222.2	6,855	6,855	192.5
A-n44-k7-30	4	8,560	8,560	73.2	7,287	7,287	92.8	6,572	6,572	1,468.2
A-n45-k6-30	7	8,380	8,380	36.7	7,903	7,903	26.8	7,233	7,233	727.8
A-n45-k7-30	6	9,720	9,720	83.6	9,128	9,128	326.8	8,397	8,396	82.4
A-n46-k7-30	5	8,320	8,320	52.4	7,589	7,589	871.9	6,740	6,740	2,067.8
A-n48-k7-30	6	9,600	9,600	15.1	9,053	9,038	*TL*	8,107	8,107	247.9
A-n53-k7-30	6	8,060	8,060	301.9	7,691	7,626	*TL*	6,852	6,829	*TL*
A-n54-k7-30	5	9,200	9,200	990.1	8,803	8,760	*TL*	7,993	7,993	281.7
A-n55-k9-30	8	8,320	8,320	176.2	7,286	7,286	14.5	6,783	6,783	1,522.9
A-n60-k9-30	5	7,960	7,960	171.9	7,480	7,480	2,084.4	6,755	6,705	*TL*
A-n61-k9-30	5	6,760	6,760	134.0	6,134	6,134	187.6	5,370	5,370	64.2

Table A.7: Detailed results for instances with $m = 29$ customers for cost minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n44-k7-40	7	10,780	10,718	*TL*	9,800	9,800	1,330.8	8,992	8,979	*TL*
A-n45-k6-40	9	10,940	10,940	917.8	10,006	10,006	111.5	9,299	9,297	*TL*
A-n45-k7-40	7	12,080	12,010	*TL*	11,311	11,310	*TL*	10,235	10,235	195.4
A-n46-k7-40	6	10,000	10,000	2,157.7	9,054	9,054	772.5	8,287	8,162	*TL*
A-n48-k7-40	8	11,640	11,640	141.6	10,809	10,809	371.6	10,009	10,008	1,260.7
A-n53-k7-40	9	9,800	9,800	1,379.2	9,285	9,271	*TL*	8,954	8,703	*TL*
A-n54-k7-40	5	10,440	10,440	961.4	10,135	10,086	*TL*	9,418	9,288	*TL*
A-n55-k9-40	11	10,260	10,224	*TL*	9,760	9,744	*TL*	9,129	9,128	1,532.0
A-n60-k9-40	5	10,040	10,039	965.8	9,318	9,182	*TL*	8,537	8,270	*TL*
A-n61-k9-40	6	9,360	9,358	*TL*	8,394	8,394	654.9	7,558	7,503	*TL*
A-n62-k8-40	4	10,340	10,298	*TL*	9,501	9,501	*TL*	8,519	8,439	*TL*
A-n63-k9-40	9	13,720	13,720	1,003.8	12,882	12,864	*TL*	11,824	11,794	*TL*
A-n63-k10-40	11	11,680	11,680	540.9	11,331	11,330	1,302.2	10,531	10,530	1,666.1
A-n64-k9-40	10	11,656	11,395	*TL*	10,832	10,795	*TL*	10,047	9,973	*TL*
A-n65-k9-40	8	10,040	9,872	*TL*	9,553	9,553	555.7	9,051	9,051	596.3
A-n69-k9-40	5	9,320	9,187	*TL*	7,962	7,962	610.8	7,116	7,113	*TL*
A-n80-k10-40	10	13,900	13,777	*TL*	12,939	12,939	1,511.2	12,098	12,098	473.4
B-n41-k6-40	7	10,220	10,220	750.4	9,846	9,738	*TL*	9,241	9,140	*TL*
B-n43-k6-40	8	8,787	8,174	*TL*	7,853	7,853	1,331.0	7,509	7,509	182.7
B-n44-k7-40	8	10,500	10,481	*TL*	10,280	10,224	*TL*	9,796	9,793	*TL*

Table A.8: Detailed results for instances with $m = 39$ customers for cost minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n53-k7-50	13	12,240	12,240	1,681.4	11,996	11,674	*TL*	11,012	10,847	*TL*
A-n54-k7-50	7	13,480	13,146	*TL*	12,734	12,534	*TL*	11,657	11,616	*TL*
A-n55-k9-50	13	12,700	12,700	946.8	12,073	12,073	993.8	11,502	11,502	1,505.0
A-n60-k9-50	9	13,880	13,612	*TL*	12,891	12,621	*TL*	11,862	11,741	*TL*
A-n61-k9-50	6	10,880	10,876	*TL*	9,992	9,992	840.1	9,096	9,020	*TL*
A-n62-k8-50	9	13,360	13,190	*TL*	13,003	12,591	*TL*	12,395	11,581	*TL*
A-n63-k9-50	12	16,240	16,240	370.0	15,747	15,651	*TL*	15,043	14,627	*TL*
A-n63-k10-50	13	13,940	13,840	*TL*	13,146	12,977	*TL*	12,453	12,245	*TL*
A-n64-k9-50	10	13,720	13,134	*TL*	12,729	12,400	*TL*	12,218	11,525	*TL*
A-n65-k9-50	10	11,640	11,639	728.9	11,297	11,226	*TL*	10,799	10,615	*TL*
A-n69-k9-50	6	10,600	10,600	737.5	10,030	9,764	*TL*	8,895	8,716	*TL*
A-n80-k10-50	12	15,600	15,315	*TL*	14,982	14,576	*TL*	13,737	13,681	*TL*
B-n50-k7-50	9	9,608	9,345	*TL*	8,626	8,626	367.3	8,304	8,304	482.2
B-n50-k8-50	13	17,200	16,153	*TL*	15,777	15,685	*TL*	15,046	14,994	*TL*
B-n51-k7-50	10	13,220	12,935	*TL*	12,556	12,120	*TL*	12,025	11,762	*TL*
B-n52-k7-50	9	7,980	7,980	3,342.8	7,866	7,849	*TL*	7,920	7,594	*TL*
B-n56-k7-50	7	7,734	7,613	*TL*	8,190	7,278	*TL*	7,727	6,886	*TL*
B-n57-k7-50	9	13,154	13,018	*TL*	12,506	12,115	*TL*	12,651	11,551	*TL*
B-n57-k9-50	10	19,220	17,932	*TL*	19,354	17,256	*TL*	18,509	16,419	*TL*
B-n63-k10-50	12	16,072	14,997	*TL*	14,351	14,183	*TL*	13,436	13,321	*TL*

Table A.9: Detailed results for instances with $m = 49$ customers for cost minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n32-k5-20	4	6,042	6,042	248.5	5,139	5,139	89.5	5,100	5,100	1.4
A-n33-k5-20	3	4,932	4,932	23.9	4,320	4,319	1,215.3	4,120	4,119	3.8
A-n33-k6-20	5	4,672	4,672	35.1	3,544	3,544	106.7	3,480	3,480	55.2
A-n34-k5-20	5	5,017	5,017	346.8	4,430	4,410	*TL*	4,400	4,367	47.8
A-n36-k5-20	3	5,240	5,240	2,932.3	3,653	3,653	1,758.7	3,077	3,077	247.6
A-n37-k5-20	4	4,584	4,584	3,013.8	3,836	3,836	1,707.8	3,742	3,742	6.1
A-n37-k6-20	3	5,380	5,380	618.1	4,089	4,088	150.3	3,848	3,848	32.7
A-n38-k5-20	2	4,560	4,560	23.7	3,295	3,295	55.9	2,942	2,942	131.3
A-n39-k5-20	5	3,945	3,944	398.5	3,240	3,240	17.7	3,208	3,207	43.3
A-n39-k6-20	6	5,630	5,630	604.7	4,627	4,626	252.9	4,218	4,218	24.4
A-n44-k7-20	2	5,017	5,017	85.5	3,512	3,512	9.0	3,480	3,480	2.2
A-n45-k6-20	3	5,219	5,218	1,535.3	3,967	3,967	112.0	3,866	3,866	25.0
A-n45-k7-20	3	5,254	5,254	497.3	3,840	3,840	89.4	3,560	3,559	44.1
A-n46-k7-20	3	4,842	4,779	*TL*	3,535	3,535	68.4	3,520	3,519	*TL*
A-n48-k7-20	6	5,994	5,994	62.5	4,689	4,689	209.2	3,934	3,934	19.4
A-n53-k7-20	5	5,757	5,621	*TL*	4,309	4,308	339.0	4,100	4,100	203.6
A-n54-k7-20	3	5,558	5,235	*TL*	4,080	4,079	558.8	3,851	3,850	109.5
A-n55-k9-20	4	4,615	4,614	227.7	3,280	3,280	10.8	3,212	3,212	2.4
A-n60-k9-20	3	5,173	5,128	*TL*	3,470	3,470	159.9	3,082	3,082	100.4
A-n61-k9-20	4	3,869	3,868	565.6	3,080	3,080	5.7	3,080	3,080	1.6

Table A.10: Detailed results for instances with $m = 19$ customers for duration minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n32-k5-30	6	8,163	7,974	*TL*	6,462	6,446	*TL*	6,215	6,203	37.6
A-n33-k5-30	4	6,390	6,251	*TL*	5,079	5,052	*TL*	4,723	4,723	472.1
A-n33-k6-30	10	7,321	7,321	1,400.6	6,028	6,028	838.4	5,866	5,866	1,661.9
A-n34-k5-30	7	7,105	7,087	*TL*	6,640	6,452	*TL*	6,600	6,242	556.7
A-n36-k5-30	3	6,759	6,539	*TL*	4,612	4,599	*TL*	4,380	4,091	795.9
A-n37-k5-30	5	5,957	5,597	*TL*	4,598	4,438	*TL*	4,378	4,214	500.9
A-n37-k6-30	3	7,627	7,382	*TL*	5,357	5,356	1,645.2	4,932	4,896	985.7
A-n38-k5-30	4	6,400	6,286	*TL*	4,819	4,819	1,411.3	4,434	4,434	260.0
A-n39-k5-30	7	6,401	6,395	*TL*	5,493	5,338	*TL*	5,086	5,017	185.9
A-n39-k6-30	8	6,952	6,787	*TL*	5,507	5,457	*TL*	5,276	5,073	192.5
A-n44-k7-30	4	6,982	6,643	*TL*	4,993	4,874	*TL*	4,969	4,709	1,468.2
A-n45-k6-30	7	7,287	7,033	*TL*	5,439	5,398	*TL*	5,202	5,176	727.8
A-n45-k7-30	6	8,440	8,217	*TL*	6,814	6,796	*TL*	6,461	6,414	82.4
A-n46-k7-30	5	7,168	6,574	*TL*	4,956	4,954	*TL*	4,880	4,684	2,067.8
A-n48-k7-30	6	8,011	7,872	*TL*	6,565	6,250	*TL*	5,730	5,611	247.9
A-n53-k7-30	6	7,424	6,443	*TL*	5,193	5,089	*TL*	5,180	4,878	*TL*
A-n54-k7-30	5	7,708	7,662	*TL*	6,578	6,578	581.2	6,140	6,140	281.7
A-n55-k9-30	8	6,411	6,410	658.8	5,300	5,287	*TL*	5,080	5,054	1,522.9
A-n60-k9-30	5	6,654	6,601	*TL*	4,986	4,849	*TL*	4,522	4,482	*TL*
A-n61-k9-30	5	5,437	5,377	*TL*	4,071	4,070	1,127.2	3,805	3,709	64.2

Table A.11: Detailed results for instances with $m = 29$ customers for duration minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n44-k7-40	7	8,783	8,714	*TL*	6,817	6,727	*TL*	6,440	6,319	*TL*
A-n45-k6-40	9	9,340	9,095	*TL*	6,843	6,820	*TL*	6,488	6,463	*TL*
A-n45-k7-40	7	10,432	10,241	*TL*	8,617	8,448	*TL*	7,611	7,605	195.4
A-n46-k7-40	6	8,446	8,003	*TL*	6,107	5,956	*TL*	5,667	5,570	*TL*
A-n48-k7-40	8	9,903	—	*TL*	7,849	7,733	*TL*	7,261	7,050	1,260.7
A-n53-k7-40	9	9,240	—	*TL*	6,485	6,457	*TL*	6,320	6,236	*TL*
A-n54-k7-40	5	9,843	8,920	*TL*	7,748	7,396	*TL*	6,923	6,792	*TL*
A-n55-k9-40	11	8,808	8,521	*TL*	7,379	7,320	*TL*	7,040	7,021	1,532.0
A-n60-k9-40	5	8,800	—	*TL*	6,049	5,936	*TL*	5,362	5,303	*TL*
A-n61-k9-40	6	7,912	7,382	*TL*	5,726	5,678	*TL*	5,271	5,255	*TL*
A-n62-k8-40	4	9,020	—	*TL*	—	6,378	*TL*	—	5,715	*TL*
A-n63-k9-40	9	11,931	11,565	*TL*	10,067	9,573	*TL*	9,122	8,791	*TL*
A-n63-k10-40	11	10,654	10,110	*TL*	8,766	8,724	*TL*	8,433	8,361	1,666.1
A-n64-k9-40	10	10,270	9,610	*TL*	7,929	7,736	*TL*	7,595	7,435	*TL*
A-n65-k9-40	8	8,736	8,401	*TL*	7,372	7,316	*TL*	7,240	7,159	596.3
A-n69-k9-40	5	7,566	—	*TL*	5,426	5,167	*TL*	5,207	4,764	*TL*
A-n80-k10-40	10	11,965	—	*TL*	9,665	9,484	*TL*	9,628	9,285	473.4
B-n41-k6-40	7	9,407	8,861	*TL*	7,696	7,639	*TL*	6,724	6,724	*TL*
B-n43-k6-40	8	8,021	7,397	*TL*	6,932	6,840	*TL*	6,609	6,547	182.7
B-n44-k7-40	8	10,480	9,451	*TL*	8,367	8,323	*TL*	7,941	7,855	*TL*

Table A.12: Detailed results for instances with $m = 39$ customers for duration minimization.

		$\beta = 1$			$\beta = 3$			$\beta = 5$		
Instance	#To	UB	LB	Time	UB	LB	Time	UB	LB	Time
A-n53-k7-50	13	10,950	—	*TL*	9,001	8,874	*TL*	8,866	8,573	*TL*
A-n54-k7-50	7	11,472	—	*TL*	9,481	9,152	*TL*	8,596	8,440	*TL*
A-n55-k9-50	13	10,829	10,674	*TL*	9,598	9,489	*TL*	9,179	9,124	1,505.0
A-n60-k9-50	9	12,178	—	*TL*	9,474	9,386	*TL*	9,009	8,721	*TL*
A-n61-k9-50	6	9,483	—	*TL*	6,999	6,874	*TL*	6,353	6,223	*TL*
A-n62-k8-50	9	12,168	—	*TL*	9,518	8,870	*TL*	—	8,167	*TL*
A-n63-k9-50	12	14,812	—	*TL*	12,181	11,798	*TL*	11,306	11,003	*TL*
A-n63-k10-50	13	12,568	11,765	*TL*	10,517	10,231	*TL*	10,123	9,768	*TL*
A-n64-k9-50	10	11,715	—	*TL*	9,182	8,772	*TL*	8,628	8,348	*TL*
A-n65-k9-50	10	10,403	9,903	*TL*	9,032	8,514	*TL*	8,331	8,255	*TL*
A-n69-k9-50	6	9,458	—	*TL*	6,631	6,475	*TL*	6,359	6,116	*TL*
A-n80-k10-50	12	13,871	—	*TL*	11,693	10,718	*TL*	—	10,178	*TL*
B-n50-k7-50	9	8,240	—	*TL*	7,326	—	*TL*	7,292	—	482.2
B-n50-k8-50	13	15,523	—	*TL*	13,765	13,214	*TL*	13,467	12,545	*TL*
B-n51-k7-50	10	12,260	—	*TL*	10,581	9,915	*TL*	10,461	9,795	*TL*
B-n52-k7-50	9	7,563	—	*TL*	—	—	*TL*	6,042	—	*TL*
B-n56-k7-50	7	7,659	—	*TL*	5,867	5,459	*TL*	4,943	4,795	*TL*
B-n57-k7-50	9	13,100	—	*TL*	10,604	10,228	*TL*	—	9,955	*TL*
B-n57-k9-50	10	18,993	—	*TL*	13,910	13,589	*TL*	13,533	13,324	*TL*
B-n63-k10-50	12	13,911	—	*TL*	11,478	10,927	*TL*	11,091	10,352	*TL*

Table A.13: Detailed results for instances with $m = 49$ customers for duration minimization.