The Last-mile Vehicle Routing Problem with Delivery Options

Christian Tilk[a,*], Katharina Olkis[a], 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*

Abstract

The ongoing rise in e-commerce comes along with an increasing number of first-time delivery failures due to the absence of the customer at the delivery location. Failed deliveries result in rework which in turn has a large impact on the carriers' delivery cost. In the classical vehicle routing problem (VRP) with time windows, each customer request has only one location and one time window describing where and when shipments need to be delivered. In contrast, we introduce and analyze the vehicle routing problem with delivery options (VRPDO), in which some requests can be shipped to alternative locations with possibly different time windows. Furthermore, customers may prefer some delivery options. The carrier must then select, for each request, one delivery option such that the carriers' overall cost is minimized and a given service level regarding customer preferences is achieved. Moreover, when delivery options share a common location, e.g., a locker, capacities must be respected when assigning shipments. The VRPDO generalizes several known extensions of the VRP with time windows, e.g., the generalized VRP with time windows, the multi-vehicle traveling purchaser problem, and the VRP with roaming delivery locations. To solve the VRPDO exactly, we present a new branch-price-and-cut algorithm. The associated pricing subproblem is a shortest-path problem with resource constraints that we solve with a bidirectional labeling algorithm on an auxiliary network. We focus on the comparison of two alternative modeling approaches for the auxiliary network and present optimal solutions for instances with up to 100 delivery options. Moreover, we provide 17 new optimal solutions for the benchmark set for the VRP with roaming delivery locations.

Keywords: routing, vehicle routing, city logistics, branch-price-and-cut, service level, capacitated location

1. Introduction

Over the past decade, mail-order trade has shown a strong compound annual growth rate, e.g., of 9.6 % in Germany with an overall revenue of 68.1 billion euros in 2018 (Furchheim *et al.*, 2020). Due to the increasing e-commerce, over 3.5 billion deliveries had to be handled in Germany in 2018 resulting in 12 million deliveries per operations day on average (BIEK, 2019). These considerable numbers raise the question of how to cope with the strongly growing demand and general challenges in last-mile logistics. While minimizing logistic costs, operators are also faced with issues such as the trend of smaller and smaller truckloads, restrictions imposed by urban development and environmental policies, and operational issues like traffic congestion and strict parking regulation (Zhou *et al.*, 2018). Furthermore, new challenges arise from *logsumers* (DHL, 2014) who are allowed to individualize their orders by choosing a preferred price, quality, time window, and options for environmental friendliness. Consequently, last-mile logistic has been referred to as 'the bottleneck of e-commerce' (Wang *et al.*, 2014) and 'the logistic service providers' nightmare(s)' (Savelsbergh and Van Woensel, 2016).

The paper at hand introduces the *vehicle routing problem with delivery options* (VRPDO) which captures one of the most recent trends in last-mile package delivery related to the introduction of delivery options.

*Corresponding author.

Email addresses: tilk@uni-mainz.de (Christian Tilk), kaolkis@uni-mainz.de (Katharina Olkis), irnich@uni-mainz.de (Stefan Irnich)

The VRPDO is obviously a generalization of the *vehicle routing problem with time windows* (VRPTW, Costa *et al.*, 2019) and the *generalized vehicle routing problem with time windows* (GVRPTW, Moccia *et al.*, 2012) in which each customer request is represented by one or several *delivery options*. The delivery options of a customer differ within the designated location and delivery time window. Exactly one delivery option of each customer has to be selected.

The VRPDO extends the GVRPTW by two important real-world aspects: First, customers can individually prioritize their different delivery options beforehand, and the overall customer satisfaction level is taken into account by a given service level that must be achieved. Second, some delivery options may share a common location, e.g., pick-up points or postal boxes (Janjevic *et al.*, 2019). The capacity of these locations is limited, in particular in densely populated areas of cities where space is scarce and expensive. For finding an optimal set of routes, both extensions lead to a non-trivial *interdependence problem*, where modifying one route can make another route infeasible regarding location capacities or required service level (Drexl, 2012). The objective of the VRPDO is to minimize the overall cost while ensuring a minimum customer satisfaction level as well as not violating location capacity restrictions.

The VRPDO also generalizes some other vehicle routing problems. In the VRP with multiple time windows (Doerner *et al.*, 2008), the delivery options of a customer have an identical location. The multiple-vehicle traveling purchaser problem (Manerba *et al.*, 2017) can be modeled by considering products as customers/requests, the suppliers of a certain product as the delivery options, and the visited purchasers as the routes performed by different vehicles. Moreover, to model the multi-vehicle covering tour problem (Hachicha *et al.*, 2000), one can introduce a delivery option for each customer and each service point covering that customer. Last, the *vehicle routing problem with (home and) roaming delivery locations* (VRP(H)RDL, Ozbaygin *et al.*, 2017) is an application-specific variant of the GVRPTW and therefore also a special case of the VRPDO. Note that none of these problems include location capacities or service-level constraints.

The two main contributions of this work are the following: We introduce the VRPDO and present a *branch-price-and-cut* (BPC, Costa *et al.*, 2019) algorithm for its solution. In particular, we present a set-partitioning problem for the VRPDO, develop and analyze two different network structures for the solution of the pricing problem, and adapt cutting planes and branching rules. The second contribution is an extensive computational study with three parts: First, we evaluate the performance of our algorithm for the two different network structures on a newly introduced benchmark instance set. Second, we compare our BPC with the state-of-the-art algorithm from the literature on benchmark instances for the VRPHRDL and VRPRDL. Last, we conduct a sensitivity analysis to determine the impact of varying service levels and location capacities on routing costs and solution times.

The remainder of the paper is structured as follows: Section 2 introduces the VRPDO and all necessary notation. Section 3 describes the BPC algorithm with subsections on the set partitioning formulation, the pricing subproblem, valid inequalities, and branching. In Section 4, the results of the computational study are shown and discussed. Final conclusions are drawn in Section 5.

2. Definition of the VRPDO

The VRPDO can be defined as follows: Let N denote the set of all *(customer) requests*, L the set of all *locations*, and $P = \{1, \dots, p^{\max}\}$ the set of $p^{\max} \geq 1$ different *(delivery) priorities*. A *delivery option* is a triplet $o = (n, \ell, p) \in N \times L \times P$ and let $O \subset N \times L \times P$ denote the set of all delivery options. In order to identify for an option $o \in O$ the associated request, location, and priority, we write $o = (n_o, \ell_o, p_o)$. A request $n \in N$ is fulfilled by selecting exactly one of the options $O_n^N = \{(n_o, \ell_o, p_o) \in O : n_o = n\}$. For a feasible VRPDO instance, there must exist at least one option per request, i.e., $|O_n^N| \geq 1$ for all $n \in N$. Similarly, we define $O_\ell^L = \{(n_o, \ell_o, p_o) \in O : \ell_o = \ell\}$ as the set of options belonging to location $\ell \in L$ and $O_p^P = \{(n_o, \ell_o, p_o) \in O : p_o \leq p\}$ as the set of options with priority level $p \in P$ or smaller (=better). Moreover, each option $o \in O$ has a non-negative service time s_o.

For each location $\ell \in L$, a time window $[a_\ell, b_\ell]$ represents the time period in which all deliveries to this location must take place. Additionally, the capacity C_ℓ limits the number of shipments that can be delivered to that location. A location with more than one delivery option is called *multiple-delivery location*, all other

locations are called *single-delivery locations*. $L^m = \{\ell \in L : |O^L_\ell| > 1\}$ denotes the set of all multiple-delivery locations.

A fleet of K homogeneous vehicles is housed at the depot 0 for which $\ell_0 \in L$ denotes the depot location. A vehicle has a *capacity* Q to serve the demands q_n of the requests $n \in N$ and *fixed cost* c^f when used. We assume the time window $[a_{\ell_0}, b_{\ell_0}]$ of the depot location to span the whole planning horizon. Traveling between location ℓ and $\ell' \in L$ consumes a *travel time* of $t_{\ell\ell'}$ and a *travel cost* of $c_{\ell\ell'}$. The travel time $t_{\ell\ell'}$ also includes a required access time needed, e.g., for parking a vehicle at ℓ' before the actual service (=delivery) at ℓ' can start.

Each priority level $p \in P$ can be characterized by a percentage β_p that indicates that at least $\lceil \beta_p |N| \rceil$ delivery requests must be served with priority level p or smaller (=better). For this purpose, we assume that priorities are nested and the set O^P_p contains all options o with $p_o \leq p$. Note that the percentage $\beta_{p^{\max}}$ refers to the set $O^P_{p^{\max}} = O$ and is therefore irrelevant.

A *(vehicle) route* $r = (0, o_1, \ldots, o_h, 0')$ is as sequence of options in which the artificial options $o_0 = 0$ and $o_{h+1} = 0'$ represent the visit of the depot location ℓ_0 at the start and end of the route, respectively. A route r is *feasible* if it fulfils the capacity and time-window constraints that we define as follows. The demand served by route r is $q(r) = \sum_{j=1}^{h} q_{n_{o_j}}$, so that r is capacity-feasible if $q(r) \leq Q$ holds true. A route is time-window feasible if there exists a schedule $(T_0, T_1, \ldots, T_h, T_{h+1}) \in \mathbb{R}^{h+2}$ which complies with the option service times, travel times, and time windows, i.e., if $T_{j-1} + t_{\ell_{o_{j-1}}, \ell_{o_j}} + s_{o_{j-1}} \leq T_j$ for all $1 \leq j \leq h+1$ (assuming $s_{o_0} = 0$) and $T_j \in [a_{\ell_{o_j}}, b_{\ell_{o_j}}]$ for all $0 \leq j \leq h+1$. This definition has the following consequences: a subsequence $o_i, \ldots, o_j$ of options with identical locations $\ell = \ell_{o_i} = \cdots = \ell_{o_j}$ models a single physical stop of a vehicle at this location. The above time-window feasibility conditions impose that (i) $T_i, \ldots, T_j$ can be considered as the start times when the associated requests are served, (ii) $T_i + s_{o_i}, \ldots, T_j + s_{o_j}$ are the respective service end times, and (iii) the time window $[a_\ell, b_\ell]$ of location ℓ must cover all service times entirely. We stress that we have chosen this definition of the time windows (diverging from the standard definition for the VRPTW referring to possible service start times) because in the VRPDO the total service time at a location is a variable. It results from the selection of options that are together served during the one stop of a vehicle at the location.

The cost c_r of a route r is the sum of the fixed cost and the travel costs between the visited locations, i.e., $c_r = c^f + \sum_{j=1}^{h+1} c_{\ell_{o_{j-1}}, \ell_{o_j}}$. The objective of the VRPDO is to find a least-cost set of feasible routes together covering exactly one option of O^N_n for all customers $n \in N$ respecting the fleet size and location capacities as well as achieving the required service level.

3. Branch-Price-and-Cut

We use a BPC algorithm in order to solve the VRPDO exactly. According to the recent survey by Costa *et al.* (2019), BPC is the leading exact methodology for solving many types of vehicle routing problems. Section 3.1 presents the extensive formulation of the VRPDO and Section 3.2 the pricing subproblem and approaches for its resolution. In particular, we describe two competing approaches for modeling the underlying network including a discussion of expected pros and cons and a summary of their properties. Section 3.3 then briefly describes known valid inequalities for strengthening the linear relaxation, their adaption for the VRPDO, and separation algorithms. The branching scheme developed in Section 3.4 finally ensure integer solutions.

3.1. Extensive Formulation

Let Ω denote the set of all feasible routes. The following extensive formulation of the VRPDO comprises one binary variables λ_r per possible route $r \in \Omega$ indicating whether r is part of the solution ($\lambda_r = 1$) or not (=0). The binary coefficients α_{or} indicate whether route $r \in \Omega$ serves option $o \in O$. The model is an

extended set-partitioning formulation and reads as follows:

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

$$\text{subject to} \quad \sum_{r\in\Omega}\sum_{o\in O_n^N} \alpha_{or}\lambda_r = 1 \qquad \forall n \in N \tag{1b}$$

$$\sum_{r\in\Omega}\sum_{o\in O_\ell^L} \alpha_{or}\lambda_r \le C_\ell \qquad \forall \ell \in L^m \tag{1c}$$

$$\sum_{r\in\Omega}\sum_{o\in O_p^P} \alpha_{or}\lambda_r \ge \lceil \beta_p |N| \rceil \qquad \forall p \in P \tag{1d}$$

$$\sum_{r\in\Omega} \lambda_r \le K \tag{1e}$$

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

The objective (1a) minimizes the overall costs of the routes that are selected and performed. Constraints (1b) ensure that each request is served exactly once. For each multiple-delivery location, the number of shipments to this locations is bounded by constraints (1c). The required service level per priority level is enforced by constraints (1d). Finally, constraint (1e) restricts the number of employed vehicles.

For solving the linear relaxation of (1), a column-generation algorithm is used (Desaulniers *et al.*, 2005). In the following, the linear relaxation of formulation (1) in which the set of all feasible routes Ω is replaced by a subset $\bar{\Omega} \subset \Omega$ is denoted as *restricted master program* (RMP). Column generation alternates between the reoptimization of the RMP and the solution of the pricing subproblem that either generates new negative reduced-cost variables (=routes) to be added to $\bar{\Omega}$, or proves that none exist. In the latter case, the column-generation process terminates with a solution to the linear relaxation of the extensive formulation.

3.2. Column Generation

Let $(\pi_n)_{n\in N}$ denote the dual prices of the constraints (1b), $(\rho_\ell)_{\ell\in L^m}$ of the constraints (1c), $(\nu_p)_{p\in P}$ of constraints (1d), and μ of the fleet-size constraint (1e). Then, the reduced cost $\bar{c}_r$ of a route $r \in \Omega$ is given by

$$\bar{c}_r = c_r - \mu - \sum_{n\in N}\sum_{o\in O_n^N} \alpha_{or}\pi_n - \sum_{\ell\in L^m}\sum_{o\in O_\ell^L} \alpha_{or}\rho_\ell - \sum_{p\in P}\sum_{o\in O_p^P} \alpha_{or}\nu_p. \tag{2}$$

The pricing problem asks for a feasible route with minimal reduced cost. We show in the following that it is a variant of the *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). SPPRCs can be solved with a dynamic-programming labeling algorithm that creates partial paths starting from an origin vertex moving forward to a destination vertex of the network.

Next, we present two different possibilities to model the underlying network. A unified labeling algorithm for both networks is described in Section 3.2.2. Standard acceleration techniques are presented in Section 3.2.3.

3.2.1. Network Modeling

An important characteristic of the VRPDO is that in many realistic instances several options share the same physical delivery location. This happens, e.g., when different customers choose the same delivery locker or the same shop as a potential option. Moreover, customers living together in the same apartment building and allowing home delivery is another case of identical locations.

One of our leading questions was whether these identical locations can be exploited so that a tailored solution approach works better than one that does not anticipate the identical locations. The *commodity-constrained split delivery vehicle routing problem* (C-SDVRP, Archetti *et al.*, 2016) is an example of a VRP in which the solution approach can be tailored to exploit identical locations. In this problem, the total demand of a customer can be split into given smaller demands (the different commodities requested) which

can be served by one or several visits to the customer. A straightforward approach for modeling and solving the C-SDVRP is to reduce it to a standard CVRP, as done in (Archetti *et al.*, 2016), where each customer vertex is duplicated as many times as the number of commodities requested by the customer. Each duplicated vertex has a demand given by the weight of the corresponding commodity. A solution approach may either disregard or exploit the fact that this new CVRP instance has customers with identical locations. The recently presented approach of Gschwind *et al.* (2019) is of the latter type and heavily benefits from working on a transformed underlying graph with a macroscopic level considering locations and a microscopic level considering the commodities of each customer separately.

In the same spirit, we present two modeling approaches for the underlying network. The first one uses an *option-based network* $G^{opt} = (V^{opt}, A^{opt})$. It is rather straightforward and only utilizes options and depots as vertices. No information about identical locations is taken into account. Formally, the vertex set V^{opt} of the option-based network is given by $\{0, 0'\} \cup O$. The arc set A^{opt} contains all feasible direct connections between vertices, i.e., $A^{opt} \subset V^{opt} \times V^{opt}$.

The second network, the *location-based network* $G^{loc} = (V^{loc}, A^{loc})$, is more sophisticated as it makes use of the location information of options. All options are grouped according to their respective location. The idea of the location-based network is to significantly reduce the number of arcs at the cost of adding two artificial vertices e_ℓ and f_ℓ for each multiple-delivery location $\ell \in L^m$ modeling the entry and exit point of that location, respectively.

Formally,

$$E = \{e_\ell : \ell \in L^m\} \qquad \text{and} \qquad F = \{f_\ell : \ell \in L^m\}$$

denote the sets of all *entry* and *exit vertices*, respectively. (Conversely, to refer to an entry or exit's location, we write $\ell_e = \ell_f = \ell$ for $\ell \in L^m$.) Then, the vertex set is given by $V^{loc} = \{0, 0'\} \cup O \cup E \cup F = V^{opt} \cup E \cup F$. Entering (exiting) a multiple-delivery location is only possible by using the entry (exit) vertex of that location. Thus, the arc set can be characterized by

$$\begin{aligned} A^{loc} \quad \subseteq \quad & \{(i,j) \in A^{opt} : \ell_i, \ell_j \notin L^m\} \\ & \cup \{(e_\ell, j) \in E \times V^{opt} : \ell_j = \ell\} \quad \cup \quad \{(i, f_\ell) \in V^{opt} \times F : \ell_i = \ell\} \\ & \cup \{(f_\ell, j) \in F \times V^{opt} : \ell_j \notin L^m, j \neq 0\} \quad \cup \quad \{(i, e_\ell) \in V^{opt} \times E : i \neq 0', \ell_i \notin L^m\} \\ & \cup \{(f_\ell, e'_\ell) \in F \times E : \ell \neq \ell'\}, \end{aligned}$$

where the first row refers direct connections between options at single-delivery locations, the second row internal connections of a multiple-delivery location, the third row connections between exit/entry vertices and single-delivery locations, and the last row connections between exit and entry vertices. For the sake of clarity, all arcs of the location-based network are summarized in Table 1. Note that typically time-window constraints and demands allow to eliminate infeasible arcs so that A^{loc} is a proper subset of the indicated arc set (this is also the reason why we write $A^{loc} \subseteq \dots$).

Table 1: Arcs in the location-based network; entry "–" if no such arc exists, entry "✓" if all such arcs exist, and a condition under which some of the arcs exist.

from ↓ to →	$0'$	$e_{\ell'} \in E$	$f_{\ell'} \in F$	$o' \in O : \ell_{o'} \notin L^m$	$o' \in O : \ell_{o'} \in L^m$
0	–	✓	–	✓	–
$e_\ell \in E$	–	–	–	–	$\ell_{o'} = \ell$
$f_\ell \in F$	✓	$\ell \neq \ell'$	–	✓	–
$o \in O : \ell_o \notin L^m$	✓	✓	–	$n_o \neq n_{o'}$	–
$o \in O : \ell_o \in L^m$	–	–	$\ell_o = \ell'$	–	$\ell_o = \ell_{o'},\ n_o \neq n_{o'}$

Example 1. *We consider the situation that a route serves six options $o_1, o_2, o_3, o_4, o_5, o_6$ in the given order. Associated with these options are four different locations ℓ_{o_1}, $\ell_{o_2} = \ell_{o_3} = \ell_{o_4}$, ℓ_{o_5}, and ℓ_{o_6}. Moreover, we assume that the second and penultimate location, i.e., $\ell = \ell_{o_2}$ and $\ell' = \ell_{o_5}$, are the only multiple-delivery locations in this example.*

In the option-based network, the resulting route is represented by the path $\mathcal{P}^{opt} = (0, o_1, o_2, o_3, o_4, o_5, o_6, 0')$. *In the location-based network, the path is* $\mathcal{P}^{loc} = (0, o_1, e_\ell, o_2, o_3, o_4, f_\ell, e_{\ell'}, o_5, f_{\ell'}, o_6, 0')$.

Table 2: Comparison of the option-based network and the location-based network.

	Option-based Network $G^{opt} = (V^{opt}, A^{opt})$	Location-based Network $G^{loc} = (V^{loc}, A^{loc})$
Vertices	less $(-2\|L^m\|)$	more $(+2\|L^m\|)$
Arcs	many more; $\mathcal{O}(\|O\|^2)$	much less; $\mathcal{O}(\|L\|^2) + \mathcal{O}(\|L^m\|\Delta^2)$
Dominance	weaker	stronger

Note: Δ denotes the size of the largest set O^L_ℓ for $\ell \in L^m$.

Table 2 compares the option-based and location-based networks regarding the sizes of the vertex and arc sets as well as regarding the strength of the dominance that is used in the dynamic-programming labeling algorithm for solving the pricing problem. The following example explains what is meant by strength of dominance:

Example 2. *We consider two feasible routes that have no options in common. Moreover, option* o_1 *is served by the first route while option* o_2 *is served by the second route. In addition, we assume that the options have the identical delivery location* $\ell = \ell_{o_1} = \ell_{o_2}$.

Then, in the option-based network, the paths $\mathcal{P}_1$ *and* $\mathcal{P}_2$ *representing the two routes are vertex disjoint, except for the two depot vertices* 0 *and* $0'$. *As a consequence, no dominance can occur between labels associated with proper partial paths that occur when constructing the two paths.*

In contrast, in the location-based network, the two paths are

$$(0, \ldots, e_\ell, \ldots, o_1, \ldots, f_\ell, \ldots, 0')$$

and

$$(0, \ldots, e_\ell, \ldots, o_2, \ldots, f_\ell, \ldots, 0'),$$

i.e., they have at least the additional entry and exit vertices is common. At these vertices e_ℓ *and* f_ℓ, *partial paths associated with the two routes are compared by the dominance algorithm.*

We can summarize that, in the location-based network, more partial paths are supposed to meet at the entry and exit vertices of their common multiple-delivery locations. Consequently, there are more possibilities that labels dominate each other so that the dominance ist stronger.

Moreover, the main driver for the computational effort of a labeling algorithm is the network size, because it impacts the number of labels that are generated and compared against each other using the dominance algorithm. Since labels are extended along arcs, also the two last points in combination with Table 2 speak in favor of the location-based network. Therefore, we initially expected that a BPC algorithm with pricing over the location-based network would clearly outperform the other BPC algorithm using the option-based network. The later computational experiments presented in Section 4.2 will however show that results are less clear cut. This came very unexpectedly for us.

3.2.2. Unified Labeling Algorithm

We now present the labeling algorithm in a unified way so that the description is correct for both networks. To this end, let $G = (V, A) \in \{G^{opt}, G^{loc}\}$ be the option-based or location-based digraph.

It is helpful to first introduce the dual price of a vertex $i \in V$ that we defined as

$$dual(i) = \begin{cases} \pi_{n_i} + \rho_{\ell_i} + \sum\limits_{p \le p_i} \nu_p, & \text{if } i = (n_i, \ell_i, p_i) \in O \\ 0, & \text{otherwise} \end{cases} + \begin{cases} \mu - c^f, & \text{if } i = 0 \text{ or } i = 0' \\ 0, & \text{otherwise} \end{cases}. \tag{3}$$

Then, the travel time and reduced cost of an arc $(i,j) \in A$ are defined as

$$\bar{t}_{ij} = t_{\ell_i,\ell_j} + \begin{cases} s_i, & \text{if } i = (n_i, \ell_i, p_i) \in O \\ 0, & \text{otherwise} \end{cases} \qquad \text{and} \qquad \bar{c}_{ij} = c_{\ell_i,\ell_j} - \frac{1}{2}\left(dual(i) + dual(j)\right), \tag{4}$$

respectively.

As in all VRPTW variants, we can remove infeasible arcs. Regarding the time window constraints, we can remove all arcs $(i,j) \in A$ with $a_{\ell_i} + \bar{t}_{ij} > b_{\ell_j}$. Moreover, for $(i,j) \in O \times O$ the arcs with $q_{n_i} + q_{n_j} > Q\}$ can be eliminated. Note that the conditions in Table 1 already ensure that $n_i \neq n_j$ holds.

Since the routing cost between two options of the same multiple-delivery location is 0, the order in which these options are served does not matter. In the following, we therefore reduce the symmetry of the network by eliminating the arcs in one direction. Let a total ordering $<$ of all options be given, i.e., either $i < j$ or $j < i$ holds for all $i, j \in O, i \neq j$. The arcs of set $(i,j) \in O \times O$ with $\ell_i = \ell_j$ and $i > j$ can be removed from A. Note that with this symmetry reduction we partially exploit location-specific information also in the option-based network.

A partial path $P_i = (0, \ldots, i)$ starts at the depot 0 and ends at some vertex $i \in V$. The associated label $L_i = (i, C_i, Q_i, T_i, S_i)$ comprises the attributes

i: the last visited vertex,
C_i: the accumulated reduced cost,
Q_i: the accumulated load,
T_i: the earliest arrival time at i, and
S_i: the set of the requests served along the partial path P_i.

For the trivial partial path (0), the initial label is given by $L_0 = (0, C_0, Q_0, T_0, S_0) = (0, 0, 0, a_{\ell_0}, \varnothing)$. Labels are propagated over arcs toward the destination vertex with the help of so-called *resource extension functions* (REFs, Irnich and Desaulniers, 2005). An arbitrary label $L_i = (i, C_i, Q_i, T_i, S_i)$ is extended along an arc $(i,j) \in A$ creating a new label (j, C_j, Q_j, T_j, S_j) using the following REFs:

$$C_j = C_i + \bar{c}_{ij} \tag{5a}$$

$$Q_j = \begin{cases} Q_i + q_{n_j}, & \text{if } j = (n_j, \ell_j, p_j) \in O \\ Q_i, & \text{otherwise} \end{cases} \tag{5b}$$

$$T_j = \max\{a_{\ell_j}, T_i + \bar{t}_{ij}\} \tag{5c}$$

$$S_j = \begin{cases} S_i \cup \{n_j\}, & \text{if } j = (n_j, \ell_j, p_j) \in O \\ S_i, & \text{otherwise} \end{cases} \tag{5d}$$

The extended partial path $P_j = (0, \ldots, i, j)$ is feasible if the following constraints are fulfilled:

$$\begin{aligned} &Q_j \leq Q \\ &T_j \leq b_j \\ &n_j \notin S_i, \text{if } j = (n_j, \ell_j, p_j) \in O \end{aligned}$$

To avoid the enumeration of all feasible paths, provable redundant labels are eliminated through a dominance criterion. Let two labels $L_i = (i, C_i, Q_i, T_i, S_i)$ and $L'_i = (i, C'_i, Q'_i, T'_i, S'_i)$ of two different partial paths ending at the same vertex $i \in V$ be given. Label L_i dominates L'_i if $C_i \leq C'_i$, $Q_i \leq Q'_i$, $T_i \leq T'_i$, and $S_i \subseteq S'_i$ holds. A dominated label can be discarded as long as a dominating label is kept.

Note that the use of a dominance rule always plays with the tradeoff between the effort resulting from less labels to be generated and further extended and the effort from applying the dominance algorithm itself. For the location-based network, we will investigate different dominance strategies varying the vertices at which the dominance is applied. Dominance can be checked at every vertex (full dominance) or only at some vertices (reduced dominance). Examples for the latter are dominance only at option vertices or dominance at all vertices but not at the option vertices of multiple-delivery locations. These three strategies are computationally evaluated in Section 4.2.

3.2.3. Acceleration of the Labeling

We apply bidirectional labeling which has become a quasi-standard for accelerating the solution of SPPRC labeling algoriithms (Righini and Salani, 2006). In a bidirectional labeling algorithm, forward and backward labels are only extended up to a half-way point that splits the domain of a monotone resource into two intervals, one for extension of forward and one for the extension of backward labeling.

Backward labels refer to backward partial paths $(j, \dots, 0')$ staring at a vertex $j \in V$ and ending at the destination depot $0'$. For the trivial backward partial path $(0')$, the initial label is $B_{0'} = (0', C_{0'}, Q_{0'}, T_{0'}, S_{0'}) = (0', 0, 0, b_{\ell_0}, \varnothing)$. Note that the time attribute in the backward case represents an as-late-as-possible schedule. Backward labels are propagated against the arc direction toward the source vertex. An arbitrary label $B_j = (j, C_j, Q_j, T_j, S_j)$ is extended backwards over an arc $(i, j) \in A$ creating a new label $B_i = (i, C_i, Q_i, T_i, S_i)$ defined by:

$$C_i = C_j + \bar{c}_{ij} \tag{6a}$$

$$Q_i = \begin{cases} Q_j + q_{n_i}, & \text{if } i = (n_i, \ell_i, p_i) \in O \\ Q_j, & \text{otherwise} \end{cases} \tag{6b}$$

$$T_i = \min\{b_{\ell_i}, T_j - \bar{t}_{ij}\} \tag{6c}$$

$$S_i = \begin{cases} S_j \cup \{n_i\}, & \text{if } i = (n_i, \ell_i, p_i) \in O \\ S_j, & \text{otherwise} \end{cases} \tag{6d}$$

The extended partial path $P_i = (i, j, \dots, 0')$ is feasible if $Q_i \le Q$, $T_i \ge a_i$, and $n_i \notin S_j$ if $i \in O$.

Let two backward labels $B_j = (j, C_j, Q_j, T_j, S_j)$ and $B'_j = (j, C'_j, Q'_j, T'_j, S'_j)$ of two different backward partial paths starting at the same vertex $j \in V$ be given. Label L_j dominates L'_j if $C_j \le C'_j$, $Q_j \le Q'_j$, $T_j \ge T'_j$, and $S_j \subseteq S'_j$ holds.

The acceleration of a bidirectional labeling approach results from the consideration of a monotone resource, the time attribute here, for which a half-way point H is defined: Forward labels with a time attribute $T > H$ as well as backward labels with a time attribute $T \le H$ are not further extended. This mitigates the often observed combinatorial explosion that happens when partial paths with many arcs are generated. We use a dynamic half-way point as defined in (Tilk *et al.*, 2017) to balance the labels generated in forward and backward labeling.

The merge step considers a forward label $L_i = (i, C_i, Q_i, T_i, S_i)$ for a partial path $(0, \dots, i)$ and a backward label $B_i = (i, C_i^{bw}, Q_i^{bw}, T_i^{bw}, S_i^{bw})$ for a backward partial path $(i, \dots, 0')$. The two labels can be merged, i.e., the concatenation $(0, \dots, i, \dots, 0')$ represents a feasible VRPDO route r if the following conditions hold:

$$Q_i + Q_i^{bw} \le \begin{cases} Q - q_{n_i}, & \text{if } i = (n_i, \ell_i, p_i) \in O \\ Q, & \text{otherwise} \end{cases}$$

$$T_i \le T_i^{bw}$$

$$S_i \cap S_i^{bw} = \begin{cases} \{n_i\}, & \text{if } i = (n_i, \ell_i, p_i) \in O \\ \varnothing, & \text{otherwise} \end{cases}$$

The conditions take into account that both (5) and (6) model the operations at the merge point i so that correction terms are needed to not double count. The reduced cost of the route r is $\bar{c}_r = C_i + C_i^{bw}$.

The bidirectional labeling takes by far the largest portion of the computation time in the overall BPC algorithm. To further speed up the solution process, two additional techniques are used. First, we adapt the *ng*-path relaxation (Baldacci *et al.*, 2011) with a more powerful dominance rule that comes at the cost of introducing non-elementary routes as solutions to the pricing subproblem leading to a weaker linear relaxation bound. We adopt the *ng*-path relaxation as follows: For each location ℓ, we define a neighborhood $N_\ell \subset N$ that contains the κ closest requests to ℓ (ties are broken arbitrarily). We define the distance of a request n to a location ℓ as $\min\{c_{\ell,\ell_o} : o \in O_n^N\}$.

Second, the labeling is solved heuristically but typically faster using a *limited discrepancy search* (Feillet *et al.*, 2007). In the heuristic labeling algorithm, the set of outgoing arcs of each vertex is partitioned in

good and *bad arcs*. Then the total number of bad arcs that can be traversed in a route is limited by some upper bound. Section 4 provides further details.

3.3. Valid Inequalities

To strengthen the linear relaxation of (1), two classes of valid inequalities for the VRPTW are adapted: *k-path inequalities* (KPIs, Kohl *et al.*, 1999) and *limited memory subset-row inequalities* (LmSRIs, Pecin *et al.*, 2017). We briefly describe them in the following.

Let $S \subseteq V$ be a subset of vertices. We say that S *contains a request* $n \in N$ if it contains all options associated with n, i.e., $O_n \subseteq S$. The KPI for S is given by

$$\sum_{r \in \Omega} \sum_{(i,j) \in \delta^-(S)} e_{ij}^r \lambda_r \geq k(S), \tag{7}$$

where $\delta^-(S)$ is the set of all in-going arcs into vertex set S, e_{ij}^r an integer coefficient counting how often arc $(i,j) \in A$ is used by route r, and $k(S)$ the minimum number of vehicles need to serve all requests contained in S. KPIs are robust, meaning they do not change the structure of the pricing problem: the dual price of the KPI for S must simply be subtracted from the reduced cost of the arcs in $\delta^-(S)$. The minimum number $k(S)$ of vehicles can be substituted by any lower bound. This lower bound can be determined either considering the demand or time-window constraints. In the first case, we separate the KPIs by applying two shrinking heuristics presented by Belenguer *et al.* (2000) and Ralphs *et al.* (2003), the extended shrinking heuristic and the greedy shrinking heuristic, respectively. In the second case, we restrict ourselves to the case $k(S) = 2$ and use the heuristic proposed by Kohl *et al.* (1999) to generate candidate sets S.

SRIs were first introduced for the VRPTW by Jepsen *et al.* (2008). Let $S \subset N$ be any subset of requests and let h_r^S be the number of times the route r serves a request in S. As proposed by Jepsen *et al.* (2008), we restrict ourselves to SRIs defined on three requests, i.e., $|S| = 3$, because violated SRIs of this type can be separated by straightforward enumeration. The corresponding SRI for S is then given by

$$\sum_{r \in \Omega} \left\lfloor \frac{h_r^S}{2} \right\rfloor \lambda_r \leq 1. \tag{8}$$

SRIs comprise a family of non-robust cuts meaning that for each SRI with a positive dual value an additional binary attribute has to be added in the labeling algorithm. Moreover, the standard dominance rule of Section 3.2.2 has to be modified to effectively cope with the additional attributes (we refer to Jepsen *et al.*, 2008, for details).

The presence of many SRIs often drastically increases the practical difficulty of the pricing problem. To alleviate these negative effects, Pecin *et al.* (2017) introduced LmSRIs that use an S-specific memory for the associated binary attribute. The role of the memory is very similar to the neighborhoods in the ng-path relaxation. With a complete memory, LmSRIs are identical to standard SRIs. However, with a S-specific memory, the difficulty of the pricing subproblem is typically reduced. We use the same separation algorithm and vertex memory as described by Pecin *et al.*.

3.4. Branching

To finally compute integer solutions for (1), branching may be necessary. We use a hierarchical four-level branching scheme and apply a best-first search to determine the next branch-and-bound node to process. Let $(\bar{\lambda}_r)$ be the current solution of the RMP.

First, we branch on the total number of vehicles used, whenever $\bar{F} = \sum_{r \in \Omega} \bar{\lambda}_r$ is fractional. We create two branches enforcing either $\sum_{r \in \Omega} \lambda_r \leq \lfloor \bar{F} \rfloor$ or $\sum_{r \in \Omega} \lambda_r \geq \lceil \bar{F} \rceil$.

At the second level, we branch on whether an option is used to fulfill a customer request. We select an option $o^* \in O$ for which the value $\sum_{r \in \Omega} \alpha_{o^* r} \bar{\lambda}_r$ is fractional and closest to 0.5 (ties are broken arbitrarily). Instead of adding a constraint, we directly implement the branching decision by manipulating the vertex set V of the underlying network. In the first branch, option o^* is removed from V. In the second branch, all

options $\{o \in O : o \neq o^*, n_o = n_{o^*}\}$ are removed from V. Moreover, all route variables that do not comply with the branching decision are removed from the RMP.

The third level decides on the integer flow over arcs $(i,j) \in A$. For all arcs $(i,j) \in A$, we compute the value $\bar{e}_{ij} = \sum_{r \in \Omega} e^r_{ij} \bar{\lambda}_r$ and select the arc (i^*, j^*) for which the value $\bar{e}_{i^*j^*} - \lfloor \bar{e}_{i^*j^*} \rfloor$ is fractional and closest to 0.5. Two branches are created by adding the constraint $\sum_{r \in \Omega} e^r_{i^*j^*} \lambda_r \leq \lfloor \bar{e}_{ij} \rfloor$ and $\sum_{r \in \Omega} e^r_{i^*j^*} \lambda_r \geq \lceil \bar{e}_{ij} \rceil$ to the RMP, respectively. Note that this branching rule reduces to standard binary arc branching in the option-based network where it can be enforced directly on the underlying network by removing arcs. Note also that, for the option-based network, this branching rule guarantees integer route variables.

For the VRPDO and the location-based network, integer flows on arcs do not guarantee that the route variables are integer. Therefore, the fourth level applies the additional branching rule based on the flow-splitting method proposed by Feillet *et al.* (2005). Given an subpath (i, f, e, j) with $f \in F$ and $e \in E$, we branch on the flow induced by all routes that exactly traverse this subpath. This flow value must either be zero or one, which can be seen as follows. The flow on the two arcs (i, f) and (e, j) is necessarily binary, because the network structure ensures that i and j are option vertices, i.e., $i, j \in O$. If the two arcs are traversed by a route in this order, the flow on the subpath is one. Otherwise, the two arcs are traversed by different routes or not in the given order so that the flow on the subpath is zero. We always select a subpath (i^*, f^*, e^*, j^*) with flow closest to 0.5. These branching decisions change the structure of the pricing problem and a new resource for each of these branching decision has to be added to the label. It is worth mentioning that, in our computational experiments, it was never necessary to apply the fourth branching rule (for a more detailed discussion we refer to Desaulniers *et al.*, 1998; Jans, 2010).

4. Computational Results

The BPC algorithm is implemented in C++ and compiled with MS Visual Studio 2015 into 64-bit single-thread code. The callable library of CPLEX 12.9.0 was used for (re)optimizing the RMPs and to determine an integer solution based on the columns generated so far when reaching the time limit. All results were obtained using a standard PC with an Intel® Core™ i7-5930K processor clocked at 3.5 GHz and 64 GB RAM running Microsoft Windows 10 Education.

This computational results section is structured as follows: In Section 4.1, we introduce the considered benchmark instances. Section 4.2 compares the results obtained with the two networks used for solving the subproblem. To evaluate the performance of our algorithm, we also solve benchmark instances of the VRPRDL and the VRPHRDL and compare our results with those using the current state-of-the-art algorithm from the literature in Section 4.3. Finally, we explore the impact of different service levels and location capacities on total costs, computation times, and number of optimally solved instances in Section 4.4.

4.1. Benchmark Instances

We test our BPC algorithm described in Section 3 on two benchmark sets. The first set is specifically generated for this study, since the VRPDO is a new problem. The instance set is divided into twelve instance classes with ten instances each differing in the number of requests (25 or 50), the number of options (on average 1.5 options per request in class V and 2 options in class U), and the time-window widths (small, medium, or large). Priorities between 1 and 3 are uniformly distributed over the options of a request. All instances have a planning horizon of 12 hours (=720 minutes) and the average time-window width for single-delivery and multiple-delivery locations is either 60 and 240 (small), 120 and 480 (medium), or 240 and 600 (large) minutes, respectively. Locations are randomly placed in a 50 times 50 grid. Travel costs and times are computed as Euclidean distances, multiplied by 10 and rounded up.

Travel times include a location preparation time, e.g., for parking, which is 6 minutes for single-delivery locations and 4 minutes for multiple-delivery locations. The number of multiple-delivery locations is $|N|/5$, each location has a capacity of between 3 and 5 requests. Options at multiple-delivery locations have a service time of 2 minutes, while options at single-delivery locations have a service time of 5 minutes. The vehicle capacity is 150 and demands are drawn uniformly at random from the interval $[10, 20]$. We assume that there is a sufficient number of vehicles available at the depot. The fixed cost of a vehicle

are set to 100 000 so that they are dominating the travel costs. As a result, we model the hierarchical objective of minimizing the number of vehicles first and the total travel cost second. Note that the service-level parameters (β_p for $p = 1, 2$) are not part of the instance data. The benchmark set is available at https://logistik.bwl.uni-mainz.de/research/benchmarks.

The second benchmark set has been introduced by Reyes *et al.* (2017) and is divided into 60 instances for the VRPRDL and 60 instances for the VRPHRDL. All 120 instances are randomly generated with a size ranging from 15 to 120 requests clustered to a maximum of 6 options per request. We adjust the data of this benchmark set as suggested by Ozbaygin *et al.* (2017) such that the triangle inequality for travel cost and travel times holds.

4.2. Comparison of the Option-based and Location-based Networks

Recall from Section 3.2.1 that the pricing problem can be modeled and solved on two different networks, the option-based network and the location-based network. The consideration of the network size (number of arcs) and of potential dominance between labels suggested that the location-based network may be superior over the option-based network.

However, the following experiments are designed with the intention to make the comparison as fair and clear as possible. To this end, we separately analyze the solution of the linear relaxation of the master problem (1) (Secion 4.2.1) and the behaviour in the branch-and-bound tree when cutting and branching take place (Section 4.2.2). One point is that results for the linear relaxation are significantly less scattered regarding computation times, because slightly different trajectories of the column-generation processes often lead to completely different branching decisions and hereby to strongly varying branch-and-bound trees.

Moreover, for the linear relaxation, the two networks can be used interchangeably. In contrast, the possibilities of making cutting and branching decisions differ in the two networks. For every arc $a \in A^{loc}$ in the location-based network, there is a corresponding set $B \subset A^{opt}$ of arcs in the option-based network so that any constraint considering the flow over a can also be formulated as a flow constraint over B. This correspondence is detailed in Table 3. As a result, any cutting and branching decision formulated as a flow constraint in the location-based network can also be formulated as a flow constraint in the option-based network. However, the reverse is *not* true, because arcs in the option-based network may refer to subpaths in the location-based network. For example, an option-based arc $(i, j) \in A^{opt}$ with $\ell_i, \ell_j \in L^m$ and $\ell_i \neq \ell_j$ refers to the subpath $(i, f_{\ell_i}, e_{\ell_j}, j)$ in the location-based network. The use of such a subpath cannot be formulated directly with arc-flow variables of the location-based network. The experiments for producing integer solutions must therefore be designed so that results are still comparable.

Table 3: Correspondence between arcs of the location-based network and arc sets in the option-based network.

Location-based network arc $a \in A^{loc}$			Option-based network corresponding set $B \subset A^{opt}$ of arcs
(o, o')	$o \in V^{opt}$,	$o' \in V^{opt}$	$\{(o, o')\}$
(o, e_ℓ)	$o \in V^{opt}$,	$e_\ell \in E$	$\{(o, i) \in A^{opt} : \ell_i = \ell\}$
(o, f_ℓ)	$o \in V^{opt}$,	$f_\ell \in F$	$\{(o, i) \in A^{opt} : \ell_i \neq \ell\}$
(e_ℓ, o)	$e_\ell \in E$,	$o \in V^{opt}$	$\{(i, o) \in A^{opt} : \ell_i = \ell\}$
(f_ℓ, o)	$f_\ell \in F$,	$o \in V^{opt}$	$\{(i, o) \in A^{opt} : \ell_i \neq \ell\}$
$(f_\ell, e_{\ell'})$	$f_\ell \in F$,	$e_{\ell'} \in E$	$\{(i, j) \in A^{opt} : \ell_i = \ell$ and $\ell_j = \ell'\}$

4.2.1. Linear Relaxation Results

In the first experiment, we compare the linear-relaxation solutions when either the option-based or the location-based network is used in the pricing problem. We use the first benchmark, i.e., the self-generated VRPDO instances with the service levels of $\beta_1 = 0.8$ and $\beta_2 = 0.9$. Pretests have shown that 10 is a reasonable *ng*-neighborhood size. Furthermore, for the limited discrepancy search, good arcs are determined

in each pricing iteration according to the current reduced cost of the arcs. Only one bad arc per path is admissible.

We use four pricing heuristics, where the maximum number of outgoing good arcs per vertex is 2, 4, 8, and 14, respectively. Note that a reduced option-based network and a reduced location-based network are not at all equivalent. In order to eliminated the impact of the heuristical network reduction, all four pricing heuristics use the option-based network. The exact pricing is then performed with the two different networks. Note that typically the exact solution of the pricing problems consumes more than 25 % of the total computation time. This implies that with this setup any substantial impact of the underlying network should become observable.

Finally, the RMP is initialized with one big-M variable that covers all requests, fulfills all priority constraints, and has no impact on other constraints. The time limit is two hours.

Table 4: Aggregated linear relaxation results for the VRPDO benchmark.

	Option-based network		Location-based network					
			default strategy		unwise strategy		clever strategy	
Class	#sol	Time [s]	#sol	Time [s]	#sol	Time [s]	#sol	Time [s]
V.25.small	10	5.7	10	5.8	10	6.2	10	6.5
V.25.medium	10	12.5	10	14.0	10	13.3	10	14.6
V.25.large	10	16.8	10	17.4	10	17.2	10	17.4
V.50.small	10	350.7	10	337.6	10	404.5	10	313.9
V.50.medium	10	647.2	10	664.1	10	683.8	10	622.4
V.50.large	8	2534.0	8	2558.8	8	2670.9	8	2560.3
U.25.small	10	2.0	10	1.9	10	2.3	10	2.0
U.25.medium	10	85.1	10	94.1	10	92.7	10	88.2
U.25.large	10	176.4	10	182.0	10	237.9	10	177.9
U.50.small	10	52.9	10	51.1	10	60.3	10	53.0
U.50.medium	9	1327.0	9	1324.6	9	1671.6	9	1328.3
U.50.large	7	3142.5	7	3169.2	7	3196.6	7	3159.0
Total/Avg.	114	696.1	114	701.7	114	754.8	114	695.3

Table 4 displays aggregated results of the experiments grouped by classes of instances. Columns "#sol" give the number of solved linear relaxations per group of 10 instances and "Time [s]" the average consumed computation time in seconds (unsolved instances are considered with 7200 seconds). For the setup described above, the comparison of the option-based network and the location-based network can be found in the columns headlined "Option-based network" and "Location-based network, default strategy". The same 114 of 120 VRPDO instances are solved. Moreover, the computation times per class and in total are very similar.

We have carefully checked and analyzed the results and have found out the following: The majority of the computation time is spent in the dominance algorithm. Neither the creation of new labels in the extension step, nor the merge procedure of the bidirectional labeling account for a considerable share of the computation time. Moreover, the computation time of the dominance algorithm differs for different types of vertices. For example, in the location-based network, the dominance on average consumes more time at entry and exit vertices than at option vertices.

We have design a second experiment in order to highlight the impact that the dominance algorithm has on the overall performance. Recall that the *default strategy* is to apply dominance at every vertex. We define two alternative dominance strategies, one unwise and one clever.

In the *unwise strategy*, the dominance algorithm is only applied at option vertices. This eliminates the potentially superior possibility of the location-based network to compare paths not comparable in the option-based network (see Example 2). We can expect a worse performance compared to the default strategy.

In the *clever strategy*, the dominance algorithm is omitted at option vertices of multiple-delivery locations. The reasoning behind this strategy is that already the initial and inevitable dominance performed at the

entry vertex of a multiple-delivery location eliminates most of the labels that are discarded at an associated option vertex when the default strategy were used. Moreover, the dominance at the exit vertex of a multiple-delivery location actually ensures the elimination of all extensions of labels that were eliminated at option vertices of the same location in the default strategy. Hence, the clever strategy can only extend useless labels internally at a multiple-delivery location.

The last two blocks of Table 4 contain aggregated results for the unwise and the clever strategy. These results confirm the expectations, i.e., the column-generation algorithm on the location-based network using the clever strategy is superior to the one using the default strategy and, in turn, the latter is superior to the one using the unwise strategy. The differences are however not really large, the average computation times are 695.3 seconds, 701.7 seconds, and 754.8 seconds, respectively. It is more important that even with the clever strategy, the difference to the approach using the option-based network is minor.

4.2.2. Integer Results

For the full BPC algorithm, we rely on the clever strategy when using the location-based network. Moreover, the following reasonable parameters have been identified during pretests: A maximum of 10 SRIs in total and 5 per cutting iteration are allowed, since pricing problems rapidly become harder after adding SRIs. For KPIs, only sets S that contain not more than 20 requests are allowed. All valid inequalities are separated up to the second level of the branch-and-bound tree. For all branching and cutting constraints, we use the location-based network as a basis since each arc of the location-based network corresponds to a set of arcs in the option-based network as summarized in Table 3. Hence, in that direction all dual prices of arcs can be transferred in a straightforward way.

Table 5 summarizes in aggregated form the results for the integer problem (1). As additional information, columns headlined with "#BB" give the average number of branch-and-bound nodes solved and columns "#SRIs" and "#KPIs" the number of separated valid inequalities of the respective type. The average number of branch-and-bound nodes solved is computed only over those instances that are solved to proven optimality.

As for the linear relaxation, results do not differ significantly for the two network types. In both cases, the BPC algorithm solves 78 instances to optimality in less than 40 minutes on average.

Table 5: Aggregated integer results for the VRPDO benchmark.

	Option-based network					Location-based network				
Class	#sol	Time [s]	#BB	#SRIs	#KPIs	#sol	Time [s]	#BB	#SRIs	#KPIs
V.25.small	10	33.9	27.0	8.9	6.6	10	44.7	25.0	8.9	6.6
V.25.medium	10	489.5	26.9	10.0	5.2	10	563.3	26.9	10.0	5.2
V.25.large	10	938.3	104.9	8.0	4.6	10	875.6	83.0	8.0	4.6
V.50.small	6	4090.9	682.2	10.0	5.8	6	3908.8	450.5	10.0	5.9
V.50.medium	4	4861.7	63.5	9.5	7.2	4	4861.7	64.3	9.5	7.0
V.50.large	2	6227.6	48.0	7.0	4.2	2	6248.9	45.0	7.0	4.4
U.25.small	10	23.6	13.8	9.5	1.7	10	26.5	13.9	9.5	1.7
U.25.medium	9	1089.3	44.1	9.0	4.8	9	1116.3	44.0	9.0	5.4
U.25.large	9	1595.0	44.6	10.0	6.1	9	1680.6	48.1	10.0	6.1
U.50.small	6	4020.6	280.3	10.0	4.5	6	4165.8	478.8	10.0	4.3
U.50.medium	1	7014.2	49.0	9.0	5.2	1	7049.4	49.0	9.0	4.3
U.50.large	1	6559.4	5.0	7.0	0.2	1	6622.9	9.0	7.0	0.3
Total/Avg.	78	2337.1	111.6	9.2	5.1	78	2349.2	106.4	9.2	5.1

The integer results, compared over the different classes of instances, reveal the normal behavior of a BPC approach applied to a VRPTW variant: With increasing time-window widths and number of requests, the computation times grow even stronger. While all instances with small time windows and 25 options can be solved exactly within the time limit, medium and large time windows in combination with more options provide more challenging instances. At the end, only 3 of 20 instances with large time windows

and 50 options are solved to optimality. Detailed results for all 120 VRPDO instances can be found in the Appendix in Section I.

4.3. Results for the VRP with Roaming Delivery Locations

We now report the results we obtained by applying our BPC algorithm to the VRPRDL and VRPHRDL benchmarks. The current state of the art for exactly solving these problems is the branch-and-price algorithm of Ozbaygin *et al.* (2017).

These benchmarks differ significantly from the VRPDO benchmark, because time windows are more narrow so that instances require less computational effort. Therefore, we adjust our branching and cutting strategies in the following way: First, we allow up to 200 SRIs in total and 10 per round of separation. Second, we do not use the second-level branching rules (deciding whether a specific option is used), since there are many more options per request on average compared to the VRPDO instances making these branching decision less effective. Third, we apply *strong branching* with up to ten branching candidates (Achterberg, 2007). For each candidate, a heuristic evaluation of the lower bound of the child node is performed by using only the first two pricing heuristics to generate routes. We then choose the branching decision that maximizes the minimum of the lower bounds of its children.

Table 6: Comparison with results of Ozbaygin *et al.* (2017) on VRPRDL and VRPHRDL benchmarks.

		Ozbaygin *et al.* (2017)			Our results				
Problem	$\lvert N\rvert$	#sol	Time [s]	#BB	#sol	Time [s]	#BB	#SRIs	#KPIs
VRPRDL	< 30	20	1.8	247.9	20	0.4	3.0	6.7	0.0
	40	19	765.7	1896.4	20	7.5	14.1	69.7	2.6
	60	10	76.4	949.6	10	3.0	5.4	21.7	0.1
	120	3	16154.5	4481.2	10	872.2	113.9	127.1	0.6
VRPHRDL	<30	19	371.2	1550.8	20	1.3	3.1	9.9	0.0
	40	— not available —			20	10.4	11.0	38.1	0.4
	60	5	4006.0	771.4	10	150.1	16.7	95.2	0.5
	120	0	21600.0	0.0	3	17371.1	61.4	150.1	1.7
		76			93(+20)				

As done in (Ozbaygin *et al.*, 2017), we set the computation time limit to 2 hours for the instance with up to 60 requests and 6 hours for the 120-request instances. Table 6 shows aggregated results comparing the two solution approaches. The column entries have the same meaning as in the sections before. For the VRPRDL, our BPC solves all instances with an average computation time less than 15 minutes while Ozbaygin *et al.* (2017) solved 52 of the 60 instances where the largest class however requires more than 4 hours on average. For the VRPHRDL, Ozbaygin *et al.* (2017) did not report solution times for the 40-request instances. Their branch-and-price algorithm solves 24 of the 40 remaining VRPHRDL instances, while our BPC solves 33 of them and all 40-request instances. Summarizing, we find eight optimal solutions for previously unsolved VRPRDL instances and nine for the VRPHRDL. In comparison over all instances solved by both algorithms, ours is on average more than 20 times faster than the one of Ozbaygin *et al.* (2017).

4.4. Sensitivity Analysis

During the experiments, we observed that the presence of service-level and location-capacity constraints often makes VRPDO instances practically more difficult to solve. Therefore, we perform and present a sensitivity analysis to more precisely explore the impact of different service levels and location capacities on total costs, computation times, and number of optimally solved instances.

Regarding the service-level constraints (1d), we vary the required service level for the first priority between 60 % and 100 % in steps of 4 %, i.e., $\beta_1 \in \{0.6, 0.64, 0.68, \ldots, 0.96, 1.0\}$. Note that an increase of

4 % means that one more customer has to be served with first priority for the 25-customer instances and two more for the 50-customer instances. In order to keep the scenarios simple, the second priorities are set to $\beta_2 = 0$. Regarding the location-capacity constraints (1c), we only consider two scenarios where either capacities remain as given or are completely disregarded. The latter scenario assumes that there always is sufficient capacity. With these parameter variations, we create 22 scenarios for each original VRPDO instance. The one with $\beta_1 = 1.0$ and without location capacities leads to a standard VRPTW in which all first-priority options are the customers.

To limit the computational burden, we a priori restrict ourselves to those VRPDO instances solved in less than 3600 seconds when setting the service levels to $\beta_1 = 0.8$ and $\beta_2 = 0.9$. This results in 69 instances for each scenario. For all scenarios, we set the computation time limit to 3600 seconds.

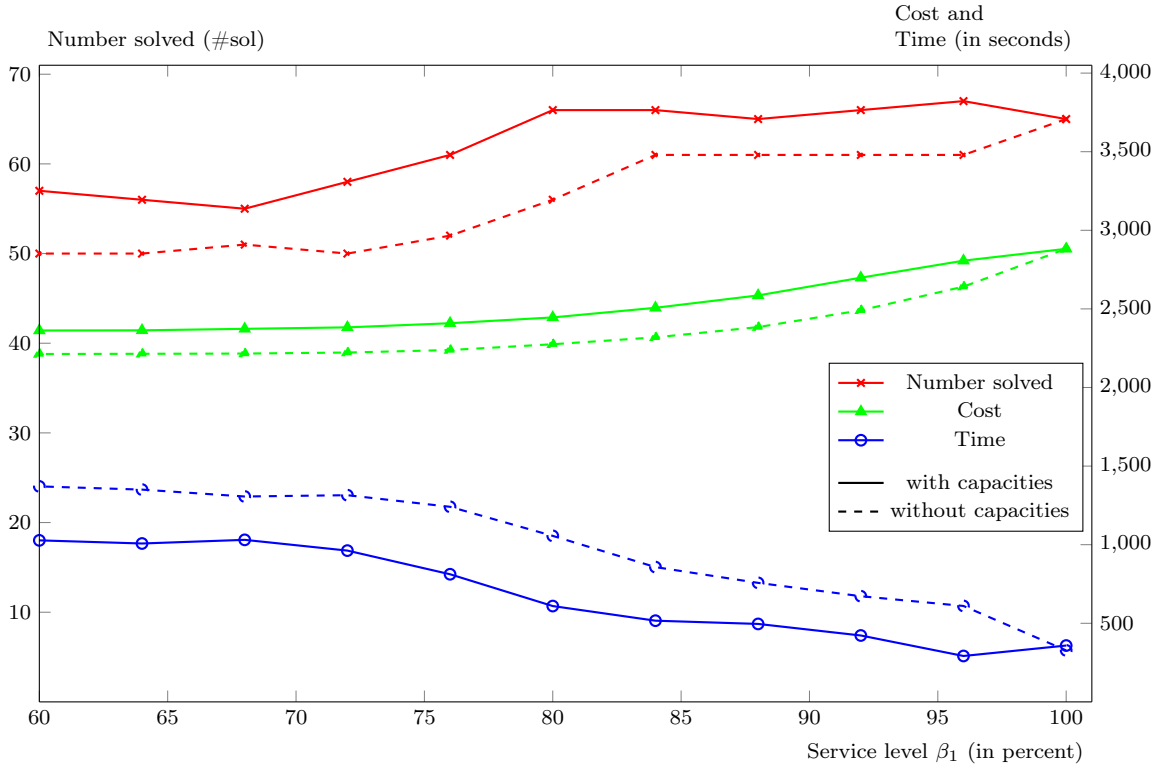

Figure 1: Sensitivity analysis, impact of service-level and location-capacity constraints.

Figure 1 summarizes the results of the sensitivity analysis. We can see that between 50 and 65 instances per scenario can be solved to optimality (#sol) and the trend is that the number increases for higher service levels. Moreover, consistently more instances can be solved when locations have restricting capacities. When looking into the details, we find that 41 original VRPDO instances are solved to optimality in all 22 scenarios.

Only for these 41 VRPDO instances, we analyze the impact on total costs and computation times. (We think that the analysis over this restricted test set makes the averages less biased.) Figure 1 also shows the average runtime and confirms that the problem difficulty decreases when the service level increases. For example, average run times decrease by around 75 % over the 60–100 % service-level interval for the scenarios without capacity. Moreover, the runtime per service level is on average 25 % higher when location capacities are disregarded.

Note also that for average computation times and number of solved instances (#sol), the impact of the service level is strongest, i.e., the absolute slope of the curves is highest, for values of β_1 between 0.7 and 0.85, both in the capacitated and uncapacitated case in our benchmark.

The average routing costs for the 41 instances that were solved to optimality in all scenarios are also depicted in Figure 1. As expected, increasing the service level leads to an increase in total costs (around 22 % higher for $\beta_1 = 1.0$ than for $\beta_1 = 0.6$ with location-capacity constraints). With uncapacitated locations, the average cost decreases by around 7 %.

In summary, the practical difficulty of the VRPDO increases when constraints are relaxed. This is true for service-level requirements as well as location-capacity constraints. We interpret these observations in the following way: the possibility to really choose between more options is the main driver of the computation time and therefore also for the capability to actually solve VRPDO instances.

5. Conclusions and Outlook

In this paper, we have introduced the vehicle routing problem with delivery options (VRPDO) as an extension of the generalized VRPTW. The extension consists of service-level constraints that consider the given priorities for the different options of fulfilling a delivery request. Moreover, subsets of options may refer to the same physical location for which capacity constraints can be specified. Both service-level and location-capacity constraints are inter-route constraints as defined in (Hempsch and Irnich, 2008) and (Irnich *et al.*, 2014, Sect. 1.3.5) making VRPDOs much more difficult than VRPTWs.

This statement is supported by the comprehensive sensitivity analysis that we performed. In a nutshell, the possibility to choose between options is the main driver of the practical difficulty of the VRPDO. We found also that more binding service-level and location-capacity constraints make instances less difficult, i.e., average computation times decrease and within a given time limit relatively more instances can be solved.

Additionally, we applied a sensitivity analysis of the effects of using different service levels and restricting or non-restricting location capacities. We have seen that otherwise identical instances tend to become easier when location capacities are restricted. Analyzing costs, we conclude that offering customers the choice of different delivery options can reduce routing costs while a reasonable service level can be provided.

We see an important value in the fact that the VRPDO model allows us to quantify the impact that service-level and location-capacity constraints have on the routing costs. Service levels are typically not naturally given, but the result of negotiations between the carrier and its partners or found by trading costs against customer expectations. In this spirit, the new model allows us to more deeply study independencies and to finally make cost-based decisions.

The exact solution approach we propose for solving VRPDO instances is based on branch-price-and-cut (BPC). It utilizes a route-based extended set-partitioning formulation. For generating routes in the pricing subproblem, we proposed a unified labeling algorithm and offered two alternative networks. On the one hand, the option-based network is straightforward because only options and depots are represented by vertices. On the other hand, the location-based network makes use of the fact that often different options share the same physical location. It has the superficial advantages of having fewer arcs and allowing for a stronger dominance than the location-based network. Surprisingly, the use of the location-based network does not pay off in the BPC algorithm, since both networks perform equally well when used for pricing. Our experiments revealed that the computational effort of the dominance algorithm is the key factor in the performance comparison. In the location-based network, the additional effort of checking dominance at the extra entry and exit vertices of each multiple-delivery location is not overcompensated by the above-named advantages. We suspect another factor to diminish the otherwise positive effects of grouping options of a location. In the VRPDO, one cannot perform the effective preprocessing on the arcs inside the same location as done in (Archetti *et al.*, 2015; Gschwind *et al.*, 2019) for the commodity-constrained split delivery VRP.

The computational experiments have shown that the BPC algorithm is highly competitive. For the comparison of the pricing networks and the sensitivity analysis, we introduced 120 new benchmark instances of the VRPDO with a variety of characteristics such as different number of requests, number of options per request, and time-window widths. We were able to compute provably optimal solutions for 78 of the 120 instances. To compare our BPC algorithm against the only other exact approach for a slightly simpler problem, we solved the benchmark sets for the VRP with roaming delivery locations. On this benchmark, we computed 17 new optimal solutions with an average computation time that is approximately 20 times faster than the former state of the art.

We see several interesting paths for future research extending the here studied VRPDO. First, revenue related aspects could be integrated in the sense that customers may pay for preferred options or carriers may give a discount or bonus when customers accept lower-prioritized delivery options. Second, we have introduced the VRPDO as a static and deterministic problem. However, in reality, location capacities, e.g., at lockers, are not completely known in advance. Over the day, customers retrieve their deliveries from lockers making the capacity time-dependent and stochastic. Third, we have observed that integrality gaps of the VRPDO instances are larger compared, e.g., to those of typical VRPTW instances. Therefore, research on finding tighter formulations with the help of problem-specific valid inequalities is worthwhile.

Acknowledgement

This research was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/8-1. This support is gratefully acknowledged.

References

Achterberg, T. (2007). *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften, Berlin, Germany.

Archetti, C., Bianchessi, N., and Speranza, M. G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research*, **64**, 1–10.

Archetti, C., Campbell, A. M., and Speranza, M. G. (2016). Multicommodity vs. single-commodity routing. *Transportation Science*, **50**(2), 461–472.

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.

Belenguer, J. M., Martinez, M. C., and Mota, E. (2000). A lower bound for the split delivery vehicle routing problem. *Operations Research*, **48**(5), 801–810.

BIEK (2019). KEP-Studie 2019: Analyse des Marktes in Deutschland: Clever verpackt, effizient zugestellt. Bundesverband Paket und Expresslogistik e. V. (BIEK) press report, 2019-26-06, https://www.biek.de/presse/meldung/kep-studie-2019.html, Berlin, Germany (in German).

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

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M. M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, Boston.

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

DHL (2014). Logistics Trend Radar: Delivering Insight today. Creating value tomorrow! https://post-und-telekommunikation.de/PuT/1Fundus/Dokumente/Studien/Deutsche_Post/2014-DHL_Logistics-TrendRadar_2014.pdf, DHL Trend Research, Troisdorf, Germany.

Doerner, K. F., Gronalt, M., Hartl, R. F., Kiechle, G., and Reimann, M. (2008). Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. *Computers & Operations Research*, **35**(9), 3034–3048.

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

Feillet, D., Dejax, P., and Gendreau, M. (2005). The profitable arc tour problem: Solution with a branch-and-price algorithm. *Transportation Science*, **39**(4), 539–552.

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, **45**(4), 239–256.

Furchheim, G., Wenk-Fischer, C., and Groß-Albenhausen, M. (2020). E-Commerce — Rekordwachstum, Nachhaltigkeit, Globalisierung & Plattform. Presentation of bevh (Bundesverband E-Commerce und Versandhandel Deutschland e.V.). https://www.bevh.org/fileadmin/content/05_presse/Pressemitteilungen_2020/200121_-_Pra__sentaion_fu__r_PK_FINAL.pdf, Berlin, Germany (in German).

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Hachicha, M., Hodgson, M. J., Laporte, G., and Semet, F. (2000). Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, **27**(1), 29 – 42.

Hempsch, C. and Irnich, S. (2008). Vehicle routing problems with inter-tour resource constraints. In B. L. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 421–444. Springer US.

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

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

Janjevic, M., Winkenbach, M., and Merchán, D. (2019). Integrating collection-and-delivery points in the strategic design of urban last-mile e-commerce distribution networks. *Transportation Research Part E: Logistics and Transportation Review*, **131**, 37–67.

Jans, R. (2010). Classification of Dantzig-Wolfe reformulations for binary mixed integer programming problems. *European Journal of Operational Research*, **204**(2), 251–254.

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.

Kohl, N., Desrosiers, J., Madsen, O. B. G., Solomon, M. M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, **33**(1), 101–116.

Manerba, D., Mansini, R., and Riera-Ledesma, J. (2017). The traveling purchaser problem and its variants. *European Journal of Operational Research*, **259**(1), 1–18.

Moccia, L., Cordeau, J.-F., and Laporte, G. (2012). An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society*, **63**(2), 232–244.

Ozbaygin, G., Ekin Karasan, O., Savelsbergh, M., and Yaman, H. (2017). A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, **100**, 115–137.

Pecin, D., Contardo, C., Desaulniers, G., and Uchoa, E. (2017). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, **29**(3), 489–502.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., and Trotter, L. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, **94**(2-3), 343–359.

Reyes, D., Savelsbergh, M., and Toriello, A. (2017). Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, **80**, 71–91.

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.

Savelsbergh, M. and Van Woensel, T. (2016). 50th anniversary invited article—city logistics: Challenges and opportunities. *Transportation Science*, **50**(2), 579–590.

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.

Wang, X., Zhan, L., Ruan, J., and Zhang, J. (2014). How to choose "last mile" delivery modes for e-fulfillment. *Mathematical Problems in Engineering*, **2014**, 1–11.

Zhou, L., Baldacci, R., Vigo, D., and Wang, X. (2018). A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. *European Journal of Operational Research*, **265**(2), 765–778.

Online Supplementary Material

Tables 7–14 present detailed computational results for all tested instances. These results have been computed with the option-based network. The first column indicates the name of the instance, the second gives the best found upper bound (UB). Columns LB Root and LB Tree contain the lower bound at the root node and at the end of the optimization, respectively. Columns Time LP and Time IP give the time the BPC algorithm takes for the solution of the root node and the complete integer program, respectively. Columns headlined with "#BB" give the number of branch-and-bound nodes solved and columns "#SRIs" and "#KPIs" the number of separated valid inequalities of the respective type.

I. Results for the VRPDO Benchmark

Table 7: Detailed Results for VRPDO instances, class V with 25 requests

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
V.25.small.1	$2876+3c^f$	$2760+3c^f$	$2876+3c^f$	0.9	25.5	31	10	1
V.25.small.2	$2638+3c^f$	$2573+3c^f$	$2638+3c^f$	4.6	46.0	13	10	6
V.25.small.3	$2716+3c^f$	$2532+3c^f$	$2716+3c^f$	4.8	50.8	51	10	14
V.25.small.4	$2781+3c^f$	$2688+3c^f$	$2781+3c^f$	33.5	179.6	96	10	15
V.25.small.5	$2401+3c^f$	$2339+3c^f$	$2401+3c^f$	0.6	8.6	18	10	4
V.25.small.6	$2723+3c^f$	$2623+3c^f$	$2723+3c^f$	2.3	7.2	8	10	2
V.25.small.7	$1941+3c^f$	$1941+3c^f$	$1941+3c^f$	1.4	1.4	1	0	0
V.25.small.8	$2366+3c^f$	$2345+3c^f$	$2366+3c^f$	7.8	8.6	6	9	12
V.25.small.9	$2817+3c^f$	$2771+3c^f$	$2817+3c^f$	0.4	3.0	13	10	10
V.25.small.10	$2910+3c^f$	$2788+3c^f$	$2910+3c^f$	0.5	8.7	33	10	2
V.25.medium.1	$2320+3c^f$	$2280+3c^f$	$2320+3c^f$	2.2	62.6	23	10	0
V.25.medium.2	$2177+3c^f$	$2129+3c^f$	$2177+3c^f$	4.1	18.2	12	10	8
V.25.medium.3	$2299+3c^f$	$2200+3c^f$	$2299+3c^f$	8.0	49.7	14	10	9
V.25.medium.4	$2904+3c^f$	$2889+3c^f$	$2904+3c^f$	1.4	3.5	6	10	2
V.25.medium.5	$2797+3c^f$	$2744+3c^f$	$2797+3c^f$	1.2	3.5	4	10	4
V.25.medium.6	$2501+3c^f$	$2420+3c^f$	$2501+3c^f$	0.7	10.0	36	10	7
V.25.medium.7	$2474+3c^f$	$2423+3c^f$	$2474+3c^f$	44.8	427.1	13	10	7
V.25.medium.8	$2423+3c^f$	$2396+3c^f$	$2423+3c^f$	3.7	25.9	12	10	5
V.25.medium.9	$2099+3c^f$	$1976+3c^f$	$2099+3c^f$	60.3	4287.6	142	10	10
V.25.medium.10	$2442+3c^f$	$2426+3c^f$	$2442+3c^f$	1.9	7.3	7	10	0
V.25.large.1	$1990+3c^f$	$1963+3c^f$	$1990+3c^f$	2.0	11.4	9	0	2
V.25.large.2	$2342+3c^f$	$2230+3c^f$	$2342+3c^f$	9.5	161.1	45	10	14
V.25.large.3	$1627+3c^f$	$1584+3c^f$	$1627+3c^f$	3.0	26.1	36	10	9
V.25.large.4	$2444+3c^f$	$2349+3c^f$	$2444+3c^f$	13.8	1371.9	126	10	4
V.25.large.5	$2064+3c^f$	$2039+3c^f$	$2064+3c^f$	5.2	51.1	9	10	3
V.25.large.6	$2129+3c^f$	$2127+3c^f$	$2129+3c^f$	3.6	5.6	2	0	1
V.25.large.7	$1533+3c^f$	$1504+3c^f$	$1533+3c^f$	2.0	10.0	9	10	1
V.25.large.8	$1930+3c^f$	$1919+3c^f$	$1930+3c^f$	3.6	22.4	6	10	1
V.25.large.9	$2484+3c^f$	$2302+3c^f$	$2484+3c^f$	71.4	3006.6	120	10	6
V.25.large.10	$2604+3c^f$	$2493+3c^f$	$2604+3c^f$	53.5	4716.9	687	10	5

Table 8: Detailed Results c^f or VRPDO instances, class U with 25 requests

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
V.50.small.1	$5111+5c^f$	$4932+5c^f$	$4977+5c^f$	320.5	7200.0	31	10	1
V.50.small.2	$5117+7c^f$	$4517+6c^f$	$4535+6c^f$	2502.5	7200.0	2	10	0
V.50.small.3	$3934+6c^f$	$3808+6c^f$	$3934+6c^f$	4.4	567.0	408	10	14
V.50.small.4	$3387+5c^f$	$3262+5c^f$	$3387+5c^f$	16.5	711.5	37	10	14
V.50.small.5	$3353+5c^f$	$3108+5c^f$	$3160+5c^f$	322.7	7200.0	38	10	7
V.50.small.6	$4343+6c^f$	$4207+6c^f$	$4343+6c^f$	14.2	6481.0	3.075	10	6
V.50.small.7	$4726+5c^f$	$4503+5c^f$	$4556+5c^f$	261.1	7200.0	26	10	5
V.50.small.8	$5338+6c^f$	$5276+6c^f$	$5338+6c^f$	7.4	349.4	81	10	0
V.50.small.9	$4082+6c^f$	$3976+6c^f$	$4082+6c^f$	1.4	161.7	185	10	4
V.50.small.10	$4531+6c^f$	$4444+6c^f$	$4531+6c^f$	40.3	3837.9	307	10	7
V.50.medium.1	—	$3972+5c^f$	$4014+5c^f$	296.2	7200.0	28	10	0
V.50.medium.2	$3576+5c^f$	$3532+5c^f$	$3576+5c^f$	36.7	1272.8	73	10	9
V.50.medium.3	$3526+5c^f$	$3486+5c^f$	$3526+5c^f$	394.3	2790.9	9	10	1
V.50.medium.4	$4275+6c^f$	$4258+6c^f$	$4258+6c^f$	4252.2	7200.0	2	5	1
V.50.medium.5	$3945+5c^f$	$3857+5c^f$	$3917+5c^f$	71.7	7200.0	76	10	8
V.50.medium.6	—	$4116+5c^f$	$4173+5c^f$	495.1	7200.0	13	10	14
V.50.medium.7	$3078+6c^f$	$2996+6c^f$	$3072+6c^f$	45.8	7200.0	196	10	5
V.50.medium.8	$3574+5c^f$	$3512+5c^f$	$3574+5c^f$	17.5	548.8	23	10	5
V.50.medium.9	$3709+6c^f$	$3610+6c^f$	$3709+6c^f$	20.7	804.8	149	10	25
V.50.medium.10	$4098+5c^f$	$3852+5c^f$	$3900+5c^f$	746.6	7200.0	15	10	4
V.50.large.1	—	$4064+5c^f$	$4064+5c^f$	7200.0	7200.0	1	0	1
V.50.large.2	$3634+7c^f$	$3383+6c^f$	$3423+6c^f$	108.6	7200.0	89	10	8
V.50.large.3	$4021+5c^f$	$3902+5c^f$	$3964+5c^f$	300.4	7200.0	33	10	4
V.50.large.4	$3882+5c^f$	—	—	7200.0	7200.0	0	0	0
V.50.large.5	—	$3210+5c^f$	$3241+5c^f$	1614.9	7200.0	12	10	6
V.50.large.6	—	—	—	7200.0	7200.0	0	0	0
V.50.large.7	$3766+5c^f$	$3579+5c^f$	$3593+5c^f$	1799.8	7200.0	6	10	0
V.50.large.8	$3893+6c^f$	$3831+6c^f$	$3893+6c^f$	130.8	4370.5	92	10	12
V.50.large.9	$3777+6c^f$	$3632+6c^f$	$3646+6c^f$	1840.7	7200.0	7	10	7
V.50.large.10	$3745+6c^f$	$3730+6c^f$	$3745+6c^f$	83.6	305.6	4	10	4

Table 9: Detailed Results for VRPDO instances, class V with 50 requests

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
U.25.small.1	$2188+3c^f$	$2144+3c^f$	$2188+3c^f$	2.4	68.8	10	10	2
U.25.small.2	$2805+3c^f$	$2727+3c^f$	$2805+3c^f$	0.9	26.2	52	10	1
U.25.small.3	$2460+3c^f$	$2441+3c^f$	$2460+3c^f$	7.4	78.0	7	10	0
U.25.small.4	$2994+3c^f$	$2983+3c^f$	$2994+3c^f$	1.2	2.4	2	5	0
U.25.small.5	$2580+3c^f$	$2526+3c^f$	$2580+3c^f$	0.4	2.4	12	10	1
U.25.small.6	$1689+3c^f$	$1665+3c^f$	$1689+3c^f$	0.8	3.8	7	10	2
U.25.small.7	$2454+3c^f$	$2382+3c^f$	$2454+3c^f$	4.1	30.5	12	10	1
U.25.small.8	$1861+3c^f$	$1803+3c^f$	$1861+3c^f$	1.1	13.0	16	10	9
U.25.small.9	$2992+3c^f$	$2936+3c^f$	$2992+3c^f$	1.0	4.0	11	10	0
U.25.small.10	$2535+3c^f$	$2513+3c^f$	$2535+3c^f$	1.0	6.5	9	10	1
U.25.medium.1	$2400+3c^f$	$2298+3c^f$	$2400+3c^f$	24.1	1163.6	72	10	5
U.25.medium.2	$2290+3c^f$	$2224+3c^f$	$2290+3c^f$	6.0	73.6	22	10	16
U.25.medium.3	$1525+3c^f$	$1525+3c^f$	$1525+3c^f$	0.7	0.7	1	0	0
U.25.medium.4	$2805+4c^f$	$2351+3c^f$	$2375+3c^f$	579.1	7200.0	5	10	1
U.25.medium.5	$2074+3c^f$	$1989+3c^f$	$2074+3c^f$	40.7	660.4	33	10	3
U.25.medium.6	$2566+3c^f$	$2434+3c^f$	$2566+3c^f$	72.5	762.1	54	10	6
U.25.medium.7	$1566+3c^f$	$1503+3c^f$	$1566+3c^f$	72.9	395.6	18	10	11
U.25.medium.8	$2065+3c^f$	$2039+3c^f$	$2065+3c^f$	28.9	184.3	7	10	1
U.25.medium.9	$1916+3c^f$	$1847+3c^f$	$1916+3c^f$	4.5	68.4	27	10	3
U.25.medium.10	$2716+3c^f$	$2563+3c^f$	$2716+3c^f$	5.4	384.6	163	10	2
U.25.large.1	$2423+3c^f$	$2359+3c^f$	$2423+3c^f$	2.3	70.0	59	10	9
U.25.large.2	$2797+3c^f$	$2633+3c^f$	$2797+3c^f$	7.7	78.5	42	10	15
U.25.large.3	$2107+3c^f$	$2096+3c^f$	$2107+3c^f$	565.0	2412.7	6	10	1
U.25.large.4	$2411+3c^f$	$2295+3c^f$	$2411+3c^f$	84.7	585.8	49	10	3
U.25.large.5	$1995+3c^f$	$1931+3c^f$	$1995+3c^f$	13.7	319.0	31	10	7
U.25.large.6	$2316+3c^f$	$2150+3c^f$	$2244+3c^f$	194.7	7200.0	210	10	7
U.25.large.7	$2237+3c^f$	$2155+3c^f$	$2237+3c^f$	13.4	431.1	88	10	1
U.25.large.8	$2586+3c^f$	$2532+3c^f$	$2586+3c^f$	836.6	3930.7	6	10	7
U.25.large.9	$2996+3c^f$	$2869+3c^f$	$2996+3c^f$	7.9	203.2	49	10	6
U.25.large.10	$2538+3c^f$	$2418+3c^f$	$2538+3c^f$	28.8	719.0	71	10	5

Table 10: Detailed Results c^f or VRPDO instances, class V with 50 requests

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
U.50.small.1	—	$4338+5c^f$	$4430+5c^f$	118.2	7200.0	81	10	4
U.50.small.2	$5303+7c^f$	$4413+6c^f$	$4554+6c^f$	18.9	7200.0	1.047	10	5
U.50.small.3	$3940+5c^f$	$3635+5c^f$	$3726+5c^f$	85.0	7200.0	291	10	0
U.50.small.4	$3732+6c^f$	$3659+6c^f$	$3732+6c^f$	6.3	200.6	69	10	2
U.50.small.5	—	$3462+5c^f$	$3568+5c^f$	52.6	7200.0	162	10	2
U.50.small.6	$3188+6c^f$	$3166+6c^f$	$3188+6c^f$	121.7	989.3	26	10	18
U.50.small.7	$3066+6c^f$	$2987+6c^f$	$3066+6c^f$	5.9	1994.0	838	10	5
U.50.small.8	$4488+6c^f$	$4373+6c^f$	$4488+6c^f$	33.8	450.1	53	10	7
U.50.small.9	$3460+5c^f$	$3383+5c^f$	$3460+5c^f$	56.5	3974.7	102	10	1
U.50.small.10	$3387+6c^f$	$3315+6c^f$	$3387+6c^f$	19.7	3797.6	594	10	1
U.50.medium.1	$4706+6c^f$	—	—	7200.0	7200.0	0	0	0
U.50.medium.2	$3432+6c^f$	$3290+6c^f$	$3312+6c^f$	478.9	7200.0	10	10	18
U.50.medium.3	$4296+6c^f$	$4253+6c^f$	$4296+6c^f$	120.1	5342.4	49	10	0
U.50.medium.4	$3662+6c^f$	$3288+6c^f$	$3321+6c^f$	423.5	7200.0	10	10	2
U.50.medium.5	$3387+5c^f$	$3300+5c^f$	$3308+5c^f$	948.9	7200.0	9	10	1
U.50.medium.6	$6136+6c^f$	$5251+6c^f$	$5341+6c^f$	326.2	7200.0	14	10	19
U.50.medium.7	$4652+5c^f$	$4394+5c^f$	$4441+5c^f$	1559.6	7200.0	7	10	4
U.50.medium.8	$3979+5c^f$	$3451+5c^f$	$3503+5c^f$	92.0	7200.0	114	10	3
U.50.medium.9	—	$2853+5c^f$	$2881+5c^f$	1193.4	7200.0	7	10	4
U.50.medium.10	$4332+6c^f$	$4118+5c^f$	$4162+5c^f$	1121.8	7200.0	23	10	1
U.50.large.1	—	$4217+6c^f$	$4222+6c^f$	4000.0	7200.0	2	10	0
U.50.large.2	—	—	—	7200.0	7200.0	0	0	0
U.50.large.3	—	—	—	7200.0	7200.0	0	0	0
U.50.large.4	—	—	—	7200.0	7200.0	0	0	0
U.50.large.5	$3678+6c^f$	$3466+6c^f$	$3500+6c^f$	466.5	7200.0	33	10	0
U.50.large.6	—	$3929+5c^f$	$3934+5c^f$	2618.5	7200.0	3	10	0
U.50.large.7	$3394+5c^f$	$3296+5c^f$	$3326+5c^f$	640.2	7200.0	25	10	1
U.50.large.8	$3917+5c^f$	$3774+5c^f$	$3819+5c^f$	1262.4	7200.0	10	10	1
U.50.large.9	$3803+5c^f$	$3541+5c^f$	$3586+5c^f$	612.0	7200.0	19	10	0
U.50.large.10	$4244+6c^f$	$4229+6c^f$	$4244+6c^f$	361.1	793.6	5	10	0

II. Results for the VRPRDL Benchmark

Table 11: Detailed Results for VRPRDL instances (1/2)

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
41.v1	3203	3132.7	3203	0.5	27.4	49	200	0
42.v1	2799	2737.0	2799	0.7	6.4	26	121	4
43.v1	2603	2503.0	2603	1.3	25.2	23	144	0
44.v1	2261	2182.1	2261	0.7	3.8	11	53	10
45.v1	3217	3217.0	3217	0.4	0.5	2	2	0
46.v1	2805	2769.5	2805	0.4	1.3	8	28	0
47.v1	3339	3192.7	3339	0.5	6.7	20	115	8
48.v1	3325	3325.0	3325	0.4	0.4	1	0	0
49.v1	3534	3463.0	3534	0.5	3.2	19	61	1
50.v1	2752	2714.5	2752	0.8	16.1	24	157	0
41.v2	2133	2084.5	2133	1.3	13.0	14	112	0
42.v2	1946	1863.8	1946	0.9	8.8	14	85	1
43.v2	1966	1916.3	1966	1.1	11.5	10	69	1
44.v2	1610	1588.5	1610	0.9	3.4	5	10	20
45.v2	2478	2478.0	2478	1.3	1.3	1	0	0
46.v2	2469	2437.4	2469	0.7	2.7	9	12	2
47.v2	1946	1918.0	1946	0.9	4.3	10	27	4
48.v2	2380	2341.3	2380	0.6	6.2	19	90	0
49.v2	2492	2477.9	2492	0.6	4.2	9	55	0
50.v2	2443	2418.9	2443	1.2	4.5	8	53	0

Table 12: Detailed Results for VRPRDL instances (2/2)

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
instance.0-triangle	901	901.0	901	0.1	0.1	1	0	0
instance.1-triangle	1286	1286.0	1286	0.1	0.1	2	2	0
instance.2-triangle	991	991.0	991	0.1	0.1	1	0	0
instance.3-triangle	1062	1062.0	1062	0.1	0.1	1	0	0
instance.4-triangle	1832	1832.0	1832	0.1	0.1	1	0	0
instance.5-triangle	1294	1272.0	1294	0.1	0.2	5	6	0
instance.6-triangle	1155	1140.1	1155	0.1	0.9	5	17	0
instance.7-triangle	1455	1455.0	1455	0.1	0.1	1	0	0
instance.8-triangle	1260	1228.0	1260	0.1	0.3	5	8	0
instance.9-triangle	1684	1684.0	1684	0.1	0.1	2	1	0
instance.10-triangle	1922	1916.0	1922	0.2	0.6	4	8	0
instance.11-triangle	2324	2266.8	2324	0.1	0.6	6	18	0
instance.12-triangle	1747	1747.0	1747	0.2	0.2	1	0	0
instance.13-triangle	1273	1265.5	1273	0.3	0.9	4	12	0
instance.14-triangle	1694	1683.3	1694	0.2	0.7	4	21	0
instance.15-triangle	1938	1938.0	1938	0.2	0.2	1	0	0
instance.16-triangle	1965	1963.0	1965	0.2	0.3	2	4	0
instance.17-triangle	1827	1827.0	1827	0.1	0.1	1	0	0
instance.18-triangle	2083	2033.0	2083	0.3	1.5	10	37	0
instance.19-triangle	1822	1813.0	1822	0.2	0.5	3	0	0
instance.20-triangle	3761	3761.0	3761	1.3	1.3	1	0	0
instance.21-triangle	2828	2828.0	2828	1.8	1.8	1	0	0
instance.22-triangle	4440	4383.3	4440	0.6	5.3	19	101	1
instance.23-triangle	3378	3362.5	3378	1.3	1.8	2	10	0
instance.24-triangle	3161	3132.5	3161	2.6	5.1	5	10	0
instance.25-triangle	4536	4536.0	4536	1.0	1.0	1	0	0
instance.26-triangle	2865	2859.8	2865	1.9	3.7	3	6	0
instance.27-triangle	4173	4168.0	4173	2.2	5.1	9	29	0
instance.28-triangle	3964	3918.5	3964	0.6	4.1	11	57	0
instance.29-triangle	4107	4107.0	4107	0.9	1.3	2	4	0
instance.30-triangle	4935	4934.7	4935	26.9	71.6	17	99	0
instance.31-triangle	5258	5126.9	5258	19.3	2001.3	211	200	0
instance.32-triangle	5061	4939.8	5061	27.6	5931.4	754	200	4
instance.33-triangle	5218	5190.1	5218	26.8	63.2	9	64	0
instance.34-triangle	5498	5428.8	5498	22.6	120.6	24	200	1
instance.35-triangle	6498	6498.0	6498	10.7	13.3	3	6	0
instance.36-triangle	4830	4800.6	4830	17.1	49.1	10	75	1
instance.37-triangle	5604	5553.0	5604	17.6	319.8	72	143	0
instance.38-triangle	5841	5798.7	5841	11.0	34.0	13	84	0
instance.39-triangle	4995	4922.3	4995	16.0	117.7	26	200	0

III. Results for the VRPHRDL Benchmark

Table 13: Detailed Results for VRPHRDL instances (1/2)

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
41.v1	2662	2662.0	2662	0.9	0.9	1	0	0
42.v1	2610	2610.0	2610	0.6	0.6	1	0	0
43.v1	2260	2245.0	2260	2.6	4.8	4	12	0
44.v1	2147	2091.0	2147	0.8	3.2	20	39	0
45.v1	3172	3172.0	3172	0.6	0.6	1	0	0
46.v1	2616	2578.5	2616	0.7	2.3	8	20	0
47.v1	3010	2943.8	3010	0.7	5.4	15	65	6
48.v1	3278	3278.0	3278	0.7	0.9	2	2	0
49.v1	3514	3430.0	3514	0.3	10.7	50	154	0
50.v1	2727	2690.5	2727	0.8	27.5	58	185	0
41.v2	1998	1998.0	1998	3.0	3.0	1	0	0
42.v2	1927	1830.5	1927	2.1	47.0	17	136	0
43.v2	1830	1812.7	1830	3.1	51.3	8	30	2
44.v2	1478	1477.1	1478	2.7	22.5	5	16	0
45.v2	2466	2466.0	2466	1.3	2.4	3	10	0
46.v2	2388	2348.5	2388	1.1	2.3	5	12	0
47.v2	1848	1804.5	1848	2.3	16.4	12	57	0
48.v2	2264	2263.5	2264	0.7	1.0	2	2	0
49.v2	2457	2444.7	2457	0.9	3.4	5	17	0
50.v2	2302	2302.0	2302	1.9	2.5	2	4	0

Table 14: Detailed Results for VRPHRDL instances (2/2)

		LB		Time		Number of		
Instance	UB	Root	Tree	LP	IP	#BB	#SRIs	#KPIs
instance.0-triangle	773	773.0	773	0.1	0.1	1	0	0
instance.1-triangle	1065	1065.0	1065	0.1	0.1	1	0	0
instance.2-triangle	988	988.0	988	0.1	0.1	1	0	0
instance.3-triangle	914	914.0	914	0.1	0.1	1	0	0
instance.4-triangle	1710	1710.0	1710	0.0	0.0	1	0	0
instance.5-triangle	1099	1077.0	1099	0.1	0.3	5	6	0
instance.6-triangle	996	989.3	996	0.4	2.0	6	14	0
instance.7-triangle	1346	1346.0	1346	0.1	0.1	1	0	0
instance.8-triangle	997	997.0	997	0.2	0.2	1	0	0
instance.9-triangle	1166	1166.0	1166	0.1	0.1	1	0	0
instance.10-triangle	1587	1584.8	1587	0.7	1.0	2	10	0
instance.11-triangle	1808	1808.0	1808	0.4	0.4	1	0	0
instance.12-triangle	1563	1563.0	1563	0.5	0.5	1	0	0
instance.13-triangle	1058	1058.0	1058	0.4	0.4	1	0	0
instance.14-triangle	1347	1319.1	1347	0.4	15.8	21	106	0
instance.15-triangle	1517	1486.0	1517	0.5	1.4	6	25	0
instance.16-triangle	1445	1445.0	1445	0.6	0.6	1	0	0
instance.17-triangle	1627	1611.7	1627	0.3	1.4	8	36	0
instance.18-triangle	1461	1461.0	1461	0.4	0.4	1	0	0
instance.19-triangle	1715	1715.0	1715	0.3	0.3	1	0	0
instance.20-triangle	2580	2580.0	2580	5.1	7.3	2	6	0
instance.21-triangle	2206	2106.8	2206	8.3	773.7	59	200	0
instance.22-triangle	3363	3357.0	3363	2.5	3.8	3	12	0
instance.23-triangle	2569	2458.5	2569	5.5	82.1	33	164	1
instance.24-triangle	2378	2277.5	2378	25.0	124.8	17	168	0
instance.25-triangle	2845	2823.5	2845	4.5	13.2	10	86	0
instance.26-triangle	2518	2490.0	2518	3.9	16.7	17	82	4
instance.27-triangle	2758	2749.5	2758	20.0	341.6	7	44	0
instance.28-triangle	2892	2715.9	2892	6.0	133.0	18	190	0
instance.29-triangle	2691	2691.0	2691	4.6	4.6	1	0	0
instance.30-triangle	3666	3665.5	3666	501.3	933.6	4	28	0
instance.31-triangle	—	3706.7	3824	268.1	21600.0	45	200	2
instance.32-triangle	—	3467.0	3493	514.2	21600.0	9	78	0
instance.33-triangle	3694	3636.4	3694	253.4	20931.4	63	200	5
instance.34-triangle	3298	3080.7	3118	435.8	21600.0	31	200	1
instance.35-triangle	4570	4152.6	4188	190.9	21600.0	41	200	1
instance.36-triangle	3217	3189.9	3217	230.9	646.1	7	44	0
instance.37-triangle	4070	3711.7	3835	111.8	21600.0	188	200	5
instance.38-triangle	4324	4209.3	4274	59.5	21600.0	211	200	3
instance.39-triangle	—	3332.4	3427	948.5	21600.0	15	151	0