




A New Exact Algorithm for Single-Commodity Vehicle Routing with Split Pickups and Deliveries

Jiliu Li,^a Zhixing Luo,^b Roberto Baldacci,^c Hu Qin,^{d,*} Zhou Xu^e

^aSchool of Management, Northwestern Polytechnical University, Xi'an 710072, China; ^bSchool of Management and Engineering, Nanjing University, Nanjing 210008, China; ^cEngineering Management and Decision Sciences, College of Science and Engineering, Hamad Bin Khalifa University, Doha 34110, Qatar; ^dSchool of Management, Huazhong University of Science and Technology, Wuhan 430074, China; ^eFaculty of Business, The Hong Kong Polytechnic University, Hong Kong

*Corresponding author

Contact: lijiliu1992@outlook.com,  <https://orcid.org/0000-0002-2459-1234> (JL); luozx.hkphd@gmail.com,  <https://orcid.org/0000-0002-0491-1501> (ZL); rbaldacci@hbku.edu.qa (RB); tigerqin1980@qq.com (HQ); lgtzx@polyu.edu.hk,  <https://orcid.org/0000-0003-1528-116X> (ZX)

Received: July 23, 2021

Revised: March 2, 2022; September 8, 2022

Accepted: September 9, 2022

Published Online in Articles in Advance:
November 18, 2022

<https://doi.org/10.1287/ijoc.2022.1249>

Copyright: © 2022 INFORMS

Abstract. We present a new exact algorithm to solve a challenging vehicle routing problem with split pickups and deliveries, named as the single-commodity split-pickup and split-delivery vehicle routing problem (SPDVRP). In the SPDVRP, any amount of a product collected from a pickup customer can be supplied to any delivery customer, and the demand of each customer can be collected or delivered multiple times by the same or different vehicles. The vehicle fleet is homogeneous with limited capacity and maximum route duration. This problem arises regularly in inventory and routing rebalancing applications, such as in bike-sharing systems, where bikes must be rebalanced over time such that the appropriate number of bikes and open docks are available to users. The solution of the SPDVRP requires determining the number of visits to each customer, the relevant portions of the demands to be collected from or delivered to the customers, and the routing of the vehicles. These three decisions are intertwined, contributing to the hardness of the problem. Our new exact algorithm for the SPDVRP is a branch-price-and-cut algorithm based on a pattern-based mathematical formulation. The SPDVRP relies on a novel label-setting algorithm used to solve the pricing problem associated with the pattern-based formulation, where the label components embed reduced cost functions, unlike those classical components that embed delivered or collected quantities, thus significantly reducing the dimension of the corresponding state space. Extensive computational results on different classes of benchmark instances illustrate that the newly proposed exact algorithm solves several open SPDVRP instances and significantly improves the running times of state-of-the-art algorithms.

History: Accepted by Andrea Lodi, Area Editor for Design and Analysis of Algorithms–Discrete.

Funding: This work was supported by the National Natural Science Foundation of China [Grants 72222011, 71971090, 71821001, 72171112], by the Young Elite Scientists Sponsorship Program by CAST [Grant 2019QNR001], and by the Research Grants Council of Hong Kong SAR, China [Grant 15221619].

Supplemental Material: The e-companion is available at <https://doi.org/10.1287/ijoc.2022.1249>.

Keywords: vehicle routing • single commodity • split pickups • split deliveries • branch-price-and-cut • label-setting algorithm • exact algorithm

1. Introduction

In this paper, we investigate a challenging vehicle routing problem with split pickups and deliveries, known as the single-commodity split-pickup and split-delivery vehicle routing problem (SPDVRP). Following the literature (Casazza et al. 2021), the problem can be defined as follows:

Consider a given digraph $G = (V, A)$, where $V = \{0, 1, \dots, n, n+1\}$ is a vertex set and A is an arc set. Let $N = V \setminus \{0, n+1\}$ represent a set of n customers, and let vertex 0 represent a depot. For the sake of convenience, we introduce a dummy depot $n' = n+1$ positioned at the same location as depot 0, and we regard depots 0

and n' as the start and end depots, respectively. The set of arcs A is defined as $A = \{(i, j) : i, j \in V, i \neq j, i \neq n+1, j \neq 0\} \setminus \{(0, n+1)\}$. Each vertex $i \in V$ has a demand q_i , which represents the amount of a product to be collected, if $q_i > 0$, or to be delivered, if $q_i < 0$. We assume that $q_0 = q_{n'} = 0$ and that the problem is balanced, namely, that $\sum_{i \in N} q_i = 0$. As shown by Hernández-Pérez and Salazar-González (2004a), the *unbalanced* case, where $\sum_{i \in N} q_i \neq 0$, might be taken into account by adding a dummy customer positioned at the same location as depot 0 and having a demand equal to $-\sum_{i \in N} q_i$. Let $N^+ = \{i : i \in N, q_i > 0\}$ be the set of *pickup* customers, and let $N^- = \{i : i \in N, q_i < 0\}$ be the set of *delivery* customers.

Associated with each arc $(i, j) \in A$ is a routing or travel cost d_{ij} and a travel time t_{ij} , the latter including the service time at vertex i . We assume that matrices $[d_{ij}]$ and $[t_{ij}]$ satisfy the triangle inequality.

A fleet of identical vehicles having capacity Q stationed at depot 0 and represented by the index set $K = \{1, \dots, |K|\}$ has to fulfill the customer demands. Each vehicle must start its route from vertex 0 and end at vertex n' , and each route has a total duration not greater than a given limit T . We assume that each vehicle route departs from depot 0 and arrives at depot n' with an empty load, and we therefore also assume that the set of arcs A does not contain the arcs in $\{(0, i) : i \in N^-\} \cup \{(i, n') : i \in N^+\}$.

In the SPDVRP, any amount of product collected from a pickup customer can be supplied to any delivery customer; that is, a *single commodity* is considered. The demand of each customer may be collected or delivered multiple times by the same or different vehicles; that is, *split* deliveries or collections are allowed. In addition, it is assumed that temporary storage is not allowed (*non-preemptive* case), meaning that the demands cannot be temporarily dropped off at any customer.

The SPDVRP consists of designing at most $|K|$ routes of minimum total cost such that, subject to the vehicle capacity and duration constraints, the demand of all the customers is satisfied.

Based on the classification scheme proposed by Berbeglia et al. (2007), the SPDVRP is a *many-to-many, single-commodity, multivehicle* pickup and delivery problem $[M-M | P/D || K]$. The SPDVRP is a generalization of the split-delivery vehicle routing problem (SDVRP) (Archetti and Speranza 2008, 2012; Irnich et al. 2014) in which the depot is the pickup location, the customers are delivery locations, and the demand of a customer is allowed to be served in more than one visit by different vehicles. The SPDVRP further extends the SDVRP by considering both pickup and delivery customers and maximum duration constraints in addition to vehicle capacity constraints. Because the SDVRP is \mathcal{NP} -hard, so is the SPDVRP.

The SPDVRP has practical applications in many routing problems, especially in inventory repositioning, where the inventories of a product of a set of retailers must be rebalanced to fit the nature of the demand. Practical applications of the SPDVRP arise in the context of milk transportation, money transfer between branch offices of a bank (Hernández-Pérez and Salazar-González 2004a), self-service bike-sharing systems (Laporte et al. 2015, Espegren et al. 2016, Laporte et al. 2018), and transportation of crude oil (Izuno et al. 2012), to mention just a few. Important problems related to the SPDVRP are multicommodity pickup and delivery problems, and comprehensive surveys of the different variants studied in the literature can be found in Berbeglia et al. (2007) and Parragh et al. (2008a, b). We also refer to Battarra

et al. (2014) for a survey about pickup and delivery problems.

VRPs with split delivery are notoriously hard combinatorial optimization problems, and the design of exact algorithms is not straightforward because the amount to be delivered at each customer visit is also a decision variable. State-of-the-art branch-price-and-cut (BPC) algorithms for classical nonsplit VRPs (see, e.g., Costa et al. 2019) cannot be easily adapted to solve split-delivery problems because handling the quantities to deliver either requires an exponential number of constraints in the master problem, or must be handled at the subproblem level, thus complicating the structure of the pricing algorithm.

In this paper, we present a new exact algorithm for the SPDVRP. The algorithm is developed from a pattern-based formulation of the problem where the LP-relaxation of the formulation is solved in a column generation fashion and embedded in an exact BPC solution approach. To solve the pricing problem associated with the pattern-based formulation, we develop a novel label-setting algorithm specifically designed for handling split deliveries. In this algorithm, label components embed reduced cost functions, unlike those classical components that embed delivered or collected quantities, thus significantly reducing the dimension of the corresponding state space. To further reduce the number of labels generated for speeding up the algorithm, we also derive and apply new dominance rules based on the reduced cost functions. We report extensive computational results on different sets of benchmark instances, and we compare our results with the results of state-of-the-art algorithms for the SPDVRP. The results show that our new exact algorithm solves several open SPDVRP instances and significantly outperforms current state-of-the-art algorithms.

The remainder of this paper is organized as follows. The next section reviews problems related to the SPDVRP. Section 3 presents the pattern-based mathematical formulation of the SPDVRP. Section 4 describes the details of the procedure adopted to solve the LP-relaxation of the formulation, with emphasis on the definition of the reduced cost functions and the label-setting algorithm used to solve the pricing problem. Details of the exact algorithm are given in Section 5, followed by its computational evaluation in Section 6. Finally, we conclude the paper and indicate future research directions in Section 7.

2. Literature Review

The SDVRP, a special case of the SPDVRP, has been addressed by many articles in the literature, most of them dealing with heuristic techniques, and surveys on the SDVRP and related problems can be found in Archetti and Speranza (2008, 2012) and in Irnich et al. (2014). The most recent exact algorithms for the SDVRP have been proposed by Archetti et al. (2011a) and Archetti

et al. (2014). In Archetti et al. (2011a), a BPC algorithm is described for both the case where the fleet of vehicles is unlimited and the case where the fleet is limited to the minimum possible number of vehicles. Instances with up to 48 customers were consistently solved to optimality, and an instance with 144 customers was also solved to optimality. Archetti et al. (2014) described two exact branch-and-cut (BC) algorithms based on two relaxed formulations and on procedures to obtain feasible solutions to the SDVRP from feasible solutions to the relaxed formulations. The results show that one of the two BC algorithms was capable of solving most of the instances with up to 50 customers and two instances with 75 and 100 customers.

One of the most studied variants of the SDVRP is the SDVRP with time windows (SDVRPTW). Desaulniers (2010) described a BPC algorithm to solve the SDVRPTW. Archetti et al. (2011b) enhanced the BPC algorithm of Desaulniers (2010) by introducing a tabu search algorithm to heuristically solve the pricing problem, several classes of valid inequalities, and a new separation algorithm for the k -path inequalities. Instances with up to 100 customers were solved to optimality within a one-hour time limit. Recently, Luo et al. (2017) investigated a BPC algorithm for solving an extension of the SDVRPTW, called the SDVRPTW with linear weight-related cost (SDVRPTWL), in which the travel cost per unit distance is charged based on a linear function of the load weight. The algorithm was tested on both SDVRPTW and SDVRPTWL instances, and instances with up to 100 customers for both of the two variants were solved to optimality within a one-hour time limit. Bianchessi and Irnich (2019) presented a BC algorithm for the SDVRPTW. The algorithm is based on a relaxed compact model, in which some integer solutions are infeasible for the SDVRPTW, and classes of valid inequalities are introduced to cut them. The computational experiments reported proved the optimality for several previously unsolved instances from the literature.

The SPDVRP is a generalization of the one-commodity pickup-and-delivery traveling salesman problem (1PDTSP) introduced in Hernández-Pérez and Salazar-González (2004a, b), where one-commodity and one capacitated vehicle is considered, and each customer must be visited exactly once. As for the SPDVRP, in the 1PDTSP any amount of product collected from a pickup customer can be supplied to any delivery customer. The SPDVRP is more challenging than the 1PDTSP since multiple vehicles and multiple visits to the customers are considered. Hernández-Pérez and Salazar-González (2004a) described a BC algorithm based on different valid inequalities, which was further improved by Hernández-Pérez and Salazar-González (2007) with new valid inequalities and which proved to be effective in solving to optimality instances with up to 100 customers. Salazar-González and Santos-Hernández (2015) further extended the 1PDTSP by introducing a more general

problem, the split-demand one-commodity pickup-and-delivery traveling salesman problem (1-SPDTSP). The 1-SPDTSP is a many-to-many, single-commodity, multi-vehicle problem, where the number of times that a vehicle visits a customer is limited and preemption is also allowed. The depot is considered as a standard customer having its own demand and capacity, and the vehicle is not forced to depart from or arrive at the depot empty. The problem has been modeled using a single-commodity flow formulation, and both a heuristic algorithm, based on the heuristic proposed by Hernández-Pérez and Salazar-González (2004b), and a BC approach, were described for its solution. The authors reported optimal solutions for 1-SPDTSP instances with up to 50 customers. Recently, Hernández-Pérez et al. (2018) described a matheuristic algorithm that iteratively applies a constructive procedure and a refinement procedure with the aim of solving large-sized 1-SPDTSP instances with up to 500 customers.

Problems that are closely related to the SPDVRP and corresponding applications arise in the context of self-service bike-sharing systems. Due to the growth in popularity in recent years of self-service bike-sharing and scooter-sharing systems, there is a rapidly growing amount of literature on these problems. The survey paper of Laporte et al. (2015), and its updated version of Laporte et al. (2018), classify the relevant literature by considering station location, fleet dimensioning, station inventory, rebalancing incentives, and vehicle repositioning. Below, we briefly review the most recent exact algorithms in this area, and the reader is referred to the surveys of Laporte et al. (2015, 2018) for an in-depth analysis of the literature.

In the *static* variant of these problems, it is assumed that bicycles are not moved by users and that capacitated vehicles must be used to reallocate the bicycles between the different stations or locations. This corresponds to the practical scenario where every night the bicycle inventories at the locations must be restored to fit the demand at minimum travel cost. The problems are then classified according to (i) the number of maximum visits allowed to the stations (i.e., *single* or *multiple visits*), (ii) the *preemptive* or *nonpreemptive* cases, and (iii) the number of vehicles available (*single* or *multiple vehicles*). In the *multiple visits* case, a station can be visited more than once by different vehicles, and a station is also allowed to be visited multiple times by the same vehicle. Two special cases of the multiple visits case arise: (i) a vehicle is allowed to visit a customer at most once, hereafter referred to as a *multiple-visit, different-vehicle* case; and (ii) a customer can be visited (even more than once) by only one vehicle, hereafter referred to as a *multiple-visit, same-vehicle* case.

Static, single-visit, nonpreemptive problems were investigated by Erdoğan et al. (2014) and Dell'Amico et al. (2014). Erdoğan et al. (2014) addressed a single-vehicle

variant, where stations might be visited at most once and the resulting numbers of bikes at stations after the rebalancing should lie within a given interval instead of exactly corresponding to a fixed target value. The objective function includes a commodity handling cost. The authors developed an integer programming formulation and described valid inequalities that have been used to develop a BC algorithm as well as a Benders decomposition scheme. The BC algorithm was capable of solving instances with up to 50 locations. Dell’Amico et al. (2014) considered a multiple vehicle problem, called the bike rebalancing problem (BRP), where initially balanced stations are to be visited as well, to ensure a daily inspection. The authors described four mathematical formulations for the BRP and proposed a BC algorithm for its solution capable of solving to optimality BRP instances involving up to 50 locations or stations. Dell’Amico et al. (2016) solved the BRP by developing a destroy and repair heuristic, improving the best-known solutions for several instances from the literature.

Chemla et al. (2013), Erdoğan et al. (2015), Bulhões et al. (2018), Casazza et al. (2019), and Bruck et al. (2019) investigated static and multiple-visit variants. Chemla et al. (2013) studied a single-vehicle, preemptive problem. The authors proposed a mathematical formulation from which two relaxations are derived. A tabu search algorithm is also described to solve instances involving up to 100 stations. The same problem was also considered by Cruz et al. (2017), who obtained improved results on the existing benchmark instances by means of an iterated local search algorithm. Erdoğan et al. (2015) presented a BC algorithm for the preemptive single-vehicle variant and reported results of computational tests on benchmark instances from the literature showing that instances with up to 60 stations can be solved to optimality in less than two hours of computing time. Casazza et al. (2019) considered a multiple-vehicle non-preemptive problem, called the multiple-vehicle balancing problem (MVBP), with an additional constraint imposing a maximum number of visits per station. Based on theoretical properties of the problem, the authors proposed an integer linear programming formulation and introduced some valid inequalities that were used to compute valid lower bounds in a column generation fashion. Upper bounds were also computed by means of a rounding heuristic and a Memetic algorithm, and the lower and upper bounds computed were used to solve to prove optimality instances with up to 20 stations. Any instance of the MVBP can be transformed into an equivalent SPDVRP instance simply by modeling the additional constraint on the maximum number of visits (say, α), by setting the travel time $t_{ij} = 1$, $\forall (i, j) \in A$, and $T = \alpha$. Bulhões et al. (2018) investigated a multiple-visit, multiple-vehicle, nonpreemptive problem with an upper limit on the maximum number of visits to a station. In addition, a station could only be served (even multiple

times) by one of the fleet vehicles, and the vehicle workload constraint was also considered. The authors designed a BC algorithm based on an extended network-based mathematical formulation and an iterated local search-based heuristic. The BC was capable of solving to optimality several instances with 20 stations and one instance with 30 stations and three vehicles within the time limit of one hour. Bruck et al. (2019) also investigated a single-vehicle, nonpreemptive problem. The authors investigated theoretical results concerning problem complexity and worst-case analysis, and then proposed three exact algorithms based on different mathematical formulations. Computational experiments were reported for instances involving up to 80 stations.

To the best of our knowledge, the only work considering the SPDVRP can be found in Casazza et al. (2021). The authors proposed a formulation where routes are decomposed into sequences of simpler substructures called *clusters*. Valid inequalities, a rounding heuristic, and a branch-and-price (BP) algorithm are also described. The proposed algorithm is competitive with the algorithm proposed in Casazza et al. (2019) for the MVBP, and instances with up to 20 stations were solved to optimality. In the computational results in Section 6, we compare our results with the results reported in Casazza et al. (2019) and Casazza et al. (2021).

Table 1 provides a summary of the main problem features of works closely related to the SPDVRP. As shown in the table, the SPDVRP addresses the most general case in terms of types of visits to the customers, which is regarded as the main factor that influences the complexity of split-delivery problems.

3. A Pattern-Based Formulation for the SPDVRP

The pattern-based (*PB*) formulation is based on the set-partitioning formulation originally proposed by Balinski and Quandt (1964) for the capacitated VRP (CVRP), and it associates a decision variable with each feasible route of the SPDVRP.

A *route* in graph G is defined as a (not necessarily elementary) path $R = (0, i_1, \dots, i_b, n')$ starting at depot 0, ending at depot n' , visiting a set of vertices $\{i_1, \dots, i_b\} \subseteq N$ (maybe more than once), and being such that the total travel time of the route is less than or equal to T . Associated with a route R are (i) the set of *quantities* $\{l_{i_1}, \dots, l_{i_b}\}$ delivered (negative values) and collected (positive values) to the visited vertices, and (ii) a *demand pattern*, or simply *pattern* $\alpha \in \mathbb{R}^{|N|}$, where α_i for $i \in N^+$ with $0 \leq \alpha_i \leq q_i$ represents the total amount of the product picked up from customer i , and α_i for $i \in N^-$ with $q_i \leq \alpha_i \leq 0$ represents the total quantity delivered to customer i . Note that if the route is elementary, for a customer $i \in N$ we have $\alpha_i = l_i$ if $i \in \{i_1, \dots, i_b\}$, and $\alpha_i = 0$ otherwise.

Table 1. Features of the Main Works Related to the SPDVRP

Work	Visits	Vehicles	Type of visits	Preemptive	Constraints
	(s)ingle	(s)ingle	(sv) same vehicle	(y)es	(c)apacity
	(m)ultiple (o)ptional	(m)ultiple	(df) different vehicle	(n)o	(m)ax. number of visits (d)uration
SPDVRP	m	m	sv, dv	n	c, d
Hernández-Pérez and Salazar-González (2004a)	s	s	—	n	c
Chemla et al. (2013)	m	s	sv	y	c
Erdoğan et al. (2014)	o	s	sv	n	c
Dell’Amico et al. (2014)	s	m	—	n	c
Erdoğan et al. (2015)	m	s	sv	y	c
Salazar-González and Santos-Hernández (2015)	m	s	sv	y	c, m
Bulhões et al. (2018)	m	m	sv	n	c, m, d
Casazza et al. (2019)	m	m	sv, dv	n	c, m
Bruck et al. (2019)	m	s	sv	n	c
Casazza et al. (2021)	m	m	sv, dv	n	c, d

A set of quantities and a demand pattern are feasible for route $R = (0, i_1, \dots, i_b, n')$, and we simply refer to it as a *feasible pattern*, if (i) the vehicle starts and ends the route R with an empty load, (ii) along the route R the total demand collected from the pickup customers is delivered to the delivery customers (i.e., $\sum_{i \in N} \alpha_i = 0$), and (iii) the load of the vehicle after it visits the h th vertex of the route R is nonnegative and does not exceed the vehicle capacity Q (i.e., $0 \leq \sum_{s=1}^h l_{i_s} \leq Q, h = 1, \dots, b-1$).

Let \mathcal{R} be the index set of all routes, where the route of index $r \in \mathcal{R}$ is denoted by R_r . For each $r \in \mathcal{R}$, let \mathcal{P}_r be the index set of all feasible patterns associated with route R_r . Given a route $r \in \mathcal{R}$ and a pattern index $p \in \mathcal{P}_r$, let α_{ipr} be a continuous coefficient equal to the total demand collected from or delivered to customer i by route R_r according to the pattern with index p . In addition, we denote by c_r the routing cost associated with route R_r , computed as $c_r = \sum_{(i,j) \in A} \gamma_{ijr} d_{ij}$, where γ_{ijr} is the number of times that arc (i, j) is traversed by route R_r . Due to the constraint on the maximum route duration, each route R_r satisfies that $\sum_{(i,j) \in A} \gamma_{ijr} t_{ij} \leq T$.

Let θ_{pr} be a nonnegative integer variable representing the number of vehicles assigned to pattern $p \in \mathcal{P}_r$ of route R_r for $r \in \mathcal{R}$. Formulation *PB* is as follows:

$$(PB) \quad \min \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} c_r \theta_{pr} \quad (1a)$$

$$\text{subject to (s.t.) } \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \alpha_{ipr} \theta_{pr} \leq q_i, \quad \forall i \in N, \quad (1b)$$

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \theta_{pr} \leq |K|, \quad (1c)$$

$$\theta_{pr} \geq 0 \text{ and integer, } \quad \forall p \in \mathcal{P}_r, r \in \mathcal{R}. \quad (1d)$$

The objective function in (1a) states to minimize the total routing cost. The constraints in (1b) impose that the demand of each customer is satisfied by the demand patterns associated with the routes selected in the solution, and their correctness arises from the fact that $\sum_{i \in N} q_i = 0$.

The constraints in (1c) limit the number of vehicles used to the number of vehicles available.

Casazza et al. (2019) also described a pattern-based formulation to model the MVBP. However, the authors assume that $|q_i| \leq Q, \forall i \in N$, and showed that, under this assumption, variables θ_{pr} can be stated as binary variables. To remove the limit $|q_i| \leq Q, i \in N$, their algorithm requires the introduction of $k = \lceil |q_i|/Q \rceil$ vertices for vertex i , say, $\{i_1, \dots, i_k\}$, such that $|q_{i_h}| = Q, h = 1, \dots, k-1$, and $|q_{i_k}| = |q_i| - Q(k-1)$.

The LP-relaxation of formulation *PB* can be solved in a column generation fashion by iteratively solving a *restricted master problem* (RMP) and the pricing problem, which determines whether there exists a variable θ_{pr} to be added to the RMP to improve its current solution. For a thorough description of the column generation technique and corresponding solution approaches, the reader is referred to the book of Desaulniers et al. (2005) and the reviews of Barnhart et al. (1998) and Lübbecke and Desrosiers (2005).

3.1. Pricing Problem

Let $\mu = (\mu_0, \mu_1, \mu_2, \dots, \mu_n)$, where $\mu_1, \mu_2, \dots, \mu_n \leq 0$, and $\mu_0 \leq 0$ are the dual variables associated with the constraints in (1b) and (1c), respectively. Given a route $r \in \mathcal{R}$ represented by path $R_r = (0, i_1, \dots, i_b, n')$, the most negative reduced cost pattern α associated with the route can be computed by solving the following LP problem:

$$c_r - \mu_0 + \min \left(- \sum_{i \in V(R_r)} \alpha_i \mu_i \right)$$

$$\text{s.t. } 0 \leq \sum_{s=1}^h l_{i_s} \leq Q, \quad h = 1, \dots, b-1, \quad (2a)$$

$$\sum_{i \in V(R_r)} \alpha_i = 0, \quad (2b)$$

$$0 \leq \alpha_i = \sum_{s \in R_r(i)} l_{i_s} \leq q_i, \quad \forall i \in V(R_r) \cap N^+, \quad (2c)$$

$$q_i \leq \alpha_i = \sum_{s \in R_r(i)} l_{is} \leq 0, \quad \forall i \in V(R_r) \cap N^-, \quad (2d)$$

$$0 \leq l_{is} \leq q_i, \quad \forall s \in R_r(i), i \in N^+, \quad (2e)$$

$$q_i \leq l_{is} \leq 0, \quad \forall s \in R_r(i), i \in N^-, \quad (2f)$$

$$\alpha_i = 0, \quad \forall i \in N \setminus V(R_r), \quad (2g)$$

where $V(R_r)$ is the set of customers visited by route R_r and $R_r(i) \subset \{1, 2, \dots, b\}$ for $i \in V(R_r)$ is the set of vertex indices of route R_r corresponding to customer $i \in N$ (i.e., $i_h = i_k, \forall i, k \in R_r(i), h \neq k$). In the formulation, the constraints in (2b) impose the vehicle capacity constraints, and the constraints in (2f) and (2g) are redundant due to the definition of variables α_i given by the constraints in (2d) and (2e). It can be seen that the formulation admits a feasible solution where each customer $i \in V(R_r)$ can be associated with $\alpha_i = 0$.

Based on the above formulation, the pricing problem therefore calls for the joint definition of route R and the associated pattern α , a variant of the shortest path problem with resource constraints (SPPRC) (Irnich and Desaulniers 2005, Irnich and Villeneuve 2006) that allows for the generation of paths with cycles (i.e., routes with multiple visits to the same customer). The SPPRC is a relaxation of the elementary SPPRC (ESPPRC), a strongly \mathcal{NP} -hard problem, where elementarity requirements are also considered. The ESPPRC was studied in the context of vehicle routing problems by several researchers (see Feillet et al. 2004; Chabrier 2006; Righini and Salani 2006, 2008), who developed effective dynamic programming algorithms for solving it. The SPPRC is easier to solve than the ESPPRC, as it can be solved by a pseudo-polynomial algorithm, and a number of SPPRC relaxations have been proposed in the VRP literature (see Costa et al. 2019).

As shown by the literature, the presence of the split deliveries and collections further complicates the ESPPRC (Desaulniers 2010, Archetti et al. 2011b, Casazza et al. 2021) and the SPPRC (Casazza et al. 2019). Indeed, in these variants the quantity delivered or collected at each visited customer is a decision variable.

The algorithm of Casazza et al. (2019) requires a state variable for each customer that accounts for the total number of units picked up or delivered to the customer. The algorithm follows the procedure proposed by Baldacci et al. (2008) for the pickup and delivery problem with time windows where forward paths are first generated and combined to form complete (not necessarily elementary) routes.

The pricing algorithm of Desaulniers (2010) solves an ESPPRC and exploits certain properties of the pricing problem related to the knapsack problem. The algorithm requires state variables indicating (i) the total quantity delivered in the full deliveries, (ii) whether a split delivery occurred, (iii) the maximum quantity that can be

delivered in the split delivery, and (iv) the unit dual price for the split delivery. In this way, when extending a label from a vertex i to a vertex j , their pricing algorithm creates up to three labels associated with a zero delivery, a split delivery, and a full delivery to j , respectively. The pricing algorithm relies on the use of reduced (linear) cost functions. More precisely, a state variable is used to compute the maximal reduced cost that is obtained by replacing the split delivery, if any, by a zero delivery. The reduced cost function of a label (and associated path) is then defined as $c - \pi(x - L)$, where (i) c is the maximal reduced cost, (ii) π is the unit dual price of the split delivery, (iii) L is the total quantity delivered in the full deliveries, and (iv) $x \in [L, L + l]$ is the total quantity delivered, including the split delivery if any, where l is the maximum quantity that can be delivered in the split delivery. Given the reduced cost function of a label, the dominance rule designed for classical ESPPRC (Irnich and Desaulniers 2005) cannot be used directly. Desaulniers (2010) therefore described new dominance rules based on the comparison of the reduced cost functions associated with two labels.

The algorithm of Casazza et al. (2021) is inspired by the one proposed by Desaulniers (2010). Casazza et al. (2021) solve the pricing problem of an alternative SPDVRP formulation, where routes are decomposed into sequences of simpler substructures called *clusters*. Their pricing algorithm exploits properties of the pricing problem related to the fractional knapsack problem with penalties and, similarly to Desaulniers (2010), requires state variables representing the residual capacity of a partial path and the existence (in the partial path) of a fractional vertex, that is, a vertex having its demands fractionally serviced.

The label-setting algorithm that we propose also relies on reduced cost functions, but they are expressed as general piecewise-linear convex functions and embedded as components for the labels. The main advantage is that a label can be defined without the use of components related to the quantities delivered or collected, thus reducing the dimension of the state space. Furthermore, we propose some set-dominance rules used to reduce the number of labels and therefore increase the performance of the label-setting algorithm.

The use of general functions as label attributes has also been adopted by Liberatore et al. (2011) for the VRP with soft time windows, by Dabia et al. (2013) for solving the time-dependent VRPTW, and by Luo et al. (2017) for the SDVRPTWL. The label-setting algorithms of Liberatore et al. (2011) and Luo et al. (2017) use piecewise-linear reduced cost functions of the start of service time and of the quantity delivered, respectively, whereas in Dabia et al. (2013) the function represents the ready time at the last node visited by the label as a function of the departure time at the depot.

3.2. Relaxation \overline{PB}

To overcome these drawbacks and in order to reduce the complexity of the pricing problem, we consider a relaxation of formulation PB where the index set of patterns \mathcal{P}_r of route R_r for $r \in \mathcal{R}$ is enlarged to set $\overline{\mathcal{P}}_r$ containing also indices of patterns where the total demand collected from or delivered to a customer can exceed the demand of the customer; that is, coefficients α_{ipr} can exceed the demand associated with customer i (either positively or negatively). More specifically, we relax the constraints in (2d) and (2e) that define a feasible pattern. In the new pattern definition, the amount of the product collected from or delivered to a customer each time the customer is visited cannot exceed the customer demand, as imposed by the constraints in (2f) and (2g). We denote by \overline{PB} the formulation obtained by substituting the index set of patterns \mathcal{P}_r , $\forall r \in \mathcal{R}$, with $\overline{\mathcal{P}}_r$. It is worth noting that formulation \overline{PB} under the integrality requirements in (1d) provides a correct SPDVRP formulation. In the following, we denote by \overline{LPB} the LP-relaxation of formulation \overline{PB} .

4. Solving the Pricing Problem of Formulation \overline{LPB}

This section describes the algorithm adopted to solve the pricing problem associated with formulation \overline{LPB} , which is a key component of our exact algorithm.

We next give the details of the reduced cost functions used together with their properties, followed by a description of the label-setting algorithm.

4.1. Forward, Backward Paths, and Reduced Cost Functions

We define a *forward path* $P = (i_0 = 0, i_1, \dots, i_{s-1}, i_s)$ as a (not necessarily elementary) path starting at depot 0, visiting vertices $V(P) = \{0, i_1, \dots, i_{s-1}, i_s\}$ and ending at customer i_s with a total duration $t(P)$ less than or equal to T . Similarly, a *backward path* $\overline{P} = (i_s, i_{s-1}, \dots, i_1, i_0 = n')$ is a (not necessarily elementary) path starting at vertex i_s , visiting vertices $V(\overline{P}) = \{i_s, i_{s-1}, \dots, i_1, 0\}$ and ending at the depot n' with a total duration $t(\overline{P})$ less than or equal to T .

The algorithm is based on the following observations:

(i) Any route R represented by path $R = (0, i_1, \dots, i_b, n')$ can be decomposed, for every $i = i_s$, $s \in \{1, \dots, b-1\}$, into a forward path P ending at i_s and a backward path \overline{P} starting at $j = i_{s+1}$ such that $t(P) + t_{ij} + t(\overline{P}) \leq T$.

(ii) The set of patterns α associated with route R satisfying the constraints in (2b), (2c), (2f), (2g), and (2h), can be partitioned by the amount w of demand picked up from the pickup customers of path P and left for the delivery customers of path \overline{P} , that is to say that the vehicle load along arc (i, j) joining paths P and \overline{P} is equal to w .

(iii) Let $\overline{c}_P^f : \mathbb{R}_+ \rightarrow \mathbb{R}$ be a *forward reduced cost function*, where $\overline{c}_P^f(w)$ is equal to the cost of the minimum reduced cost path P and associated demand pattern $\alpha(P)$ if a quantity w is delivered to the customers of path \overline{P} . In addition, let $\overline{c}_{\overline{P}}^b : \mathbb{R}_+ \rightarrow \mathbb{R}$ be a *backward reduced cost function*, where $\overline{c}_{\overline{P}}^b(w)$ is equal to the cost of the minimum reduced cost path \overline{P} and associated demand pattern $\alpha(\overline{P})$ if a quantity w is collected from the customers of path P . The reduced cost $\overline{c}(R)$ of the route and corresponding demand pattern having minimum reduced cost can be computed as

$$\overline{c}(R) = \min_{0 \leq w \leq \min\{W(P), W(\overline{P})\}} \left\{ \overline{c}_P^f(w) + d_{ij} + \overline{c}_{\overline{P}}^b(w) \right\},$$

where $W(P) = \min\{Q, \sum_{i \in V(P): i \in N^+} q_i\}$ is the maximum amount of product collected from the pickup customers along path P and $W(\overline{P}) = \min\{Q, \sum_{i \in V(\overline{P}): i \in N^-} -q_i\}$ is the maximum amount of the product required by the delivery customers along path \overline{P} .

The following theorems state that functions \overline{c}_P^f and $\overline{c}_{\overline{P}}^b$ admit piecewise-linear convex representations. For the sake of the exposition, we consider a generic piecewise-linear convex function $f : \mathbb{R}_+ \rightarrow \mathbb{R}$ expressed as $f(x) = \max_{h \in H} \{a_h x + b_h\}$, where $H = \{1, 2, \dots, m\}$ is the index set of its *breakpoints*.

Theorem 1. Function \overline{c}_P^f is a piecewise-linear convex nondecreasing function.

Proof. The proof is provided in the e-companion to this paper (see Section EC.1.1). \square

Theorem 2. Function $\overline{c}_{\overline{P}}^b$ is a piecewise-linear convex nonincreasing function.

Proof. The proof is provided in the e-companion to this paper (see Section EC.1.2). \square

Based on the above observations, our pricing algorithm considers in each label a piecewise-linear reduced cost function of the quantity w . The functions are properly initialized and updated with each extension. When performing an extension along an arc (i, j) , the reduced cost function of the new label is obtained by summing up the function from the previous label and the function associated with the delivery or pickup service at j . The various components of the algorithm are described in detail below.

4.2. A Label-Setting Algorithm for the Pricing Problem

The algorithm is based on the *bidirectional search* algorithm proposed by Righini and Salani (2006), where *forward* and *backward* paths are first generated and then combined to form complete routes by a *joining procedure*.

4.2.1. Forward Extension. A forward path $P = (i_0 = 0, i_1, \dots, i_{s-1}, i_s)$ is represented by a label L^f associated

with the last vertex i_s , where $L^f = (v^f, s^f, V^f, W^f, \{(a_h^f, b_h^f)\}_{h \in H^f})$ is composed of the following information:

- $v^f \in V$: the last vertex visited.
- s^f : the total duration.
- $V^f \subseteq V$: the set of vertices that could be reached from the last vertex v^f .
- W^f : the maximum amount of product that can be delivered to the delivery customers to be appended to the path.
- $\{(a_h^f, b_h^f)\}_{h \in H^f}$: the representation of the piecewise-linear convex nondecreasing function.

In the label L^f , for a given $w \in [0, W^f]$ representing the demand picked up by path P and left for the vertices to be appended to vertex v^f , the reduced cost associated with the path can be computed as $\max_{h \in H^f} \{a_h^f w + b_h^f\}$.

A forward label L_1^f with $i = v_1^f$ can be extended to a vertex $j \in V_1^f$ yielding a new label L_2^f based on the following extension functions. We have two cases:

(i) $j \in N^+$ (pickup customer). The components of label L_2^f are computed as follows:

- $v_2^f = j$.
- $s_2^f = s_1^f + t_{ij}$.
- $V_2^f = V_1^f \setminus \{l : (j, l) \in A, s_2^f + t_{jl} + t_{ln'} > T\} \setminus \{j\}$.
- $W_2^f = \min\{Q, W_1^f + q_j\}$.
- $\{(a_h^f, b_h^f)\}_{h \in H_2^f} = \{(a_h^f, b_h^f + d_{ij})\}_{h \in H_1^f} \cup \{(a_h^f, b_h^f - (a_h^f + \mu_j)q_j + d_{ij})\}_{h \in H_1^{f+}} \cup \{(-\mu_j, \max\{z_1^*, \max_{h \in H_1^f} \{(a_h^f + \mu_j)W_1^f + b_h^f\}, \max_{h \in H_1^{f+}} \{b_h^f\} + d_{ij}\})\}$,

where $H_1^{f-} = \{h \in H_1^f : a_h^f + \mu_j < 0\}$, $H_1^{f+} = \{h \in H_1^f : a_h^f + \mu_j \geq 0\}$ and $z_1^* = \min\{\max_{h \in H_1^f} \{a_h^f x + b_h^f\}\}$.

(ii) $j \in N^-$ (delivery customer). Components v_2^f, s_2^f and V_2^f are computed as in the previous case, whereas the remaining components are updated as follows:

- $W_2^f = W_1^f$.
- $\{(a_h^f, b_h^f)\}_{h \in H_2^f} = \{(a_h^f, b_h^f - (a_h^f + \mu_j)q_j + d_{ij})\}_{h \in H_1^f} \cup \{(a_h^f, b_h^f + d_{ij})\}_{h \in H_1^{f+}} \cup \{(-\mu_j, \max\{z_1^*, \max_{h \in H_1^f} \{(a_h^f + \mu_j)W_1^f + b_h^f\} + d_{ij}\})\}$,

where H_1^{f-}, H_1^{f+} and z_1^* are defined as in the previous case.

In the expansion, label W^f is updated only for vertices in N^+ , and its upper limit Q represents the maximum quantity that can be picked up and left for the vertices to be appended to the path. Note that the update of the reduced cost functions follows the scheme of the proof of Theorem 1.

The effectiveness of a label-setting algorithm strongly relies on the use of dominance rules aimed at removing dominated paths that cannot be part of any optimal solution. In the above definition, with the label L^f , each path P is associated with a set of infinite loading patterns defined by values $w \in [0, W^f]$. As illustrated by Irnich and Desaulniers (2005), given two labels L_1^f and L_2^f ,

respectively, label L_1^f is dominated by label L_2^f if we have the following:

- $v_1^f = v_2^f$.
- Any feasible extension of label L_1^f is also feasible for L_2^f .
- Extending label L_1^f always results in a route having a reduced cost greater than or equal to the reduced cost of the route obtained from label L_2^f and the extension of L_1^f .

However, verifying the above conditions is not straightforward. Given values $w_1 \in [0, W_1^f]$ and $w_2 \in [0, W_2^f]$, a sufficient condition for L_2^f to dominate L_1^f is

- $v_1^f = v_2^f$.
- $s_2^f \leq s_1^f$.
- $V_2^f \subseteq V_1^f$.
- $w_2 = w_1$.
- $\max_{h \in H_2^f} \{a_h^f w_2 + b_h^f\} \leq \max_{h \in H_1^f} \{a_h^f w_1 + b_h^f\}$.

For our algorithm, such a dominance rule is not applicable, because the quantities w_1 and w_2 are not components of the labels. Nevertheless, given two labels L_1^f and L_2^f such that conditions (i)–(iii) hold, L_1^f is dominated by label L_2^f if $W_2^f \geq W_1^f$ and $\max_{h \in H_2^f} \{a_h^f w + b_h^f\} \leq \max_{h \in H_1^f} \{a_h^f w + b_h^f\}$, for all $w \in [0, W_1^f]$. In practice, the latter dominance rule can be quite weak since it requires that the reduced cost function of L_2^f lie below the one of L_1^f on the interval $[0, W_1^f]$. We therefore extend the dominance rule to compare L_1^f with a set of labels for which we compute the lower envelope of the set of piecewise-linear convex functions associated with the labels. The dominance rule, called a *set-dominance* rule, is defined as follows.

Let $L_1^f = (v_1^f, s_1^f, V_1^f, W_1^f, \{(a_h^f, b_h^f)\}_{h \in H_1^f})$, and define \mathcal{L}_1^f to be the set of labels $L^f = (v^f, s^f, V^f, W^f, \{(a_h^f, b_h^f)\}_{h \in H^f})$ such that (i) $v^f = v_1^f$, (ii) $s^f \leq s_1^f$, and (iii) $V_1^f \subseteq V^f$.

Given a value w with $0 \leq w \leq Q$, for a label $L^f \in \mathcal{L}_1^f$ define

$$g_{L^f}(w) = \begin{cases} \max_{h \in H^f} \{a_h^f w + b_h^f\}, & w \leq W^f, \\ +\infty, & W^f < w \leq Q, \end{cases} \quad (3)$$

and, for the set of labels \mathcal{L}_1^f , define

$$\bar{g}_{\mathcal{L}_1^f}(w) = \min_{L^f \in \mathcal{L}_1^f} \{g_{L^f}(w)\}.$$

Function $\bar{g}_{\mathcal{L}_1^f}(\cdot)$ computes the lower envelope of the set of reduced cost functions associated with the label set \mathcal{L}_1^f . The following dominance can be used to reduce the number of labels.

Dominance 1 (Forward Set-Dominance). Label L_1^f is dominated by the label set \mathcal{L}_1^f if $g_{L_1^f}(w) \geq \bar{g}_{\mathcal{L}_1^f}(w)$, $\forall w, 0 \leq w \leq Q$.

Proof. The proof is provided in the e-companion to this paper (see Section EC.1.3 in the e-companion). \square

Figure 1. (Color online) Illustration of the Forward Set-Dominance Defined in Dominance 1

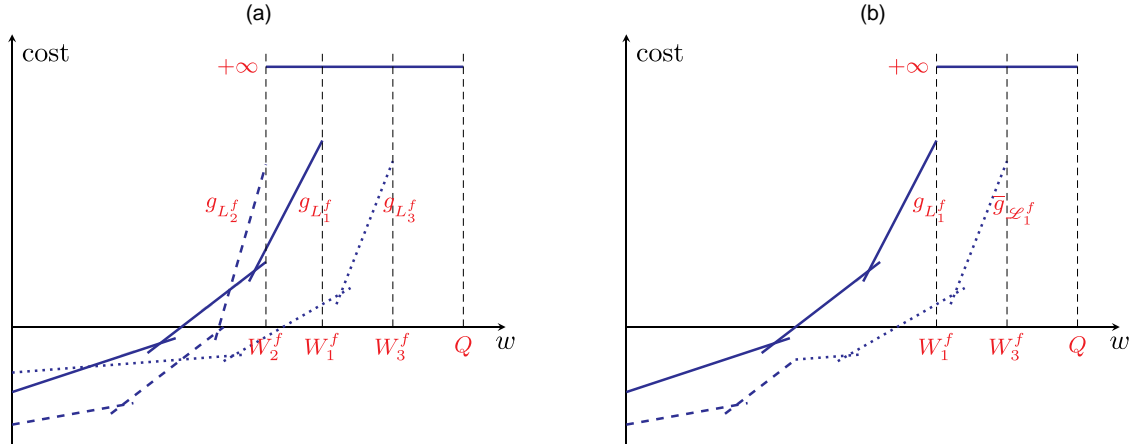


Figure 1 illustrates the use of Dominance 1. Figure 1(a) depicts functions $g_{L_x^f}(\cdot)$, $x = 1, 2, 3$, associated with three labels, L_1^f , L_2^f , and L_3^f , respectively, with $\mathcal{L}_1^f = \{L_2^f, L_3^f\}$. The figure shows that label L_1^f is dominated by neither label L_2^f nor label L_3^f in the interval $[0, W_1^f]$. The plot of function $\bar{g}_{\mathcal{L}_1^f}(w)$ given by Figure 1(b) clearly shows that the function is always inferior to function $g_{L_1^f}(\cdot)$ in the interval $[0, W_1^f]$; thus, label L_1^f is dominated by the combination of labels L_2^f and L_3^f , and can be safely discarded.

Forward labels are generated by starting from the initial label $L^f = (0, \{0\}, 0, N, 0, \{(0, -v_0)\})$. In addition, as described for bidirectional labeling by Righini and Salani (2006), time is considered to be the critical resource, and labels with a capacity consumption larger than or equal to $\lceil T/2 \rceil$ are not generated.

To check the dominance rule, we associate with a pair (i, s) , $i \in N$, $1 \leq s \leq T$, a bucket $\mathcal{F}_i(s)$ containing all nondominated labels $L^f = (v^f, s^f, \{(a_h^f, b_h^f)\}_{h \in H^f})$, with $v^f = i$ and $s^f = s$. Each bucket $\mathcal{F}_i(s)$ has an associated function $\bar{g}_{\mathcal{F}_i(s)}(w) = \min_{L^f \in \mathcal{F}_i(s)} \{g_{L^f}(w)\}$ (see also (3)), where $\mathcal{L}(s) = \{L^f : L^f \in \mathcal{F}_i(s'), s' \leq s\}$. Whenever a new label L^f is generated, the label is checked for Dominance 1 on bucket $\mathcal{F}_{v^f}(s^f)$; that is, if $g_{L^f}(w) \geq \bar{g}_{\mathcal{F}_{v^f}(s^f)}, \forall w, 0 \leq w \leq Q$, then label L^f is dominated. If the label is nondominated, then it is inserted in bucket $\mathcal{F}_{v^f}(s^f)$, and the functions of bucket $\mathcal{F}_{v^f}(s^f)$ and of all buckets $\mathcal{F}_{v^f}(s), s > s^f$ are updated as follows:

$$\bar{g}_{\mathcal{F}_{v^f}(s)}(w) = \min\{\bar{g}_{\mathcal{F}_{v^f}(s)}(w), g_{L^f}(w)\}, \forall w, 0 \leq w \leq Q, s \geq s^f.$$

Labels in the updated buckets, which are now dominated due to the addition of the new label, are removed from the corresponding buckets.

Below, we describe the backward extension following the scheme used for the forward one.

4.2.2. Backward Extension. A backward path $\bar{P} = (i_s, i_{s-1}, \dots, i_1, i_0 = n')$ is represented by a label L^b associated with the first vertex i_s , where $L^b = (v^b, s^b, V^b, W^b, \{(a_h^b, b_h^b)\}_{h \in H^b})$ is composed of the following information:

- $v^b \in V$: the first vertex visited.
- s^b : the total duration.
- $V^b \subseteq V$: the set of vertices that could be appended before the first vertex v^b .
- W^b : the maximum amount of product required by the delivery customers of the path.
- $\{(a_h^b, b_h^b)\}_{h \in H^b}$: the representation of the piecewise-linear convex nonincreasing function.

A backward label L_1^b with $j = v_1^b$ can be extended to a vertex $i \in V_1^b$, yielding a new label L_2^b based on the following extension functions. We have two cases (see also the proof of Theorem 2):

(i) $i \in N^-$ (pickup customer). The components of label L_2^b are computed as follows:

- $v_2^b = i$.
- $s_2^b = s_1^b + t_{ij}$.
- $V_2^b = V_1^b \setminus \{l : (j, l) \in A, s_1^b + t_{li} + t_{0l} > T\} \setminus \{i\}$.
- $W_2^b = \min\{Q, W_1^b - q_i\}$.
- $\{(a_h^b, b_h^b)\}_{h \in H_2^b} = \{(a_h^b, b_h^b + d_{ij})\}_{h \in H_1^b} \cup \{(a_h^b, b_h^b + (a_h^b - \mu_i)q_i + d_{ij})\}_{h \in H_1^{b+} \cup \{(\mu_i, \max\{z_1^*, \max_{h \in H_1^{b-}} \{(a_h^b - \mu_i)W_1^b + b_h^b\}, \max_{h \in H_1^{b+}} \{b_h^b\} + d_{ij})\}}$,

where $H_1^{b-} = \{h \in H_1^b : a_h^b - \mu_i < 0\}$, $H_1^{b+} = \{h \in H_1^b : a_h^b - \mu_i \geq 0\}$, and $z_1^* = \min_{h \in H_1^b} \{a_h^b x - b_h^b\}$.

(ii) $i \in N^+$ (delivery customer). Components v_2^b, s_2^b and V_2^b are computed as for the previous case, whereas the remaining components are updated as follows:

- $W_2^b = W_1^b$.
- $\{(a_h^b, b_h^b)\}_{h \in H_2^b} = \{(a_h^b, b_h^b + (a_h^b - \mu_i)q_i + d_{ij})\}_{h \in H_1^{b-} \cup \{(a_h^b, b_h^b + d_{ij})\}_{h \in H_1^{b+} \cup \{(-\mu_i, \max\{z_1^*, \max_{h \in H_1^{b-}} \{(a_h^b - \mu_i)W_1^b + b_h^b\} + d_{ij})\}}$,

where H_1^{b-} , H_1^{b+} , and z_1^* are defined as in the previous case.

Backward labels are generated by starting from the initial label $L^b = (n', \{n'\}, 0, N, 0, \{(0, 0)\})$, and labels with a capacity consumption larger than or equal to $\lceil T/2 \rceil$ are not generated.

Similar to the forward expansion, the following set-dominance rule is also used to speed up the computation. Let $L_1^b = (v_1^b, s_1^b, V_1^b, W_1^b, \{(a_h^b, b_h^b)\}_{h \in H_1^b})$, and define \mathcal{L}_1^b to be the set of labels $L^b = (v^b, s^b, V^b, W^b, \{(a_h^b, b_h^b)\}_{h \in H^b})$ such that $v^b = v_1^b$, $s^b \geq s_1^b$, and $V_1^b \subseteq V^b$.

Dominance 2 (Backward Set-Dominance). Label L_1^b is dominated by the label set \mathcal{L}_1^b if $g_{L_1^b}(w) \geq \bar{g}_{\mathcal{L}_1^b}(w)$, $\forall w$, $0 \leq w \leq Q$.

We omit the corresponding proof, it being similar to the proof of Dominance 1.

4.2.3. Joining Forward and Backward States. The set of forward and backward labels generated can be combined to form complete routes as follows.

A forward path $P = (i_0 = 0, i_1, \dots, i_{s-1}, i_s)$ associated with label $L^f = (v^f, s^f, V^f, W^f, \{(a_h^f, b_h^f)\}_{h \in H^f})$ with $i = i_s$ can be combined with a backward path $\bar{P} = (j_l, j_{l-1}, \dots, j_1, j_0 = n')$ associated with label $L^b = (v^b, s^b, V^b, W^b, \{(a_h^b, b_h^b)\}_{h \in H^b})$ with $j = j_l$ if the total duration is less than or equal to T ; that is, $s^f + t_{ij} + s^b \leq T$.

The reduced cost function of the resulting route $R = (0, i_1, \dots, i_{s-1}, i, j, j_{l-1}, \dots, j_1, n')$ can be computed as

$$\bar{c}(R, w) = \max_{h \in H^f} \{a_h^f w + b_h^f\} + d_{ij} + \max_{h \in H^b} \{a_h^b w + b_h^b\},$$

where $0 \leq w \leq \min\{W^f, W^b\}$ is the demand delivered by the partial forward path P associated with label L^f to the backward path \bar{P} associated with label L^b . The value w^* minimizing function $\bar{c}(R, w)$ can therefore be computed as

$$w^* = \arg \min_{0 \leq w \leq \min\{W^f, W^b\}} \{\bar{c}(R, w)\}.$$

The set of quantities $\{l_{i_1}, \dots, l_{j_1}\}$ associated with route R and value w^* can be determined by a backtracking procedure starting with the optimal value w^* applied to paths P and \bar{P} , as described below.

For the forward path $P = (i_0 = 0, i_1, \dots, i_{s-1}, i_s)$ and associated label $L^f = (v^f, s^f, V^f, W^f, \{(a_h^f, b_h^f)\}_{h \in H^f})$, let $L_p^f = (v_p^f, s_p^f, V_p^f, W_p^f, \{(a_h^f, b_h^f)\}_{h \in H_p^f})$ be the label immediately preceding label L^f , and let $P' = (i_0 = 0, i_1, \dots, i_{s-1})$ be the corresponding forward path. The backtracking procedure considers the following two cases:

(i) $i_s \in N^+$ (pickup customer). We have

$$\begin{aligned} \bar{c}_p^f(w^*) &= \min_{\max\{0, w^* - q_{i_s}\} \leq x \leq \min\{W_p^f, w^*\}} \\ &\quad \{\bar{c}_{p'}^f(x) - \mu_{i_s}(w^* - x) + d_{i_{s-1}i_s}\}, \\ &= \min_{\max\{0, w^* - q_{i_s}\} \leq x \leq \min\{W_p^f, w^*\}} \\ &\quad \left\{ \max_{h \in H_p^f} \{(a_h^f + \mu_{i_s})x + b_h^f\} - \mu_{i_s}w^* + d_{i_{s-1}i_s} \right\}. \end{aligned}$$

Define $f(x) = \max_{h \in H_p^f} \{(a_h^f + \mu_{i_s})x + b_h^f\} - \mu_{i_s}w^* + d_{i_{s-1}i_s}$, $H_p^{f+} = \{h \in H_p^f : a_h + \mu_{i_s} \geq 0\}$, and $H_p^{f-} = \{h \in H_p^f : a_h + \mu_{i_s} < 0\}$. Let (x_{\min}, z_{\min}) be the intersection point of the last line in H_p^{f-} and the first line in H_p^{f+} . We set $x_{\min} = -\infty$ and $z_{\min} = -\infty$ if $H_p^{f-} = \emptyset$ or $H_p^{f+} = \emptyset$. To compute the demand $l_{i_s} = w^* - x^*$ associated with i_s , we have the following three cases, as shown by Figure 2:

- (a) If $x_{\min} < \max\{0, w^* - q_{i_s}\}$, then $x^* = \max\{0, w^* - q_{i_s}\}$.
- (b) If $x_{\min} > \min\{W_p^f, w^*\}$, then $x^* = \min\{W_p^f, w^*\}$.
- (c) If $\max\{0, w^* - q_{i_s}\} \leq x_{\min} \leq \min\{W_p^f, w^*\}$, then $x^* = x_{\min}$.

(ii) $i_s \in N^-$ (delivery customer). We have

$$\begin{aligned} \bar{c}_p^f(w^*) &= \min_{w^* \leq x \leq \min\{W_p^f, w^* - q_{i_s}\}} \\ &\quad \{\bar{c}_{p'}^f(x) - \mu_{i_s}(x - w^*) + d_{i_{s-1}i_s}\}, \\ &= \min_{w^* \leq x \leq \min\{W_p^f, w^* - q_{i_s}\}} \\ &\quad \left\{ \max_{h \in H_p^f} \{(a_h^f - \mu_{i_s})x + b_h^f\} + \mu_{i_s}w^* + d_{i_{s-1}i_s} \right\}. \end{aligned}$$

Let $H_p^{f+} = \{h \in H_p^f : a_h - \mu_{i_s} \geq 0\}$ and $H_p^{f-} = \{h \in H_p^f : a_h - \mu_{i_s} < 0\}$. To compute the demand $l_{i_s} = x^* - w^*$, we have the following three cases:

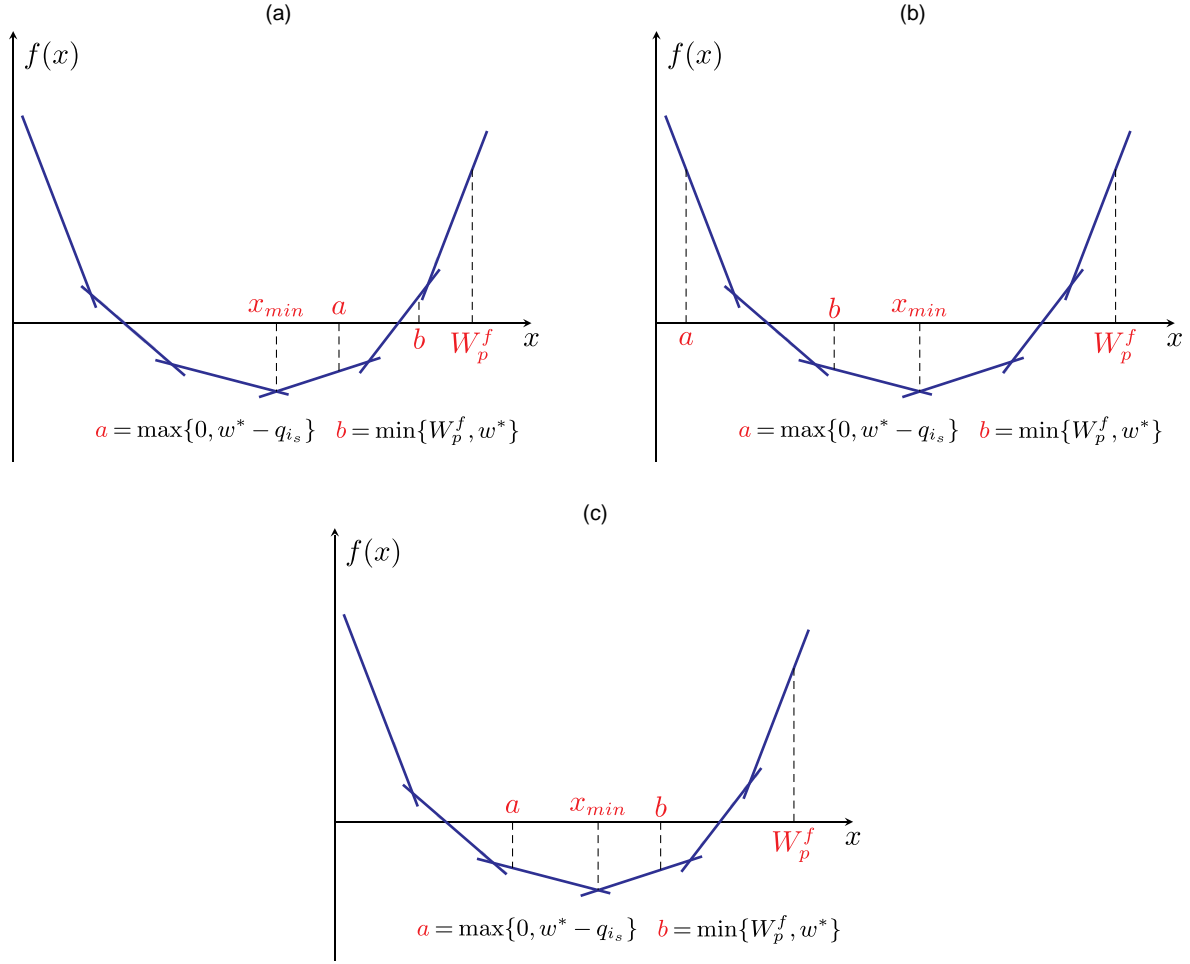
- (a) If $x_{\min} < w^*$, then $x^* = w^*$.
- (b) If $x_{\min} > \min\{W_p^f, w^* - q_{i_s}\}$, then $x^* = \min\{W_p^f, w^* - q_{i_s}\}$.
- (c) If $w^* \leq x_{\min} \leq \min\{W_p^f, w^* - q_{i_s}\}$, then $x^* = x_{\min}$.

For the backward path $\bar{P} = (j_l, j_{l-1}, \dots, j_1, j_0 = n')$ and associated label $L^b = (v^b, s^b, V^b, W^b, \{(a_h^b, b_h^b)\}_{h \in H^b})$, let $L_p^b = (v_p^b, s_p^b, V_p^b, W_p^b, \{(a_h^b, b_h^b)\}_{h \in H_p^b})$ be the label immediately preceding label L^b , and let $\bar{P}' = (j_{l-1}, \dots, j_1, j_0 = n')$ be the corresponding backward path. The backtracking procedure considers the following two cases:

(i) $j_l \in N^+$ (pickup customer). We have

$$\begin{aligned} \bar{c}_p^b(w^*) &= \min_{w^* \leq x \leq \min\{W_p^b, w^* + q_{j_l}\}} \\ &\quad \{\bar{c}_{p'}^b(x) - \mu_{j_l}(x - w^*) + d_{j_{l-1}j_l}\}, \\ &= \min_{w^* \leq x \leq \min\{W_p^b, w^* + q_{j_l}\}} \\ &\quad \left\{ \max_{h \in H_p^b} \{(a_h^b - \mu_{j_l})x + b_h^b\} + \mu_{j_l}w^* + d_{j_{l-1}j_l} \right\}. \end{aligned}$$

Figure 2. (Color online) Computing the Optimal Value x^*



Let $H_p^{b+} = \{h \in H_p^b : a_h - \mu_{j_l} \geq 0\}$ and $H_p^{b-} = \{h \in H_p^b : a_h - \mu_{j_l} < 0\}$. We have the following three cases:

(a) If $x_{min} < w^*$, then $x^* = w^*$.

(b) If $x_{min} > \min\{W_p^b, w^* + q_{j_l}\}$, then $x^* = \min\{W_p^b, w^* + q_{j_l}\}$.

(c) If $w^* \leq x_{min} \leq \min\{W_p^b, w^* + q_{j_l}\}$, then $x^* = x_{min}$.

The demand l_{i_l} is set equal to $x^* - w^*$.

(ii) $j_l \in N^-$ (delivery customer). We have

$$\begin{aligned} \bar{c}_p^b(w^*) &= \min_{\max\{0, w^* + q_{j_l}\} \leq x \leq \min\{W_p^b, w^*\}} \\ &\quad \{\bar{c}_p^b(x) - \mu_{j_l}(w^* - x) + d_{j_l j_{l-1}}\}, \\ &= \min_{\max\{0, w^* + q_{j_l}\} \leq x \leq \min\{W_p^b, w^*\}} \\ &\quad \left\{ \max_{h \in H_p^b} \{(a_h^b + \mu_{j_l})x + b_h^b\} - \mu_{j_l}w^* + d_{j_l j_{l-1}} \right\}. \end{aligned}$$

Let $H_p^{b+} = \{h \in H_p^b : a_h + \mu_{j_l} \geq 0\}$ and $H_p^{b-} = \{h \in H_p^b : a_h + \mu_{j_l} < 0\}$. We have the following three cases:

(a) If $x_{min} < \max\{0, w^* + q_{j_l}\}$, then $x^* = \max\{0, w^* + q_{j_l}\}$.

(b) If $x_{min} > \min\{W_p^b, w^*\}$, then $x^* = \min\{W_p^b, w^*\}$.

(c) If $\max\{0, w^* + q_{j_l}\} \leq x_{min} \leq \min\{W_p^b, w^*\}$, then $x^* = x_{min}$.

The demand l_{i_l} is set equal to $w^* - x^*$.

The minimum reduced cost among all complete routes is the optimal solution value of the pricing subproblem.

5. An Exact Algorithm for the SPDVRP

In this section, we describe an exact algorithm for the SPDVRP based on a BPC algorithm. The reader is referred to Costa et al. (2019) and Pecin et al. (2017) for a recent review about the main methodological and modeling contributions made over the years on BPC algorithms for routing problems, and for an exact BPC algorithm that includes state-of-the-art features for the VRPTW, respectively.

5.1. Column-and-Row Generation

At a given node of the branch-and-bound search tree, the LP-relaxation corresponds to formulation \overline{LPB} , augmented by the applicable branching decisions and the following valid inequalities.

Given a route $r \in \mathcal{R}$ and a pattern index $p \in \mathcal{P}_r$, coefficient β_{ir} is a nonnegative positive integer representing the number of times that route R_r visits vertex i (we assume that $\beta_{0r} = \beta_{n'r} = 1$), and we let set \mathcal{S} be defined as $\mathcal{S} \subseteq \{S : S \subseteq N, |S| \geq 2\}$.

Formulation \overline{LPB} can be strengthened with the following valid inequalities called capacity cuts (CCs) that are based on similar inequalities designed for the CVRP (see, e.g., Baldacci et al. 2012), where the set of customers comprises only delivery customers and split deliveries are not allowed.

The CCs for the CVRP assume that, given a subset $S \in \mathcal{S}$ of customers, one can estimate a lower bound $k(S)$ on the number of vehicles required to serve all customers in S . In this case, at least $k(S)$ paths must enter S in a feasible solution. Let $(x_{ij})_{(i,j) \in A}$ be arc flow variables. The inequality for the CVRP is as follows:

$$\sum_{i \in V \setminus S, j \in S} x_{ij} \geq k(S),$$

where the left-hand-side of the inequality is the total flow entering S . Given a solution θ_{pr} of formulation \overline{LPB} , the arc flow variables can be computed using the following expression:

$$x_{ij} = \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \gamma_{ijr} \theta_{pr},$$

and, therefore, by setting $k(S) = \max \left\{ 1, \left\lceil \frac{\sum_{i \in S} q_{il}}{Q} \right\rceil \right\}$ we derive the following valid inequalities:

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \rho_r(S) \theta_{pr} \geq k(S), \quad \forall S \in \mathcal{S}, \quad (4)$$

where $\rho_r(S) = \sum_{(i,j) \in A: i \in S, j \in V \setminus S} \gamma_{ijr}$. To separate CCs, we use partial enumeration and extended shrinking separation heuristics from the literature (Desaulniers 2010, Archetti et al. 2011b, Luo et al. 2021).

The LP-relaxation is solved in a column-and-row generation fashion, where, at each iteration, the primal simplex algorithm is used to solve the RMP and to provide a primal and a dual solution. Then the pricing subproblem (*column generation*) is solved using the label-setting algorithm in order to find negative reduced cost columns or variables. If no negative reduced cost columns can be found, then the current primal solution is optimal for the master problem. Otherwise, one or several negative reduced cost columns are added to the RMP before beginning a new iteration. If the current primal solution is optimal, then the valid inequalities in (4) are separated in a cutting plane fashion (*row generation*). The cutting plane algorithm terminates when no additional valid inequalities are identified by the separation algorithms, and a new iteration of the column-and-row generation is executed. The lower-bound computation stops when both the column and the row generation algorithms

terminate without having found new columns/rows to be added to the RMP.

In the computational experiments reported in Section 6, at each column generation iteration we identify at most 50 columns with negative reduced cost. In the following, we describe how the RMP is initialized at the root node of the enumeration tree, the branching scheme adopted in the algorithm, how the CCs and the branching rules are handled in the pricing algorithm, and the computation of primal bounds.

5.2. Initializing the RMP

The set of CCs \mathcal{S} is initialized as the empty set, and the initial columns forming the RMP are computed by means of a constructive algorithm, of which details are given in Algorithm 1.

In the algorithm, S_P and S_D denote the sets of pickup and delivery vertices, respectively, \bar{q}_i represents the residual demand of customer $i \in N$, and ld_i is the load collected from or delivered to customer $i \in N$. In addition, δ_i represents the load that has been collected from $i \in S_P$ and has not been delivered yet. The final set of routes, with the corresponding demand patterns built by the algorithm and used to initialize the RMP, is denoted by \mathcal{B} .

At each main iteration of the algorithm, the pickup customer having the maximum residual demand \bar{q}_i is selected to initialize a new route with the corresponding demand pattern. The route is then expanded with delivery customers until the load that has been collected from i (and has not been delivered yet) is fully distributed. The algorithm terminates whenever the total demand of the pickup customers has been delivered and, since $\sum_{i \in V} q_i = 0$, also the total demand of the delivery customers has been satisfied.

Algorithm 1 (Initialization of the RMP)

- 1: Set $\mathcal{B} \leftarrow \emptyset$, $S_P \leftarrow N^+$, $S_D \leftarrow N^-$, $\bar{q}_i \leftarrow q_i$, $\forall i \in N$, and $\mathcal{L} \leftarrow \emptyset$;
- 2: **while** $S_P \neq \emptyset$ **do**
- 3: Select $i \in S_P$ with the largest \bar{q}_i and define the single-customer route $R \leftarrow (0, i, n')$;
- 4: $ld_i \leftarrow \min\{Q, \bar{q}_i\}$, $\delta_i \leftarrow ld_i$, $\bar{q}_i \leftarrow \bar{q}_i - ld_i$;
- 5: **if** $\bar{q}_i = 0$ **then**
- 6: $S_P \leftarrow S_P \setminus \{i\}$;
- 7: **end if**
- 8: **while** $\delta_i > 0$ **do**
- 9: Select $j \in S_D$ as the nearest customers to the last customer of route R ;
- 10: $ld_j \leftarrow \max\{-\delta_i, \bar{q}_j\}$, $\delta_i \leftarrow \delta_i + ld_j$, $\bar{q}_j \leftarrow \bar{q}_j - ld_j$;
- 11: **if** $\bar{q}_j = 0$ **then**
- 12: $S_D \leftarrow S_D \setminus \{j\}$;
- 13: **end if**
- 14: Append j after the last customer of route R ;
- 15: **end while**
- 16: $\mathcal{B} \leftarrow \mathcal{B} \cup \{R\}$;
- 17: **end while**

The routes and corresponding demand patterns computed by Algorithm 1 do not necessarily define a feasible solution of the RMP, since the constraint in (1c) on the maximum number of vehicles available can be violated. To find an initial basic feasible solution of the RMP, we add a single artificial variable with a large positive cost and with an associated column consisting of all ones for entries in correspondence of constraints (1b) and (1c). This artificial variable ensures that a feasible solution to the LP-relaxation exists.

5.3. Branching Strategy

The branching strategy adopted is based on the strategy described by Desaulniers (2010) for the SDVRPTW and uses four types of branching decisions defined as follows.

Let $\bar{\theta}$ be an optimal solution of the RMP. The following four types of branching decisions are considered in sequence, and if a branching decision cannot be applied, then the next decision in the sequence is considered in order to perform branching. For any type of decision made, two child nodes are created by performing dichotomic branching.

(1) *Number of vehicles used.* Define $\bar{k} = \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \bar{\theta}_{pr}$. If \bar{k} is fractional, then we branch on the value of \bar{k} ; that is, on a first branch we impose $\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \bar{\theta}_{pr} \geq \lceil \bar{k} \rceil$, whereas on a second branch we impose $\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \bar{\theta}_{pr} \leq \lfloor \bar{k} \rfloor$.

(2) *Number of vehicles visiting each customer.* Define $\bar{\sigma}_i = \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \beta_{ir} \bar{\theta}_{pr}$, $\forall i \in N$. If there exists at least one customer $i \in N$ such that $\bar{\sigma}_i$ is fractional, then we select the customer i^* for which the fractional part of $\bar{\sigma}_i$ is the closest to 0.5. On a first branch, we impose $\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \beta_{i^*r} \bar{\theta}_{pr} \geq \lceil \bar{\sigma}_{i^*} \rceil$, whereas on a second branch, we impose $\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \beta_{i^*r} \bar{\theta}_{pr} \leq \lfloor \bar{\sigma}_{i^*} \rfloor$.

(3) *Branching on arcs.* Let $\bar{\omega}_{ij} = \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \gamma_{ijr} \bar{\theta}_{pr}$. We consider the following two cases, which are considered in sequence.

(i) Let $A^+ = \{(i, j) : (i, j) \in A, i, j \in N^+\}$ and $A^- = \{(i, j) : (i, j) \in A, i, j \in N^-\}$. It can be shown that, based on a similar property arising for the SDVRP (Dror and Trudeau 1989), in any optimal SPDVRP solution, any arc $(i, j) \in A^+ \cup A^-$ can be used at most once. Hence, if there exists at least one arc $(i, j) \in A^+ \cup A^-$ such that $\bar{\omega}_{ij}$ is fractional, then we select the arc (i^*, j^*) for which the fractional part of (i, j) is the closest to 0.5. On a first branch, we impose $\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \gamma_{i^*j^*r} \bar{\theta}_{pr} = 0$, and on a second branch we impose $\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \gamma_{i^*j^*r} \bar{\theta}_{pr} = 1$.

(ii) If there exists at least one arc $(i, j) \in A \setminus (A^+ \cup A^-)$ such that $\bar{\omega}_{ij}$ is fractional, then we select the arc (i^*, j^*) as before, and we impose the two branches

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \gamma_{i^*j^*r} \bar{\theta}_{pr} \geq \lceil \bar{\omega}_{i^*j^*} \rceil \quad \text{and} \quad \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} \gamma_{i^*j^*r} \bar{\theta}_{pr} \leq \lfloor \bar{\omega}_{i^*j^*} \rfloor.$$

(4) *Branching on two consecutive arcs.* Let $\bar{\omega}_{ijl} = \sum_{r \in \mathcal{R}_{ijl}} \sum_{p \in \mathcal{P}_r} \bar{\theta}_{pr}$, where \mathcal{R}_{ijl} denotes the index set of routes containing arc (j, l) immediately after arc (i, j) . If there exist two arcs (i, j) and (j, l) such that $\bar{\omega}_{ijl}$ is fractional, and either (i, j) or (j, l) belong to $A^+ \cup A^-$, then we select (i^*, j^*) and (j^*, l^*) such that $\bar{\omega}_{i^*j^*l^*}$ is the closest to 0.5, and on a first branch we impose $\sum_{r \in \mathcal{R}_{i^*j^*l^*}} \sum_{p \in \mathcal{P}_r} \bar{\theta}_{pr} = 0$; that is, the use of arc (j^*, l^*) immediately after arc (i^*, j^*) is forbidden, and on a second branch we impose $\sum_{r \in \mathcal{R}_{i^*j^*l^*}} \sum_{p \in \mathcal{P}_r} \bar{\theta}_{pr} = 0$, $\forall (j^*, l), l \neq l^*, \forall (j^*, l) \in A$; that is, if arc (i^*, j^*) is used, then arc (j^*, l^*) must also be used.

The above branching decisions are not sufficient to fulfill the integrality requirements of formulation \overline{PB} , as shown by the example reported in the e-companion to this paper (see Section EC.2). If none of the decisions (1)–(4) can be applied at a given node of the enumeration tree, then we solve the SPDVRP subproblem associated with the node by means of a BC algorithm based on a three-index vehicle flow formulation (see Section EC.3 in the e-companion) strengthened by the branching constraints associated with the node. In the computational experiments reported on in Section 6, the above four branching rules were sufficient to impose the integrality requirements, and the BC algorithm was therefore never applied.

To choose a node-selection rule, we performed some preliminary experiments with different rules and, based on these results, we adopted the *best-first strategy*.

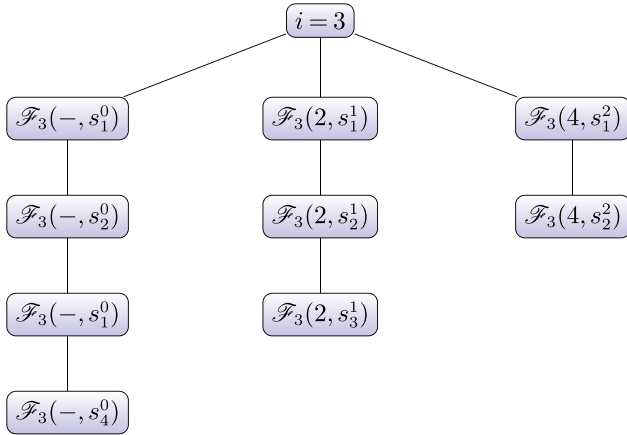
5.4. Handling the CCs and the Branching Rules in the Pricing Algorithm

According to the classification of Poggi and Uchoa (2003), the CCs are called *robust cuts* because they do not increase the complexity of the pricing problem. Indeed, let $v_S \geq 0$, $\forall S \in \mathcal{S}$, be the dual variables associated with the constraints in (4). The dual variables of the CCs can be easily considered in the label-setting algorithm by defining the modified arc costs \bar{d}_{ij} as follows,

$$\bar{d}_{ij} = d_{ij} - \sum_{S \in \mathcal{S}_{ij}} v_S, \quad \forall (i, j) \in A,$$

where $\mathcal{S}_{ij} = \{S \in \mathcal{S} : (i, j) \in A, i \in S, j \in V \setminus S\}$, and by substituting arc costs d_{ij} with the modified costs \bar{d}_{ij} in the label-setting algorithm described in Section 4.2.

Regarding the branching rules, branching rules (1)–(3) preserve the structure of the pricing problem; that is, the corresponding decisions can be easily handled in the label-setting algorithm, whereas branching rule (4) requires modifying the label-setting algorithm to impose suitable extensions based on the corresponding branching constraints. Indeed, regarding rule (4), the procedure used to check the dominance rules described in Section

Figure 3. (Color online) Example of a Bucket Tree

4.2 must be modified to ensure that a label generated as a result of a branching constraint on arc (i, j) can only dominate labels generated via label extensions along arc (i, j) .

Let Π_i be the set of vertices such that arcs (j, i) for $j \in \Pi_i$ have been selected as arcs in branching decision (4). The labels ending at customer i are decomposed in buckets $\mathcal{F}_i(j, s)$ for $\forall j \in \Pi_i$ and an additional bucket $\mathcal{F}_i(-, s)$, which are organized in a bucket tree structure. A label $L^f = (v^f, s^f, \{(a_h^f, b_h^f)\}_{h \in H^f})$, with $v^f = i$ and predecessor vertex j , is checked for dominance (see Section 4.2) in bucket $\mathcal{F}_i(j, s^f)$ if $j \in \Pi_i$; otherwise, bucket $\mathcal{F}_i(-, s^f)$ is selected. Figure 3 gives an example of a bucket tree for labels ending at customer $i = 3$ with $\Pi_3 = \{2, 4\}$, where $s_1^0 < s_2^0 < s_3^0 < s_4^0$, $s_1^1 < s_2^1 < s_3^1$ and $s_1^2 < s_2^2$.

5.5. Primal Heuristic

As shown by previous works (Joncour et al. 2010, Sadykov and Vanderbeck 2013, Sadykov et al. 2019, Wei et al. 2020), column-generation-based heuristics can help to improve the performance significantly. More specifically, Joncour et al. (2010) described a generic solution framework based on a depth-first heuristic search in the BP tree, named the *diving heuristic*. In the framework proposed by Joncour et al. (2010), the solution obtained through the initial depth-first exploration of the tree is considered as a reference incumbent solution, and it is further explored using limited backtracking as a diversification mechanism and limited discrepancy search. For the SPDVRP, we designed a diving heuristic based on a BP tree, where we adopted the branching and search strategies described below.

At each BP node, the master is solved by column generation where the separation algorithm of the valid inequalities in (4) is disabled. The BP enumeration is driven by a binary branching by first selecting a variable θ_{pr} whose fractional value θ_{pr}^* is as close as possible to 0.5, and ties are broken by selecting the variable having

the minimum cost. Then two branches are generated by adding constraints $\theta_{pr} \geq \lceil \theta_{pr}^* \rceil$ (left branch) or $\theta_{pr} \leq \lfloor \theta_{pr}^* \rfloor$ (right branch) to the RMP. In the two branches, the demand pattern α_{ipr} associated with variable θ_{pr} is eventually modified to ensure that, for each vertex I of route R , the demand $|\alpha_{ipr}|v$ is less than or equal to $|q_i|$, where v is either equal to $\lceil \theta_{pr}^* \rceil$ or $\lfloor \theta_{pr}^* \rfloor$, thus ensuring the feasibility of the resulting RMP. The pricing algorithm is not modified to include the branching constraints, and if a column composing the RMP is regenerated during the solution of the pricing problem, then the column is not added to the RMP.

The tree is explored using a modified depth-first strategy, where we restrict the search to a limited number of nodes. More specifically, suppose that (n_1, \dots, n_i) is a node sequence where n_1 is the root node and node n_{j-1} is the father of node n_j , $j = 2, \dots, i$. Thus, if n_j is the left child node of n_{j-1} , $j = 2, \dots, i$, then the next node explored by the algorithm (i.e., node n_{i+1}) is the right child of n_i ; otherwise, n_{i+1} can only be the left child node of n_i and the right child is discarded. Therefore, the search is restricted by choosing at most one right child node in the path from the root node to any visited node.

Based on preliminary experiments, we found it computationally convenient to use the primal heuristic only at the root node of the BPC algorithm.

6. Computational Experiments

This section reports on the computational results of the exact algorithm described in Section 5, hereafter called EXM.

Method EXM was coded in Java, and all experiments were conducted on a personal computer equipped with an Intel Core i7-6700 3.40-GHz CPU and 32 GB of RAM, running under Windows 8.1 operating system. The CPLEX 12.6.4 callable library (IBM CPLEX 2019) was used as the linear programming solver.

The next section reports on the description of the instances used in our numerical experiments, whereas Section 6.2 gives a summary of the results obtained. We conclude this section (see Section 6.3) with an analysis of the performance of the pricing algorithm of EXM. Additional details are given in the e-companion to this paper (see Section EC.3).

6.1. Benchmark Instances

We considered three classes of test instances, called S_A , S_B , and S_C , respectively. In classes S_A and S_C , the route-duration constraints are used to impose an upper limit on the number of stops on each route, whereas class S_C considers general route-duration constraints.

Class S_A has been obtained from the literature and contains the set of MVB instances used by Casazza et al. (2019) and Casazza et al. (2021), which were derived from the instances proposed by Chemla et al. (2013). In

these instances, the customers are randomly located in the square $[-500, 500]$ and have integer demands randomly chosen in the interval $[-10, 10]$; the depot is located at point $(0, 0)$. The travel costs are computed as the Euclidean distances, rounded to the closest integer numbers. Casazza et al. (2019) and Casazza et al. (2021) considered instances with the number of customers n in $\{10, 20, 30\}$ and with $T = 10$ (a maximum number of stops on each route is imposed). In our experiments, we also considered instances with $n = 40$ and $T = 15$. The vehicle capacity Q and the number of vehicles $|K|$ are chosen from sets $\{10, 20\}$ and $\{5, 8\}$, respectively. To model a maximum number of stops b in the SPDVRP, we set $T = b$ and $t_{ij} = 1$ for all $(i, j) \in A$. A total of 200 instances have been used, with names of the form $\langle nXY \rangle$, where X is the number of stations or customers and Y a letter identifying the type of network used in deriving the instance.

Class S_B of instances corresponds to the set of instances used by Casazza et al. (2021) and is also derived from the instances of Chemla et al. (2013) by properly defining the travel time matrices $[t_{ij}]$. More specifically, instances in the data set of Chemla et al. (2013) were considered in lexicographical order, and, for each instance, the traveling times t_{ij} were set as the Euclidean distance between vertices i and j in the subsequent instance according to the lexicographical order, thus obtaining unrelated travel times and costs. The maximum duration T was set equal to either 3,600 or 7,200. Casazza et al. (2021) considered instances with up to $n = 30$ customers, and we also considered instances involving $n = 40$ customers. A total of 100 instances were considered, and this set of instances uses the same naming convention as set S_A .

Class S_C was derived from the set of real-world instances considered by Dell'Amico et al. (2014) and based on different data sets. More specifically, the authors collected data from a seven-month period concerning the bike-sharing system of the city of Reggio Emilia, in Northern Italy, the data being drawn from Capital Bikeshare (see <https://www.capitalbikeshare.com>), as well as from Divvy, located in Chicago, USA (see <https://www.divvybikes.com>). All the instances are available at <http://www.or.unimore.it/site/home/online-resources.html>. From this set, we considered a total of 47 instances, with the number of customers ranging in $[13, 55]$ and the vehicle capacity in $[10, 30]$. For these instances, we imposed a maximum number of stops T equal to 10 or 15, and the number of available vehicles was set equal to $|K| = 8$ for all instances. A total of 94 instances were used, with names in the form of $\langle iXQ \rangle$, where i is the instance index, X the name of the corresponding city, and Q the vehicle capacity.

For the convenience of readers, we summarize the three classes of instances used and corresponding results solved by EXM at <https://github.com/huster-ljl/SPDVRP>.

6.2. Results

This section summarizes the results obtained by EXM and compares its performances with the results reported in Casazza et al. (2019) and Casazza et al. (2021) on classes S_A and S_B . The results of Casazza et al. (2019) and Casazza et al. (2021) were obtained on a PC equipped with an Intel Core i76700K CPU clocked at 4.00 GHz, thus having similar performance to the machine used for our experiments, and a time limit of three hours of computation was imposed in their experiments. In the following, we denote by C18 and C21 the algorithms of Casazza et al. (2019) and Casazza et al. (2021), respectively. All the computing times reported in this section are expressed in seconds, and a time limit of three hours was also imposed to each execution of EXM.

Table 2 summarizes the results obtained by the algorithms on the three classes of instances. For each class of instances and for each subgroup of instances in the set, the table gives the range of the number of customers (n), the number of instances in the group (ni), the values of parameters Q and T , and, for each algorithm, the number of instances solved to optimality (opt), the size of the largest instance solved to optimality in terms of number of customers (\hat{n}) and the average computing time in seconds (t) computed over all instances solved to optimality. In addition, Figure 4 gives an overview of the percentages of the number of instances solved to optimality by the different algorithms grouped by number of customers (horizontal axis) and by considering, for each algorithm, the set of instances addressed in the corresponding experiments. As already mentioned, algorithms C18 and C21 were tested on instances with up to 30 customers.

The table shows that about 70% of the instances of each class were solved to optimality by EXM within the imposed time limit. Overall, EXM solved to optimality instances with up to 41 customers. As to the set of instances considered by C18 and C21, EXM is capable of solving far more instances to optimality. A comparison of the results given in Casazza et al. (2021) about C18 and C21 with the detailed results reported on in the e-companion to this paper shows that all instances solved to optimality by C18 and C21 are also solved by EXM. It is worth noting that there are no dominance relations between C18 and C21; that is, there are instances solved by C18 within the imposed time limit that cannot be solved by C21, and vice versa.

Regarding classes S_B and S_C , Figure 4 clearly shows that EXM is capable of solving larger size instances than both C18 and C21. In particular, the detailed results show that EXM solved all but one instance involving $n = 20$ customers. Several instances with up to 40 customers were solved by EXM, thus doubling the size of the instances that can be solved.

Table 3 reports average results about the lower and upper bounds computed by EXM at the root node of the

Table 2. Summary of the Results on the Three Classes of Instances

Class	n	ni	Q	T	C18			C21			EXM		
					opt	\hat{n}	t	opt	\hat{n}	t	opt	\hat{n}	t
S_A	10-40	50	10	10	25	20	1,000.9	24	20	265.7	35	30	664.3
		50	10	15	—	—	—	—	—	—	32	30	265.5
		50	20	10	24	20	2,483.9	27	20	1,016.0	34	40	679.9
		50	20	15	—	—	—	—	—	—	33	40	368.5
S_B	10-40	50	10	3,600	—	—	—	21	20	319.3	33	30	531.9
		50	10	7,200	—	—	—	21	20	125.6	31	30	177.2
S_C	13-55	47	10-30	10	—	—	—	—	—	—	34	28	695.2
		47	10-30	15	—	—	—	—	—	—	32	41	1,001.1

enumeration tree. These data are not reported in Casazza et al. (2019) and Casazza et al. (2021). The table shows the average percentage deviations of the upper bound ($\%UB$) computed by the primal heuristic and of the lower bound obtained at the root node ($\%LB_r$) by the column-and-row generation procedure. The table also reports the corresponding average computing times in seconds (t_{UB} and t_r). The averages are computed separately for the instances solved to optimality and for the instances not solved to optimality. For each instance, the percentage deviation is computed as $100.0 \times x/z^*$, where x is the target value and z^* is the cost of the best solution found by EXM.

For the instances solved to optimality, EXM computes tight lower and upper bounds for the real-world instances of class S_C , whereas the bounds computed on classes S_A and S_B show that these instances are more difficult for EXM. We observed that the computation of the upper bound by the primal heuristic helps in improving the performance of EXM and, as shown by the table, the upper bounds computed can be further improved by the BPC.

Based on the results obtained, the two main parameter data that affect the computational complexity of EXM are clearly the number of customer n , as shown by Figure 4 and the detailed results, and the maximum duration T . Indeed, Table 2 shows that, for each class and for fixed values of Q , the higher the value of T , the

lesser is the number of instances solved to optimality. Both values Q and T affect the computational complexity of the pricing problem, but, as also shown by Table 2 and by the analyses reported on the parameter Q in the next section, for fixed values of T and varying values of Q , the pricing algorithm is particularly stable. In the e-companion to this paper (see Section EC.4.1), we give an overview of the use of the different branching rules adopted in EXM for classes S_A , S_B , and S_C , respectively. Finally, we also analyze the structure of the solutions obtained in the e-companion (see Section EC.4.2), with the aim of learning what motivates the differences in our results between the real-world instances of class S_C and classes S_A and S_B .

Finally, the detailed results reported in the e-companion to this paper (see Section EC.5) show that the CCs are particularly effective in strengthening the lower bound obtained from the LP-relaxation of formulation \overline{PB} .

6.3. Effectiveness of the Pricing Algorithm

In this section, we report a comparison between the pricing algorithm of EXM and our implementation of the pricing algorithm described by Casazza et al. (2019). The pricing algorithm of Casazza et al. (2019) solves the pricing problem associated with formulation PB , and we tailored it to solve the pricing problem associated with relaxation \overline{PB} (see Section 3.2).

For the comparison, we considered set S_A of instances with $Q = 10$, $T = 10$, and the following two additional sets of instances, for a total of 150 instances (50×3): (i) set S_{A-3} with $Q = 3Q$, $q_i = 3q'_i$, and $T = 10$; and (ii) set S_{B-6} with $Q = 6Q$, $q_i = 6q'_i$, and $T = 10$, where customer demands q'_i are the original demands of set S_A . The two additional instance sets were generated with the aim of evaluating the impact on the pricing algorithm for increasing values of the vehicle capacity and the customer demands.

Table 4 summarizes the results obtained on the three sets of instances. The corresponding detailed results are reported in the e-companion to this paper (see Section EC.6). For each algorithm, we computed the following average results over the different sets of instances: the time in seconds spent by the LP solver (t_{LP}), the time in

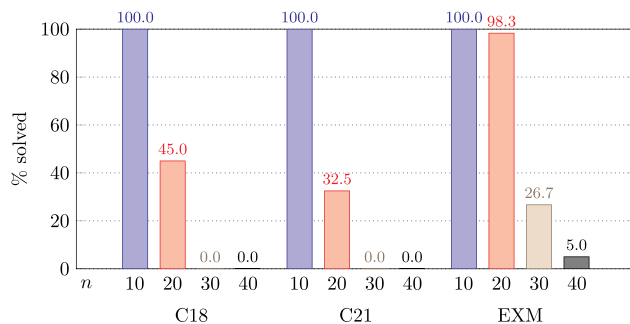
Figure 4. (Color online) Percentages of the Instances Solved to Optimality by the Different Algorithms on Classes S_A and S_B 

Table 3. Summary of Lower and Upper Bounds of EXM

Class	n	Q	T	Solved to optimality				Not solved to optimality			
				%UB	t_{UB}	%LB _r	t_r	%UB	t_{UB}	%LB _r	t_r
S_A	10-40	10	10	112.0	2.7	95.9	1.1	100.0	13.6	77.1	3.5
		10	15	120.7	3.0	98.5	1.3	104.1	43.5	78.5	13.6
		20	10	116.9	2.6	92.5	0.9	101.6	11.2	73.8	2.6
		20	15	125.9	2.8	97.1	1.0	113.4	23.6	82.2	6.0
S_B	10-40	10	3,600	112.4	3.7	96.0	1.2	101.4	49.6	79.2	8.6
		10	7,200	115.2	7.1	98.6	3.3	102.8	129.9	82.8	41.7
S_C	13-55	10-30	10	105.4	2.7	97.7	1.0	100.5	398.2	91.2	88.7
		10-30	15	104.3	4.8	98.8	1.6	101.8	208.8	92.5	41.7

seconds spent by the pricing algorithm (t_{PA}), the number of forward labels generated (fw), the number of forward labels dominated (fwd), the number of backward labels generated (bw), and the number of backward labels dominated (bwd). Table 4 shows the percentage ratios of the average values t_{LP} , t_{PA} , fw , fwd , bw , and bwd obtained by the pricing algorithm of EXM with those based on the algorithm of Casazza et al. (2019).

The table shows that, on set S_A (see column t_{PA}), our pricing algorithm is about three times faster than the algorithm based on Casazza et al. (2019). For increasing values of Q associated with sets S_{A-3} and S_{A-6} , our pricing algorithm achieves much more significant speed factors. Indeed, Table 4 and the detailed results show that, over the three sets of instances, the number of labels generated by our pricing algorithm remains quite stable, whereas the alternative pricing algorithm shows an exponential explosion of the labels generated, as also shown by the ratios over the values fw and bw .

Under the assumption that our implementation of the algorithm of Casazza et al. (2019) achieves similar performance to the implementation of Casazza et al. (2019), the results obtained clearly show that the effectiveness of EXM in solving larger instances (see Table 2) of the SPDVRP can be mainly attributed to the effectiveness of the pricing algorithm based on reduced cost functions.

In conclusion, due to the use of the reduced cost functions, our pricing algorithm requires a reduced number of labels at the cost of more complex dominance rules. However, our results show that the benefits brought about by the label reduction are greater than the loss caused by the higher complexity of our dominance rules.

Table 4. Summary of the Performance of Alternative Pricing Algorithms: Percentage Ratios of the Results Obtained by the Pricing Algorithm of EXM with Those Based on the Algorithm of Casazza et al. (2019)

	t_{LP}	t_{PA}	fw	fwd	bw	bwd
S_A	29.66	29.45	6.27	2.65	11.82	4.81
S_{A-3}	2.68	2.56	0.63	0.09	1.46	0.21
S_{A-6}	0.21	0.20	0.09	0.01	0.18	0.01

7. Conclusions and Future Research

We have presented an exact algorithm for the single-commodity split-pickup and split-delivery vehicle routing problem. The algorithm relies on a novel label-setting (pricing) algorithm, embedding reduced cost functions in the label definition, and being enhanced by new set-dominance rules. The pricing algorithm is used in a branch-price-and-cut algorithm together with additional components such as valid inequalities and a primal heuristic.

Our new exact algorithm was tested on benchmark sets from the literature and on instances derived from real-world bike-sharing problems, involving up to 55 customers. The computational results show that about 70% of the instances, involving up to 41 customers, were effectively solved to optimality by the exact algorithm, thus doubling the size of the instances that can be solved and significantly outperforming the state-of-the-art exact algorithms.

Our algorithm can be easily adapted to deal with other routing constraints, such as time-window constraints, and to solve rich variants involving, for example, a fleet of heterogeneous vehicles. However, the class of split-demand one-commodity pickup and delivery routing problems also features important variants, such as allowing preemption operations. Our future work goes in the direction of extending the current solution approach to incorporate additional important features of this class of problems.

Acknowledgments

The authors thank the anonymous reviewers and associate editor for their helpful suggestions and very thorough review of the paper. They also thank Marco Casazza for providing the instances and the corresponding details used in Casazza et al. (2021).

References

- Archetti C, Speranza MG (2008) The split delivery vehicle routing problem: A survey. Golden BL, Raghavan S, Wasil EA, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, New York), 103–122.
- Archetti C, Speranza MG (2012) Vehicle routing problems with split deliveries. *Internat. Trans. Oper. Res.* 19(1-2):3–22.

- Archetti C, Bianchessi N, Speranza MG (2011a) A column generation approach for the split delivery vehicle routing problem. *Networks* 58(4):241–254.
- Archetti C, Bianchessi N, Speranza MG (2014) Branch-and-cut algorithms for the split delivery vehicle routing problem. *Eur. J. Oper. Res.* 238(3):685–698.
- Archetti C, Bouchard M, Desaulniers G (2011b) Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Sci.* 45(3):285–298.
- Baldacci R, Christofides N, Mingozzi A (2008) An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Math. Programming* 115:351–385.
- Baldacci R, Mingozzi A, Roberti R (2012) Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *Eur. J. Oper. Res.* 218(1):1–6.
- Balinski ML, Quandt RE (1964) On an integer program for a delivery problem. *Oper. Res.* 12(2):300–304.
- Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 46(3):316–329.
- Battarra M, Cordeau J-F, Iori M (2014) Pickup-and-delivery problems for goods transportation. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications* (SIAM, Philadelphia), 161–191.
- Berbeglia G, Cordeau J-F, Gribkovskaia I, Laporte G (2007) Static pickup and delivery problems: A classification scheme and survey. *TOP* 15(1):1–31.
- Bianchessi N, Irnich S (2019) Branch-and-cut for the split delivery vehicle routing problem with time windows. *Transportation Sci.* 53(2):442–462.
- Bruck BP, Cruz F, Iori M, Subramanian A (2019) The static bike sharing rebalancing problem with forbidden temporary operations. *Transportation Sci.* 53(3):882–896.
- Bulhões T, Subramanian A, Erdoğan G, Laporte G (2018) The static bike relocation problem with multiple vehicles and visits. *Eur. J. Oper. Res.* 264(2):508–523.
- Casazza M, Ceselli A, Wolfler Calvo R (2021) A route decomposition approach for the single commodity split pickup and split delivery vehicle routing problem. *Eur. J. Oper. Res.* 289(3):897–911.
- Casazza M, Ceselli A, Chemla D, Meunier F, Wolfler Calvo R (2019) The multiple vehicle balancing problem. *Networks* 72(3):337–357.
- Chabrier A (2006) Vehicle routing problem with elementary shortest path based column generation. *Comput. Oper. Res.* 33(10):2972–2990.
- Chemla D, Meunier F, Wolfler Calvo R (2013) Bike sharing systems: Solving the static rebalancing problem. *Discrete Optim.* 10(2):120–146.
- Costa L, Contardo C, Desaulniers G (2019) Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Sci.* 53(4):946–985.
- Cruz F, Subramanian A, Bruck BP, Iori M (2017) A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. *Comput. Oper. Res.* 79:19–33.
- Dabia S, Ropke S, van Woensel T, De Kok T (2013) Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Sci.* 47(3):380–396.
- Dell’Amico M, Hadjicostantinou E, Iori M, Novellani S (2014) The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega* 45:7–19.
- Dell’Amico M, Iori M, Novellani S, Stützle T (2016) A destroy and repair algorithm for the bike sharing rebalancing problem. *Comput. Oper. Res.* 71:149–162.
- Desaulniers G (2010) Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Oper. Res.* 58(1):179–192.
- Desaulniers G, Desrosiers J, Solomon MM, eds. (2005) *Column Generation* (Springer, New York).
- Dror M, Trudeau P (1989) Savings by split delivery routing. *Transportation Sci.* 23(2):141–145.
- Erdoğan G, Battarra M, Wolfler Calvo R (2015) An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *Eur. J. Oper. Res.* 245(3):667–679.
- Erdoğan G, Laporte G, Wolfler Calvo R (2014) The static bicycle relocation problem with demand intervals. *Eur. J. Oper. Res.* 238(2):451–457.
- Espegren HM, Kristianslund J, Andersson H, Fagerholt K (2016) The static bicycle repositioning problem—literature survey and new formulation. Paias A, Ruthmair M, Voß S, eds. *Computational Logistics* (Springer, Cham, Switzerland), 337–351.
- Feillet D, Dejax P, Gendreau M, Gueguen C (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3):216–229.
- Hernández-Pérez H, Salazar-González J (2004a) A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Appl. Math.* 145(1):126–139.
- Hernández-Pérez H, Salazar-González J (2004b) Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Sci.* 38(2):245–255.
- Hernández-Pérez H, Salazar-González J (2007) The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks* 50(4):258–272.
- Hernández-Pérez H, Salazar-González J, Santos-Hernández B (2018) Heuristic algorithm for the split-demand one-commodity pickup-and-delivery travelling salesman problem. *Comput. Oper. Res.* 97:1–17.
- IBM CPLEX (2019) IBM ILOG CPLEX 12.6.4 callable library.
- Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds. *Column Generation* (Springer, New York), 33–65.
- Irnich S, Villeneuve D (2006) The shortest-path problem with resource constraints and k -cycle elimination for $k \geq 3$. *INFORMS J. Comput.* 18(3):391–406.
- Irnich S, Schneider M, Vigo D (2014) Four variants of the vehicle routing problem. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications* (SIAM, Philadelphia), 241–271.
- Izuno T, Nishi T, Yin S (2012) Column generation for split pickup and delivery vehicle routing problem for crude oil transportation. *Proc. ASME/ISCIE 2012 Internat. Sympos. Flexible Automation* (St. Louis, Missouri), 397–404.
- Joncour C, Michel S, Sadykov R, Sverdllov D, Vanderbeck F (2010) Column generation based primal heuristics. *Electronic Notes Discrete Math.* 36:695–702.
- Laporte G, Meunier F, Wolfler Calvo R (2015) Shared mobility systems. *4OR* 13(4):341–360.
- Laporte G, Meunier F, Wolfler Calvo R (2018) Shared mobility systems: An updated survey. *Ann. Oper. Res.* 271(1):105–126.
- Liberatore F, Righini G, Salani M (2011) A column generation algorithm for the vehicle routing problem with soft time windows. *4OR* 9(1):49–82.
- Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res.* 53(6):1007–1023.
- Luo Z, Qin H, Zhu W, Lim A (2017) Branch and price and cut for the split-delivery vehicle routing problem with time windows and linear weight-related cost. *Transportation Sci.* 51(2):668–687.
- Luo Z, Qin H, Cheng TCE, Wu Q, Lim A (2021) A branch-and-price-and-cut algorithm for the cable-routing problem in solar power plants. *INFORMS J. Comput.* 33(2):452–476.
- Parragh S, Doerner K, Hartl R (2008a) A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *J. Betriebswirtschaft* 58(1):21–51.
- Parragh S, Doerner K, Hartl R (2008b) A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *J. Betriebswirtschaft* 58(2):81–117.

- Pecin D, Contardo C, Desaulniers G, Uchoa E (2017) New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS J. Comput.* 29(3):489–502.
- Poggi M, Uchoa E (2003) Integer program reformulation for robust branch-and-cut-and-price algorithms. *Math. Programming Rio, Buzios, Brazil, November 9–12*, 56–61.
- Poggi M, Uchoa E (2014) New exact algorithms for the capacitated vehicle routing problem. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications* (SIAM, Philadelphia), 59–86.
- Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optim.* 3(3):255–273.
- Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks* 51(3):155–170.
- Sadykov R, Vanderbeck F (2013) Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS J. Comput.* 25(2): 244–255.
- Sadykov R, Vanderbeck F, Pessoa A, Tahiri I, Uchoa E (2019) Primal heuristics for branch and price: The assets of diving methods. *INFORMS J. Comput.* 31(2):251–267.
- Salazar-González J, Santos-Hernández B (2015) The split-demand one-commodity pickup-and-delivery travelling salesman problem. *Transportation Res. Part B Methodological.* 75: 58–73.
- Toth P, Vigo D (2014) *Vehicle Routing: Problems, Methods, and Applications* (SIAM, Philadelphia).
- Wei L, Luo Z, Baldacci R, Lim A (2020) A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS J. Comput.* 32(2):428–443.