Les Cahiers du GERAD ISSN: 0711–2440

Exact branch-price-and-cut algorithms for vehicle routing

L. Costa, C. Contardo,
G. Desaulniers

G–2018–41

June 2018

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2018
– Bibliothèque et Archives Canada, 2018

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2018
– Library and Archives Canada, 2018

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Exact branch-price-and-cut algorithms for vehicle routing

Luciano Costa [a,b]
Claudio Contardo [a,c]
Guy Desaulniers [a,b]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[c] *CIRRELT & Department of Management and Technology, ESG UQÀM, Montréal (Québec) Canada, H2X 3X2*

luciano.costa@gerad.ca
claudio.contardo@gerad.ca
guy.desaulniers@gerad.ca

June 2018
Les Cahiers du GERAD
G–2018–41

Abstract: Vehicle routing problems (VRPs) are among the most studied problems in operations research. Nowadays, the leading exact algorithms for solving many classes of VRPs are branch-price-and-cut algorithms. In this survey paper, we highlight the main methodological and modeling contributions made over the years on branch-and-price(-and-cut) algorithms for VRPs, whether they are generic or specific to a VRP variant. We focus on problems related to the classical VRP, i.e., problems in which customers must be served by several capacitated trucks, and which are not combinations of a VRP and another optimization problem.

Keywords: Branch-price-and-cut, vehicle routing, survey, generic tools, variant-specific contributions

Acknowledgments: This work was supported by Fonds de recherche du Québec – Nature et technologies (FRQNT) under the grant 181909 and the Natural Sciences and Engineering Research Council (NSERC) of Canada under the grants 2017-05683 and 2013-435824. This support is gratefully acknowledged.

1 Introduction

The vehicle routing problem (VRP) is one of the most studied problems in operations research. It was introduced by Dantzig and Ramser (1959) to solve a practical problem of delivering gasoline from a bulk terminal to service stations. Its basic version, the capacitated VRP (CVRP), consists of finding feasible routes to visit a set of customers in such a way that each customer is visited exactly once to completely satisfy its demand. A route is said to be feasible if it starts and ends at a given depot and if the sum of the demands of the visited customers does not exceed vehicle capacity. The most common objective for this problem is the minimization of the traveling costs, which is often assumed to be proportional to the total traveled distance.

Over the years, several extensions to the CVRP have been proposed. Most of them are inspired by real-life applications and consider different attributes, namely: start of service time windows at the customers, pickup and delivery requests, split deliveries, multiple depots, heterogeneous fleet, etc. For a detailed classification of the VRP variants, see Eksioglu, Volkan, and Reisman (2009), Irnich, Toth, and Vigo (2014) and Braekers, Ramaekers, and Nieuwenhuyse (2016).

VRPs have been tackled by (meta)heuristics and exact algorithms. Despite the important role played by heuristics when dealing with real-life problems, we do not review them in this survey. Most of the exact algorithms in the VRP literature rely on branch-and-bound (BB) to explore implicitly the solution space. Because the performance of the BB algorithms depends on the quality of the bounds obtained throughout the tree, it is common practice to employ some techniques to improve the quality of these bounds. BB algorithms can be combined with the generation of cutting planes, forming the so-called branch-and-cut (BC) algorithms, or with column generation (CG), resulting in branch-and-price (BP) algorithms. When both techniques are exploited simultaneously, this leads to branch-price-and-cut (BPC) algorithms.

During many years, BC algorithms were deemed the best algorithms to address VRPs (Laporte, Nobert, and Desrochers 1985, Lysgaard, Letchford, and Eglese 2004). However, since the seminal work of Desrosiers, Soumis, and Desrochers (1984), a lot of effort has been put into the development of efficient BP and BPC algorithms, which made them, nowadays, the leading algorithms for solving many classes of VRPs. BP algorithms are BB algorithms in which the linear relaxations are solved by means of CG. CG is an iterative procedure that can tackle linear programs containing a huge number of variables (see Barnhart et al. 1998, Desaulniers, Desrosiers, and Solomon 2005, Lübbecke and Desrosiers 2005). In the context of VRPs, it relies on a subproblem (called the pricing problem) to generate routes dynamically and on a master problem (MP) to select the best ones.

BPC can be seen as a generic framework to solve VRPs. In this respect, several works have proposed generic algorithmic enhancements that are applicable to most VRPs. On the other hand, tailored BPC algorithms, including specific tools/procedures to handle the particularities of a VRP variant, have also been devised. In this context, Feillet (2010) illustrates the importance of methodological studies on CG-based algorithms by pointing out some reasons that can make these algorithms hard to understand and reproduce, namely: 1) the inherent complexity associated with the methodology and the large literature in this field; 2) the variety of perspectives from which these algorithms can be explored; and 3) the lack of comprehensive descriptions of these algorithms.

To the best of our knowledge, there is no work in the literature discussing specificities of the BP/BPC algorithms designed for different classes of VRPs. The only paper providing a related analysis is that of Baldacci, Mingozzi, and Roberti (2012). Nevertheless, they limit their analysis to the CVRP and the VRP with time windows (VRPTW). The present paper is broader in the sense that it provides a survey of the methodological developments proposed for BP/BPC algorithms applied to a wide variety of VRPs. Even if several transportation problems can be modeled/addressed as routing problems, we limit our discussion to problems related to the CVRP, i.e., problems in which customers/requests must be served by several capacitated trucks, and that are not combinations of a VRP and another optimization problem. Therefore, problems such as vehicle scheduling, inventory routing, ship routing, location-routing, etc., are out of scope.

Our goal for this survey is to highlight the main methodological and modeling contributions made over the years on BP/BPC algorithms for VRPs, whether they are generic or specific to a VRP variant. Consequently, we do not provide an exhaustive review of all papers presenting a BP/BPC algorithm for a VRP. Given the large number of VRP variants covered and papers cited, we have also chosen to present and discuss no computational results.

This paper is structured as follows. The remainder of this section defines the VRPTW that will serve as our main VRP example and provides a mathematical formulation for it. Section 2 presents the main components of a basic BPC algorithm. Generic tools for improving the performance of BPC algorithms are discussed in Section 3. Contributions specific to VRP variants are reviewed in Section 4. Finally, in Section 5, we draw some conclusions.

1.1 Problem description

For the sake of simplicity, when presenting the main concepts and ingredients of BPC algorithms, the VRPTW is used as an example. The choice of this problem comes from the fact that, while it is a relatively simple problem, solving it by BPC is still quite challenging. In fact, many state-of-the-art techniques incorporated into BPC algorithms have been proposed for the VRPTW.

The VRPTW can be formally defined as follows. Let $G = (V, A)$ be a complete and directed graph, where $V = V' \cup \{0, n+1\}$ is the set of vertices and A the set of arcs. V' is the set of n customers. Vertices 0 and $n+1$ represent the depot at the start and the end of a route, respectively. Each customer $i \in V'$ is associated with a demand $q_i > 0$, a service time $s_i > 0$ and a time window $[e_i, l_i]$, with $0 \leq e_i \leq l_i$, which specifies the earliest and the latest time at which service can start at customer i. We consider $q_0 = s_0 = e_0 = q_{n+1} = s_{n+1} = e_{n+1} = 0$ and $l_0 = l_{n+1} = T$, where T represents the horizon length. The traveling cost and the traveling time associated with each arc $(i, j) \in A$ are denoted by c_{ij} and t_{ij}, respectively. Besides, an unlimited fleet of homogeneous vehicles with capacity Q is available at the depot. The VRPTW consists of designing feasible routes such that each customer $i \in V'$ is visited exactly once by a route and the sum of the costs of the routes is minimized. A feasible route corresponds to an elementary path $r = (i_0 = 0, i_1, \ldots, i_{k-1}, i_k = n+1)$ in G such that the vehicle capacity is not exceeded, i.e., $\sum_{j=0}^{k} q_{i_j} \leq Q$, and the time windows at the visited vertices are met. The latter conditions can be verified by computing recursively the start of service time t_{i_j} at every visited vertex i_j, $j \in \{0, 1, \ldots, k\}$, as follows

$$
\begin{aligned}
t_{i_0} &= e_{i_0} \\
t_{i_{j+1}} &= \max\{e_{i_{j+1}}, t_{i_j} + s_j + t_{i_{j-1}, i_j}\}, \qquad && \forall j \in \{0, 1, \ldots, k-1\},
\end{aligned}
$$

where the maximum function is used to model the possibility that a vehicle arrives before the opening of a time window and wait until its opening to start service. The time windows are met if $t_{i_j} \in [e_{i_j}, l_{i_j}]$, $\forall j \in \{0, 1, \ldots, k\}$. Given these feasibility conditions, some arcs in A can be discarded to yield arc set $A = \{(i, j) \in V \times V \mid q_i + q_j \leq Q, e_i + s_i + t_{ij} \leq l_j\} \setminus \{(0, n+1), (n+1, 0)\}$. Finally, the cost c_r of route r is given by $c_r = \sum_{j=0}^{k-1} c_{i_j, i_{j+1}}$.

The VRPTW can be formulated in terms of binary arc-flow variables x_{ij} to indicate whether a vehicle travels or not along a given arc $(i, j) \in A$ (for more details, see Desaulniers, Madsen, and Ropke 2014). Arc-flow formulations have the advantage of containing a polynomial number of variables, but they provide, in general, weak linear relaxations. An alternative way of modeling the VRPTW is by means of a set partitioning formulation (Balinski and Quandt 1964), which we present in the next section. Arc-flow and set partitioning formulations are often referred to as *compact* and *extended* formulations, respectively.

As discussed previously, the VRPTW is defined over the directed graph G. Nevertheless, other VRP variants such as the CVRP can be formulated using an undirected graph. For the sake of clarity, we will assume a directed graph G throughout this paper unless otherwise specified.

1.2 Set partitioning formulation

Let Ω be the set of all feasible routes, i.e., elementary routes that satisfy vehicle capacity and time windows. A binary parameter a_i^r specifies whether or not customer $i \in V'$ is visited on a route $r \in \Omega$. For each route $r \in \Omega$, define a binary variable λ_r which is equal to 1 if r is selected in the solution and 0 otherwise.

The VRPTW can be formulated as the following set partitioning model:

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

$$s.t. \quad \sum_{r\in\Omega} a_i^r\lambda_r = 1, \qquad \forall i \in V' \tag{2}$$

$$\lambda_r \in \{0,1\}, \qquad \forall r \in \Omega. \tag{3}$$

The objective function (1) aims at minimizing the total routing cost. Constraints (2) ensure that each customer is visited exactly once. Finally, constraints (3) define the domain of the variables.

A big advantage of formulating the VRPTW as (1)–(3) is that some of the constraints (e.g., the time windows) do not need to be explicitly expressed in the formulation: they are rather implicit from the definition of set Ω. Consequently, set partitioning formulations typically provide better lower bounds than those obtained with compact formulations. Nevertheless, they admit a huge number of variables, which makes it impossible to handle them all at once. Fortunately, BPC is a suitable technique to overcome this drawback.

2 Components of a basic BPC algorithm

In this section, we present the components of a basic BPC algorithm. For further details on BPC algorithms, the reader is referred to Barnhart et al. (1998), Desaulniers, Desrosiers, and Solomon (2005), Lübbecke and Desrosiers (2005), and Feillet (2010), where important insights about the development of BPC algorithms are highlighted.

As stated in the introduction, a BPC algorithm is a BB algorithm where the lower bounds are computed by CG and the cutting planes are added to strengthen the linear relaxations encountered in the search tree. In this context, such a linear relaxation is called a MP which is solved by CG. CG is an iterative algorithm that solves at each iteration a restricted MP (RMP) and a pricing problem. The RMP is a linear program defined as the MP restricted to a subset of its route variables. The pricing problem can be an arbitrary optimization problem which is solved either to find new variables to add to the current RMP or to prove that the current RMP solution can be extended (by setting all non-generated variables to zero) to yield an optimal solution to the MP.

Below, we discuss the MP, the pricing problem, cutting planes, and branching decisions when BPC is used to solve model (1)–(3) of the VRPTW.

2.1 The master problem

At the root node of the BB search tree, the MP is given by:

$$\min \quad \sum_{r\in\Omega} c_r\lambda_r \tag{4}$$

$$s.t. \quad \sum_{r\in\Omega} a_i^r\lambda_r = 1, \qquad \forall i \in V' \tag{5}$$

$$\lambda_r \geq 0, \qquad \forall r \in \Omega'. \tag{6}$$

In model (4)–(6), the variables λ_r, $r \in \Omega$, are implicitly upper bounded by one through constraints (5). Note that cuts can be added to this MP (see Section 2.3) and that, at other nodes of the search tree, branching decisions can modify it (see Section 2.4).

An optimal solution to (4)–(6) provides a lower bound at the root node of the search tree. When applying CG to solve the MP, a RMP, obtained by replacing Ω with a relatively small subset $\Omega' \subseteq \Omega$ in (4)–(6), is solved at each iteration. The solution process yields an optimal primal solution for this RMP and a complementary dual solution $(\pi_i)_{i \in V'}$, where π_i is the dual variable associated with constraint (5) indexed by $i \in V'$. Extending this primal solution to the MP (i.e., by setting $\lambda_r = 0$ for all $r \in \Omega \setminus \Omega'$) results in an optimal MP solution if the reduced cost $\bar{c}_r = c_r - \sum_{i \in V'} a_i^r \pi_i$ of λ_r is nonnegative for every route $r \in \Omega \setminus \Omega'$. The role of the pricing problem (see Section 2.2) is to find routes with a negative reduced cost or to prove that none exist. When negative reduced cost routes are identified, they are added to subset Ω' before starting a new iteration. Otherwise, the CG process stops with an optimal solution to the MP.

In their BPC algorithms, many authors have employed branching rules and cutting planes defined from arc flows $(x_{ij})_{(i,j) \in A}$. These arc flows can be easily computed from a solution to the MP using the following expression

$$x_{ij} = \sum_{r \in \Omega} b_{ij}^r \lambda_r, \tag{7}$$

where b_{ij}^r is a binary parameter indicating whether or not arc $(i, j) \in A$ is traversed by route $r \in \Omega$.

2.2 The pricing problem

The pricing problem consists of finding a feasible route $r \in \Omega$ with a negative reduced cost $\bar{c}_r$. This problem can be modeled as an elementary shortest path problem with resource constraints (ESPPRC) on graph G. Resources are quantities (e.g. time, load, etc.) that are used to assess the feasibility of a route or to compute the cost of a route. Their values vary along a path according to so-called *resource extension functions* (REFs), which are defined for each resource and each arc in G. In addition, at each vertex of G, *resource windows* restrict the values that can be taken by the resources. Efficient dynamic programming algorithms for solving the pricing problem might be designed when the REFs present the following two properties: 1) they only depend on the resource consumptions and on the values associated with the arc and its tail vertex, allowing the computation of resource values at each vertex; and 2) they are non-decreasing with respect to the resource values, ensuring the possible application of a dominance rule to reduce route enumeration (Desaulniers et al. 1998, Irnich and Desaulniers 2005, Irnich 2008). Irnich (2008) shows how REFs can be used to model complex route costs and constraints arising in VRPs and how they allow some algorithmic procedures to be employed, namely: path representations, efficient cost computations, and constant time feasibility checking performed while building/concatenating paths. More general REFs are presented in Section 4, where problems with many attributes are discussed.

When solving the ESPPRC, a modified cost $\bar{c}_{ij} = c_{ij} - \pi_i$ is associated with each arc (i, j) in G, with $\pi_0 = 0$. This ensures that the cost of a path in G corresponds to the reduced cost of the corresponding route $r \in \Omega$:

$$\bar{c}_r = c_r - \sum_{i \in V'} a_i^r \pi_i = \sum_{(i,j) \in A} c_{ij} b_{ij}^r - \sum_{i \in V'} a_i^r \pi_i = \sum_{(i,j) \in A} \bar{c}_{ij} b_{ij}^r. \tag{8}$$

Unlike the classical shortest path problem (Ahuja, Magnanti, and Orlin 1993), which is polynomially solvable, solving the ESPPRC is not an easy task. Because the arc costs $\bar{c}_{ij}$ may be negative, there might exist negative cost cycles. Dror (1994) showed that the ESPPRC is $\mathcal{NP}$-hard in the strong sense. For this reason, when developing BPC algorithms, some authors have replaced the ESPPRC by the shortest path problem with resource constraints (SPPRC) as the pricing problem (see Irnich and Desaulniers 2005). The SPPRC is a relaxation of the ESPPRC that allows the generation of paths with cycles (i.e., routes with multiple visits to the same customer). In this case, the set Ω of feasible routes is enlarged to include these routes and the meaning of parameter a_i^r (resp. b_{ij}^r) changes to represent the number of times customer $i \in V'$ is visited (resp. arc $(i, j) \in A$ is traversed) by a route $r \in \Omega$. On the one hand, the SPPRC is easier to solve than the ESPPRC as it can be solved by a pseudo-polynomial algorithm as shown by Desrochers, Desrosiers, and Solomon (1992) who devised the first BP algorithm for a VRP. On the other hand, solving a SPPRC

as the pricing problem may significantly reduce the quality of the lower bounds provided by the MP. We discuss in Section 3 other relaxations that have been proposed in the literature, seeking a better compromise between bound quality and total computational time.

2.3 Cutting planes

Despite the fact that extended formulations can provide better lower bounds than those obtained with compact formulations, these bounds might still be too weak to yield an efficient algorithm. Thus, when designing BP algorithms for tackling complex VRPs, it is common practice to reinforce the MP with valid inequalities.

As discussed in Section 2.1, any feasible solution to an extended model can be converted into a solution to a corresponding compact model. Thus, by applying (7) to a cutting plane of the form $\sum_{(i,j)\in A} \beta_{ij} x_{ij} \leq \beta_0$, one can rewrite this inequality in terms of the route variables as follows:

$$\sum_{r\in\Omega} \sum_{(i,j)\in A} \beta_{ij} b_{ij}^r \lambda_r \leq \beta_0, \tag{9}$$

and use it to strengthen the MP.

According to the classification of Poggi de Aragão and Uchoa (2003), cuts of the form (9) are called *robust cuts* because they do not increase the complexity of the pricing problem. Indeed, because they can be expressed in terms of the arc-flow variables, their dual values can be directly incorporated into the modified cost $\bar{c}_{ij}$ of each arc $(i,j) \in A$ as follows. Let ρ be the dual variable associated with (9). The reduced cost $\bar{c}_r$ of a variable λ_r, $r \in \Omega$, becomes:

$$\bar{c}_r = c_r - \sum_{i\in V'} a_i^r \pi_i - \rho \sum_{(i,j)\in A} \beta_{ij} b_{ij}^r = \sum_{(i,j)\in A} (c_{ij} - \pi_j - \rho\beta_{ij}) b_{ij}^r = \sum_{(i,j)\in A} \bar{c}_{ij} b_{ij}^r, \tag{10}$$

by setting $\bar{c}_{ij} = c_{ij} - \pi_j - \rho\beta_{ij}$ for all arcs $(i,j) \in A$. Note that several robust cuts, each with a dual value, can be handled simultaneously. Note also that not all cuts defined in terms of the arc-flow variables are useful in a BPC algorithm because many families of cuts are implied by the definition of the routes.

Valid inequalities can also be defined directly in terms of the route variables. Let $\sum_{r\in\Omega} \beta_r \lambda_r \leq \beta_0$ be such a generic cut and $\sigma < 0$ the dual variable associated with it. Then, the reduced cost of a route $r \in \Omega$ rewrites as follows:

$$\bar{c}_r = \sum_{(i,j)\in A} \bar{c}_{ij} b_{ij}^r - \beta_r \sigma. \tag{11}$$

Since the variable coefficients β_r, $r \in \Omega$, are not necessarily defined as linear functions of the arc flows, the dual value σ may not be directly transferred to the modified arc costs. Such cuts are, thus, said to be *non-robust* and handling their dual values increases the complexity of the pricing problem as additional unrestricted resources are required in the ESPPRC (see Desaulniers, Desrosiers, and Spoorendonk 2011). It is important to note that, regardless of this increased complexity, non-robust cuts have a great potential for reducing the integrality gaps.

In Sections 3 and 4, we discuss various families of cuts that were incorporated in BPC algorithms for solving VRPs. For more details about cutting planes in BPC algorithms, see the general framework introduced by Desaulniers, Desrosiers, and Spoorendonk (2011).

2.4 Branching decisions

To derive integer solutions, branching is the ultimate operation to perform in a BPC algorithm. In this section, we give an overview of branching decisions in BPC algorithms for VRPs. Details on the most common ones are discussed in Section 3.3.

If we consider model (1)–(3), it seems natural to branch on the route variables, that is, to choose a variable $\lambda_r \in \Omega$ such that $\lambda_r \in (0,1)$ in the current MP solution and to impose $\lambda_r = 0$ on one branch and $\lambda_r = 1$ on

the other. Imposing the latter decision is straightforward. On the other hand, imposing $\lambda_r = 0$ is not easy and often ineffective. Indeed, one must also prevent route r to be re-generated by the pricing problem which becomes a more complex ESPPRC, namely, a ESPPRC with forbidden paths (Villeneuve and Desaulniers 2005). Furthermore, fixing a unique route variable to zero does not restrict much the solution space and typically yields an unbalanced search tree.

To alleviate this drawback, more aggressive branching decisions can be devised by considering the arc-flow variables of the compact formulation, instead of the route variables of the extended formulation. These branching decisions are defined through relation (7), and have the advantage of involving many route variables simultaneously beside being robust. Examples of branching decisions performed with respect to the arc-flow variables are: branching directly on the flow on an arc, branching on the number of vehicles leaving the depot, and branching on the flow into/out of a set of vertices (see Section 3.3).

3 Generic tools

In this section, we present ideas applicable to different variants of VRPs. The topics discussed are classified as follows: pricing, cutting, branching, using upper bounds, and stabilizing dual values.

3.1 Pricing

For most VRPs, the pricing problem is an ESPPRC or a relaxation of it that is typically solved by a labeling algorithm. We start by presenting basic labeling algorithms before discussing various ESPPRC relaxations and speed up tools.

3.1.1 Basic labeling algorithms

For clarity reasons, let us start by describing a labeling algorithm for the SPPRC pricing problem arising for the VRPTW. In this algorithm, labels represent partial paths in G which all begin at the origin depot, i.e., vertex 0. Starting from vertex 0, labels are extended through the network, passing by some of the customers, until the destination depot (vertex $n+1$) is reached. In its basic version, a label L representing a path p is a tuple $L = (v, \bar{c}, q, t)$, where $v \in V$ is the last vertex in p; $\bar{c}$ the reduced cost of p; q the cumulated load along p; and t the earliest time at which service can start at vertex v if p is used to reach v. A label L also stores its predecessor label to allow to build complete paths once the algorithm is finished.

Let $L_0 = (0, 0, 0, 0)$ be the initial label at vertex 0 and denote by $v(L)$, $\bar{c}(L)$, $q(L)$, $t(L)$ the components of a label L associated with a path $p(L)$ ending at vertex $v(L) = i$. If one performs a label extension along an arc (i, j), a new label L' is obtained by applying the following relations:

$$v(L') = j \tag{12}$$
$$\bar{c}(L') = \bar{c}(L) + \bar{c}_{ij} \tag{13}$$
$$q(L') = q(L) + q_j \tag{14}$$
$$t(L') = \max\{t(L) + t_{ij}, a_j\}. \tag{15}$$

This new label represents path $p(L') = p(L) \oplus (i, j)$, where we use $\oplus$ as a concatenation symbol. After performing this extension, the feasibility of path $p(L')$ is verified by checking if the cumulated resources are within the resource windows at vertex j. Path $p(L')$ is feasible if $q(L') \in [0, Q]$ and $t(L') \in [a_j, b_j]$. Otherwise, it is infeasible and label L' is discarded.

The efficiency of a labeling algorithm depends on its ability to eliminate non-useful paths. For this reason, labels are compared through a *dominance rule* in order to identify and remove unnecessary labels. Let L be a label and $\mathscr{E}(L)$ be the set of feasible path extensions of path $p(L)$, i.e., a path w in G starting at vertex $v(L)$ belongs to $\mathscr{E}(L)$ if $p(L) \oplus w$ is a feasible path. A label L' can be discarded if there exists a set of labels $\mathcal{L} = \{L_1, L_2, \ldots, L_{|\mathcal{L}|}\}$ such that, for all feasible extensions $\omega \in \mathscr{E}(L')$, there exists a label $L \in \mathcal{L}$ for which: i) $p(L) \oplus \omega$ is feasible and ii) the reduced cost of $p(L) \oplus \omega$ is less than or equal to that of $p(L') \oplus \omega$. These two conditions specify that L' is not required to describe the set of Pareto-optimal paths ending at any vertex

$v \neq v(L)$ and can, thus, be discarded. Note that conditions i) and ii) are too difficult to be assessed because it would require an enumeration of all paths and extensions. Therefore, these two conditions are replaced by the following sufficient conditions. Let L_1 and L_2 be two labels such that $v(L_1) = v(L_2)$. L_1 is said to dominate L_2 if:

$$\bar{c}(L_1) \leq \bar{c}(L_2) \tag{16}$$
$$q(L_1) \leq q(L_2) \tag{17}$$
$$t(L_1) \leq t(L_2). \tag{18}$$

This dominance rule is valid because the REFs defined by the right-hand side of (13)–(15) are non-decreasing with respect to $\bar{c}(L)$, $q(L)$ and $t(L)$, respectively (see Desaulniers et al. 1998). As discussed in the next sections, this dominance rule needs to be modified to accommodate specificities related to different pricing problems.

Depending on the data structure used to store the labels and the order in which the labels are extended, the above labeling algorithm can have a pseudo-polynomial time complexity (see, e.g., Desrochers and Soumis 1988, Irnich and Desaulniers 2005). This is an important advantage over the algorithms that can solve the ESPPRC which is $\mathcal{NP}$-hard in the strong sense (Dror 1994).

Despite the success achieved by BPC algorithms using a SPPRC pricing problem, elementarity constraints may have a huge impact on the quality of the bounds generated by the MP (Feillet et al. 2004). For this reason, authors have turned their attention to the development of efficient algorithms that can either solve the ESPPRC or provide the so-called elementary bounds. One of the first attempts in this direction was made by Beasley and Christofides (1989), who propose a dynamic programming algorithm that manages elementarity through a *customer resource vector* ($\mathcal{E}$) which stores the customers visited along a path. With this new label definition, the dominance rule must also include:

$$\mathcal{E}(L_1) \subseteq \mathcal{E}(L_2) \tag{19}$$

in addition to relations (16)–(18). Condition (19) stipulates that label L_1 cannot dominate label L_2 if $p(L_1)$ contains a vertex j that has not been visited in $p(L_2)$, i.e., $j \in \mathcal{E}(L_1) \setminus \mathcal{E}(L_2)$. In this case, there might exist feasible extensions in $\mathscr{E}(L_2)$ which include vertex j and decrease the reduced cost. Given that these extensions are not in $\mathscr{E}(L_1)$, L_2 cannot be discarded. Improvements on this dominance rule are proposed by Chabrier (2006).

Feillet et al. (2004) present another exact algorithm to solve the ESPPRC. They extend the idea of Beasley and Christofides (1989) by introducing a new label definition that replaces set $\mathcal{E}$ with a set of unreachable vertices $\mathcal{U}$. A vertex j is added to $\mathcal{U}(L)$ if it is visited along path $p(L)$ or if it becomes unreachable due to resource limitations, i.e., $q(L) + q_j > Q$ or $t(L) + t_{ij} > b_j$. Condition (19) then becomes:

$$\mathcal{U}(L_1) \subseteq \mathcal{U}(L_2). \tag{20}$$

Through computational tests, Feillet et al. (2004) show that using elementary routes can yield much better lower bounds at the root node of the search tree.

3.1.2 ESPPRC relaxations

Given the complexity of the ESPPRC, many authors have designed BPC algorithms that rely on relaxations of the ESPPRC. In most cases, the elementarity requirements are totally or partially relaxed. Totally relaxing them gives the SPPRC which allows any cycle to be part of a generated path. We discuss below several stronger relaxations that differ by the families of cycles they forbid. Note that, although elementarity can also be enforced via cuts (e.g., strong degree and k-cycle-elimination cuts), we focus here on strategies applied to labeling algorithms. Cutting planes are discussed in Section 3.2. The reader is referred to Contardo, Desaulniers, and Lessard (2015) for a complete analysis on different ways to reach the lower bound achieved with only elementary routes.

Christofides, Mingozzi, and Toth (1981a) introduce a technique, called k-cycle elimination, that has been largely employed to avoid cycles. It consists of forbidding cycles of length k or less to be formed while solving the SPPRC. This relaxation is known as the k-cyc-SPPRC. Solving it not only increases the lower bound quality but can also accelerate the algorithm running time. For the sake of clarity, we present separately the case $k = 2$. In fact, the 2-cyc-SPPRC has been largely applied in the literature, as well as combined with state-of-the-art techniques.

2-cycle elimination: In the 2-cyc-SPPRC, cycles of the form $i - j - i$ are forbidden. The use of 2-cycle elimination is particularly interesting because it yields stronger bounds without changing the complexity of the labeling algorithm. In fact, Christofides, Mingozzi, and Toth (1981a) show that to prevent cycles of length two, it is sufficient to keep, for each possible resource state, at most two least-cost labels L_1 and L_2 whose predecessor labels are associated with different vertices. Two examples on the use of 2-cycle elimination can be found in: Desrochers, Desrosiers, and Solomon (1992), who introduce the first exact algorithm capable of solving 100-customer VRPTW instances, and Fukasawa et al. (2006), who propose a BPC algorithm to solve the CVRP and generated q-routes with the pricing problem. A q-route is a path that respects vehicle capacity but may contain cycles.

k-cycle elimination: To the best of our knowledge, k-cycle elimination with $k \geq 3$ has only been tested by Irnich and Villeneuve (2006) and Fukasawa et al. (2006). Irnich and Villeneuve propose a new representation of partial paths which is based on the notion of *set form*. The idea is to use a vector to represent all the paths in which a given customer $i \in V'$ is placed at the jth position in the path. In the case of the k-cyc-SPPRC, one uses a vector of length k to store the k last vertices of a path. Then, a set $\mathcal{H}$ formed by the union of all required set forms is defined to represent all k-cycle elimination constraints. For example, for the 4-cyc-SPPRC, consider a label L such that the four last visited vertices are (a, b, c, v). Then, $\mathcal{H}(L)$ contains the set forms: $(v, \cdot, \cdot, \cdot)$, $(\cdot, v, \cdot, \cdot)$, $(\cdot, \cdot, v, \cdot)$, $(\cdot, \cdot, \cdot, v)$, $(c, \cdot, \cdot, \cdot)$, $(\cdot, c, \cdot, \cdot)$, $(\cdot, \cdot, c, \cdot)$, $(b, \cdot, \cdot, \cdot)$, $(\cdot, b, \cdot, \cdot)$, $(a, \cdot, \cdot, \cdot)$. Any extension of $p(L)$ whose beginning matches one of these forms is forbidden as it would create a cycle of length four or less. Given that the cardinality of $\mathcal{H}(L)$ is quadratic in k, this approach may rapidly become prohibitive as k increases. Fukasawa et al. (2006) use this path representation to assess the impact of eliminating cycles with length four or less in their BPC algorithm for the CVRP.

Partial elementarity: Desaulniers, Lessard, and Hadjar (2008) introduce another ESPPRC relaxation called the partially ESPPRC. It can be seen as an ESPPRC that requires elementarity only for a subset $\mathcal{E}_{max}$ of customers, whose maximal cardinality is determined *a priori*. Set $\mathcal{E}_{max}$ is built dynamically from scratch by adding customers that are visited more than once in a route of a MP optimal solution. Each time that a customer i is added to $\mathcal{E}_{max}$, the labeling algorithm is adjusted to forbid multiple visits to customer i and all columns visiting multiple times this customer are removed from the current RMP. The MP is then re-optimized and other customers might be added to $\mathcal{E}_{max}$. Note that, if the maximal cardinality of $\mathcal{E}_{max}$ is less than n, there is no guarantee that elementary routes will be obtained at the end of the algorithm.

ng-path relaxation: Currently, state-of-the-art BPC algorithms use the *ng*-path relaxation in their pricing procedures. The *ng*-path concept proposed by Baldacci, Mingozzi, and Roberti (2011) relies on the definition of a neighborhood (also called *ng*-set) $\mathcal{N}_i$ for each customer $i \in V'$. For a given integer parameter $\Delta \geq 0$, a neighborhood $\mathcal{N}_i \subseteq V'$ contains the Δ closest customers to i and vertex i itself. Different criteria (e.g., distance, time) can be used to define the proximity of the customers. An *ng*-path is not necessarily elementary: it can contain a cycle starting and ending at a vertex j if and only if there exists a vertex i in this cycle such that $j \notin \mathcal{N}_i$. Hence, such a cycle is disallowed if and only if $j \in \mathcal{N}_i$ for every vertex i it contains. In a labeling algorithm, the neighborhoods of the customers visited along a path $p(L) = (0, i_1, \ldots, i_k)$ represented by a label L are used to form a set $\Pi(L) \subseteq V'$ of customers to which L cannot be extended without violating the *ng*-path cycling restrictions. Let $V(L) = \{i_1, \ldots, i_k\}$ be the customers visited in $p(L)$. Then, $\Pi(L)$ is given by:

$$\Pi(L) = \{i_u \in V(L) : i_u \in \bigcap_{s=u}^{k} \mathcal{N}_{i_s}\} \tag{21}$$

and is called the *memory* of $p(L)$. Contrarily to sets $\mathcal{E}(L)$ and $\mathcal{U}(L)$, once a vertex j enters a set $\Pi(L)$, it may leave the memory when $p(L)$ is extended, i.e., j can be *forgotten* and revisited.

The labeling algorithm for the ESPPRC described in Subsection 3.1.1 can be adapted to solve this new relaxation denoted *ng*-SPPRC. In the label definition, set $\mathcal{E}$ is replaced by set Π. A label L is only extended over an arc (i, j) if $j \notin \Pi(L)$. When a new label L' is created, set $\Pi(L')$ is computed as follows:

$$\Pi(L') = (\Pi(L) \cap \mathcal{N}_j) \cup \{j\}. \tag{22}$$

In the dominance rule, condition (19) is replaced by:

$$\Pi(L_1) \subseteq \Pi(L_2). \tag{23}$$

Note that set Π can be enriched by adding unreachable vertices with regard to vehicle capacity, time windows, or other side constraints. Thus, if $\mathcal{F}$ is the set of all customers where an extension is not feasible, relation (23) can also be replaced by:

$$\mathcal{F}(L_1) \subseteq \mathcal{F}(L_2). \tag{24}$$

Because the cardinality of Π is less than or equal to Δ, condition (24) is less restrictive than (20) and, consequently, more labels might be discarded than in the labeling algorithm for the ESPPRC.

When used as a pricing problem, the *ng*-SPPRC produces lower bounds that are between those derived with the SPPRC and the ESPPRC: if $\Delta = 0$, then $\mathcal{N}_i = \{i\}$, $\forall i \in V'$, and the *ng*-SPPRC is the SPPRC; if $\Delta = |V'| - 1$, then $\mathcal{N}_i = V'$, $\forall i \in V'$, and the *ng*-SPPRC is the ESPPRC. Note that, although unlikely, cycles of length two can be formed when solving the *ng*-SPPRC except when $\mathcal{N}_i = V'$, $\forall i \in V'$. According to Contardo and Martinelli (2014), the advantage of solving the *ng*-SPPRC as an alternative to the k-cyc-SPPRC can be attributed to the fact that in the former, cycles are measured in terms of distance, while in the latter, they are measured according to the number of visited customers. Given that long cycles in terms of distance have not much chance to appear in a MP optimal solution, the *ng*-SPPRC represents a powerful relaxation.

An important parameter in the *ng*-SPPRC is Δ, the size of the neighborhoods, which is normally defined *a priori* and interferes on the labeling algorithm complexity. On the one hand, the larger the value of Δ, the closer to the ESPPRC the *ng*-SPPRC becomes. On the other hand, the algorithm complexity increases exponentially with the value of Δ. Baldacci, Mingozzi, and Roberti (2011) show that $\Delta = 8$ represents a good trade-off between lower bound quality and computing time for the VRPTW. More recently, Pecin et al. (2017a) use $\Delta \in \{10, 20\}$ for Solomon's and Gehring and Homberger's VRPTW datasets.

3.1.3 Speed up tools

Given that the pricing problem is solved many times in a BPC algorithm, accelerating the solution of the pricing problem is a critical aspect for improving the performance of the algorithm (Lozano, Duque, and Medaglia 2016). For this reason, the following techniques are proposed to accelerate its solution.

Decremental state-space relaxation: Christofides, Mingozzi, and Toth (1981b) intoroduce the concept of state-space relaxation which consists of relaxing some constraints of a problem and devising an efficient algorithm for solving the resulting relaxation to produce lower bounds. When applied to the pricing problem, this approach tends to generate infeasible paths that may yield poor lower bounds. Seeking to improve the bounds provided in this way, some authors observe that by dynamically increasing the set of customers subject to elementarity restrictions, elementary routes might be obtained in reduced computational times. In this context, Boland, Dethridge, and Dumitrescu (2006) and Righini and Salani (2008) propose independently a technique called *decremental state-space relaxation* (DSSR) which includes a mechanism for re-inserting dynamically constraints that were previously relaxed.

For the ESPPRC, a DSSR algorithm provides at each iteration a lower bound on its optimal value and works as follows. Initially, a SPPRC is solved and, if the least-cost path is cycle-free, it is optimal for the

ESPPRC. Otherwise, the customer(s) visited more than once in this path are added to a set $\hat{\mathcal{E}}$. In the next iteration, one solves a more restricted SPPRC where elementary conditions are only imposed to the vertices in $\hat{\mathcal{E}}$. This procedure continues until an elementary route is found or the optimal path has a non negative (reduced) cost. Moreover, in the context of a CG algorithm, the procedure can be stopped if the labeling algorithm produces several paths and at least one of them is elementary and has a negative cost. To improve the performance of DSSR, Desaulniers, Lessard, and Hadjar (2008) initialize the set $\hat{\mathcal{E}}$ at a CG iteration to the final set of the previous iteration as they observed during their tests a large intersection between the sets $\hat{\mathcal{E}}$ generated in consecutive CG iterations.

Martinelli, Pecin, and Poggi (2014) combine the concept of *ng*-paths and DSSR. In their algorithm, neighborhoods $\mathcal{N}_i$, $i \in V'$, are computed *a priori*, but are not directly used. Instead, subsets $\tilde{\mathcal{N}}_i \subseteq \mathcal{N}_i$, $i \in V'$, are considered in the labeling algorithm. These subsets are initially empty and augmented when the labeling algorithm returns an optimal path that contains a forbidden cycle with respect to the neighborhoods $\mathcal{N}_i$ $i \in V'$. Contardo and Martinelli (2014) reinforce the sets $\tilde{\mathcal{N}}_i$ by enlarging them with the addition of vertices forming cycles of length two, even if they were not in the original sets $\mathcal{N}_i$. This approach ensures that the bounds obtained are at least as strong as those of the *ng*-SPPRC with 2-cycle elimination. Contardo, Desaulniers, and Lessard (2015) devise a similar approach where there are no initial sets $\mathcal{N}_i$, $i \in V'$, the sets $\tilde{\mathcal{N}}_i$, $i \in V'$, are initialized with a small number of closest customers and they are updated (up to a maximum size per neighborhood) only once the MP is solved, using customers that are visited more than once in a route that is part of the MP optimal solution.

3.1.4 Bidirectional labeling

Proposed by Righini and Salani (2006), *bidirectional labeling* consists of propagating labels in both directions (from vertex 0 to its successors and from vertex $n+1$ to its predecessors) and then join forward and backward labels to produce complete feasible routes. Basically, all the theory described thus far for a (forward) labeling algorithm can be applied to a backward algorithm thanks to the results in Irnich (2008) showing that REFs can be inverted. To avoid producing twice as many labels as in a monodirectional labeling algorithm, the extension of a path in one direction is stopped when it is guaranteed that its remaining part can be generated in the other direction. This can be done in two ways: by arc bounding or by resource bounding. Arc bounding consists of computing an upper bound (in polynomial time) on the number of vertices that can still be visited by the path without exceeding resource constraints. If this number is less than the number of vertices already visited by the path, the extension is stopped. Resource bounding considers a *critical resource*, whose consumption is monotone along a path, to stop the path extension. A path is only extended if the accumulated amount of the critical resource does not exceed the half-way of its availability. For example, if the load is chosen to be the critical resource, only labels whose load consumption is less than $Q/2$ can be extended. Since the number of generated labels can grow exponentially with the number of arcs in the paths, keeping the length of the paths short in both directions reduce the total number of labels and speed up the labeling process, especially when the feasible paths can contain a large number of vertices.

Note that, with bidirectional labeling, the same path can sometimes be obtained multiple times by concatenating more than one pair of forward and backward labels. To reduce as much as possible the concatenation process and avoid generating the same path multiple times, several rules may be considered. One example is given by Baldacci, Christofides, and Mingozzi (2008) who only allow the concatenation of a pair of forward and backward labels if their critical resource consumptions differ by the smallest amount possible along the resulting path.

A final aspect that is worth discussing is the potentially unbalanced numbers of labels generated by the forward and backward algorithms, i.e., one direction generates much more labels than the other), even if the half-way point is carefully chosen. This can be caused by asymmetric data coming from the VRP instances or by the dual information obtained over the CG iterations. Trying to balance the workload between the forward and backward algorithms, Tilk et al. (2017) propose a dynamic half-way point strategy that is applied at each iteration. Let H_F and H_B be resource limits associated with the forward and the backward algorithm, respectively. In a standard algorithm, $H_F = H_B = Q/2$ if load is the critical resource, and all forward extensions are performed before the backward extensions. In a dynamic half-way point algorithm,

the values of H_F and H_B are updated dynamically during the algorithm execution, always ensuring that $H_F \geq H_B$ to guarantee optimality. Moreover, forward and backward labels are extended in an alternate way. To determine which label to extend next, the algorithm chooses first the direction with the least number of generated labels, number of processed labels, or number of unprocessed labels. It then chooses the label in the selected direction according to a standard rule (e.g., least load) and extends it before updating H_F or H_B based on the critical resource values in the newly created labels. A similar strategy is conceived by Pecin et al. (2017b).

3.1.5 Completion bounds

Depending on the dominance rule used, some unpromising labels may be unnecessarily kept in a labeling algorithm. It is, however, possible to reinforce a dominance rule with the use of *completion bounds*. Let L be a label associated with path $p(L)$. A completion bound for $p(L)$ can be obtained by computing a lower bound $lb(L)$ on the reduced cost of all feasible extensions in $\mathscr{E}(L)$ that reach vertex $n+1$. If $\bar{c}(L)+lb(L) \geq 0$, then label L can be discarded without losing any negative reduced cost route. Completion bounds are used in several papers, including Lübbecke (2005), Baldacci, Christofides, and Mingozzi (2008), Baldacci, Mingozzi, and Roberti (2011), Contardo and Martinelli (2014), Martinelli, Pecin, and Poggi (2014), Pecin et al. (2017b,a).

Computing the best completion bound for every generated label would make the labeling algorithm too time consuming. Consequently, good approximations, such as the following ones, are rather used. Martinelli, Pecin, and Poggi (2014) compute completion bounds inside a DSSR procedure. They use the labels computed at the previous DSSR iteration to estimate completion bounds. When they solve symmetric CVRP instances, completion bounds are extracted directly from a best reduced cost matrix (see Baldacci, Mingozzi, and Roberti 2011). On the other hand, when solving asymmetric problems, they change the direction (forward or backward) of the labeling algorithm from one iteration to the other in the DSSR algorithm. When non-robust cuts are employed to strengthen the MP, completion bounds can be computed like in Contardo, Cordeau, and Gendron (2014) and Contardo and Martinelli (2014). They only consider at most 20% of the largest dual values associated with these cuts and to under- or overestimate the impact of the remaining ones, ensuring the bound validity. Finally, Pecin et al. (2017b,a) extend the approach of Contardo and Martinelli (2014) to apply completion bounds in bidirectional algorithms.

3.1.6 Heuristic pricing

Despite the fact that many speed up techniques have been proposed to accelerate exact dynamic programming algorithms, they can still remain very time consuming. Because there is no need to solve the pricing problem exactly, except to prove the optimality of the current solution in the last CG iteration, fast and effective heuristics have been developed to find negative reduced cost variables. Their purpose is to reduce the number of calls to the exact labeling algorithm, often yielding a substantial reduction of the total computational time.

In most cases, pricing heuristics are adaptations of exact labeling algorithms. They can be obtained by either relaxing certain dominance rules or by heuristically reducing the size of the network. Some authors only keep a few labels for each pair of time and load values at each node (Fukasawa et al. 2006, Martinelli, Pecin, and Poggi 2014, Pecin et al. 2017b,a). This implies that several non-dominated labels may be discarded, including those leading to optimal solutions. To generate q-routes, Fukasawa et al. (2006) scale down customer demands and vehicle capacity by some factor $\rho > 1$, i.e., $q'_i = \lceil q_i/\rho \rceil$, $\forall i \in V'$, and $Q' = \lceil Q/\rho \rceil$.

Another way of eliminating labels is by employing aggressive dominance rules. For instance, Desaulniers, Lessard, and Hadjar (2008) only consider a (possibly empty) restricted subset of customer resources when applying dominance test (19). When performing graph reduction, several criteria may be used to remove arcs from the network. If one decides to eliminate unpromising arcs based on their current reduced cost, it can be done by only keeping the best incoming and outcoming arcs at each customer with respect to their reduced costs (Fukasawa et al. 2006, Desaulniers, Lessard, and Hadjar 2008, Pecin et al. 2017b) or by reducing graph density using the same ranking approach (Contardo and Martinelli 2014). Alternatively, Chabrier (2006) remove from the network the arcs presenting a high resource consumption. Although heuristic labeling

algorithms are largely employed, some BPC algorithms benefit from well-known heuristics. As an example, Desaulniers, Lessard, and Hadjar (2008) and Archetti, Bouchard, and Desaulniers (2011) use tabu search to generate routes with a negative reduced cost. Finally, note that, as suggested by Desaulniers, Lessard, and Hadjar (2008), different pricing heuristics can be invoked at each CG iteration and throughout the whole CG process depending on the current phase of the process (beginning, middle or end). To ensure optimality, an exact algorithm must always be executed at least once, in the last CG iteration.

3.1.7 Miscellaneous approaches

We start this subsection by discussing a trick that can be used for the VRPTW. For some instances, vehicle capacity may not be really restrictive. For this reason, it can be neglected in the pricing problem while rounded capacity inequalities (see Subsection 3.2) may be added to the MP whenever necessary (Pecin et al. 2017a). This practice facilitates the solution of the pricing problem and can make the BPC algorithm more efficient.

Instead of using a labeling algorithm for solving the pricing problem of the VRPTW, Rousseau, Gendreau, and Pesant (2004) devise a constraint programming algorithm which allows to generate elementary routes. This algorithm includes arc elimination constraints, a dynamic programming search strategy, and a lower bounding procedure based on an assignment problem.

Feillet, Gendreau, and Rousseau (2008) incorporate concepts from constraint programming such as *limited discrepancy search* (LDS), *label loading* and *meta extensions* to dynamic programming algorithms as an attempt to quickly generate paths with negative reduced cost. LDS is a tree search method that employs a heuristic criterion to define, for each vertex, which outgoing arcs are the most promising. Each other arc yields a discrepancy and an upper bound on the number of discrepancies that a path can contain is imposed to limit the search. This upper bound is initialized to a small value and increased as needed. Label loading and meta extensions, in turn, use information from the current RMP solution to accelerate the route generation process. A metavertex is created for each vertex of each route in this solution and loaded with a label representing the end of the corresponding route. During the labeling process, these new labels can be added to partial paths to generate complete routes or can be used to compute completion bounds.

Contrarily to other authors who try to accelerate labeling algorithms, Lozano, Duque, and Medaglia (2016) develop a pulse algorithm to solve the ESPPRC by extending the work of Lozano and Medaglia (2013). In a pulse algorithm, paths connecting two vertices in a graph are found by propagating pulses through the network. It can be seen as a graph exploration procedure that follows a depth-first search strategy, where pruning is used to discard unpromising paths. Three pruning strategies are developed: infeasibility pruning, that checks for a violation of the structural constraints (vehicle capacity, time windows, elementarity); bound pruning, that uses primal solutions and lower bounds to discard sub-optimal solutions; and rollback pruning, that compares two pulses differing only by the last vertex visited. The pulse algorithm differs from a labeling algorithm in many aspects, namely: a dominance rule is not required because it does not need to handle long lists of labels; many pruning strategies in addition to pruning by infeasibility can be devised; and bidirectional search cannot be applied due its recursive nature. Finally, because recursive pulses explore one vertex at a time until it reaches the destination vertex, the algorithm can perform searches over different vertices in parallel.

Very recently, Desaulniers, Pecin, and Contardo (2017) introduce a new paradigm for the pricing problem, called *selective pricing*. Selective pricing stems from the observation that, when an ESPPRC relaxation is used as a pricing problem, the CG algorithm can be stopped when there is a proof that no elementary routes with a negative reduced cost exist, even if there are still some non-elementary routes with a negative reduced cost. Consequently, the labeling algorithm can be selective and discards cautiously non-elementary routes even if they are not dominated. As an example, the authors implement selective pricing by modifying a labeling algorithm used to solve the *ng*-SPPRC. Note that this strategy has the potential to yield better lower bounds.

Finally, Boschetti, Maniezzo, and Strappaveccia (2017) observe that, because labeling algorithms work in a stage-wise way, ESPPRC relaxations can be solved more efficiently in a GPU environment. Therefore, they

investigate the impact of using parallelization procedures to achieve time reduction for route relaxations such as q-routes and ng-routes. In particular, they develop parallel label extension procedures and a strategy to efficiently manage the sets $\Pi(L)$ of vertices that cannot be reached by extending the corresponding label L without violating the ng-restrictions.

3.2 Cutting

This section describes cutting strategies used in BPC algorithms for VRPs. Here, we limit our discussion to the CVRP and the VRPTW. Valid inequalities for other VRPs are discussed in Section 4. For the sake of clarity, we present robust and non-robust cuts separately.

3.2.1 Robust cuts

Robust cuts, as defined by Fukasawa et al. (2006), have the advantage of not changing the structure of the pricing problem. For VRPs, these cuts are often defined directly in terms of the arc-flow variables. They are effective at closing the integrality gap for some easy instances, but are not sufficient for harder ones. Note that many cuts used in BC frameworks are already implied by the structure of the routes considered in the MP. For example, Letchford and Salazar-González (2006) prove for the CVRP that all generalized large multistar cuts are implied by model (1)–(3), even if the set of routes contains q-routes. This aspect is illustrated by comparing the families of cuts used in the successful algorithms of Fukasawa et al. (2006) and Pecin et al. (2017b) for the CVRP. After solving the root node relaxation, Fukasawa et al. (2006)'s algorithm chooses between a BC and a BPC algorithm to solve the problem. The BC algorithm considers many valid inequalities used by Lysgaard, Letchford, and Eglese (2004), namely: Rounded capacity, framed capacity, strengthened comb, multistar, and extended hypotour cuts. In the BPC algorithm, only rounded capacity cuts contribute effectively to improving the lower bound obtained at the root node. On its side, the BPC algorithm of Pecin et al. (2017b) only applies rounded capacity and strengthened comb cuts.

Robust cuts can also be defined via capacity-indexed (CI) variables (see their definition below). Because these variables can be expressed by means of q-routes, the dual variables associated with cuts defined in terms of CI variables can be directly incorporated into the reduced cost of q-routes when solving a pricing problem allowing these routes.

k-path cuts, subtour elimination constraints, and rounded capacity cuts: Cutting planes belonging to the family of k-path cuts (kPCs) are applied in many algorithms to solve VRPs. The kPCs assume that, given a subset $S \subseteq V'$ 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 set S in any feasible solution. The following inequality is valid for the CVRP and the VRPTW:

$$X(S) = \sum_{(i,j)\in\delta^-(S)} x_{ij} \geq k(S), \tag{25}$$

where $\delta^-(S) = \{(i,j) \in A \,|\, i \in V \setminus S, j \in S\} \subset A$ is the subset of arcs entering S and $X(S)$ is the total flow entering S. Note that, because a route may enter S more than once, the left-hand side of (25) expressed in terms of the routing variables provides an upper bound on the number of vehicles used to serve the demand of set S. A strengthened version of these cuts is introduced by Baldacci, Christofides, and Mingozzi (2008) and discussed in Subsection 3.2.2.

How $k(S)$ is computed varies depending on the problem at hand. For the CVRP, $k(S)$ corresponds to the solution of a bin packing problem (BPP), where the bins have capacity Q and the weights of the items are given by the demands of the customers in S. Because the BPP is $\mathcal{NP}$-hard, the right-hand side of (25) can be replaced by $k(S) = \lceil d(S)/Q \rceil$ with $d(S) = \sum_{i\in S} q_i$, yielding the subtour elimination constraints (Dantzig, Fulkerson, and Johnson 1954, Laporte, Nobert, and Desrochers 1985) if $k(S) = 1$ and the rounded capacity cuts (RCCs) (Laporte, Nobert, and Desrochers 1985, Naddef and Rinaldi 2002) if $k(S) \geq 2$.

For the VRPTW, computing $k(S)$ is not an easy task because it requires solving a VRPTW restricted to subset S. Kohl et al. (1999), however, show that determining only if $k(S) > 1$ can be manageable as this corresponds to solving a traveling salesman problem (TSP) with time windows (TSPTW), which can

be done in pseudo-polynomial time using dynamic programming. If the TSPTW is infeasible for a subset S such that $X(S) < 2$, one obtains a so-called 2-path cut (2PC). In Kohl et al. (1999), the 2PCs are separated by enumeration. Desaulniers, Lessard, and Hadjar (2008) propose a generalization of the kPCs that may assign different coefficients to the arc-flow variables in (25) depending on the customers that can be visited by a path entering subset S through the corresponding arc.

Cuts defined with capacity-indexed variables: Another way of deriving robust cuts for VRPs is by using CI variables. These variables are introduced by Pessoa, Poggi de Aragão, and Uchoa (2008) and successfully employed by Pessoa, Poggi de Aragão, and Uchoa (2009) to derive robust cuts for their BPC algorithms. CI variables can be defined over a multigraph $G_Q = (V, A_Q)$, where A_Q is a set containing arcs $(i,j)^\kappa$, $\forall\,(i,j) \in A$, $\kappa = 0, \ldots, Q - d_i$ and $(0,i)^Q$, $\forall i \in V'$. With each arc $(i,j)^\kappa$ is associated a binary variable y_{ij}^κ that indicates if a vehicle traverses (i,j) with a residual capacity κ or equivalently with a load $Q - \kappa$. The CI variables can be easily written in terms of route variables when they are generated by a pricing problem involving a load resource (e.g., q-route variables). Let $b_{ij}^{r\kappa}$ be a binary coefficient that indicates if arc (i,j) is traversed by route r carrying a load $Q - \kappa$. Then, one can write:

$$y_{ij}^\kappa = \sum_{r \in \Omega} b_{ij}^{r\kappa} \lambda_r, \qquad \forall (i,j)^\kappa \in A_Q. \tag{26}$$

In a formulation defined in terms of the CI variables, a generic constraint indexed by s has the form $\sum_{(i,j)^\kappa \in A_Q} \beta_{ij}^{\kappa s} y_{ij}^\kappa \geq \beta_s$. By applying (26), we obtain the following constraint for the set partitioning model (1)–(3):

$$\sum_{r \in \Omega} \left(\sum_{(i,j)^\kappa \in A_Q} \beta_{ij}^{\kappa s} b_{ij}^{r\kappa} \right) \lambda_r \geq \beta_s. \tag{27}$$

The dual variable associated with this constraint can be easily handled in a labeling algorithm if condition (17) is replaced by $q(L_1) = q(L_2)$. Note that this new condition does not change the theoretical complexity of the labeling algorithm (see, e.g., Pecin et al. 2017b, for details).

Let us present a family of cuts defined over CI variables introduced by Pessoa, Poggi de Aragão, and Uchoa (2008). Given a set $S \subseteq V'$ of customers, denote by $\delta_Q^-(S)$ the subset of arcs in A_Q, with any capacity index, entering S. The associated RCC can be written:

$$\sum_{(i,j)^\kappa \in \delta_Q^-(S)} y_{ij}^\kappa \geq \lceil d(S)/Q \rceil = k(S). \tag{28}$$

Let $\kappa^* = d(S) - Q(k(S) - 1) - 1$ be the largest load that a vehicle can deliver to the customers in S such that the remaining demand $d(S) - q^*$ still requires at least $k(S)$ vehicles to be delivered. If less than $k(S)$ vehicles with a residual capacity $\kappa > \kappa^*$ enters set S, then at least one other vehicle must enter S. This statement yields the following strengthened RCC defined in terms of the CI variables:

$$\frac{k(S)+1}{k(S)} \sum_{(i,j)^\kappa \in \delta_Q^-(S) \,:\, \kappa > \kappa^*} y_{ij}^\kappa + \sum_{(i,j)^\kappa \in \delta_Q^-(S) \,:\, \kappa \leq \kappa^*} y_{ij}^\kappa \geq k(S) + 1. \tag{29}$$

This inequality dominates (28) and can be further strengthened by including arcs exiting S (see Pessoa, Poggi de Aragão, and Uchoa 2009).

Other families of cuts defined in terms of CI variables are the homogeneous extended capacity cuts and the triangle clique cuts. See Pessoa, Poggi de Aragão, and Uchoa (2008, 2009) for details.

3.2.2 Non-robust cuts

As mentioned in Section 2.3, handling the dual value of a non-robust cut in the pricing problem increases its complexity. In this section, we discuss some families of non-robust cuts developed in the context of VRPs and how their dual values can be handled in the labeling algorithms.

Subset row cuts: The subset-row cuts (SRCs) proposed by Jepsen et al. (2008) are Chvátal-Gomory cuts of rank 1derived from a subset of constraints (2). Given an index subset $C \subseteq V'$ of these rows, a SRC is defined as:

$$\sum_{r\in\Omega} \left\lfloor \gamma \sum_{i\in C} a_i^r \right\rfloor \lambda_r \leq \lfloor \gamma |C| \rfloor, \tag{30}$$

where $\gamma = 1/k$, with $k \in \{1, \ldots, |C|\}$. A given route variable λ_r has a non-zero coefficient in (30) if $\sum_{i\in C} a_i^r \geq k$, i.e., if route r visits more than k customers in C. Some recent research works (Pecin 2014, Pecin et al. 2017b,c,a) investigate the impact of using other values of $k \in]0, |C|[$. Moreover, Petersen, Pisinger, and Spoorendonk (2008) and Pecin et al. (2017c) consider more general rank-1 Chvátal-Gomory cuts by allowing a specific multiplier γ_i for each customer $i \in C$. In the following, we focus on the case with identical multipliers unless otherwise specified.

By varying the cardinality of C and the value of γ, different families of SRCs can be derived. However, most BPC algorithms consider SRCs obtained with $|C| \leq 5$. Larger sets C would be harder to separate and thus be only marginally useful. According to Pecin et al. (2017b), the interesting SRCs are:

- 3-SRCs: $|C| = 3$ and $\gamma = 1/2$. These are the most popular SRCs which are employed in Jepsen et al. (2008), Desaulniers, Lessard, and Hadjar (2008), Baldacci, Mingozzi, and Roberti (2011), Contardo and Martinelli (2014) and Pecin et al. (2017b);
- 4-SRCs: $|C| =$ and $\gamma = 2/3$. Used by Pecin et al. (2017b);
- 5,2-SRCs: $|C| = 5$ and $\gamma = 1/2$. Used by Pecin et al. (2017b);
- 5,1-SRCs: $|C| = 5$ and $\gamma = 1/3$. Used by Pecin et al. (2017b);
- 1-SRCs: $|C| = 1$ and $\gamma = 1/2$. These inequalities, also called strong degree cuts, are applied by Contardo and Martinelli (2014) and Pecin et al. (2017b). We discuss them later in this section.

From (11) and (30), the reduced cost of a route r in which one SRC s has been added to the MP is $\bar{c}_r = \sum_{(i,j)\in A} \bar{c}_{ij} b_{ij}^r - \sigma_s \nu_s^r$, where $\nu_s^r = \lfloor \gamma \sum_{i\in C_s} a_i^r \rfloor$, C_s is the set of vertices (customers) defining SRC s, and $\sigma_s \leq 0$ is the associated dual variable. The last term of this reduced cost can be seen as paying a *penalty* σ_s for every k visits to the customers in set C_s. To handle such a penalty, the following modifications to the basic labeling algorithms described in Section 3.1.1 are proposed by Jepsen et al. (2008).

Let Θ be the set of all SRCs s in the MP such that $\sigma_s < 0$. For every $s \in \Theta$, a new resource $\nu_s(L)$ is added to the definition of a label L to count the number of times (mod k) that a customer in C_s has been visited in the associated path. Consequently, when extending a label L along an arc $(i, j) \in A$ to create a new label L', $\nu_s(L')$ is set equal to $\nu_s(L)$ if $j \notin C_s$ and to $\nu_s(L) + 1 \pmod k$ otherwise. In the latter case, if $\nu_s(L') = 0$, then σ_s is subtracted from $\bar{c}(L')$.

The dominance rule also needs to be modified. As mentioned in Desaulniers, Desrosiers, and Spoorendonk (2011), one way to do so is to add conditions

$$\nu_s(L_1) \leq \nu_s(L_2), \qquad \forall s \in \Theta, \tag{31}$$

when checking if a label L_1 dominates a label L_2. Indeed, if $\nu_s(L_1) > \nu_s(L_2)$ for a given SRC s, then it might be possible that, for a feasible extension of L_1 and L_2, the dual value σ_s is subtracted from the reduced cost when extending L_1 but not when extending L_2. Jepsen et al. (2008) develop a stronger dominance rule which replaces conditions (16) and (31) by:

$$\bar{c}(L_1) - \sum_{s\in\Theta_{1,2}} \sigma_s \leq \bar{c}(L_2), \tag{32}$$

where $\Theta_{1,2} \subseteq \Theta$ is the subset of SRCs for which $\nu_s(L_1) > \nu_s(L_2)$.

Because Jepsen et al. (2008) showed that separating the SRCs is $\mathcal{NP}$-hard, SRCs are separated by full or restricted enumeration depending on the maximum cardinality of the sets C.

Finally, note that Baldacci, Mingozzi, and Roberti (2011) implement a weak version of the 3-SRCs (weak-SRCs). For a given set C, a weak-SRC only contains variables for which the associated route traverses at least one arc $(i, j) \in A$ such that $i, j \in C$. The weak-SRCs have the advantage of being robust.

Limited-memory SRCs: One way to reduce the impact of the SRCs on the labeling algorithms would be to re-define the cuts such that, for a larger number of labels L and SRCs s, $\nu_s(L) = 0$. Pecin et al. (2017b) exploit this observation to introduce a weaker version of the SRCs called the limited-memory SRCs (lm-SRCs). A lm-SRC is defined by a set C, a multiplier γ and a memory set M ($C \subseteq M \subseteq V'$). It writes as:

$$\sum_{r \in \Omega} \alpha(C, M, \gamma, r)\lambda_r \leq \lfloor \gamma |C| \rfloor, \tag{33}$$

where each coefficient α is computed as a function of (C, M, γ, r) and satisfies $\alpha(C, M, \gamma, r) \leq \lfloor \gamma \sum_{i \in C} a_i^r \rfloor$. On the one hand, if $M = V'$, the lm-SRC is identical to the corresponding SRC. On the other hand, if $M \subset V'$, the lm-SRC s defined by (C_s, M_s, γ) may be weaker but its dual value σ_s can be handled more easily in the pricing problem. In this case, the value of $\alpha(C_S, M_s, \gamma, r)$ is determined by applying along route r a modified version of the labeling algorithm that handles the SRCs' dual values described above. Every time that a vertex $j \in C_s$ is visited and $\nu_s(L') = 0$, the value of α increases by one and σ_s is subtracted from the reduced cost. However, every time that a vertex $j \notin M_s$ is visited, the resource value $\nu_s(L')$ is reset to 0, i.e., the previous visits to customers in C_s are forgotten. In this case, the value of α has less chances to increase than in the full-memory case and, thus, $\alpha(C_s, M_s, \gamma, r) \leq \lfloor \gamma \sum_{i \in C_s} a_i^r \rfloor$.

Observe that the smaller the sets M are, the faster the labeling algorithm and the weaker the cuts will be. Given a regular SRC associated with a set C and a multiplier γ that is violated by the current solution of the MP, Pecin et al. (2017b) convert this SRC into a lm-SRC by finding the smallest possible set M that yields the same violation. The procedure to compute M initially sets M equal to C. Then it adds to M every vertex $i \in V' \setminus M$ such that i is visited between two customers in C in a route r for which $\lfloor \gamma \sum_{i \in C} a_i^r \rfloor > 0$ and $\lambda_r > 0$ in the MP solution.

Despite the success achieved by lm-SRCs in the solution of the CVRP, they still present scalability issues for solving large-scale instances with loose structural constraints. In such instances, long feasible routes are generated, leading to large memory sets and intractable pricing problems. To address this issue, Pecin et al. (2017a) introduce new lm-SRCs where their memory is defined in terms of arcs instead of vertices. This gives rise to two families of lm-SRCs, namely, limited-vertex-memory SRCs (lvm-SRCs) and limited-arc-memory SRCs (lam-SRCs). A lam-SRC is expressed as in (33) except that the vertex memory M is replaced by an arc memory AM. A coefficient $\alpha(C, AM, \gamma, r)$ in a lam-SRC is computed similarly to a coefficient $\alpha(C, M, \gamma, r)$ in a lvm-SRC: previous visits to customers in C are forgotten when an arc $(i, j) \notin AM$ is traversed. Consequently, when comparing a lvm-SRC defined for a triplet (C, M, γ) and a lam-SRC defined for a triplet (C, AM, γ) with $AM \subset \{(i, j) \in A \,|\, i, j \in M\}$, the latter should have less impact on the labeling algorithm but be weaker. Note that a lvm-SRC defined for a customer set C, a vertex memory M and a multiplier γ can be seen as a special case of the lam-SRC with the same C and γ, and the arc memory $AM = \{(i, j) \in A \,|\, i, j \in M\}$.

Like for the lvm-SRCs, Pecin et al. (2017a) define the arc memory AM of a lam-SRC such that it is of minimal size and preserves the level of violation of the corresponding SRC. Set AM is composed of the arcs traversed between two visits to customers in C in a route r for which $\lfloor \gamma \sum_{i \in C} a_i^r \rfloor > 0$ and $\lambda_r > 0$ in the MP solution. During the separation of the lam-SRC cuts, it may happen that a violated cut is found for the same set C and multiplier γ of a previously generated cut but with different arc memories AM and AM'. These two memories can be merged for defining a unique cut with arc memory $AM \cup AM'$. A similar strategy can be applied for the lvm-SRCs.

Elementary cuts: Balas (1977) introduces the elementary cuts as logical implications of the set partitioning constraints (2). These cuts ensure that, if a route $r \in \Omega$ is chosen in a solution and does not visit a customer $i \in V'$, then no route visiting i as well as at least one customer visited by r can be choosen. Pecin et al. (2017a) propose a new family of valid inequalities that dominate those of Balas (1977). When the set of routes contains only elementary ones, this dominance is strict. Given a customer subset $C \subset V'$, a customer

$i \in V' \setminus C$, and multipliers $\gamma_i^C = (|C|-1)/|C|$ and $\gamma_j^C = 1/|C|$, $\forall j \in C$, these new rank-1 Chvátal-Gomory inequalities, also called elementary cuts in Pecin et al. (2017a), are given by:

$$\sum_{r \in \Omega} \left\lfloor \gamma_i^C a_i^r + \sum_{j \in C} \gamma_j^C a_j^r \right\rfloor \lambda_r \leq 1. \tag{34}$$

Another advantage of these rank-1 Chvátal-Gomory cuts over those proposed by Balas (1977) is that all the theory developed for the other families of rank-1 Chvátal-Gomory inequalities can be re-applied, namely, the label definition, the dominance rule, and the limited memory mechanisms. Furthermore, the elementary cuts with $|C| \in \{2,3,4\}$ correspond to general rank-1 Chvátal-Gomory cuts (Petersen, Pisinger, and Spoorendonk 2008) with $|C| \in \{3,4,5\}$ and respective vector γ of multipliers $(1/2,1/2,1/2)$, $(2/3,1/3,1/3,1/3)$ and $(3/4,1/4,1/4,1/4,1/4)$ (and their permutations). The latter are among the optimal multipliers identified by Pecin et al. (2017c). Contrarily to the other rank-1 Chvátal-Gomory cuts presented so far, a heuristic procedure based on local search is developed by Pecin et al. (2017a) for separating the elementary cuts.

Strengthened capacity cuts: Baldacci, Christofides, and Mingozzi (2008) introduce the strengthened capacity cuts (SCCs) as a strengthened version of the RCCs. If one applies relation (7) to inequality (25), a robust inequality of the form $\sum_{r \in \Omega} \rho_S^r \lambda_r \geq k(S)$ is obtained, where $\rho_S^r = \sum_{(i,j) \in \delta^-(S)} b_{ij}^r$ is the number of times that route $r \in \Omega$ enters set S. Note that, even if Ω contains only elementary routes, ρ_S^r may be greater than 1 because a route may enter and leave S more than once. Baldacci, Hadjiconstantinou, and Mingozzi (2004) redefine ρ_S^r as a binary parameter indicating whether route r visits at least one customer in S. With this new definition, a SCC can be expressed as:

$$\sum_{r \in \Omega} \rho_S^r \lambda_r \geq k(S). \tag{35}$$

When compared to RCCs (25), SCCs are stronger because they are not negatively affected by routes entering set S more than once. They are, however, non-robust cuts which require one additional binary resource in the label definition for each cut to indicate whether a route has already entered into the corresponding set S. When a route visits a customer in S for the first time, the dual value of the associated cut is subtracted from the reduced cost. The dominance rule must take these additional resources into account in a similar fashion as the resources for the SRCs.

To generate violated SCCs, Baldacci, Christofides, and Mingozzi (2008) call the heuristic CVRPSEP separation routines (Lysgaard 2003) to identify violated RCCs that are converted into SCCs. Poggi and Uchoa (2014) observe that, if there exists a set S' such that $S' \subset S$ and $k(S') = k(S)$, the SCC defined over S' dominates the one defined for S and, thus, only the cut associated with the former set needs to be added to the MP. This aspect is important in an attempt to limit the number of non-robust cuts added to the MP.

Clique cuts: The clique inequalities are well-known valid inequalities for the set partitioning problem (Balas and Padberg 1976) which induce facets. They are defined over an undirected conflict graph $G' = (\Omega, E')$, in which an edge between two vertices $r_1, r_2 \in \Omega$ exists if r_1 and r_2 visit both a customer $i \in V'$, i.e., $a_i^{r_1} \geq 1$ and $a_i^{r_2} \geq 1$. A clique $\hat{\Omega}$ in G' is a set of vertices that are conflicting pairwise and, thus, the corresponding routes are not allowed to appear simultaneously in a feasible solution to the problem. Therefore, a clique $\hat{\Omega}$ yields the following valid inequality:

$$\sum_{r \in \hat{\Omega}} \lambda_r \leq 1. \tag{36}$$

Spoorendonk and Desaulniers (2010) explore the application of clique cuts in a BPC algorithm for the VRPTW. They devise a modified labeling algorithm that handles the dual values of the clique cuts. This new labeling algorithm relies on an approximate clique representation that employs a minimal set $\chi_{min}(\hat{\Omega})$ of conflicting rows. It requires adding for each clique cut defined for a set $\hat{\Omega} \subseteq \Omega$, $|\chi_{min}(\hat{\Omega})|+1$ binary resources to the definition of a label to detect when a route contributes to this cut. The dominance rule also needs to be modified. A greedy heuristic is used to separate the cuts.

k-cycle elimination cuts and strong degree cuts: If set Ω contains non-elementary routes, some valid inequalities may be used to impose elementarity. Indeed, Contardo and Martinelli (2014) present the k-cycle elimination cuts (k-CECs) that aim at preventing routes containing cycles of length $k \geq 2$ to be part of a MP solution. Given a route $r \in \Omega$ and a vertex $j \in V'$, let α_j^{kr} be a parameter indicating the number of times that route r visits vertex j either for the first time or after at least k vertices since the last visit to j. A k-CEC associated with vertex j isß expressed as:

$$\sum_{r \in \Omega} \alpha_j^{kr} \lambda_r \geq 1. \tag{37}$$

The k-CECs ensure that no route $r \in \Omega$ visiting a vertex $j \in V'$ more than once and such that $\alpha_j^{kr} < a_j^r$ will be in the solution of the MP. To deal with the dual values of the generated k-CECs in the labeling algorithm, a new resource is added to the label definition for each generated k-CEC. For a cut associated with vertex j, this resource takes value k until reaching j for the first time. It is reset to 0 at every visit to j. Then, it counts the number of vertices different from j that are visited consecutively. The dual value associated with this cut is subtracted from the reduced cost every time that vertex j is visited and the value of this resource is greater than or equal to k. As explained in Contardo and Martinelli (2014), the dominance rule must be modified. These cuts are separated by inspection.

If one considers $k = \infty$, we obtain the strongest k-CECs, called strong degree cuts (SDCs), which are introduced by Contardo, Cordeau, and Gendron (2014) for the capacitated location-routing problem (CLRP). Let α_j^r be a binary parameter equal to 1 if route $r \in \Omega$ visits vertex $j \in V'$ at least once and 0 otherwise. The SDC for a vertex j is:

$$\sum_{r \in \Omega} \alpha_j^r \lambda_r \geq 1. \tag{38}$$

For this case, the additional resource for a SDC associated with a customer j simply indicates whether or not vertex j has already been visited. The associated dual value is subtracted from the reduced cost only at the first visit to customer j.

Contardo, Cordeau, and Gendron (2014) show that, even when non-elementary routes can be generated by the pricing problem, adding SDCs yields a lower bound equal to the lower bound obtained when considering only elementary routes. Contardo, Desaulniers, and Lessard (2015) further show empirically that combining SDCs and ng-routes is a very effective strategy to reach this bound.

3.3 Branching

Here, we present the most commonly used branching strategies in BPC algorithms for VRPs.

3.3.1 Branching on arcs/edges

Desrosiers, Soumis, and Desrochers (1984) introduce branching on arc-flow variables x_{ij}, $(i,j) \in A$, for the VRPTW. It is probably the most popular strategy due to its simplicity, ease of implementation, and robustness. In BPC algorithms, setting $x_{ij} = 0$ can be imposed by removing all variables λ_r, $r \in \Omega$, with $b_{ij}^r > 0$ that are present in the current RMP, and arc (i,j) from arc set A to avoid generating such route variables. Any other decision made on an arc-flow variable x_{ij} can be re-written using relation (7) in terms of the route variables λ_r, $r \in \Omega$, and added to the MP. Then, the associated dual variable can be incorporated in the modified cost of the arc (i,j) when solving the pricing problem.

When arc (i,j) is associated with a set partitioning constraint (2), i.e., $i \in V'$ or $j \in V'$, the addition of a new constraint to the MP can be avoided. Indeed, in this case, x_{ij} is binary and the associated branching decisions are $x_{ij} = 0$ and $x_{ij} = 1$. The former decision is treated as mentioned above. The latter decision means that arc (i,j) must be in the solution. To enforce this, we remove from the current RMP, all route variables associated with a route traversing an arc (i,k), with $k \neq j$, if $i \in V'$, and an arc (k,j), with $k \neq i$, if $j \in V'$. All these arcs must be removed from arc set A to ensure that vertex j will always be visited immediately after vertex i in any route generated by the pricing problem.

When the problem is defined over an undirected graph like the CVRP, branching on edge-flow variables can be applied. Indeed, most of the theory described above can be employed. However, given that, for such a problem, the pricing problem is often better defined as a ESPPRC on a directed graph, the flow x'_{ij} on an edge $\{i, j\}$ is computed as $x'_{ij} = x_{ij} + x_{ji}$, where x_{ij} and x_{ji} are the flows on the arcs (i, j) and (j, i) in this directed graph. In this case, setting $x'_{ij} = 0$ can be imposed as above. However, imposing $x'_{ij} = 1$ requires adding a constraint in the MP.

3.3.2 Branching on the number of vehicles

When the number of vehicles used in a solution to the MP is fractional, one can branch on this number which can be expressed as $\sum_{j \in V'} x_{0j}$, i.e., the total flow on the arcs leaving the origin depot. Note that this quantity is also equal to the sum of all the route variables when there is a single depot. Such a branching decision is implemented through the addition of a constraint in the MP. In the pricing problem, its dual value needs to be considered in the modified cost of all the arcs leaving the depot.

This branching strategy has often a significant impact on the size of the search tree. Given that there can exist fractional solutions in which the number of vehicles used is integer, it is not sufficient to explore to whole search tree and must be combined with another branching strategy. In fact, Desrochers, Desrosiers, and Solomon (1992) branch in priority on the number of vehicles used and to branch on the arc-flow variables when this number is integer.

3.3.3 Branching on cutsets

Even if branching on arcs/edges has been largely used in BPC algorithms for VRPs, it has the disadvantage of inducing only local changes to the current fractional solution. For this reason, when implementing a BC algorithm to solve the CVRP, Augerat et al. (1998) branch on cutsets as an attempt to obtain larger perturbations. Let us define this branching strategy in terms of the edge-flow variables x_e, $e \in E$, where E is the edge set. Let $S \subseteq V'$ be a subset of customers, called a *cutset*. The total flow entering S is given by $\sum_{e \in \delta(S)} x_e$, where $\delta(S)$ denotes the set of edges containing exactly one extremity in S. Assuming that at least one route enters S, we can write $\sum_{e \in \delta(S)} x_e = 2\kappa(S) + \phi(S)$, where $\kappa(S)$ is a non-negative integer and $0 \leq \phi(S) < 2$. In an integer solution, $\phi(S) = 0$. Consequently, if $\phi(S) > 0$, then one can branch on cutset S by imposing $\sum_{e \in \delta(S)} x_e \leq 2\kappa(S)$ on one branch and $\sum_{e \in \delta(S)} x_e \geq 2\kappa(S) + 2$ on the other. These decisions can be imposed as constraints in the MP. Note that most works focus on sets S with $\kappa(S) = 1$. Note also that, in their BPC algorithm, Pecin et al. (2017b) apply equivalent branching decisions by considering only the arcs entering into a cutset in a directed graph.

The challenge with this branching strategy is the exponential number of cutsets that can be used to define the branching decisions. In this regard, several simple heuristics are designed by Augerat et al. (1998), Naddef and Rinaldi (2002), Lysgaard, Letchford, and Eglese (2004), Fukasawa et al. (2006) and Pecin et al. (2017b) to select the cutsets to branch on. In general, these heuristics seek to yield balanced search trees and a higher impact on the generated lower bounds.

3.3.4 Branching on resource windows

Assuming that travel and service times are integer, Gélinas et al. (1995) branch on time windows but a similar branching scheme can be devised for other resources such as load. In a fractional solution to the MP, there might exist several routes visiting the same customer $i \in V'$ but with different start of service times. Under certain conditions, splitting the time window $[e_i, l_i]$ of customer i in two disjoint time windows $[e_i, t]$ and $[t + 1, l_i]$, with $t \in [e_i, l_i - 1]$, can make some of these routes infeasible for window $[e_i, t]$ and others infeasible for $[t+1, l_i]$. Branching on time windows consists of finding a customer for which splitting its time window in two sub-windows would eliminate the current fractional solution in both branches, and of determining how it should be split. These two intertwined choices must be carefully made to devise an efficient branching strategy. Once customer i and time t are identified, the branching decisions are applied as follows. In the branch associated with $[e_i, t]$, we replace time window $[e_i, l_i]$ by $[e_i, t]$ in the pricing problem and eliminate

from the current RMP all routes in which customer i is visited outside $[e_i, t]$. In the other branch, we proceed similarly but with the new time window $[t+1, l_i]$.

Dell'Amico, Righini, and Salani (2006) devise a similar branching technique on resources that are used to handle the vehicle capacity in a VRP with simultaneous pickups and deliveries (see Section 4.10.5). Finally, Christiansen and Lysgaard (2007) branch on the expected cumulative demand in the context of the VRP with stochastic demands (Section 4.9.1).

3.3.5 Strong branching

Some recent successful BPC algorithms rely on strong branching (Fukasawa et al. 2006, Pecin et al. 2017a,b). The idea behind strong branching is to quickly evaluate the impact of a set of branching candidates (arcs, time windows, subset of customers, etc.) on the lower bounds that would be obtained in each child node and to select the best one according to some criterion. For example, when applying arc branching at a given node, a subset of arcs $A^C \subset A$ that are candidates for branching is determined and, for each arc a in A^C, the corresponding branching decisions are successively applied to compute the lower bounds lb_a^- and lb_a^+ that would be achieved in both child nodes if this arc was selected for branching. Then, after computing all these lower bounds, one can choose the best candidate as an arc $a^* \in \arg\max_{a \in A^C} \min\{lb_a^-, lb_a^+\}$ and add to the search tree only the child nodes associated with the decisions for arc a^*.

Strong branching can reduce significantly the number of nodes to explore in the search tree but often at the expense of increasing the time to select the branching candidate at each node. Consequently, it is common practice to limit the size of the set of candidates to evaluate and to compute approximate lower bounds in the strong branching selection process. Similar to the strong branching scheme of Lysgaard, Letchford, and Eglese (2004) involving branching decisions on cutsets, Fukasawa et al. (2006) select, at each node of the search tree requiring branching, between 5 and 10 candidate cutsets S according to a criterion based on the flow entering S. Then, the lower bounds for each candidate is evaluated heuristically by performing a few CG iterations.

In Irnich (2010), the size of the candidate set is defined dynamically during the exploration of the search tree and is proportional to the relative optimality gap. Therefore, it is more selective at the beginning of the algorithm when no good solutions have been found and at the top of the search tree when the optimality gap is still high, and evaluates less candidates when the gap gets smaller. The lower bounds for each candidate are, however, computed by an exact CG algorithm.

For the CVRP, Pecin et al. (2017b) assess the impact of branching on cutsets by proposing an aggressive hierarchical hybrid strong/pseudocost branching, that consists of three phases: i) candidate set construction, ii) refinement, and iii) strong branching. In the first phase, a set of candidate cutsets is found. Half of them are chosen based on their pseudocosts (average lower bound increases when the cutset was previously selected for evaluation) and the other half are new cutsets. As in Irnich (2010), the size of the candidate set depends on the relative optimality gap. During the refinement phase, lower bound values lb^- and lb^+ are heuristically estimated for each candidate by only considering the variables in the current RMP. These quick evaluations allow to select a smaller set of candidates which are then better evaluated in the last phase using a heuristic CG algorithm.

3.4 Using upper bounds

In this section, we discuss two techniques that can be exploited in BPC algorithms for VRPs when good lower and upper bounds are available. The first technique consists of fixing to zero the value of certain variables in order to reduce the size of the model. The second is based on the enumeration of elementary routes to help close the integrality gap at a node of the search tree. This latter technique can be seen as an alternative to branching.

3.4.1 Variable fixing by reduced cost

Variable fixing can be performed in different ways. Here, we concentrate on variable fixing by reduced cost which requires a lower and an upper bound on the optimal value of the problem. The general idea behind variable fixing is as follows (see, e.g., Nemhauser and Wolsey 1988). Let $P := \{\min\ c^\top x : Ex = b, x \in \mathbb{Z}^n_+\}$ be an integer linear program, and $\bar{z}$ an upper bound on its optimal value, derived from a feasible solution. Also, let $D := \{\max\ \pi^\top b : \pi E \leq c\}$ be the dual problem associated with the linear relaxation of P and π a feasible dual solution to D of cost $\underline{z} = \pi^\top b$. The reduced cost $\bar{c}_j$ of a generic variable x_j with respect to π is given by $\bar{c}_j = c_j - \pi^\top E_j$, where E_j denotes the coefficient column of x_j in matrix E. Variable x_j can be fixed to zero if $\bar{c}_j > \bar{z} - \underline{z}$ and can thus be removed from the problem.

In BPC algorithms, variable fixing is usually applied after solving the MP at each node of the search tree, when an optimal dual solution π becomes available. However, it is not directly applied to route variables λ_r, $r \in \Omega$, because, as discussed in Section 2.4, it would require employing complex mechanisms to forbid in the pricing problem the (re-)generation of the routes associated with the removed variables. For this reason, variable fixing is rather applied to implicit arc variables (Hadjar, Marcotte, and Soumis 2006, Irnich et al. 2010, Contardo and Martinelli 2014), yielding the simultaneous elimination of a large number of route variables, namely, all those associated with a route traversing an arc to be removed. Besides, removing arcs from the network makes the pricing problem easier to solve. However, there is an inconvenient to this practice: the reduced costs of the arc-flow variables x_{ij} are not directly available in a CG algorithm. Some approaches such as the addition of coupling constraints of the form (7) to the MP (Poggi de Aragão and Uchoa 2003, Lübbecke and Desrosiers 2005) and the solution of the pricing problem directly as a linear problem (Walker 1969) have been proposed in the literature to overcome this difficulty. In the following, we highlight the approaches of Irnich et al. (2010) and Contardo and Martinelli (2014), which are designed for VRPs.

Irnich et al. (2010) fix arc-flow variables to zero as follows. Let $\Omega_{ij} \subset \Omega$ be the subset of feasible routes traversing arc $(i,j) \in A$. The reduced cost $\bar{c}_{ij}$ of arc-flow variable x_{ij} can then be computed as $\bar{c}_{ij} = \min_{r\in\Omega_{ij}} \bar{c}_r - \min_{r'\in\Omega} \bar{c}_{r'}$. Because $\min_{r'\in\Omega} \bar{c}_{r'} \leq 0$ at any CG iteration, this result implies that, if $\tilde{c}_{ij} = \min_{r\in\Omega_{ij}} \bar{c}_r > \bar{z} - \underline{z}$, then arc (i,j) can be removed from G. Alternatively, if lb_{ij} is a lower bound on $\tilde{c}_{ij}$, then $lb_{ij} > \bar{z} - \underline{z}$ is a sufficient condition to remove arc (i,j). Such a lower bound can be computed for all arcs at once by solving the pricing problem using an exact monodirectional labeling algorithm (see Hadjar, Marcotte, and Soumis 2006, Irnich et al. 2010). To compute $\tilde{c}_{ij}$ exactly and remove the maximum number of arcs, Irnich et al. (2010) find feasible shortest paths from vertex 0 to all other vertices using an exact forward labeling algorithm as well as feasible shortest paths for all vertices to vertex $n+1$ using an exact backward labeling algorithm. Given that two full labeling passes need to be performed, this process might be time-consuming. Instead, relaxed shortest paths such as *ng*-paths can be computed, yielding only lower bounds on the arc reduced costs and possibly less removed arcs depending on the quality of these bounds. The state-of-the-art BPC algorithm of Pecin et al. (2017a) eliminates arc-flow variables in the same fashion.

Although Contardo and Martinelli (2014) developed a BPC algorithm for the multi-depot VRP under capacity and route length constraints, their algorithm is also very effective at solving CVRP instances. It proceeds in two phases and variable fixing is performed in each phase. First, the algorithm solves the linear relaxation of an edge-flow formulation to quickly generate a lower bound on the optimal value of the problem. This bound is then improved by generating cutting planes. When no more cuts are found, variable fixing is performed on the edge-flow variables, i.e., edges are removed from the network. In the second phase, a BPC algorithm is applied to solve the problem using the reduced network in the pricing problem. In this algorithm, variable fixing is performed as in Irnich et al. (2010), except that the arc reduced costs are underestimated because the duals of the non-robust cuts are not fully considered in the labeling process.

3.4.2 Route enumeration

Ideally, one would like to be able to enumerate all feasible routes and solve directly model (1)–(3) using a mixed integer programming solver. This is unpractical given that the number of feasible routes grows exponentially with the size of the instance. However, given an upper bound $\bar{z}$ on the optimal value obtained from a feasible solution s^* and a dual solution π providing a lower bound $\underline{z}$ like in variable fixing, enumerating

the subset Ω'' of all the routes $r \in \Omega$ such that $\bar{c}_r < \bar{z} - \underline{z}$ and solving model (1)–(3) restricted to the routes in Ω'' is sufficient to either prove that s^* is optimal or find an optimal solution with a cost less than $\bar{z}$. This approach, called route enumeration, is presented by Baldacci, Christofides, and Mingozzi (2008) and Baldacci, Mingozzi, and Roberti (2011), who propose exact algorithms for solving the CVRP and the VRPTW. These algorithms are not BPC algorithms, as no branching is performed. In both algorithms, an additive bounding procedure which relies, among others, on CG and cutting planes is used to compute a high-quality lower bound. Route enumeration is then performed using a labeling algorithm similar to the ones applied for solving the pricing problem. Finally, if all routes can be enumerated, the MIP restricted to the subset of enumerated routes is solved by a commercial MIP solver, thus benefitting from all cutting-edge tools offered by this solver. These algorithms are very efficient when the gap $\bar{z} - \underline{z}$ is small. Otherwise, they might fail to enumerate all required routes and simply abort during the enumeration process due to a lack of memory.

To overcome this drawback, it is possible to employ a hybrid strategy which combines route enumeration and branching (Pessoa, Poggi de Aragão, and Uchoa 2008, Contardo and Martinelli 2014, Pecin et al. 2017a,b). In the BPC algorithm of Pessoa, Poggi de Aragão, and Uchoa (2008), route enumeration is performed at each node of the BB search tree. Every time that the MP is solved, possibly after adding cuts, route enumeration is attempted. If the number of generated labels or the number of generated routes exceeds a predefined threshold at a node, enumeration is stopped and branching is performed. Otherwise, the model associated with this node and restricted to the set of enumerated routes is solved using a MIP solver, and no branching is necessary. Because the optimality gap tends to decrease with the depth in the tree, route enumeration can eventually work, reducing the size of the tree and the total computational time.

Contardo and Martinelli (2014) perform route enumeration in a more aggressive way. Indeed, they allow to generate a large number of routes (e.g., up to 5 millions) that cannot be handled directly by a MIP solver. Once the MP is solved at a node of the search tree, the algorithm proceeds to route enumeration. If too many labels or too many routes are generated, the process is aborted. If the number of enumerated routes is small enough, the resulting model is solved by a MIP solver. Otherwise, the routes are stored in a pool and the algorithm continues by adding cuts and fixing variables. At this point, when cuts are added, the MP is re-optimized by CG but the pricing problem is solved by inspecting the pool of routes. This allows to separate more non-robust cuts without impacting dramatically the complexity of the pricing algorithm. Pecin et al. (2017b,a) implement route enumeration as in Contardo and Martinelli (2014). They, however, resort to branching when route enumeration is not possible.

3.5 Stabilizing dual variable values

CG algorithms are well known to suffer from convergence issues, which are mainly due to the degenerate nature of the MP and to the instability of the dual variable values from one iteration to the next (Vanderbeck 2000, Lübbecke and Desrosiers 2005). On the one hand, degenerate MPs often yield multiple optimal dual solutions and makes it difficult to prove the optimality of the primal solution. Moreover, the algorithm may perform many iterations without any or almost any improvement of the objective function value (*tailing-off effect*). On the other hand, dual variable instability causes the algorithm to generate columns unlikely to be in an optimal solution, even if some good columns have already been generated.

Simple modifications to the MP may help the convergence of a CG algorithm. For instance, by avoiding redundant constraints in the MP or removing non-active constraints, the degree of degeneracy may be reduced. In addition, if one formulates the VRP as a set-covering problem instead of a set-partitioning problem, i.e., replacing equalities (5) in the MP by inequalities, the dual solution space is cut in half (the corresponding dual variables become non-negative) and the dual variables are typically more stable (Rousseau, Gendreau, and Feillet 2007, Feillet 2010).

Dual variable stabilization strategies have also been developed to overcome these convergence issues. In general, stabilization techniques seek to: i) limit the distance traveled by the variables in the dual space, either by restricting the dual space or penalizing long displacements of the dual points (see, e.g. Marsten, Hogan, and Blankenship 1975, Ben Amor, Desrosiers, and Frangioni 2009); ii) avoid the generation of extreme dual solutions (Rousseau, Gendreau, and Feillet 2007); or iii) alleviate the impact of bad dual values (Pessoa et al.

2018). The first family of stabilization strategies either force the dual variables to remain within relatively small intervals around the current dual values or allow to go outside these intervals at the expense of paying a penalty. The intervals and penalties are adjusted dynamically throughout the CG algorithm. Such methods can yield substantial speedups if good initial dual information is available. However, this is not the case, in general, for VRPs.

Extreme dual solutions may be harmful to CG algorithms because they can lead to unrealistic route reduced costs and the generation of routes that have little chance to be part of an optimal MP solution. Thus, it may be preferable to use interior dual points to reduce the number of CG iterations. Naturally, these points can be obtained directly by solving at each iteration the RMP using an interior point method. Instead, Rousseau, Gendreau, and Feillet (2007) generate dual interior points by using convex combinations of several dual extreme points. At a given iteration, these extreme points are computed by solving the RMP several times after perturbing the right-hand side vector each time.

To reduce the impact of poor dual values and avoid large dual variable oscillations, Pessoa et al. (2018) devise a dual price smoothing technique, which considers a transformed dual vector $\tilde{\pi} = \alpha\hat{\pi} + (1-\alpha)\pi$ when solving the pricing problem. In this expression, parameter $\alpha \in (0, 1]$ indicates the level of smoothing, and vectors π and $\hat{\pi}$ are, respectively, the current dual vector and a vector containing dual information from previous iterations. Vector $\hat{\pi}$ can be the best dual vector obtained so far (Wentges 1997) or a weighted sum of dual vectors computed in previous iterations (Neame 1999). This stabilization technique proceeds by solving at each CG iteration the pricing problem defined with the vector $\tilde{\pi}$ of dual values. Three outcomes may happen: the labeling algorithm finds 1) routes that have a negative reduced cost with respect to both $\tilde{\pi}$ and π; 2) routes that have a negative reduced cost with respect to π but not with respect to $\tilde{\pi}$; 3) no routes with a negative reduced cost with respect to π. In the first two cases, the computed routes can be added to the current RMP. In the second case, the value of α is decreased as the algorithm converges towards optimal dual values. Finally, in the third case, the value of α is also decreased and the pricing problem is solved again with the updated vector $\tilde{\pi}$.

As mentioned in Rousseau, Gendreau, and Feillet (2007) and Pessoa et al. (2018), the use of a stabilization technique tends to increase the difficulty of solving the pricing problem, especially in the first CG iterations. Indeed, when no stabilization is used, the dual values are often badly distributed among the customers, yielding non-attractive customers that are rapidly disregarded during the search for negative reduced cost routes. In practice, this aspect is not a major issue because the quality of the generated routes outweighs this disadvantage.

4 Contributions to specific VRPs

In this section, we describe contributions that are specific to some VRP variants. Given the large number of contributions, we have chosen to summarize the general ideas of these specific contributions and to avoid mentioning every feature of each algorithm.

This section is divided according to the following problem features: heterogeneous fleet and multiple depots (Section 4.1), profits (Section 4.2), soft time windows (Section 4.3), multiple trips (Section 4.4), split services (Section 4.5), time dependency (Section 4.6), cumulative costs (Section 4.7), environmental aspects (Section 4.8), uncertainty (Section 4.9), and pickups and deliveries (Section 4.10).

4.1 Heterogeneous fleet and multiple depots

One of the most direct variants of the CVRP is the heterogeneous fleet VRP (HFVRP) in which vehicles of different types are available. Let $\mathcal{K}$ be the set of vehicle types. These types may differ by their capacity, fixed cost and traveling costs. The HFVRP consists of determining the fleet of vehicles to use and designing feasible routes for them so as to service a set of customers at minimum total cost. This cost is given by the sum of the vehicle fixed costs and the variable traveling costs.

Given the differences between the vehicle types, a basic BP algorithm for the HFVRP requires one pricing problem for each type $k \in \mathcal{K}$, increasing the complexity of the pricing step. Choi and Tcha (2007) develop

the first CG-based algorithm for this problem which relies on the following procedure to reduce the number of pricing problems. Let z^* be the cost of a feasible solution obtained, e.g., by a heuristic. For each vehicle type $k \in \mathcal{K}$, compute a lower bound $\underline{z}_k$ on the optimal value of the problem when forcing the utilization of at least one vehicle of type k. If $\underline{z}_k > z^*$, then no vehicle of type k can be used in an optimal solution and the corresponding pricing problem can be discarded. To compute rapidly a lower bound $\underline{z}_k$, the authors solve a small-sized integer program that aims at minimizing the total fixed cost incurred by a fleet that has sufficient capacity to cover the total demand. This lower bound might be useful to discard some vehicle types only if the fixed costs are very large compared to the traveling costs.

Pessoa, Poggi de Aragão, and Uchoa (2009) design a BPC algorithm for the HFVRP. When vehicle types may incur different traveling costs, multiple pricing problems are necessary. Otherwise, the authors show how to use a single pricing problem. They also introduce new families of robust cuts defined over CI variables, namely, homogeneous extended capacity cuts and strengthened RCCs, and also applied robust triangle clique cuts. Even if the HFVRP is usually defined over an undirected graph, the proposed algorithm relies on directed graphs in the pricing problems which are needed to handle the cuts defined over the CI variables. These networks introduce symmetry while solving the pricing problems because the same route can be traversed in both directions at the same reduced cost unless the contributions to the cuts differ with the direction. To break this symmetry, Pessoa, Poggi de Aragão, and Uchoa (2009) impose that the index of the first customer visited along a route must be less than that of its last customer and adapt their labeling algorithm to handle this requirement.

Pessoa, Sadykov, and Uchoa (2018) propose a new BPC algorithm for the HFVRP that combines the most recent methodological advances for the CVRP and the VRPTW. By employing a new pseudo-polynomially large extended formulation, stronger extended capacity cuts are devised. Furthermore, they exploit the structure of the HFVRP to propose the concept of vehicle-type-dependent memory for the SRCs (see Section 3.2.2). This new idea yields a sharper definition of the concept of lm-SRCs by defining the coefficients of the route variables in the cuts according to their associated vehicle type. Each cut has, thus, a different memory for each vehicle type and this memory is smaller than the memory that would not depend on the vehicle type. Consequently, the impact on the time required to solve each pricing problem is more limited. On the other hand, these cuts are weaker and additional rounds of cut separation may be required to achieve the same final lower bound. Finally, Pessoa, Sadykov, and Uchoa (2018) also exploit the structure of the HFVRP to develop a progressive route enumeration scheme. Indeed, given that enumerating routes for small-capacitated vehicles is easier, in general, than for large-capacitated vehicles, enumeration can be possible only for a subset of the vehicle types. Thus, when enumeration is successful for a type, the enumerated routes are added to the RMP or stored in a pool (see Section 3.4.2), the corresponding pricing problem is removed, and all the generated cuts can be lifted with respect to these route variables.

The multi-depot VRP (MDVRP) is also a direct extension of the CVRP, where the available vehicles are assigned to a set of depots. A route for a vehicle of a given depot must start and end at this depot. The MDVRP can be seen as a special case of the HFVRP by associating each depot with a vehicle type.

Baldacci and Mingozzi (2009) design a general solution method for solving the HFVRP and several of its special cases including the MDVRP. This method extends the framework introduced by Baldacci, Christofides, and Mingozzi (2008) for the CVRP (see Section 3.4.2) to deal with different vehicle types. It also incorporates mechanisms designed for the HFVRP, namely, a relaxation involving binary variables assigning customers to vehicle types and rules to resize the vehicle fleet (i.e., to update lower and upper bounds on the number of vehicles of each type). The first mechanism is especially effective when the fixed cost contribution to the total cost is significant. The overall algorithm consists of solving three different relaxations and keep the best lower bound obtained to perform route enumeration.

Bettinelli, Ceselli, and Righini (2011) tackle a multi-depot heterogeneous VRPTW with limited route duration, where the duration of a route is computed as the difference between the return time to the depot and the departure time from it. The latter is a decision variable, as some waiting time along the route may be avoided by delaying the departure from the depot. To solve this problem, Bettinelli, Ceselli, and Righini (2011) conceive a BPC algorithm with one pricing problem per vehicle type and depot. All pricing problems are, however, solved at once using a single bidirectional labeling algorithm which handles two additional

resources to impose a maximal route duration. This labeling algorithm relies on labels that contains, in theory, duplicated components for each depot. Given that two paths associated with different depots and covering the same customers in the same order only differ by their initial arcs, each label rather contains components for a single depot and, for each depot, a pointer to the initial arc of the path if it were to start at this depot. Since it is easy to retrieve the component values for each depot with these pointers, label storage requires much less memory. To deal with the multiple vehicle types that only differ by the fixed cost and the vehicle capacity, the labeling process is performed considering the largest vehicle capacity. Then, during the joining phase of the bidirectional search algorithm, the feasibility rules are verified for each vehicle type and the reduced cost is adjusted accordingly.

To solve the MDVRP with a route length constraint, Contardo and Martinelli (2014) devise a BPC algorithm which includes variable fixing and route enumeration. In a post-processing phase, variable fixing is first performed based on the solution of the linear relaxation of a two-index arc-flow formulation augmented by cutting planes. Once the BPC algorithm is started, variable fixing is also carried out in the fashion of Irnich et al. (2010). Contrarily to Bettinelli, Ceselli, and Righini (2011), Contardo and Martinelli (2014) solve one pricing problem per depot. Their BPC algorithm comprises several families of robust and non-robust cuts. In particular, they exploit the similarity between the MDVRP and the CLRP to adapt non-robust CLRP cuts for the MDVRP, namely, the y-strong capacity cuts and the strong framed capacity cuts.

4.2 Profits (optional customers)

In some applications, it may not be possible to visit all the customers due to vehicle fleet limitations (e.g. a limited number of vehicles with limited capacity is available) or to a short planning horizon. In this case, the customers to serve are part of the decisions to make and some of them might not be serviced. To determine which ones to serve, a profit is associated with each customer and profit maximization is sought. For this class of routing problems, called VRPs with profits (VRPPs), only a few BP algorithms have been designed.

The team orienteering problem (TOP) is the simplest multi-vehicle variant of the VRPP. It consists of designing routes for a homogeneous fleet of vehicles in such a way that the total profit collected is maximized and the length (duration) of the routes does not exceed a given threshold. No routing costs are considered. When solving the TOP by means of a CG algorithm, the MP is defined as the linear relaxation of a set-packing model, with an additional constraint to limit the number of routes in the solution. The pricing problem is an ESPPRC which has been solved by a labeling algorithm, except in Butt and Ryan (1999) where it is solved by a procedure that enumerates subsets of customers and checks for each subset the existence of a feasible route by solving a TSP.

Boussier, Feillet, and Gendreau (2007) present a generic BP algorithm for the TOP and the selective VRPTW (SVRPTW). The SVRPTW is a generalized TOP where vehicle capacity and customer time windows are considered. To derive integer solutions, Boussier, Feillet, and Gendreau (2007) apply two branching strategies. They branch first on a customer $i \in V'$ that is visited a fractional number of times. If this is not possible, i.e., the flow through each customer vertex is integer, an alternative branching on an arc (i, j) is performed. Two cases are considered. If one of the customers i or j has been constrained to be served, two branches are created: one imposing arc (i, j) to be traversed, and the other forbidding it. Otherwise, three branches are created: one forbidding the visit to i, and two enforcing the visit to i followed or not by j. Keshtkaran et al. (2015) enhanced the algorithm of Boussier, Feillet, and Gendreau (2007) by incorporating to the pricing algorithm several acceleration techniques discussed in Section 3 and by including SRCs to reinforce the MP. These cuts are also valid for a set-packing formulation.

Archetti et al. (2009) introduce the capacitated TOP (CTOP), a variant of the TOP that involves a vehicle capacity constraint. They also study the capacitated profitable tour problem (CPTP) which aims at minimizing the difference between the total collected profits and the total traveling cost and considers a vehicle capacity constraint but no maximum route duration constraint. To solve both the CTOP and the CPTP, the authors adapt the BP algorithm of Boussier, Feillet, and Gendreau (2007). Archetti, Bianchessi, and Speranza (2013b) enhance the BP algorithm of Archetti et al. (2009) by improving the pricing labeling algorithm with some acceleration techniques and revising the branching strategies used. The new branching

scheme involves branching on the number of vehicles used, on the flow through a customer, and on the flow on an arc. In the latter case, imposing a flow of one on an arc is performed by adding a constraint in the MP. This new algorithm employs a column-based restricted master heuristic to generate primal solutions throughout the search tree. Finally, Archetti, Bianchessi, and Speranza (2013a) address the CTOP with incomplete services, where a customer may be partially served. The problem is then formulated as a set-packing problem. The routes, possibly performing incomplete services, are generated by means of the labeling algorithm described in Archetti, Bianchessi, and Speranza (2014), which exploits an expanded network that contains vertices associated with every possible pair of customer and quantity that can be delivered to this customer.

Another problem belonging to the class of VRPPs is investigated by Bulhões et al. (2018), who introduce the VRP with service level constraints (VRP-SL). This problem arises in situations where the total customer demand exceeds the general capacity of supply. However, due to commercial agreements, a minimum service level to the customers must be ensured as follows. The set of customers V' is partitioned into m disjoint groups $\mathcal{V}_k$ (i.e., $V' = \cup_{k=1}^{m} \mathcal{V}_k$ and $\mathcal{V}_k \cap \mathcal{V}_{k'} = \emptyset$ for any $k \neq k'$) where a group may represent the deliveries to a same company. Moreover, with each customer $i \in V'$ are associated a demand q_i, a profit ρ_i and a service weight w_i. This latter represents the importance of the customer in its group. The VRP-SL consists of designing profitable routes that visit some of the customers in V' (to deliver their full demands) such that, for each subset $\mathcal{V}_k$, $k = 1, \ldots, m$, a minimum service level $\phi_k \in [0, 1]$ is reached. The service level of a group k is computed as $\sum_{i \in \mathcal{V}_k^*} w_i / \sum_{i \in \mathcal{V}_k} w_i$, where $\mathcal{V}_k^* \subseteq \mathcal{V}_k$ denotes the subset of customers in $\mathcal{V}_k^*$ that are serviced in the solution. The objective function aims at minimizing the total cost and the lost profits incurred by not servicing some of the customers. Bulhões et al. (2018) formulate the VRP-SL using a set partitioning model that contains binary slack variables to identify which customers are not serviced and the service level constraints. For solving the problem, they devise a rather straightforward BP algorithm.

Other VRPP variants are studied by Azi, Gendreau, and Potvin (2010), Archetti, Bianchessi, and Speranza (2014), Archetti et al. (2014), Gutiérrez-Jarpa et al. (2010), and Parragh, Sousa, and Almada-Lobo (2015). Given that the focus of these works is on the treatment of multiple trips, split services or pickups and deliveries, they are reviewed in Sections 4.4, 4.5 and 4.10.

4.3 Soft time windows

Soft time windows, unlike traditional time windows that pose structural restrictions on the feasible solution space, do not impact the feasibility of the routes. These restrictions are thus relaxed in the soft variant and transferred as penalties in the objective function (Liberatore, Righini, and Salani 2011). More precisely, in the VRP with soft time windows, a vehicle can serve a customer $i \in V'$ without paying a penalty if the service is performed within $[e_i, l_i]$, but it can also serve a customer before e_i, or after l_i by paying a linear penalty proportional to the anticipation or the delay.

From a computational viewpoint, soft time windows are harder to handle than the hard time windows because the feasible solution space in the former is larger. Liberatore, Righini, and Salani (2011) present the first exact BP algorithm that handles soft time windows in an efficient manner. In their algorithm, the labeling algorithm considers in each label a piecewise linear reduced cost function $\bar{c}(t)$ of the start of service time t. This function is equal to 0 at the source vertex 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 penalty function associated with the possible start of service times at j. In order to reduce the number of feasible extensions, the authors introduce the notion of *profitable time windows*, which are the reduced intervals of time in which a visit to a customer may be beneficial. Arrival times out of this interval are deemed unpromising for the current pricing problem and discarded.

Bettinelli, Ceselli, and Righini (2014) adapt this algorithm to solve the multi-depot heterogeneous fleet pickup and delivery problem with soft time windows. Abdallah and Jang (2014) consider a hybrid variant of this problem in which penalties are accepted only in a limited zone around the hard time windows, out of which arrivals are deemed infeasible. Thus, structural constraints are not completely lost. These authors develop a tailored labeling algorithm that is shown to be efficient to limit the number of non-dominated labels.

Taş et al. (2014) study the VRP with soft time windows and stochastic travel times. This work is reviewed in Section 4.9.1.

4.4 Multiple trips

In some applications, vehicles may return to the depot in the middle of their route to be replenished before performing additional deliveries. Such returns may be caused by a limited vehicle capacity compared to the volume of the demands, by some duration constraints imposed by the drivers' regulations or because the products delivered/collected are perishable. This feature gives rise to a class of problems, called the multi-trip VRPs (MTVRPs), where a trip is defined by a sequence of visits to customers between two visits to the depot and a route contains one or several trips. These problems are worthy of investigation if the number of vehicles is limited (or they incur fixed costs) and some constraints (such as a maximal route length or time windows) restrict the feasibility of the trip sequences. Otherwise, there exists an optimal solution with only single-trip routes and the problem becomes equivalent to the VRP.

Despite the practical importance of the MTVRP, no exact solution algorithms have been proposed for this problem prior to the recent work of Mingozzi, Roberti, and Toth (2013). The MTVRP they consider is a direct extension of the CVRP with a limited number of vehicles and a maximum driving time per route. Their algorithm extends the one of Baldacci, Christofides, and Mingozzi (2008), as it relies on bounding procedures to solve relaxations of two set-partitioning formulations and trip/route enumeration. In the first set-partitioning formulation, denoted SPF_1, the variables are associated with feasible trips; in the second, denoted SPF_2, they represent feasible routes. The proposed algorithm exploits the strengths of both formulations. On the one hand, generating trips is easier than generating routes. On the other hand, the bounds provided by SPF_2 are better than those yielded by SPF_1. The algorithm starts by computing a lower bound from SPF_1 by CG before performing trip enumeration and solving the resulting reduced model. If this approach does not succeed (e.g., when too many trips are enumerated), a lower bound from SPF_2 is computed by CG using only the trips enumerated in the first phase. Route enumeration is then performed to derive a reduced SPF_2 model that is solved by a MIP solver. Both set partitioning formulations are reinforced with non-robust cuts. In particular, a strengthened version of the knapsack inequalities, called working time inequalities, is introduced for SPF_1.

Considering the delivery of perishable products, Azi, Gendreau, and Potvin (2010) tackle a MTVRP with time windows and a limited trip duration (MTVRPTW-LD) in which customers' service is optional. The problem can also be seen as a VRPP variant. It is formulated as a set packing problem where the variables are associated with feasible routes and the objective consists of minimizing the total profit collected from the serviced customers minus the total traveling cost. To solve this problem, the authors propose a two-phase method where a complete enumeration of all the feasible trips is carried out in the first phase and routes obtained by concatenating feasible trips are generated by CG in the second phase. Complete trip enumeration is achievable in the first phase because the number of customers in a trip is small due to the maximum trip duration. This enumeration is performed using the algorithm of Azi, Gendreau, and Potvin (2007) for the single-vehicle case and in which a dominance rule is applied to eliminate non-Pareto-optimal trips. Each trip is associated with a (minimum) duration, a loading time that needs to be taken into account before starting the trip, and a starting time window which preserves the trip duration. A BP algorithm is applied in the second phase. The pricing problem is an ESPPRC defined over a directed graph where the vertices represent the trips enumerated in the first phase.

Hernandez et al. (2014) address the MTVRPTW-LD with mandatory services to the customers. They model the problem as a set-covering problem where the variables are associated with feasible trips and so-called *mutual exclusion constraints* (MECs) ensure that the number of available vehicles is met at all times. If the discretization of the planning horizon is too fine-grained, the number of MECs explodes and the model becomes intractable. In the proposed BP algorithm, this issue is dealt with by solving first a relaxation that considers a coarser discretization (thus, less MECs) and, if necessary, subsequent stronger relaxations obtained by refining the discretization. The pricing problem consists of finding trips with negative reduced costs or proving that none exist. This includes determining the exact schedule of each trip. To solve this problem, Hernandez et al. (2014) devise a two-phase algorithm. In the first phase, trips without schedule

are enumerated using a corrected version of the enumeration algorithm of Azi, Gendreau, and Potvin (2010). Each trip is defined by a sequence of customers, a minimum duration, a loading time, and a starting time window. In the second phase, a minimal reduced cost schedule is determined for each enumerated trip. Given that imposing integrality requirements on the arc flow between two customers is not sufficient to derive an optimal integer solution (a convex combination of schedules can be chosen for a given trip), the authors also develop a specific branching scheme. Given a fractional solution, branching on the arc flow between two customers is favored. If this is not possible (i.e., all arc flows are integer), then a special case of a VRPTW is defined from this solution (each trip in the solution represents a customer) and solved by an ad hoc BP algorithm to find a feasible solution to the MTVRPTW-LD if one exists. If so, the cost of this solution is equal to the current RMP solution cost and the node can be pruned. Otherwise, multiple branching nodes are created, each forbidding the usage of an arc in the current solution.

Contrarily to the previous works, Hernandez et al. (2016) study the MTVRP with time windows in which no limitation on the trip duration is imposed. They present two different formulations and two BP algorithms. In the first formulation, variables are associated with routes, whereas in the second, they are associated with trips. Despite the similarity with previous methods, Hernandez et al. (2016) do not benefit from trips with limited duration and, therefore, complete trip enumeration becomes impractical. Thus, the authors develop tailored labeling algorithms to solve the pricing problem ensuing from each formulation. In the BP algorithm for the first model, routes are generated by solving an ESPPRC with possible returns to the depot. Mono-directional backward labeling is performed to handle the loading time before each trip which depends on the total quantity to be delivered along the trip. In the BP algorithm for the second model which includes MECs in the MP, trips are also generated by solving an ESPPRC but, in this case, the trip schedule impacts its reduced cost because of the dual values of the MECs. To reduce the number of generated labels, Hernandez et al. (2016) establish some theoretical results which allow to group together labels corresponding to the same sequence of visited customers and keep a single representative label for each group. Here again, backward labeling is used.

Finally, Muter, Cordeau, and Laporte (2014) consider a generalization of the MTVRP and the MDVRP called the MDVRP with inter-depot routes (MDVRPI). In the MDVRPI, a route starts and ends at the same depot, and can also stop at any intermediate depot to replenish. The routes can be seen again as sequences of trips where the starting depot of a trip may differ from its ending depot. Route duration is limited but not trip duration. Route duration is computed as the sum of the traveling times, customer service times and fixed loading times required before starting the trips. The MDVRPI is modeled as a set covering problem where the variables are associated with routes and two different BP algorithms are developed to solve it. They differ by the algorithm used to solve the ESPPRC pricing problems, one for each depot. The first algorithm is a labeling algorithm similar to that of Feillet et al. (2004), which is applied on a network that contains additional vertices to represent intermediate visits at a depot. Because a vehicle can be replenished along its route, a customer can be temporarily considered unreachable if this status is only due to vehicle capacity. The second algorithm is a two-phase algorithm. In the first phase, all non-dominated trips are enumerated like in the work of Azi, Gendreau, and Potvin (2010). Furthermore, among the trips that link the same depot, only those with a negative reduced cost are kept. In the second phase, a labeling algorithm is applied on a network where the vertices represent the depots and the arcs the enumerated trips.

4.5 Split services

In routing problems with split services (split deliveries or split pickups), customers are allowed to be visited more than once in order to have their demand satisfied. The fact that each demand can be split among several vehicles can yield substantial economic savings, operational flexibility, not to mention, the possibility of customers with a demand larger than a vehicle capacity to be served (Archetti and Speranza 2012). Taking this option into account, the split delivery VRP (SDVRP) and the split delivery VRPTW (SDVRPTW) extend the CVRP and the VRPTW, respectively. The design of BP algorithms for these problems is not straightforward because the amount to deliver at each customer visit is also a variable decision. In the following, we start by discussing works on the SDVRPTW which is the first split delivery VRP variant to be solved by means of a BP algorithm. Note that, unless otherwise specified, the service time at a customer does not depend on the quantity delivered.

Gendreau et al. (2006) propose the first BPC algorithm for the SDVRPTW and, assuming that travel costs and times satisfy the triangle inequality, prove the following properties for the SDVRPTW (the first two ensue from identical properties for the SDVRP): 1) two routes cannot have more than two customers in common; 2) no arc can appear more than once in a solution; and 3) each route visits a customer at most once. In the proposed BPC algorithm, the pricing problem only determines the vehicles routes, not the quantities to deliver. These decisions are made at the MP level using additional quantity variables and constraints, which depend on the routes generated. Therefore, the MP involves an exponential number of constraints that are generated dynamically with the generated route variables. The authors develop sufficient conditions on the optimality of a MP solution in this context. These conditions are used to define the objective function of the pricing problem which is an ESPPRC that is solved by an adapted version of the labeling algorithm of Feillet et al. (2004). To speed up the solution process, Gendreau et al. (2006) solve first a relaxed MP which allows to generate columns more rapidly. Because customers may be visited more than once in the SDVRPTW, a solution may be fractional even if the arc flows are integer. When this occurs, a sophisticated branching rule on a subset of variables described in Feillet, Dejax, and Gendreau (2005) is applied. Two other branching rules can be applied, namely, on the number of visits to a customer and on an arc flow.

Desaulniers (2010) design a BPC algorithm for the SDVRPTW where the quantities to be delivered to the customers visited in a route (forming a so-called *delivery pattern*) are determined in the pricing problem, which corresponds to an ESPPRC combined with the linear relaxation of a bounded knapsack problem (LP-BKP). To solve this pricing problem, a tailored labeling algorithm is developed. It exploits the property of an optimal basic solution to the LP-BKP: all variables are either at their lower or upper bound except possibly one. In the SDVRPTW context, this property translates to: in a delivery pattern of a route yielding an optimal solution to the pricing problem (called an *extreme delivery pattern*), all customers receive a full or a zero delivery, except at most one customer that can receive a partial delivery. Because convex combinations of extreme delivery patterns for the same route are allowed in the MP, all customers in a route receive a positive quantity in an optimal solution and multiple partial deliveries are possible. In the labeling algorithm, a label can be extended up to three times along an arc reaching a customer, namely, for a full delivery at this customer, a zero delivery, or a partial delivery. Because the quantity delivered in a partial delivery is a decision variable that is computed only once the route is completed, the reduced cost of a partial path is a linear function of this quantity and the dominance rule is devised to handle such reduced cost components. The MP is reinforced with adapted kPCs exploiting the structure of the SDVRPTW and with edge-flow cuts which ensue from the application to a single edge of the first property stated by Gendreau et al. (2006). Four branching strategies are applied, namely: on the total number of vehicles, on the number of times a customer is visited, on an arc flow, and on the possibility of using or not two arcs consecutively. Decisions of this last type are non robust in the sense that they require additional resources in the labeling algorithm. They are, however, necessary to ensure an exhaustive exploration of the search tree.

Archetti, Bouchard, and Desaulniers (2011) improve the BPC algorithm of Desaulniers (2010). They design a tabu search algorithm that combines the procedure of Desaulniers, Lessard, and Hadjar (2008) with an LP-BKP solver to generate negative reduced cost columns. They also develop three efficient separation heuristics for the kPCs. Finally, to further strengthen the MP lower bounds, they introduce three new classes of non-robust valid inequalities for the SDVRPTW: strong k-path cuts (SkPCs), strong minimum number of vehicles inequalities (SMVCs), and adapted SRCs. The SkPCs are a strengthened version of the kPCs which are not weakened by paths entering a given subset of customers more than once. The SMVCs are based on the idea that, if a customer does not receive a full delivery, it has to be visited at least twice. Finally, even if the MP does not include set partitioning constraints, SRCs can be defined by observing that each customer cannot receive more than one full delivery.

Salani and Vacca (2011) study an extension of the SDVRPTW, called the discrete SDVRPTW (DSDVRPTW), in which the demand of the customers is composed of a discrete set of items which can be delivered separately. The items delivered by a vehicle to a customer form an order and the service time at a customer depends on the delivered order. Because of this feature, several properties defined for the SDVRP(TW) are no longer valid for the DSDVRPTW. For instance, two routes can have more than one split customer in common. The authors formulate the DSDVRPTW as a set partitioning model where the set partitioning constraints are associated with the items of each customer. The pricing problem is an ESPPRC defined on a network

identical to that used for the VRPTW except that the vertices are associated with feasible orders instead of customers. This network structure permits to include the service time for each possible order on the arcs.

Luo et al. (2017) introduce an extension of the SDVRPTW called the SDVRPTW with linear weight-related cost, in which the travel costs per unit distance are charged based on a linear function of the load weight. To solve this problem, they devise a BPC algorithm that borrows ideas from Desaulniers (2010) and Archetti, Bouchard, and Desaulniers (2011) and relies on an efficient labeling algorithm involving an aggressive dominance rule. In this labeling algorithm, each label contains a reduced cost component, which is also a function of the vehicle load but encodes information regarding the extreme delivery patterns for the corresponding route. With this reduced cost function, each label is extended once along each arc contrarily to the algorithm of Desaulniers (2010) that extends it up to three times. The proposed dominance rule allows to dominate partially or fully a label by comparing it with a set of labels. In fact, all labels are stored in a dominance graph that permits efficient label comparisons.

Archetti, Bianchessi, and Speranza (2011) present the first BPC algorithm for solving the SDVRP. The approach is similar to the one proposed by Desaulniers (2010) since the pricing problem generates simultaneously the routes and their delivery patterns. The main difference is that the pricing problem is defined over an expanded network, where each customer is represented by a set of vertices, one for each possible quantity that can be delivered to it. With this network, the dual values of the customer demand constraints can be easily incorporated into the arc costs and the labeling algorithm is much simpler than that of Desaulniers (2010), especially a standard dominance rule can be used. On the other hand, the performance of the algorithm heavily depends on the size of the customer demands.

Moreno, Poggi de Aragão, and Uchoa (2010) focus on improving the lower bounds for the SDVRP. They design a robust CG algorithm that employs cutting planes defined in terms of CI variables. The algorithm considers three families of valid inequalities: the homogeneous extended capacity cuts of Pessoa, Poggi de Aragão, and Uchoa (2009), edge-flow cuts and split delivery cuts. The edge-flow cuts are the same as those proposed independently in Desaulniers (2010). The split delivery cuts are Chvátal-Gomory cuts derived from a single customer demand cut. The authors point out that, by formulating the SDVRP using CI variables, the knapsack structure of the problem can be exploited, enabling stronger cuts to be devised.

Archetti, Bianchessi, and Speranza (2015) address the commodity constrained SDVRP (C-SDVRP) in which a customer can be visited more than once if its demand is composed by multiple commodities that must be treated separately. This situation occurs, for example, when different products require specific temperatures (e.g., frozen, fresh and dry food), or when different commodities must be transported in separated compartments (e.g., food and hazardous products). Like in the DSDVRPTW, the C-SDVRP allows multiple visits to a customer, but does not allow a specific commodity to be split. The authors model the C-SDVRP as a CVRP where the set of customers is replaced by the set of all requested commodities for all customers. They design a BPC algorithm where the MP ensures that all commodities are delivered and the pricing problem is defined over an expanded network, where most vertices represent a commodity/customer pair. In fact, for each customer, a subnetwork containing the related vertices is created to reduce symmetry. Furthermore, at each column generation iteration, a preprocessing step is applied to each subnetwork to deduce non-dominated subpaths with respect to vehicle capacity and the current dual values. Besides the branching strategies proposed by Desaulniers (2010), a new strategy inspired by Ryan and Foster (1981)'s rule is presented. It branches on whether or not two pairs of commodity/customer must be delivered in the same route. Each such decision requires adding a new resource in the label definition.

Some authors study the impact of split services in VRPP variants. Archetti, Bianchessi, and Speranza (2014) develop a BP algorithm to solve the split delivery CTOP (SDCTOP), where each customer can be visited multiples times in order to have its demand completely fulfilled. This algorithm incorporates features from the algorithms of Archetti et al. (2009) and Archetti, Bouchard, and Desaulniers (2011), in particular, it uses the expanded network defined in Archetti, Bouchard, and Desaulniers (2011). Archetti et al. (2014) investigate the SDCTOP with incomplete service, where a customer may be served only partially and can be visited by more than one vehicle. This problem can be modeled as the SDCTOP except that the constraints enforcing the complete service of the customers are replaced by constraints bounding the total quantity

delivered to each customer by its demand. Therefore, the authors present an algorithm similar to the one described in Archetti, Bianchessi, and Speranza (2014).

Parragh, Sousa, and Almada-Lobo (2015) address a pickup and delivery routing problem with split services that is discussed in Section 4.10.

4.6 Time dependency

Classical VRP variants do not capture some important features related to real-life problems such as the presence of congested routes at different times of a day. In the literature, road congestion has been represented by the consideration of time-dependent travel times, i.e., by making the travel time on an arc depend on the moment at which it is traversed. In this context arises the time-dependent VRPTW (TDVRPTW) in which the planning horizon is divided into time zones (e.g., morning, afternoon, and night) and, for each time zone, travel times are assumed to be constant on every arc, leading to a stepwise speed function over the horizon. This speed function is then used to determine a travel time piecewise linear function that satisfies the first-in-first-out (FIFO) principle (Ichoua, Gendreau, and Potvin 2003).

Dabia et al. (2013) propose the only BP algorithm to solve a variant of the TDVRPTW that minimizes the total duration of the routes. Given that the cost of a route is its duration, the reduced cost of a route cannot be computed as the sum of arc costs in the pricing problem. Therefore, in the labeling algorithm developed by Dabia et al. (2013), a label associated with a customer vertex $i \in V'$ stores a piecewise linear function $\delta_i(t)$ to represent the time at which the service is completed (ready time) at i if the vehicle leaves the depot at time t. This function is non-decreasing and allows to compute the route duration in function of this departure time as $\delta_i(t) - t$ as well as the route reduced cost as $\delta_i(t) - t - c_\pi$, where c_π is the sum of the dual variables associated with each vertex visited along the route. The authors introduce a sharp dominance rule that takes into account the possibility to use different departure times from the depot for each compared label.

4.7 Cumulative costs

The real cost of a vehicle traversing an arc depends on many variables. Several of them are proportional to the arc distance (or travel time). Some others like the elapsed time up to each customer and the vehicle load, which impacts the fuel consumption rate, cannot be represented by the distance. In some cases, they are, however, proportional to a flow on the arc, e.g., the remaining quantity to deliver and the number of customers still to be visited, and the cost function can be defined as a product of this flow and the distance traveled. This gives rise to a class of VRPs, called the cumulative VRPs (CumVRPs) (Kara, Kara, and Yetis 2008).

Lysgaard and Wøhlk (2014) design a BPC algorithm to solve a CumVRP in which the objective aims at minimizing the sum of the arrival times at the customers. In fact, their algorithm is very similar to the algorithm of Fukasawa et al. (2006) for the CVRP. However, to assess the cost of traversing an edge e, one not only needs to know the traveling time t_e along e but also the number of customers remaining to visit after e, because t_e contributes to the arrival time of all the remaining customers. Given that this information is unavailable when building a route from the source vertex, Lysgaard and Wøhlk (2014) use a backward labeling algorithm to solve the pricing problem. In this way, the customers remaining to service after traversing an edge are those that have been visited in the current backward path and, thus, the reduced cost of traversing an edge can easily be computed.

Another CumVRP variant where the cost of traversing an arc depends on the vehicle load is tackled by Fukasawa, He, and Song (2016). This work is reviewed in the next section.

4.8 Environmental aspects

Recently, it has become critical to make transportation and logistics environmentally sustainable, and thus VRPs that include environmental aspects have gained considerable attention in the literature. These VRP variants often aim at minimizing greenhouse gas emissions/fuel consumption (Lin et al. 2014), or at managing the use of electric vehicles (Pelletier, Jabali, and Laporte 2016). Green VRPs (GVRPs) denote VRP variants

in which greenhouse gas emission or fuel consumption is considered either in the objective function or in the operational constraints (Lin et al. 2014).

Fukasawa, He, and Song (2016) design a BPC algorithm to solve the energy minimization vehicle routing problem (EMVRP). This problem belongs to the class of CumVRPs (Kara, Kara, and Yetis 2008) and extends the CVRP by defining the arc cost as the product between the arc length and the current vehicle weight when traversing the arc. The proposed algorithm is similar to that of Fukasawa et al. (2006). These algorithms differ by the network used to generate q-routes. Because in the EMVRP, the cost of traversing an arc depends on the carried load, Fukasawa, He, and Song (2016) use an expanded network G_Q where each arc appears $Q+1$ times in the network. In G_Q, an arc (i, j, q) represents a traversal of the arc (i, j) with a load of q units. The labeling algorithm is adapted to be executed on G_Q.

The pollution routing problem (PRP, Bektaş and Laporte (2011)) is a GVRP variant in which one must not only design vehicle routes but also determine at which speed each arc in each selected route should be traveled. The problem aims at minimizing the vehicle consumption and the drivers' wages that are proportional to route duration. Both objectives are conflicting since higher speeds yield shorter routes but larger fuel consumptions. To address a variant of the PRP which assumes constant speed along a route and variable departure time from the depot, Dabia, Demir, and Van Woensel (2017) adapt the BP algorithm of Dabia et al. (2013) (see Section 4.6). The modifications concern the labeling algorithm which also considers a piecewise linear function to represent the ready time $\delta_i(t, v)$ at a customer $i \in V'$, that does not only depend on the departure time t from the depot, but also on the vehicle speed v along the route. Because computing this function dynamically is not straightforward, the authors rather generate two functions, obtained by fixing $t = 0$ and $v = v_{max}$ (the maximum speed), that can be combined to provide all necessary information. The labeling algorithm requires new resources to compute the fuel consumption and a tailored dominance rule.

In addition to the GVRPs discussed above, another VRP variant that has environmental motivations is the electric VRP (EVRP), in which vehicles are powered by electricity and suffer from limited battery autonomy constraints (Pelletier, Jabali, and Laporte 2016). When time windows are associated with the customers, the EVRP with time windows (EVRPTW) arises. In the EVRPTW, one considers all features of the VRPTW plus the following ones: additional vertices representing recharging stations, a recharging time depending on the amount of energy recharged, a battery capacity for each vehicle and, for each arc, the energy consumed by a vehicle traversing it.

Depending on the number of allowed visits to the recharging stations in each route —single (S) or multiple (M)—, and the type of battery recharging policy —full (F) or partial (P)—, four VRP variants of the EVRPTW may arise, namely, the EVRPTW-SF, the EVRPTW-MF, the EVRPTW-SP, and the EVRPTW-MP. Desaulniers et al. (2016) study all these variants and propose different BPC algorithms that differ by the labeling algorithm used to generate routes. Assuming that the energy consumed along each arc of a route can be converted into a required recharging time, the battery capacity constraint is expressed in terms of the time required to recharge the consumed energy. For the EVRPTW-SF and the EVRPTW-MF, the labeling algorithms are similar (the former problem requires an additional resource to impose at most one recharge per route). In the forward algorithms, a single resource is needed to cumulate the required recharging time which also monitors the battery capacity. Given that the time required for a recharge at a station depends on the energy consumed, this time is unknown in a backward labeling algorithm. Nevertheless, Desaulniers et al. (2016) devise a bidirectional labeling algorithm where the backward algorithm relies on additional resources to ensure that this time is well computed. When partial recharges are allowed in the EVRPTW-SP and the EVRPTW-MP, the trade-off between the amount recharged and the time spent for recharging leads to express the start of service time at a customer visited after a recharge as a linear function of the recharging time. This relation is modeled using three resources and requires an ad hoc dominance rule. In this case, a backward labeling algorithm symmetric to the forward one can be devised and used in a bidirectional search.

4.9 Uncertainty

All VRPs discussed so far assume that all required input data is known in advance. However, when solving real-life problems, this is not always the case as several data may be subject to different sources of uncertainty coming from expected variations and/or unexpected events. The demands, the service times, the

traveling times, and the presence of the customers are commonly considered to be stochastic according to Gendreau, Jabali, and Rei (2014). In the last two decades (see, e.g., Oyola, Arntzen, and Woodruff (2016)), many researchers have turned their attention to the exact solution of stochastic VRPs (SVRPs) and robust VRPs (RVRPs).

4.9.1 Stochastic VRPs

SVRPs are often considered as *a priori* optimization (or two-stage optimization) problems. In the first stage when complete information is unknown, *a priori* decisions (e.g., planned routes) must be taken. Once the uncertain information is revealed in the second stage, the planned routes may become infeasible in which case they can be revised according to predefined recourse actions or they can simply be deemed failures. In the literature, two main modeling approaches have been proposed for the SDVRPs: chance-constrained programs (CCP) or stochastic programs with recourse (SPR) (Gendreau, Laporte, and Séguin 1996, Oyola, Arntzen, and Woodruff 2016). In a CCP, a lower bound on the probability that a given plan will be feasible once the stochastic information becomes known is imposed. This probability can be considered while solving the pricing problem (Dinh, Fukasawa, and Luedtke 2017) or by adding chance constraints directly in the MP (Gendreau, Jabali, and Rei 2014). In turn, in a SPR, the objective function consists of minimizing the expected costs, namely, the travel costs of the planned routes plus the expected recourse costs ensuing from the application of recourse actions when the routes become infeasible in the second stage. Note that the recourse actions are not arbitrary as they must obey to the chosen recourse policy for the problem at hand. Even if the SPRs are typically more difficult to solve than the CCPs, their objective function is more meaningful (Gendreau, Laporte, and Séguin 1996) as it considers the cost of turning the infeasible planned routes into feasible ones. On the other hand, the main advantage of the CCPs is that they allow to control directly the probability of failure which may be important in certain cases to maintain the image of the company.

The most studied SVRP variant is the VRP with stochastic demands (VRPSD) (Gendreau, Laporte, and Séguin 1996), where the demand of each customer $i \in V'$ is expressed by a random variable with an expected value $\xi(q_i)$ and a variance $\vartheta(q_i)$. Normally, the demands are assumed to be independent and to follow an additive probability distribution such as the Normal or the Poisson distribution (Oyola, Arntzen, and Woodruff 2016). For a route $r = (0, i_1, \ldots, i_k, n+1)$, one can thus define the cumulative expected demand $\mu_h = \sum_{\ell=1}^{h} \xi(q_{i_\ell})$ and the cumulated variance $\sigma_h^2 = \sum_{\ell=1}^{h} \vartheta(q_{i_\ell})$ at any given customer i_h, $h \in \{1, \ldots, k\}$. In the VRPSD, route r is said to be feasible if it is elementary and $\mu_h \leq Q$ for all $h \in \{1, \ldots, k\}$. This latter condition prevents routes from systematically *failing* when their feasibility is assessed. By considering a cumulative probability distribution with mean μ_h and variance σ_h^2 for customer i_h, one can determine the probability $P(\mu_h > Q)$ that a *failure* occurs at this customer. In this case, if the problem is modeled as an SPR, a recourse action (e.g., returning to the depot before continuing the route) should be applied. The objective of the VRPSD consists of minimizing the total expected cost of the routes, that can be decomposed in two parts: the deterministic cost and the *expected failure cost* (EFC). This latter is given by the probability that a route fails at a customer multiplied by the cost of returning to the depot before continuing the route (or of another recourse action).

Christiansen and Lysgaard (2007) introduce the first BP algorithm for a SVRP, namely, the VRPSD. The pricing problem is a shortest path problem with 2-cycle elimination defined on an expanded network $G_S = (V_S, A_S)$. The vertex set V_S is defined as $V_S = \{(0,0,0)\} \cup \{(\mu, \sigma^2, i) : i \in V \cup \{0\}, \mu = 1, \ldots, Q, \sigma^2 = 1, \ldots, \vartheta_{\max}\}$, where $(0,0,0)$ is the source vertex and each vertex (μ, σ^2, i) can only be reached by paths representing a (partial) route ending at customer i (or at the depot if $i = 0$) and whose expected cumulative demand and variance are μ and σ^2, respectively. In this definition, $\vartheta_{\max}$ is the maximum total variance of a feasible route, i.e., with a mean demand $\mu \leq Q$, which is pre-computed by solving a knapsack problem. In arc set A_S, there exists an arc between two vertices (μ_i, σ_i^2, i) and (μ_j, σ_j^2, j) only if $\mu_j = \mu_i + \xi(q_j)$ and $\sigma_j^2 = \sigma_i^2 + \vartheta(q_j)$ with $\xi(q_j) = \vartheta(q_j) = 0$ if $j = 0$. This network structure allows to incorporate the EFCs directly in the arc costs, giving a huge advantage over the approaches requiring dynamic calculations of the EFCs. Note, however, that the construction of network G_S requires a finite number of possible expected cumulative demands and variances. For their computational tests, Christiansen and Lysgaard (2007) assume that the demands are Poisson distributed, which implies $\mu = \sigma^2$. This allows to considerably reduce the size of the network G_S by eliminating all vertices (μ, σ^2, i) for which $\mu \neq \sigma^2$. The authors propose a branching strategy inspired by that of Gélinas et al. (1995) (see Section 3.3.4). Instead of branching on time windows,

they branch on the expected cumulative demand at a customer $i \in V'$ that is visited by two routes containing the vertices (μ_1, σ_1^2, i) and (μ_2, σ_2^2, i), where $\mu_1 < \mu_2$. Choosing a value $\delta \in [\mu_1, \mu_2)$, one can impose that expected cumulative demand μ at customer i is such that $\mu > \delta$ in one branch and $\mu \leq \delta$ in the other.

Addressing the same VRPSD, Gauvin, Desaulniers, and Gendreau (2014) develop a BPC algorithm that improves the algorithm of Christiansen and Lysgaard (2007) by adding several features (*ng*-routes, heuristic pricing, bidirectional labeling, cuts) found in state-of-the-art algorithms for VRPs. Moreover, they introduce a new dominance rule which allows the comparison of labels associated with two different vertices, as long as these vertices represent the same customer i. Finally, to derive integer solutions, they branch on customer sequences or cutsets (see Section 3.3.3).

Dinh, Fukasawa, and Luedtke (2017) study the chance-constrained VRPSD (CCVRPSD) and develop a BPC algorithm for solving it which is derived from that of Fukasawa et al. (2006). They introduce the concept of *chance-constraint feasible route* (CC route), which is a route for which the probability of satisfying the capacity constraint is greater than or equal to a threshold $1-\epsilon$ for a given $\epsilon \in (0, 1)$. Contrary to previous methods used to solve the VRPSD, the algorithm of Dinh, Fukasawa, and Luedtke (2017) does not require the customer demands to be independent, though it needs the quantiles of the random variable defined by the sum of the demands in any subset of customers. Because the pricing problem remains strongly $\mathcal{NP}$-hard, even if the q-route relaxation is used, the application of the BPC algorithm of Fukasawa et al. (2006) to solve the CCVRPSD is not straightforward. To circumvent this issue, the authors propose to relax the capacity chance constraint in the pricing problem and to handle it through capacity constraints in the MP. To strengthen the relaxed pricing problem, they add a knapsack constraint which ensures that all CC routes remain feasible. Regarding the capacity inequalities added to the MP, Dinh, Fukasawa, and Luedtke (2017) discuss how strong lower bounds on the number of vehicles required to serve a subset of customers can be computed cheaply considering the quantiles of the probability distribution of the total demand for this customer subset.

Noorizadegan and Chen (2018) devise a BP algorithm for the CCVRPSD, assuming that every customer demand follows a Poisson distribution. They, however, mention that, if verifying the satisfaction of the chance capacity constraint for a route is possible, the algorithm can be adapted to other distributions for which the sum of their random variables follows a known distribution. These requirements are necessary because the pricing problem is solved by means of a labeling algorithm, in which the chance capacity constraint is verified after every label extension. In the case of the Poisson distribution, a single additional resource specifying the average demand along the route (which is equal to its variance) is required. Other resources may be required for other distributions. Standard dominance rule is then applied.

Taş et al. (2014) address a VRP with soft time windows and stochastic travel times. In this problem, the travel times follow Gamma probability distributions and the customers may be served outside their time windows under the payment of a penalty. The objective consists of minimizing a weighted cost function of the total transportation costs and the service costs. The transportation costs depend on the total distance traveled, the number of vehicles used, and the total expected overtime of the drivers, whereas the service costs are due to early and late arrivals at the customers. Waiting at customers is forbidden. The authors developed a BP algorithm where the pricing problem is solved by a labeling algorithm derived from that of Feillet et al. (2004) and in which routes cannot be deemed infeasible because of a time window violation. A label includes an expected reduced cost component which is computed by adjusting the departure time from the depot using a golden section search method. The dominance rule is standard except that it includes an additional condition which compares the expected reduced cost of the routes if they were both starting from the depot at the optimal departure time of the potentially dominated route.

Errico et al. (2016a,b) study the VRPTW with stochastic service times (VRPTW-ST) which is defined like the VRPTW except that the customer service times are expressed by mutually independent probability distributions. Both works address the problem of dispatching technicians to perform maintenance operations, where vehicle capacity is not a concern. Nevertheless, the proposed BPC algorithms can easily be adapted to handle vehicle capacity if required. The works of Errico et al. (2016a,b) differ in the way the VRPTW-ST is modeled, namely, as a CCP in the former and as a SPR in the latter.

In Errico et al. (2016a), the VRPTW-ST is formulated as a set partitioning model with an additional constraint imposing a lower bound on the success probability of the whole route plan. Applying logarithm properties and by taking advantage of the service times being mutually independent, the authors show how this non-linear constraint can be linearized. Errico et al. (2016a) develop a BPC algorithm where the pricing algorithm must handle the dual from this additional constraint considering a stochastic time component. Indeed, the coefficient of a route variable λ_r in this constraint depends on its success probability. To determine this probability as the route is being built in the labeling algorithm, the mass function of the earliest start of service time at each vertex along the route must be computed. Consequently, the authors replace the usual time component in a label associated with a vertex $i \in V'$ by $l_i - e_i + 1$ components, namely, one for each integer time t in the time window $[e_i, l_i]$ which is denoted $\bar{M}_i(t)$. These components define the cumulative probability distribution of the earliest start of service time at i restricted to $[e_i, l_i]$. They also allow to decompose the reduced cost of a route into single arc contributions obtained from conditional probabilities. In the labeling algorithm, the dominance rule combines the traditional dominance criteria with the concept of *stochastic dominance* which compares the components $\bar{M}_i(t)$ for each time $t \in [e_i, l_i]$.

When formulating the VRPTW-ST as a SPR, Errico et al. (2016b) propose two alternative recourse policies. Both policies assume that the real service time at a customer is first evaluated just before starting the service at a customer. If this evaluation indicates that the time window at the next customer will be missed, then the service is skipped at the current customer in the first policy, denoted `C`, or at the next customer in the second one, denoted `N`. In both policies, a penalty is paid for skipping the service at a customer. To ensure a high-level service, a route is considered feasible if it requires at most one recourse action and its success probability is larger than or equal to a given threshold. Each recourse policy induces a different pricing problem, which is solved by a tailored labeling algorithm adapted from that devised in Errico et al. (2016a). To handle policy `C`, two new resources are added to the label definition. Policy `N` is more complex because skipping the service at the next customer i_N due to a long service time at a customer i_C implies taking a shortcut in the route from i_C to the customer i_F (or depot) visited after i_N, yielding a smaller traveling cost. Given that this situation is observed at vertex i_N in the labeling algorithm, vertex i_F is unknown at that time and the cost saving realized by cutting short from i_C to i_F cannot be evaluated at vertex i_N. Consequently, Errico et al. (2016b) replace the reduced cost component of a label by an *incomplete reduced cost* component, and compute lower and upper bounds on the expected reduced cost of any extension of the corresponding route that reaches the depot. In the dominance rule, these bounds are used to modify the reduced cost criterion. The rest of the labeling algorithm is similar to the algorithm used for policy `C`.

4.9.2 Robust VRPs

Stochastic programming approaches require the knowledge of the probability distributions of the uncertain data. These distributions are not always available and, when they are, they must respect certain assumptions to yield tractable SVRPs. Robust optimization is an alternative approach that expresses the uncertainty of the data by means of an uncertainty set Ψ. This set is generally parameterized by a perturbation vector that indicates how the problem data diverge from their nominal values. It can be defined by taking into account past observations or considering some existing knowledge about the probability distributions. Given a set Ψ, a solution is said to be *robust feasible* if it satisfies all the realizations of the constraints defined over the uncertainty set. The problem of finding the best robust feasible solution is called the *robust counterpart problem* (Ben-Tal, Ghaoui, and Nemirovski 2009). It is shown by Bertsimas, Pachamanova, and Sim (2004) that the robust counterpart problem of a linear problem is also a linear problem of polynomially-bounded size.

Very few BP algorithms exist for solving RVRPs. Lee, Lee, and Park (2012) consider the VRP with customer deadlines and travel time/demand uncertainty, and develop a BP algorithm in which the robust aspect is embedded in the pricing problem. Two uncertainty sets Ψ_t and Ψ_d associated with the travel times and the demands, respectively, are defined as follows. For each arc $(i, j) \in A$, the travel time can take a value in the interval $[\hat{t}_{ij}, \hat{t}_{ij} + \delta_{ij}]$, where $\hat{t}_{ij}$ is the nominal travel time along this arc and δ_{ij} is the maximum deviation from this nominal value. Because it is unlikely that a vehicle will be delayed on every arc of a route, the uncertainty set is limited by the maximum number of delayed segments Γ_t which ensures a certain degree of robustness of the routes. Set Ψ_t is defined as: $\Psi_t = \{(\tilde{t}_{ij})_{(i,j)\in A} \,|\, \tilde{t}_{ij} = \hat{t}_{ij} + \delta_{ij}\nu_{ij}, 0 \leq \nu_{ij} \leq 1,\ \forall (i,j) \in A,\ \sum_{(i,j)\in A} \nu_{ij} \leq \Gamma_t\}$. Set Ψ_d is defined similarly. To ensure that the generated routes are feasible for all

possible realizations of the travel times and demands in these uncertainty sets, the labeling algorithm relies on $\Gamma_t + \Gamma_d$ additional resources. The Γ_t resources store the sums of the k largest travel time deviations δ_{ij} associated with the arcs traversed in a route, for $k = 1, \ldots, \Gamma_t$. The other Γ_d resources play the same role for the demands. With these resources, a standard dominance rule can be used.

Souyris et al. (2013) also suggest a robust approach to tackle a real-life problem of designing routes for maintenance technicians where the customer service times are uncertain and a soft start of service time deadline is imposed at each customer. Furthermore, it is assumed that the technicians have different initial locations and do not have to return to a depot at the end of the day. Given that the routes are subject to a maximum duration, some customers might not be serviced. The objective function is a weighted sum of the traveling costs, penalties for delayed customer service, and penalties for unserviced customers. The authors present two models. The first assumes that the customer service times are independent, resulting in a VRP with soft time windows where all service times are replaced by their worst case. The second model assumes that the service times of the customers visited by each technician are correlated, i.e., deviations will not occur at all customers visited along a route. Assuming that the service time at a customer $i \in V'$ belongs to the interval $[\hat{s}_i, \hat{s}_i + \delta_i]$, the uncertainty set is given by $\Psi^k = \{(\tilde{s}_i)_{i \in V'} \mid \tilde{s}_i = \hat{s} + \nu_i, 0 \leq \nu_i \leq \delta_i, \forall i \in V', \sum_{i \in V'} \nu_i \leq \Gamma^k\}$, where Γ^k is an upper bound on the total service time deviation for technician k. To solve this RVRP, Souyris et al. (2013) develop a BP algorithm. They formulate the resulting pricing problem as a mixed integer linear program but solve it using constraint programming.

Dinh, Fukasawa, and Luedtke (2017) and Noorizadegan and Chen (2018) mention how their BP algorithms discussed in Section 4.9.1 can be adapted to solve a *distributionally robust* extension of the CCVRPSD for which the probability distributions are not known precisely, but they are assumed to belong to an ambiguity set of possible distributions. Routes must be robust in the sense that they must satisfy the chance capacity constraint for all probability distributions in this set.

4.10 Pickups and deliveries

Pickup and delivery problems (PDPs) consist of finding least-cost vehicle routes to satisfy a set of requests. Each request is defined by a pickup location, a delivery location, and a volume of merchandise (or a number of persons) to be transported from the pickup location to the delivery location. Several requests can be transported simultaneously by a vehicle as long as the onboard load does not exceed vehicle capacity. A large number of PDP variants have been studied in the literature and classified by Berbeglia et al. (2007). So far, BP algorithms have been devised for *one-to-one PDPs* and *one-to-many-to-one PDPs*. In the former class, each request corresponds to a specific commodity and, in general, the pickup and delivery locations differ from the depot. In the latter class, commodities originating from the depot (e.g., filled beer bottles) must be delivered to a set of customers, while other commodities destined to the depot (e.g., empty beer bottles) must be picked up at a possibly different set of customers.

Given the large number of works on PDPs, we divide our review in the following five subsections: time windows (Subsection 4.10.1), ride time constraints (Subsection 4.10.2), transfers (Subsection 4.10.3), loading constraints (Subsection 4.10.4), and simultaneous pickups and deliveries (Subsection 4.10.5). The first four subsections are devoted to one-to-one PDPs, whereas the last one concerns one-to-many-to-one PDPs.

4.10.1 Time windows

Most BP algorithms for PDPs have been developed for PDPs subject to time window constraints. Here, we focus our attention on the one-to-one PDP with time windows (PDPTW), which can be defined on the following network $G = (V, A)$. Let n be the number of requests to satisfy. Let $P = \{1, \ldots, n\}$ be the set of pickups and $D = \{n + 1, \ldots, 2n\}$ the set of deliveries such that $i \in P$ and $n + i \in D$ are associated with the same request. Set V is then given by $V = P \cup D \cup \{0, 2n + 1\}$, where the vertices 0 and $2n + 1$ denote the origin and the destination depot, respectively. Set A is formed by the feasible links connecting the vertices in V. In particular, for $i \in P$, there is no arc $(n+i, i)$ given that the pickup of a request must be performed before its delivery. Note that set P is often used to represent the set of requests. When designing BP algorithms to solve the PDPTW, one of the most complicating aspects is the presence of *pairing* and

precedence constraints. The pairing constraints specify that, if a vehicle performs the pickup of a request, it must also perform its delivery. The precedence constraints impose that pickups be performed before the corresponding deliveries. These constraints must be dealt with when constructing the routes in the pricing problem. Note that the MP does not need to include a set partitioning constraint for each pickup and each delivery, as one for each request suffices. This allows some flexibility when defining the modified arc costs. Indeed, the dual cost for fulfilling a request can be associated with its pickup vertex i, its delivery vertex $n+i$, or be arbitrarily distributed among both of them (see Gschwind et al. 2018).

The first BP algorithm for the PDPTW is due to Dumas, Desrosiers, and Soumis (1991) who design a labeling algorithm in which the label definition is extended by adding a set of visited vertices $V(L)$ and a set of open requests $\mathcal{O}(L)$, i.e., pickup vertices whose corresponding delivery vertices have not been visited yet. Strong dominance rules can be obtained if one assumes that the arc modified cost matrix in the pricing problem respects the delivery triangle inequality (DTI), i.e., a visit to a delivery vertex never lowers the path reduced cost. To ensure that this condition is satisfied, the dual variables of the request covering constraints are transferred to the arcs exiting a pickup vertex. When comparing two labels L_1 and L_2 for dominance, the previous property allows to discard label L_2 if the condition $\mathcal{O}(L_1) \subseteq \mathcal{O}(L_2)$ holds (together with the usual conditions), rather than $\mathcal{O}(L_1) = \mathcal{O}(L_2)$. Dumas, Desrosiers, and Soumis (1991) also propose preprocessing techniques to tighten the time windows and eliminate infeasible arcs, as well as unreachability criteria that can be applied while solving the pricing problem. Finally, they devise a branching strategy based on request ordering. Savelsbergh and Sol (1998) improve this BP algorithm by developing two heuristics for generating feasible routes: one considering a reduced network and another based on construction and improvement algorithms.

Ropke and Cordeau (2009) present the first BPC algorithm for the PDPTW that adds valid inequalities to the algorithm of Dumas, Desrosiers, and Soumis (1991). This practice, however, yields a modified cost matrix that no longer satisfies the DTI, which is required to apply the enhanced dominance rule. The authors then propose a procedure to transform any arbitrary cost matrix into an equivalent matrix where the DTI is ensured.

Baldacci, Bartolini, and Mingozzi (2011) extend the exact solution framework of Baldacci, Christofides, and Mingozzi (2008) to deal with the PDPTW. They develop two new bounding procedures that exploit the properties of the PDPTW.

The application of bidirectional search in the context of the PDPTW is not straightforward. The strong dominance rules of Dumas, Desrosiers, and Soumis (1991) are not compatible with a backward labeling. When the direction of the labeling changes from forward to backward, the cost matrix must respect the pickup triangle inequality (PTI). Two techniques are suggested to overcome this issue. First, Bettinelli, Ceselli, and Righini (2014) redistribute the weights over the arcs of the network, so that the two above properties hold simultaneously. It requires solving a linear program for every label extension in the labeling algorithm, which is computationally costly. Second, Gschwind et al. (2018) use two modified cost matrices when employing bidirectional search: one for the forward labeling, that respects the DTI, and another for the backward labeling, where the PTI holds. With these matrices, the bidirectional search is performed normally except that, when merging the partial paths, the reduced cost of the complete route must be adjusted.

4.10.2 Ride time constraints

The dial-a-ride problem (DARP) is an important variant of the PDP in which the hard time windows are replaced totally or partially by ride time constraints that prevent the commodities from staying too long inside the vehicle (Cordeau and Laporte 2007). These constraints are also called *dynamic time windows*, in contrast to the *static time windows* used in the PDPTW. Important applications of the DARP can be found in the context of people transportation, or in the delivery of perishable products. Ropke (2005) propose a simple way to adapt a BPC algorithm for the PDPTW to the DARP. It consists of adding *infeasible path elimination constraints* (IPECs) to the MP to handle the ride time constraints.

Gschwind and Irnich (2015) develop a BPC algorithm for the DARP that handles the ride time constraints in the pricing problem. They observe that although harder subproblems need to be solved, much better

bounds can be achieved, resulting in a positive trade-off. The additional burden in the pricing problem arises from the following two observations: 1) to ensure time window feasibility, every vertex should be visited as early as possible; 2) to ensure ride time feasibility, every pickup vertex should be visited as late as possible. These two observations are clearly in conflict with each other. To address this issue, the authors design two labeling algorithms. In the first, the label definition is the same as for the PDPTW but a label L_1 can dominate another label L_2 only if $|\mathcal{O}(L_1)| \leq 1$. In the second labeling algorithm, each label keeps track of the *latest possible delivery time* for each open request in function of the start of service time at the vertex associated with the label. This allows to define a strong dominance rule that can be applied for any value of $|\mathcal{O}(L_1)|$. Several valid inequalities devised for the PDPTW (Ropke, Cordeau, and Laporte 2007) and for the DARP (Cordeau 2006) are used to reinforce the MP.

Gschwind (2015) introduce the synchronized PDP (SPDP), a generalization of the DARP in which a delivery vertex must be visited within a time interval defined in terms of minimum and maximum ride times associated with its corresponding pickup vertex. Because the pickup and the delivery vertices have to be visited by the same vehicle, these additional constraints are also called *intra-route synchronization constraints* (general synchronization aspects are discussed in Drexl 2012). Due to the complexity of the resulting pricing problems when considering simultaneously minimum and maximum ride times, the author proposes four BPC algorithm variants which consider different pricing problems obtained by relaxing none or some features: the $\text{PP}^{\text{max}}_{\text{min}}$ that handles all the route constraints of the SPDP; the PP_{min} that only addresses minimum ride times; the PP_{max} that is similar to the one proposed by Gschwind and Irnich (2015) and consider only maximum ride times; and the PP_{relax} that relaxes both ride time constraints. Contrarily to the PP_{max}, in which maximum ride times may yield conflicting decisions with customer starting service times, minimum ride times are in conformity with the decisions of visiting customers as early as possible in the PP_{min}. Thus, the proposed labeling algorithm is similar to the one used to solve the pricing problem of the PDPTW. Regarding the $\text{PP}^{\text{max}}_{\text{min}}$, the presence of both minimum and maximum ride time constraints complicates considerably the pricing problem and a complex dominance rule is devised for the labeling algorithm. Except for the first BPC algorithm in which all routing constraints are handled in the pricing problem, the other three algorithms use IPECs in the MP to enforce the relaxed constraints as proposed by Ropke (2005).

Parragh, Sousa, and Almada-Lobo (2015) tackle the DARP with split requests and profits (DARPSRP) in the context of people transportation. In the DARPSRP, a revenue is associated with each request $i \in P$ which might remain unserved. If served, all the associated people must be transported to their destination, though they can be split into different vehicles. The objective aims at maximizing the total profit which is expressed by the total revenues collected minus the traveling costs. Due to the presence of paired pickups and deliveries in the DARPSRP, some of the known SDVRP solution properties (Gendreau et al. 2006, Desaulniers 2010) are no longer valid and cannot be exploited. In particular, an optimal solution might necessarily contain a route which visits more than once the same pickup vertex. In the BP algorithm of Parragh, Sousa, and Almada-Lobo (2015), the pricing problem consists of generating routes with positive reduced cost, in which all route feasibility constraints are respected, namely: vehicle capacity, customer time windows, maximum ride time, precedence and pairing. The authors devise a labeling algorithm inspired by that of Ropke and Cordeau (2009). A set of resources is defined to take into account the number of people associated with the onboard requests. Besides, two additional resources are used to compute the minimum possible duration of a route that would complete all open requests and, consequently, to check the feasibility of a label extension. When performing an extension over an arc (i, j), two cases are analyzed: if $j \in P$ (pickup vertex), as many labels as different possible splits of the people remaining to pick up at j may be generated. If $j \in D$, only a label associated with the number of people to be delivered is created. Moreover, unpromising extensions are avoided by applying a tailored dominance rule along with completion bounds.

Finally, Qu and Bard (2015) study a variant of the PDPTW that arises in the context of people transportation and considers a heterogeneous fleet, multiple shipment types and configurable vehicle capacities. No ride time constraint are imposed but the objective function penalizes each minute of ride time and of waiting before a pickup. In this problem, each pickup point (customer) is associated with a set of load requirements, i.e., specific conditions for transporting the items. For example, such requirements may ask for enough space for transporting a passenger and its wheelchair or walker. In turn, each vehicle type can be set a priori in different configurations to accommodate different types of demand. For example, a given

vehicle can transport three seating passengers or only two if they bring wheelchairs. The problem consists of designing feasible routes such that the number of used vehicles, the total traveled distance and the customers traveling times are minimized. The decisions also include selecting the configuration of each vehicle to be set up before starting its route. There is a pricing problem for each vehicle type. Each problem is solved by a labeling algorithm where a label includes the list of feasible vehicle configurations. To determine the list of feasible configurations, a generalized assignment problem is solved.

4.10.3 Transfers

Classic PDPs are subject to pairing and precedence relations which impose that the pickup and the delivery of a request be performed by the same vehicle. However, in some applications, some requests can be transferred from one vehicle to another at predetermined locations called *transfer points*. This extension of the PDP is called the PDP with transfers (PDPT) and is particularly challenging when the pickup and delivery points have time windows. Indeed, in this case, the PDPT involves a new type of precedence constraint stipulating that any transferred request must be delivered to its transfer point by a first vehicle before being picked up by a second vehicle which will transport it to its final destination.

Masson et al. (2013) address a special case of the PDTP with time windows called the PDP with shuttle routes (PDPS), where it is assumed that the requests have individual pickup points but they share a limited number of delivery points. In the PDPS, *pickup routes* visit first pickup locations and end by a visit to a single delivery point which may not be the final destination of the picked up requests, i.e., it may be a transfer point for some of these requests. In this case, direct *shuttle routes* are used to transport the transferred requests to their delivery point. The PDPS arises when, e.g., people need to be transported from their home to schools or rehabilitation centers. Masson et al. (2013) design two different set partitioning formulations with additional constraints for the PDPS. Both formulations involve binary variables associated with the pickup routes. In the first, nonnegative integer variables indicate the number of shuttles required between each pair of delivery points without specifying the transported requests. In the second which provides better lower bounds, binary variables associated with shuttle routes and their transported requests are used. A BPC algorithm is designed for each formulation. In both algorithms, there is one pricing problem for each possible delivery (transfer) point. It is an ESPPRC that involves a resource indicating the latest time at which the vehicle can arrive at the delivery vertex without violating the time windows of the picked up requests. In the second algorithm, a pricing problem is also needed to generate the shuttle routes. This problem corresponds, in general, to a binary knapsack problem. However, if a single person is associated with each request, it can be solved more efficiently using a sorting algorithm for each pair of delivery points.

Motivated by situations in which part of the operations can be performed at a certain cost by the public transit available, Ghilas, Cordeau, and Demir (2017) introduce a generalization of the PDPT called the PDPTW with scheduled lines (PDPTW-SL). In this problem, each request can be served directly by a single vehicle or indirectly using two vehicles and a scheduled public transit line. In the former case, the request is picked up by a vehicle which brings it to a transfer point that is serviced by a scheduled line. This request is then transported by the public transit to another transfer point. Finally, a second vehicle picks up the request at this second transfer point to deliver it to its final destination. Such indirect trips may be beneficial for transportation companies when pickup nodes are located far from the delivery nodes. To solve the PDPTW-SL, Ghilas, Cordeau, and Demir (2017) present a BPC algorithm that relies on a set partitioning MP with several sets of additional constraints that enforce the capacity of the scheduled lines and proper synchronization between the vehicle routes and the scheduled lines transporting the transferred requests. Furthermore, the authors develop a forward labeling algorithm for solving the ESPPRC pricing problems, one for each vehicle type and depot. This labeling algorithm uses labels that store three sets of requests: the open requests that have been picked up but not yet delivered, the completed requests, and the requests that have been pickup up at a transfer point. Note that a picked up request may be delivered to its final destination or to a transfer point. Finally, the authors devise a bidirectional search labeling algorithm for these pricing problems.

4.10.4 Loading constraints

In some contexts, the order in which the requests are collected or the shape (volume and/or size) of their items may have operational implications that must be taken into account by the models and algorithms. This feature is, in general, referred to as *loading constraints*. The following works concern loading constraints induced by the order of the collected requests.

Cherkesly, Desaulniers, and Laporte (2015) address a PDPTW subject to a LIFO (last-in-first-out) loading constraint, denoted PDPTWL, which arises when handling operations should be avoided, for instance, in the transportation of heavy, dangerous or large items. This LIFO constraint assumes that, in a vehicle, the items of the requests are stored in a single stack (e.g., from the front of the vehicle towards the access door at the back). Consequently, when requests are picked up, they are put on top of the stack. Furthermore, a delivery point can only be visited if its corresponding request is on top of the stack. The authors propose three BPC algorithms that differ in the way the LIFO constraint is handled. In the first, the LIFO constraint is imposed through *LIFO-infeasible path cuts* in the MP. In the second algorithm, this constraint is enforced directly in the pricing problem which requires the development of a new labeling algorithm. In particular, new label components indicating the position in the stack of each onboard request are considered and a specialized dominance rule is devised. This second approach yields better lower bounds than the first one, but the pricing problem is much harder to solve. Seeking a trade-off between these two algorithms, the authors introduce a third algorithm that incorporates LIFO-infeasible path cuts in the MP whenever needed and a labeling algorithm which partially imposes the LIFO policy. Indeed, given a positive integer κ, the LIFO constraint must be respected by all onboard requests that have remained in the top κ positions in the stack. Consequently, as soon as a request falls below the top κ positions, it is ejected from the stack and can be delivered in any order with respect to the other requests that have also been ejected.

The last two algorithms of Cherkesly, Desaulniers, and Laporte (2015) mentioned above are adapted by Cherkesly et al. (2016) to solve the PDPTW with multiple stacks. In this problem, each stack has a limited capacity and the LIFO loading constraint applies to each individual stack. The presence of multiple stacks is addressed in the labeling algorithms by means of components that indicate: the accumulated load under each request in a given stack, the relative position between requests inside the same stack, and the simultaneous presence of two requests in the same vehicle, but not in the same stack. Besides handling the loading constraints correctly, the use of these label components favors the elimination of symmetry between the identical stacks as a stack can be identified by its top request. When extending a label to a delivery vertex, a single label is created. However, when it is extended to a pickup vertex, multiple labels can be created, namely, one for each stack on which the request can be put. In the algorithm where the LIFO loading constraints are partially imposed in the pricing problem, it is possible to generate a path that does not respect these constraints with the proposed loading plan (assignments of the requests to the stacks). Nevertheless, with a different loading plan, the path may be feasible. Therefore, once a path with an infeasible loading plan is generated, the algorithm first checks if there exists a feasible loading plan for this path by solving a shortest path problem with resource constraints. This allows to reduce the number of LIFO-infeasible path cuts added to the MP. Finally, let us mention that these algorithms rely on valid inequalities including the RCCs that were adapted by Côté et al. (2012) for the pickup and delivery TSP with multiple stacks.

It may be beneficial in some cases to allow the rehandling of the load at intermediate points of a route, if this brings extra savings when compared with a strict LIFO policy. In this context, Veenstra et al. (2017) introduce the PDPTW with handling operations as an extension of the PDPTWL. In their study, rehandling operations are only allowed at delivery points. Two rehandling policies are analyzed: one that only allows compulsory rehandling, i.e., only the requests on top of the delivered request must be rehandled; and another that accepts compulsory and preventive rehandlings, meaning that all the requests in a vehicle can be rehandled at once. No cost is incurred for rehandling but extra service time, which depends on the number of items rehandled, must be accounted for each rehandling operation, yielding infeasible extensions due to the time windows. Veenstra et al. (2017) propose a BPC algorithm for each handling policy. In both cases, the labeling algorithm takes into account the relative position between any pair of requests like in Cherkesly et al. (2016). However, two requests are considered to be at the same position if their most recent rehandling operation was carried out at the same location.

4.10.5 Simultaneous pickups and deliveries

The VRP with simultaneous pickups and deliveries (VRPSPD) is an extension of the CVRP, where each customer requires a delivery, a pickup, or both. This problem belongs to the class of one-to-many-to-one PDPs because all delivered items originate from the depot, whereas all picked up items must be transported to the depot. It models real-life situations related to reverse logistics activities such as the distribution of beverages and the collection of empty cans and bottles.

Angelelli and Mansini (2002) develop the first BP algorithm to tackle the VRSPD with time windows (VRPTWSPD). They assume that each customer $i \in V'$ is associated with a nonnegative delivery demand q_i^D and a nonnegative pickup demand q_i^P. They notice that, if either $q_i^D > q_i^P, \forall i \in V'$, or $q_i^P > q_i^D, \forall i \in V'$, then an optimal solution to the VRPTWSPD can be found by solving a VRPTW in which the demand of every customer is given by its net demand. In the algorithm proposed for the general case, the pricing problem is solved by a labeling algorithm which relies on two dependent resources to enforce vehicle capacity: one stores the total demand collected in the route and another indicates the minimum capacity required along the route, i.e., the maximum load carried simultaneously. These resources were previously stated in Desaulniers et al. (1998).

Dell'Amico, Righini, and Salani (2006) design a BP algorithm to solve the VRPSPD, where the pricing problem is also solved by a labeling algorithm. To manage vehicle capacity in this algorithm, they rely on two resources that are different but equivalent to those used by Angelelli and Mansini (2002). Dell'Amico, Righini, and Salani (2006) propose various strategies to accelerate the search for negative reduced cost routes. In particular, they apply bidirectional search and limit the number of customers that can be visited in each forward and backward route to $\lceil \bar{\sigma}/2 \rceil$, where $\bar{\sigma}$ is an upper bound on the number of customers that can be visited in a route. This upper bound is computed by solving two binary knapsack problems, namely, one for the pickup demands and one for the delivery demands. Furthermore, they use a state-space relaxation obtained by replacing in the label definition the customer resource vector $\mathcal{E}$ with a single component $|\mathcal{E}|$ counting the number of customers visited. This relaxation allows the generation of routes with cycles. To keep the RMP tractable, variables associated with routes having a reduced cost greater than a given threshold are removed from the RMP when it reaches a maximum size. These columns are then stored in a pool for a maximum number of iterations. While in the pool, these routes are priced directly and added to the RMP if their reduced cost becomes negative. Finally, two new branching strategies are proposed: branching on cycles and branching on resource windows. When a cycle occurs in a fractional solution, it can be eliminated by creating multiple child nodes obtained by imposing/forbidding the use of some arcs in the cycle. The second type of branching is only applied when a customer vertex is involved in two cycles of two different routes. It is an adaptation of the branching proposed by Gélinas et al. (1995) and corresponds to branching on the resource windows of the two resources handling the vehicle capacity constraint. Using a label definition similar to the one considered in Dell'Amico, Righini, and Salani (2006), Righini and Salani (2008) show how to apply bidirectional search for solving the pricing problem arising in the VRPSPD.

Subramanian et al. (2013) extend the BPC algorithm of Fukasawa et al. (2006) for the CVRP to the VRPSPD. They introduce the concept of *pd*-routes. A *pd*-route starts and ends at the depot, and at any visited customer, the sum of all collected items plus the sum of all items to be delivered does not exceed vehicle capacity. In the BPC algorithm, the pricing problem consists of finding *pq*-routes with negative reduced cost. To speed up the labeling algorithm, the authors use completion bounds that are computed by finding the least-cost path from each customer to the depot, but only considering deliveries.

Finally, Gutiérrez-Jarpa et al. (2010) study the VRP with deliveries, selective pickups and time windows (VRPDSPTW), where all deliveries are mandatory whereas pickups are optional and partial pickups are not allowed. Performing a pickup at a customer yields a revenue. The authors propose a BP algorithm that not only solves the VRPDSPTW, but also five other variants of the problem involving: single or combined (pickup and delivery) demands, single or multiple visits; and backhauls. The MP includes set partitioning constraints for the deliveries and set packing constraints for the pickups. The labeling algorithm handles the vehicle capacity using the two resources proposed in Desaulniers et al. (1998). To address the variants allowing combined demands and backhauls, changes in the underlying network are required: in particular, one pickup and one delivery vertex is created for each customer with a combined demand. In the variants

where the combined demands of a customer may be satisfied with multiple visits, the elementarity of the routes is ensured by considering two resources for each customer, i.e., one per vertex.

5 Conclusion

The VRP was introduced in 1959 by Dantzig and Ramser (1959). Since then, a large number of VRPs have been studied, leading to one of the most prolific research areas in operations research. VRPs are, in general, $\mathcal{NP}$-hard and large-sized practical instances are usually solved using heuristics. Nevertheless, the advances achieved in the past decades on the exact BPC algorithms have led to the exact solutions of instances with more than 300 customers for the CVRP and 200 customers for the VRPTW. In this paper, we surveyed the methodological and modeling contributions made on the BP/BPC algorithms for VRPs since the 1990s. In Section 3, we discussed in details the proposed generic tools that can be applied for most VRP variants. In Section 4, we reviewed the main contributions that are specific to a VRP variant. We believe that this survey paper will help the researchers to identify more easily the contributions made on BP/BPC algorithms for vehicle routing so that they can further improve state-of-the-art algorithms and work on unexplored ideas and topics. Given the large number of papers that have recently been published on this subject, we can only expect that the literature will continue to grow in this domain, especially to address complex VRP variants including, e.g., synchronization constraints or further uncertainty factors.

References

Abdallah KS, Jang J, 2014 An exact solution for vehicle routing problems with semi-hard resource constraints. Computers & Industrial Engineering 76:366–377.

Ahuja RK, Magnanti TL, Orlin JB, 1993 Network flows: Theory, algorithms, and applications (Upper Saddle River, NJ, USA: Prentice-Hall, Inc.).

Angelelli E, Mansini R, 2002 The vehicle routing problem with time windows and simultaneous pick-up and delivery. Klose A, Speranza MG, Van Wassenhove LN, eds., Quantitative Approaches to Distribution Logistics and Supply Chain Management, 249–267 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Archetti C, Bianchessi N, Speranza MG, 2011 A column generation approach for the split delivery vehicle routing problem. Networks 58(4):241–254.

Archetti C, Bianchessi N, Speranza MG, 2013a The capacitated team orienteering problem with incomplete service. Optimization Letters 7:1405–1417.

Archetti C, Bianchessi N, Speranza MG, 2013b Optimal solutions for routing problems with profits. Discrete Applied Mathematics 161(4-5):547–557.

Archetti C, Bianchessi N, Speranza MG, 2014 The split delivery capacitated team orienteering problem. Networks 63(1):16–33.

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

Archetti C, Bianchessi N, Speranza MG, Hertz A, 2014 Incomplete service and split deliveries in a routing problem with profits. Networks 63(2):135–145.

Archetti C, Bouchard M, Desaulniers G, 2011 Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. Transportation Science 45(3):285–298.

Archetti C, Feillet D, Hertz A, Speranza MG, 2009 The capacitated team orientering and profitable tour problems. Journal of the Operational Research Society 60(6):831–842.

Archetti C, Speranza MG, 2012 Vehicle routing problems with split deliveries. International Transactions in Operational Research 19(1-2):3–22.

Augerat P, Belenguer J, Benavent E, Coberan A, Naddef D, Rinaldi G, 1998 Computational results with a branch-and-cut code for the capacitated vehicle routing problem. Technical report, Istituto di Analisi dei Sistemi ed Informatica, URL http://www.iasi.cnr.it/reports{_}html/R495.

Azi N, Gendreau M, Potvin JY, 2007 An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. European Journal of Operational Research 178(3):755–766.

Azi N, Gendreau M, Potvin JY, 2010 An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. European Journal of Operational Research 202(3):756–763.

Balas E, 1977 Some valid inequalities for the set partitioning problem. Annals of Discrete Mathematics 1:13–47.

Balas E, Padberg MW, 1976 Set partitioning: A survey. SIAM Review 18(4):710–760.

Baldacci R, Bartolini E, Mingozzi A, 2011 An exact algorithm for the pickup and delivery problem with time windows. Operations Research 59(2):414–426.

Baldacci R, Christofides N, Mingozzi A, 2008 An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming 115(2):351–385.

Baldacci R, Hadjiconstantinou E, Mingozzi A, 2004 An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Operations Research 52(5):723–738.

Baldacci R, Mingozzi A, 2009 A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming 120(2):347–380.

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

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

Balinski ML, Quandt RE, 1964 On an integer program for a delivery problem. Operations Research 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. Operations Research 46(3):316–329.

Beasley JE, Christofides N, 1989 An algorithm for the resource constrained shortest path problem. Networks 19(4):379–394.

Bektaş T, Laporte G, 2011 The pollution-routing problem. Transportation Research Part B: Methodological 45(8):1232–1250, supply chain disruption and risk management.

Ben Amor HMT, Desrosiers J, Frangioni A, 2009 On the choice of explicit stabilizing terms in column generation. Discrete Applied Mathematics 157(6):1167–1184.

Ben-Tal A, Ghaoui LE, Nemirovski A, 2009 Robust Optimization (Princeton University Press).

Berbeglia G, Cordeau JF, Gribkovskaia I, Laporte G, 2007 Static pickup and delivery problems: A classification scheme and survey. TOP 15(1):1–31.

Bertsimas D, Pachamanova D, Sim M, 2004 Robust linear optimization under general norms. Operations Research Letters 32(6):510–516.

Bettinelli A, Ceselli A, Righini G, 2011 A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. Transportation Research Part C: Emerging Technologies 19(5):723–740.

Bettinelli A, Ceselli A, Righini G, 2014 A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. Mathematical Programming Computation 6(2):171–197.

Boland N, Dethridge J, Dumitrescu I, 2006 Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 34(1):58–68.

Boschetti MA, Maniezzo V, Strappaveccia F, 2017 Route relaxations on GPU for vehicle routing problems. European Journal of Operational Research 258(2):456–466.

Boussier S, Feillet D, Gendreau M, 2007 An exact algorithm for team orienteering problems. 4OR 5(3):211–230.

Braekers K, Ramaekers K, Nieuwenhuyse IV, 2016 The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99:300–313.

Bulhões T, Hà MH, Martinelli R, Vidal T, 2018 The vehicle routing problem with service level constraints. European Journal of Operational Research 265(2):544–558.

Butt SE, Ryan DM, 1999 An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers & Operations Research 26(4):427–441.

Chabrier A, 2006 Vehicle routing problem with elementary shortest path based column generation. Computers & Operations Research 33(10):2972–2990.

Cherkesly M, Desaulniers G, Irnich S, Laporte G, 2016 Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. European Journal of Operational Research 250:782–793.

Cherkesly M, Desaulniers G, Laporte G, 2015 Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. Transportation Science 49(4):752–766.

Choi E, Tcha DW, 2007 A column generation approach to the heterogeneous fleet vehicle routing problem. Computers & Operations Research 34(7):2080–2095.

Christiansen CH, Lysgaard J, 2007 A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. Operations Research Letters 35(6):773–781.

Christofides N, Mingozzi A, Toth P, 1981a Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming 20(1):255–282.

Christofides N, Mingozzi A, Toth P, 1981b State-space relaxation procedures for the computation of bounds to routing problems. Networks 11:145–164.

Contardo C, Cordeau JF, Gendron B, 2014 An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. INFORMS Journal on Computing 26(1):88–102.

Contardo C, Desaulniers G, Lessard F, 2015 Reaching the elementary lower bound in the vehicle routing problem with time windows. Networks 65(1):88–99.

Contardo C, Martinelli R, 2014 A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discrete Optimization 12:129–146.

Cordeau JF, 2006 A branch-and-cut algorithm for the dial-a-ride problem. Operations Research 54(3):573–586.

Cordeau JF, Laporte G, 2007 The dial-a-ride problem: models and algorithms. Annals of Operations Research 153(1):29–46.

Côté JF, Archetti C, Speranza MG, Gendreau M, Potvin JY, 2012 A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks. Networks 60(4):212–226.

Dabia S, Demir E, Van Woensel T, 2017 An exact approach for a variant of the pollution-routing problem. Transportation Science 51(2):607–628.

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 Science 47(3):380–396.

Dantzig G, Fulkerson R, Johnson S, 1954 Solution of a large-scale travelling-salesman problem. Journal of the Operations Research Society of America 2(4):393–410.

Dantzig GB, Ramser JH, 1959 The truck dispatching problem. Management Science 6(1):80–91.

Dell'Amico M, Righini G, Salani M, 2006 A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. Transportation Science 40(2):235–247.

Desaulniers G, 2010 Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations Research 58(1):179–192.

Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D, 1998 A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. Crainic TG, Laporte G, eds., Fleet Management and Logistics, chapter 3, 57–93 (Boston, MA: Springer US).

Desaulniers G, Desrosiers J, Solomon MM, eds., 2005 Column Generation (Springer US), 1st edition.

Desaulniers G, Desrosiers J, Spoorendonk S, 2011 Cutting planes for branch-and-price algorithms. Networks 58(4):301–310.

Desaulniers G, Errico F, Irnich S, Schneider M, 2016 Exact algorithms for electric vehicle-routing problems with time windows. Operations Research 64(6):1388–1405.

Desaulniers G, Lessard F, Hadjar A, 2008 Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science 42(3):387–404.

Desaulniers G, Madsen OBG, Ropke S, 2014 The vehicle routing problem with time windows. Toth P, Vigo D, eds., Vehicle Routing: Problems, Methods and Applications, chapter 5, 119–159 (Philadelphia, PA: Society for Industrial and Applied Mathematics), 2nd edition.

Desaulniers G, Pecin D, Contardo C, 2017 Selective pricing in branch-price-and-cut algorithms for vehicle routing. EURO Journal on Transportation and Logistics Forthcoming:1–22.

Desrochers M, Desrosiers J, Solomon M, 1992 A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40(2):342–354.

Desrochers M, Soumis F, 1988 A generalized permanent labeling algorithm for the shortest path problem with time windows. INFOR 26:191–212.

Desrosiers J, Soumis F, Desrochers M, 1984 Routing with time windows by column generation. Networks 14(4):545–565.

Dinh T, Fukasawa R, Luedtke J, 2017 Exact algorithms for the chance-constrained vehicle routing problem. Mathematical Programming Forthcoming:1–34.

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

Dror M, 1994 Note on the complexity of the shortest path models for column generation in VRPTW. Operations Research 42(5):977–978.

Dumas Y, Desrosiers J, Soumis F, 1991 The pickup and delivery problem with time windows. European Journal of Operational Research 54(1):7–22.

Eksioglu B, Volkan A, Reisman A, 2009 The vehicle routing problem : A taxonomic review. Computers & Industrial Engineering 57(4):1472–1483.

Errico F, Desaulniers G, Gendreau G, Rei W, Roussau LM, 2016a The vehicle routing problem with hard time windows and stochastic service times. EURO Journal on Transportation and Logistics Forthcoming:1–29.

Errico F, Desaulniers G, Gendreau M, Rei W, Rousseau LM, 2016b A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times. European Journal of Operational Research 249(1):55–66.

Feillet D, 2010 A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR 8(4):407–424.

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

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.

Feillet D, Gendreau M, Rousseau LM, 2008 New refinements for the solution of vehicle routing problems with branch and price. INFOR: Information Systems and Operational Research 45(4):239–256.

Fukasawa R, He Q, Song Y, 2016 A branch-cut-and-price algorithm for the energy minimization vehicle routing problem. Transportation Science 50(1):23–34.

Fukasawa R, Longo H, Lysgaard J, Poggi de Aragão M, Reis M, Uchoa E, Werneck RF, 2006 Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming 106(3):491–511.

Gauvin C, Desaulniers G, Gendreau M, 2014 A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. Computers & Operations Research 50:141–153.

Gélinas S, Desrochers M, Desrosiers J, Solomon MM, 1995 A new branching strategy for time constrained routing problems with application to backhauling. Annals of Operations Research 61(1):91–109.

Gendreau M, Dejax P, Feillet D, Gueguen C, 2006 Vehicle routing with time windows and split deliveries. Technical Report 2006–851, Laboratoire Informatique d'Avignon, Avignon, France.

Gendreau M, Jabali O, Rei W, 2014 Stochastic vehicle routing problems. Toth P, Vigo D, eds., Vehicle Routing: Problems, Methods and Applications, chapter 8, 213–239 (Philadelphia, PA: Society for Industrial and Applied Mathematics), 2nd edition.

Gendreau M, Laporte G, Séguin R, 1996 Stochastic vehicle routing. European Journal of Operational Research 7(1992):3–12.

Ghilas V, Cordeau JF, Demir E, 2017 Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. Transportation Science *Forthcoming*.

Gschwind T, 2015 A comparison of column-generation approaches to the synchronized pickup and delivery problem. European Journal of Operational Research 247(1):60–71.

Gschwind T, Irnich S, 2015 Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. Transportation Science 49(2):335–354.

Gschwind T, Irnich S, Rothenbächer AK, Tilk C, 2018 Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research 266(1):521–530.

Gutiérrez-Jarpa G, Desaulniers G, Laporte G, Marianov V, 2010 A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. European Journal of Operational Research 206(2):341–349.

Hadjar A, Marcotte O, Soumis F, 2006 A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Operations Research 54(1):130–149.

Hernandez F, Feillet D, Giroudeau R, Naud O, 2014 A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration. 4OR 12(3):235–259.

Hernandez F, Feillet D, Giroudeau R, Naud O, 2016 Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. European Journal of Operational Research 249(2):551–559.

Ichoua S, Gendreau M, Potvin JY, 2003 Vehicle dispatching with time-dependent travel times. European Journal of Operational Research 144(2):379–396.

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

Irnich S, 2010 A new branch-and-price algorithm for the traveling tournament problem. European Journal of Operational Research 204(2):218–228.

Irnich S, Desaulniers G, 2005 Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds., Column Generation, chapter 2, 33–65 (Boston, MA: Springer US), 1st edition.

Irnich S, Desaulniers G, Desrosiers J, Hadjar A, 2010 Path-reduced costs for eliminating arcs in routing and scheduling. INFORMS Journal on Computing 22(2):297–313.

Irnich S, Toth P, Vigo D, 2014 The family of vehicle routing problems. Toth P, Vigo D, eds., Vehicle Routing: Problems, Methods and Applications, chapter 1, 1–33 (Philadelphia, PA: Society for Industrial and Applied Mathematics), 2nd edition.

Irnich S, Villeneuve D, 2006 The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. INFORMS Journal on Computing 18(3):391–406.

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

Kara I, Kara B, Yetis K, 2008 Cumulative vehicle routing problems. Caric T, Gold H, eds., Vehicle Routing Problem, chapter 6, 85–98 (Vienna: I-Tech Education and Publishing KG).

Keshtkaran M, Ziarati K, Bettinelli A, Vigo D, 2015 Enhanced exact solution methods for the team orienteering problem. International Journal of Production Research 54(2):591–601.

Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F, 1999 2-path cuts for the vehicle routing problem with time windows. Transportation Science 33:101–116.

Laporte G, Nobert Y, Desrochers M, 1985 Optimal routing under capacity and distance restrictions. Operations Research 33(5):1050–1073.

Lee C, Lee K, Park S, 2012 Robust vehicle routing problem with deadlines and travel time/demand uncertainty. Journal of the Operational Research Society 63:1294–1306.

Letchford AN, Salazar-González JJ, 2006 Projection results for vehicle routing. Mathematical Programming 105(2):251–274.

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.

Lin C, Choy K, Ho G, Chung S, Lam H, 2014 Survey of green vehicle routing problem: Past and future trends. Expert Systems with Applications 41(4, Part 1):1118–1138.

Lozano L, Duque D, Medaglia AL, 2016 An exact algorithm for the elementary shortest path problem with resource constraints. Transportation Science 50(1):348–357.

Lozano L, Medaglia AL, 2013 On an exact method for the constrained shortest path problem. Computers & Operations Research 40(1):378–384.

Lübbecke ME, 2005 Dual variable based fathoming in dynamic programs for column generation. European Journal of Operational Research 162(1):122–125.

Lübbecke ME, Desrosiers J, 2005 Selected topics in column generation. Operations Research 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 Science 51(2):668–687.

Lysgaard J, 2003 CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Technical report, Department of Management Science and Logistics, Aarhus School of Business.

Lysgaard J, Letchford AN, Eglese RW, 2004 A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100(2):423–445.

Lysgaard J, Wøhlk S, 2014 A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. European Journal of Operational Research 236(3):800–810.

Marsten RE, Hogan WW, Blankenship JW, 1975 The boxstep method for large-scale optimization. Operations Research 23(3):389–405.

Martinelli R, Pecin D, Poggi M, 2014 Efficient elementary and restricted non-elementary route pricing. European Journal of Operational Research 239(1):102–111.

Masson R, Ropke S, Lehuédé F, Péton O, 2013 A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. European Journal of Operational Research 236(3):849–862.

Mingozzi A, Roberti R, Toth P, 2013 An exact algorithm for the multitrip vehicle routing problem. INFORMS Journal on Computing 25(2):193–207.

Moreno L, Poggi de Aragão M, Uchoa E, 2010 Improved lower bounds for the split delivery vehicle routing problem. Operations Research Letters 38(4):302–306.

Muter I, Cordeau JF, Laporte G, 2014 A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. Transportation Science 48(3):425–441.

Naddef D, Rinaldi G, 2002 Branch-and-cut algorithms for the capacitated vrp. Toth P, Vigo D, eds., The vehicle routing problem, chapter 3, 53–84 (Philadelphia, PA, USA: Society for Industrial and Applied Mathematics), 1 edition.

Neame PJ, 1999 Nonsmooth Dual Methods in Integer Programming. Ph.D. thesis, University of Melbourne, Department of Mathematics and Statistics.

Nemhauser GL, Wolsey LA, 1988 Integer and combinatorial optimization (New York, NY, USA: Wiley-Interscience).

Noorizadegan M, Chen B, 2018 Vehicle routing with probabilistic capacity constraints. European Journal of Operational Research 270(2):544–555.

Oyola J, Arntzen H, Woodruff DL, 2016 The stochastic vehicle routing problem, a literature review, part I: Models. EURO Journal on Transportation and Logistics Forthcoming:1–29.

Parragh SN, Sousa JP, Almada-Lobo B, 2015 The dial-a-ride problem with split requests and profits. Transportation Science 49(2):311–334.

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

Pecin D, Pessoa A, Poggi M, Uchoa E, 2017b Improved branch-cut-and-price for capacitated vehicle routing. Mathematical Programming Computation 9(1):61–100.

Pecin D, Pessoa A, Poggi M, Uchoa E, Santos H, 2017c Limited memory rank-1 cuts for vehicle routing problems. Operations Research Letters 45(3):206–209.

Pecin DG, 2014 Exact algorithms for the capacitated vehicle routing problem. Ph.D. thesis, PUC-Rio.

Pelletier S, Jabali O, Laporte G, 2016 50th anniversary invited article - goods distribution with electric vehicles: Review and research perspectives. Transportation Science 50(1):3–22.

Pessoa A, Poggi de Aragão M, Uchoa E, 2008 Robust branch-cut-and-price algorithms for vehicle routing problems. Golden B, , Raghavan S, , Wasil E, eds., The Vehicle Routing Problem: Latest Advances and New Challenges, 297–325 (Boston, MA: Springer US).

Pessoa A, Poggi de Aragão M, Uchoa E, 2009 A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. Networks 54(4):167–177.

Pessoa A, Sadykov R, Uchoa E, 2018 Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. European Journal of Operational Research 270(2):530–543.

Pessoa A, Sadykov R, Uchoa E, Vanderbeck F, 2018 Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS Journal on Computing 30(2):339–360.

Petersen B, Pisinger D, Spoorendonk S, 2008 Chvátal-Gomory Rank-1 Cuts Used in a Dantzig-Wolfe Decomposition of the Vehicle Routing Problem with Time Windows, 397–419 (Boston, MA: Springer US).

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, chapter 3, 59–86 (Philadelphia, PA: Society for Industrial and Applied Mathematics), 2nd edition.

Poggi de Aragão M, Uchoa E, 2003 Integer program reformulation for robust branch-and-cut-and-price algorithms. Proceedings of the Conference Mathematical Programming in Rio: A Conference in Honour of Nelson Maculan, 56–61.

Qu Y, Bard JF, 2015 A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. Transportation Science 49(2):254–270.

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

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

Ropke S, 2005 Heuristic and exact algorithms for vehicle routing problems. Ph.D. thesis, University of Copenhagen, URL http://www.diku.dk/{~}sropke/Papers/PHDThesis.pdf.

Ropke S, Cordeau JF, 2009 Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science 43(3):267–286.

Ropke S, Cordeau JF, Laporte G, 2007 Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 49(4):258–272.

Rousseau LM, Gendreau M, Feillet D, 2007 Interior point stabilization for column generation. Operations Research Letters 35(5):660–668.

Rousseau LM, Gendreau M, Pesant G, 2004 Solving VRPTWs with constraint programming based column generation. Annals of Operations Research 130(1–4):199–216.

Ryan DM, Foster BA, 1981 An integer programming approach to scheduling. Wren A, ed., Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling, 269–280 (North-Holland).

Salani M, Vacca I, 2011 Branch and price for the vehicle routing problem with discrete split deliveries and time windows. European Journal of Operational Research 213(3):470–477.

Savelsbergh M, Sol M, 1998 Drive: Dynamic routing of independent vehicles. Operations Research 46(4):474–490.

Souyris S, Cortés CE, Ordóñez F, Weintraub A, 2013 A robust optimization approach to dispatching technicians under stochastic service times. Optimization Letters 7(7):1549–1568.

Spoorendonk S, Desaulniers G, 2010 Clique inequalities applied to the vehicle routing problem with time windows branch-cut-and-price. INFOR Journal 48(1):53–67.

Subramanian A, Uchoa E, Pessoa AA, Ochi LS, 2013 Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. Optimization Letters 7(7):1569–1581.

Taş D, Gendreau M, Dellaert N, Van Woensel T, de Kok AG, 2014 Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. European Journal of Operational Research 236(3):789–799.

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

Vanderbeck F, 2000 On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research 48(1):111–128.

Veenstra M, Cherkesly M, Desaulniers G, Laporte G, 2017 The pickup and delivery problem with time windows and handling operations. Computers & Operations Research 77:127–140.

Villeneuve D, Desaulniers G, 2005 The shortest path problem with forbidden paths. European Journal of Operational Research 53(4):97–107.

Walker WE, 1969 A method for obtaining the optimal dual solution to a linear program using the Dantzig-Wolfe decomposition. Operations Research 17(2):368–370.

Wentges P, 1997 Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming. International Transactions in Operational Research 4(2):151–162.