Les Cahiers du GERAD ISSN: 0711–2440

Selective pricing in branch-price-and-cut algorithms for vehicle routing

G. Desaulniers, D. Pecin,
C. Contardo

G–2016–110

November 2016

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

Selective pricing in branch-price-and-cut algorithms for vehicle routing

Guy Desaulniers [a]
Diego Pecin [a]
Claudio Contardo [c]

[a] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal, Canada*

[b] *GERAD, CIRRELT & École des Sciences de la Gestion, Université du Québec à Montréal, Montréal, Canada*

guy.desaulniers@gerad.ca
diegopecin@gmail.com
claudio.contardo@gerad.ca

November 2016

Les Cahiers du GERAD
G–2016–110

Abstract: Branch-price-and-cut is a leading methodology for solving various vehicle routing problems (VRPs). For many VRPs, the pricing problem of a branch-price-and-cut algorithm is highly complex and, to alleviate this difficulty, a relaxed pricing problem is used. In this paper, we introduce a new paradigm, called selective pricing, that can be applied in this context to improve the solution process of hard-to-solve VRPs by branch-price-and-cut. This paradigm requires the development of an ad hoc labeling algorithm. To illustrate selective pricing, we apply it to a branch-price-and-cut algorithm for the VRP with windows where the relaxed pricing problem is a shortest *ng*-path problem with resource constraints. We develop an ad hoc labeling algorithm and show through computational experiments that it can yield substantial time reductions (up to 32%) to reach a good lower bound on certain very-hard-to-solve VRPTW instances with 200 customers. We also introduce a new labeling heuristic which is also efficient at reducing the overall computational times.

Résumé : La méthode de *branch-cut-and-price* est la plus performante pour une grande panoplie de problèmes de tournées de véhicules (PTV). Pour plusieurs d'entre eux, le problème de *pricing* associé est hautement complexe. Pour alléger la résolution du problème de *pricing*, il est normal de considérer une relaxation du problème. Dans cet article, nous introduisons un nouveau paradigme de résolution, notamment le *pricing sélectif* qui peut être utilisé afin de réduire l'effort de calcul associé. Pour illustrer ce nouveau cadre méthodologique, nous l'avons implémenté dans une méthode de *branch-cut-and-price* pour le PTV avec fenêtres de temps (PTVFT) dont le problème de *pricing* est une relaxation avec des voisinages (ng-routes). Nous développons une méthode d'étiquetage spécialement modifiée pour inclure le *pricing sélectif*, et l'avons testée sur des instances difficiles du PTVFT avec 200 clients. Nos expérimentons numériques montrent des réductions jusqu'à 32% du temps de calcul par rapport à l'approche classique de résolution. Nous introduisons aussi une heuristique capable de réduire les temps de calcul davantage.

Acknowledgments: This work has been partially financed by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds de recherche du Québec –Nature et technologies (FRQNT) and the GERAD.

1 Introduction

Vehicle routing problems (VRPs, see Toth and Vigo, 2014) consist of determining least-cost vehicle routes that visit a set of customers to service them (typically, to deliver or pickup merchandise). Widely studied for more than 50 years, these problems remain challenging, especially with new variants that incorporate complex features from real-world applications such as time windows (Desaulniers et al., 2014), split deliveries (Irnich et al., 2014), pickups and deliveries (Battarra et al., 2014; Doerner and Salazar-González, 2014), and stochastic demands (Gendreau et al., 2014), to name just a few.

A generic integer programming model for the VRPs is as follows:

$$\min \quad \sum_{r \in R} c_r x_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{rh} x_r = b_h, \qquad \forall h \in H \tag{2}$$

$$x_r \in \{0,1\}, \qquad \forall r \in R, \tag{3}$$

where R denotes the set of feasible vehicle routes, c_r the cost of route $r \in R$, H the set of constraints, a_{rh} the contribution of route $r \in R$ to constraint $h \in H$, and b_h the right-hand side of this constraint. Each binary variable x_r indicates whether or not route $r \in R$ is selected in the solution. The objective function (1) aims at minimizing the total routing costs. Constraint set (2) may include various types of constraints, depending on the problem variant considered. For example, when each customer must be visited exactly once, they include set partitioning constraints (for such a constraint h, $b_h = 1$ and $a_{rh} \in \{0,1\}$ for all $r \in R$) to force one route to visit each customer. This set of constraints can also include inequalities, for instance, to limit the number of routes selected to the number of available vehicles. Finally, let us mention that additional variables not directly associated with routes (e.g., counting the number of vehicles used from each depot) may also be part of this model.

Over the years, various exact algorithms have been developed to solve VRPs. Currently, branch-price-and-cut algorithms are the most successful ones for a wide variety of VRPs. A branch-price-and-cut algorithm (see Barnhart et al., 1998; Lübbecke and Desrosiers, 2005) is a branch-and-bound algorithm in which the linear relaxation of a mixed-integer linear problem at each node of the search tree is solved using column generation and tightened with the addition of cuts. Column generation is an iterative algorithm that solves at each iteration a restricted master problem (RMP) and a pricing problem (PP). For model (1)–(3), the RMP corresponds to its linear relaxation restricted to a small subset of the routes in R, and the PP is often an elementary shortest path problem with resource constraints (ESPPRC) (see Irnich and Desaulniers, 2005). Indeed, elementary requirements forbidding multiple visits to the same customer along a route must often be enforced. At a given iteration, the RMP is solved to yield a primal and a dual solution. Then the PP is solved to find negative reduced cost columns (variables) with respect to this dual solution. If such columns are found, they are added to the RMP before starting a new iteration. Otherwise, the algorithm stops and the cost of the current RMP solution provides a lower bound.

For many VRPs, the PP turns out to be a problem with high complexity. For instance, the ESPPRC arising for the vehicle routing problem with time windows (VRPTW) is known to be strongly NP-hard (Dror, 1994). To alleviate the difficulty of solving a complex PP at each iteration, a relaxation of it is often solved instead, yielding the generation of infeasible routes, for example, routes visiting the same customer more than once. These routes can be part of linear relaxation solutions, reducing the quality of the computed lower bounds, but they are discarded from the solution afterwards, through cutting and branching.

Several route relaxations have been proposed for various VRPs. The most notorious ones are relaxations of the ESPPRC, which can be applied to several VRPs. To replace a special case of the ESPPRC considering a single resource, namely, time, Desrosiers et al. (1984) introduced the shortest path problem with time windows that omits the elementary requirements. This relaxation was generalized by Desrochers et al. (1992) who developed a branch-and-price algorithm for the VRPTW. The resulting PP was called the shortest path problem with resource constraints (SPPRC). To strengthen this relaxation, these authors also considered the SPPRC with 2-cycle elimination that forbids cycles of the form $i - j - i$ (where i and j represent customer

nodes). Later, Irnich and Villeneuve (2006) extended this idea to eliminate all cycles of length k or less, resulting in the SPPRC with k-cycle elimination. More recently, Baldacci et al. (2011) introduced the *ng*-route relaxation that rapidly became state-of-the-art when routes must be elementary like in the ESPPRC. An *ng*-route is a route that may contain certain cycles if they meet certain conditions. More precisely, let NG_i be a neighborhood of customer i that contains i and a subset of its closest neighbors such that $|NG_i| \leq \lambda$, where λ is a predefined parameter value. An *ng*-route can visit twice a customer i if at least one customer j such that $i \notin NG_j$ is visited between the two visits to i. Using this route relaxation, the PP becomes a shortest *ng*-path problem with resource constraints (*ng*-SPPRC) which can be solved by a labeling algorithm. If λ is small, the labeling algorithm is fast but the lower bounds may be weak. On the other hand, larger λ values yield better lower bounds but the PP is harder to solve.

In this paper, we present a new paradigm, called *selective pricing*, that can be applied to branch-price-and-cut algorithms for VRPs when a relaxed PP is used to generate columns. Its application to a specific VRP requires the development of a new algorithm to solve the PP. We illustrate its usefulness by concentrating on the *ng*-route relaxation. However, we believe that our main contribution is not the development of a new labeling algorithm for the *ng*-SPPRC but rather the introduction of this paradigm that opens up a new way of designing algorithms for solving relaxed PPs. Note also that this paradigm may be applicable to other problem types that are solvable by branch-price-and-cut.

This paper is structured as follows. Section 2 presents the selective pricing paradigm. Its application when the *ng*-SPPRC is used as a relaxed PP is described in Section 3. Computational results on the VRPTW are reported in Section 4. Conclusions are drawn in Section 5.

2 Selective pricing

When using a route relaxation, the set of routes R that can be considered in the MP is enlarged to a set $\hat{R}$. Traditionally, the role of the PP is to find negative reduced cost columns in $\hat{R}$ if at least one exists even if this route contains a cycle. The PP is then formulated as the problem of finding a route in $\hat{R}$ that has the least reduced cost. More precisely, let π_h, $h \in H$, be the dual variables associated with constraints (2) and let $\bar{c}_r = c_r - \sum_{h \in H} a_{rh}\pi_h$ be the reduced cost of a route $r \in \hat{R}$. The traditional PP is defined as:

$$z^{PP}(\hat{R}) \quad = \quad \min_{r \in \hat{R}} \quad \bar{c}_r. \tag{4}$$

To guarantee convergence of the column generation algorithm, it is not necessary to always find a least reduced cost route as long as negative reduced cost routes are identified. In consequence, the PP can rather be stated as:

$$\text{If } z^{PP}(\hat{R}) < 0, \text{ find at least one route } r \in \hat{R} \text{ with } \bar{c}_r < 0. \tag{5}$$

The labeling algorithm used for solving the PP is designed to do so. The column generation algorithm stops when $z^{PP}(\hat{R}) \geq 0$, i.e., when there are no more routes in $\hat{R}$ with a negative reduced cost. In this case, the optimal value z^{RMP} of the current RMP provides a lower bound, denoted $\underline{z}(\hat{R})$, for the current node of the branch-and-bound search tree.

Observe, however, that this stopping condition is not necessary to ensure that z^{RMP} is a valid lower bound. Indeed, condition $z^{PP}(R) \geq 0$ is sufficient and does not imply that $\bar{c}_r \geq 0$ for all $r \in \hat{R} \setminus R$, i.e., there might exist an infeasible route $r \in \hat{R} \setminus R$ with $\bar{c}_r < 0$. This condition is stated and proven in the following proposition.

Proposition 1 *The optimal value z^{RMP} of the RMP at a given column generation iteration is a valid lower bound for the current node of the search tree if $z^{PP}(R) \geq 0$ or, equivalently, if $\bar{c}_r \geq 0$, $\forall r \in R$.*

Proof. Let $R' \subseteq \hat{R} \setminus R$ be the subset of infeasible routes that have been generated and that are taken into account in the current RMP. Because $R' \cup R \supseteq R$, considering only the set of routes $R' \cup R$ in the MP yields a relaxation of the original model (1)–(3) and thus its optimal solution provides a valid lower bound z^{RMP}. □

In the following, we denote by $\underline{z}(R' \cup R) = z^{RMP}$ this lower bound.

Following Proposition 1, we can re-state the PP as follows:

$$\text{If } z^{PP}(R) < 0, \text{ find at least one route } r \in \hat{R} \text{ with } \bar{c}_r < 0. \tag{6}$$

The difference between the traditional PP definition (5) and this new PP definition (6) is subtle ($\hat{R}$ is replaced by R in the condition) but may allow to define pricing algorithms that are more efficient than the existing ones. In fact, these algorithms may discard infeasible routes in $\hat{R} \setminus R$ even if they are not dominated. To ensure the exactness of the column generation algorithm, these pricing algorithms must, however, identify a negative reduced cost route in $\hat{R}$ (i.e., feasible or not) if there exists at least one feasible route in R that has a negative reduced cost. For these reasons, we say that the pricing process is *selective* or that *selective pricing* is applied.

Let us make some remarks about the usage of selective pricing in a branch-price-and-cut algorithm.

1. At a node of the search tree, selective pricing can yield a better lower bound than traditional pricing. Indeed, if R' corresponds to the set of infeasible routes in the last RMP solved in this node, then $\underline{z}(R' \cup R) \geq \underline{z}(\hat{R})$ because $R' \cup R \subseteq \hat{R}$.

2. The lower bound $\underline{z}(R' \cup R)$ computed at a node depends on R'. Therefore, different parameter configurations for the same algorithm might yield different lower bounds at a same node.

3. In a search tree, the lower bound achieved at a node may be less than the lower bound achieved at its parent node because infeasible routes that were not generated in the parent node may be generated in the child node.

To the best of our knowledge, only Cherkesly et al. (2015) have used a special case of selective pricing in the vehicle routing literature. They introduced different branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in, first-out loading constraints. In the algorithms based on an ng-path relaxation, the labeling algorithm used to solve the PP may discard routes with cycles without proving that they are dominated by other routes. Cherkesly et al. (2015) only gave a very short justification for allowing this. In this section, we have presented selective pricing in a general context and fully justified it.

3 Selective pricing with the ng-SPPRC

To illustrate selective pricing, we develop a new labeling algorithm for solving[1] the ng-SPPRC. We present this algorithm in the context of the VRPTW which is used as an illustrative example for the rest of this paper. The proposed ideas are, however, applicable to other VRPs.

This section is divided in three subsections. First, we define the VRPTW and the resulting ng-SPPRC PP. Second, we introduce the new labeling algorithm. Finally, we discuss a heuristic version of it that is less time-consuming and can be used to try to generate negative reduced cost columns rapidly.

3.1 The vehicle routing problem with time windows

The VRPTW (see Desaulniers et al., 2014) involves a set of customers N and a sufficiently large set of vehicles. With each customer $i \in N$ are associated a demand d_i, a service time s_i, and a time window $[e_i, \ell_i]$ within which the service must start. The vehicles are all identical with a capacity Q and their routes must start and end at a common depot. For each pair of locations i and j, the travel time t_{ij} from i to j and the travel cost c_{ij} are assumed to be known. The VRPTW consists of finding feasible routes such that each customer is visited once and the total routing costs are minimized. A route is said to be feasible if the sum of the demands of the visited customers does not exceed Q and if the time windows of these customers are respected. The cost of a route is computed as the sum of the costs of the links forming it.

[1] We continue to use the expression "solving the PP" even if we define the PP according to (6).

For the VRPTW, model (1)–(3) becomes:

$$\min \quad \sum_{r \in R} c_r x_r \tag{7}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ri} x_r = 1, \qquad \forall i \in N \tag{8}$$

$$x_r \in \{0,1\}, \qquad \forall r \in R, \tag{9}$$

where a_{ri} is an integer constant equal to the number of times that route $r \in R$ visits customer $i \in N$ ($a_{ri} \in \{0,1\}$ if r is elementary). Objective function (7) minimizes the total routing costs, whereas constraints (8) ensure that each customer is visited by a single route. The dual variables associated with constraints (8) are denoted π_i, $i \in N$, and the reduced cost of route variable x_r is given by $\bar{c}_r = c_r - \sum_{i \in N} a_{ri}\pi_i$. In a classical branch-price-and-cut algorithm for the VRPTW, the PP corresponds to an ESPPRC, but is often replaced by an ng-SPPRC PP in the most recent works.

The ng-SPPRC PP can be defined on a directed graph $G = (V, A)$ with node set $V = N \cup \{o, \bar{o}\}$ and arc set A. Nodes o and $\bar{o}$ represent the depot at the beginning and the end of a route, respectively. With these nodes, we also associate time windows $[e_o, \ell_o] = [0, 0]$ and $[e_{\bar{o}}, \ell_{\bar{o}}] = [0, \mathcal{T}]$, where $\mathcal{T}$ is the horizon length, and define without loss of generality $s_o = d_o = d_{\bar{o}} = \pi_o = 0$. The arc set is given by

$$A = \{(i,j) \in V \times V \mid i \neq \bar{o}, j \neq o, e_i + s_i + t_{ij} \leq \ell_j, d_i + d_j \leq Q\}.$$

With each arc $(i,j) \in A$, we define its reduced cost as $\bar{c}_{ij} = c_{ij} - \pi_i$. Finally, let $NG_i \subseteq N$ be the neighborhood of node $i \in N$, i.e., a subset of at most λ customers including i as described in Section 1.

Let $p = (v_0, v_1, \ldots, v_k)$, $v_j \in V$ for all $j = 0, 1, \ldots, k$, be a path in G. It corresponds to a feasible route in $\hat{R}$, hereafter a feasible ng-route, if it

- starts and ends at the depot: $v_0 = o$ and $v_k = \bar{o}$;
- satisfies vehicle capacity: $\sum_{j=1}^{k-1} d_{v_j} \leq Q$;
- satisfies time windows: for every node v_j, $j = 0, 1, \ldots, , k$, the earliest start of service time $T_{v_j} \in [e_{v_j}, \ell_{v_j}]$, where $T_{v_0} = e_{v_0} = 0$ and $T_{v_{j+1}} = \max\{e_{v_{j+1}}, T_{v_j} + s_{v_j} + t_{v_j, v_{j+1}}\}$ for all $j = 1, \ldots, k$;
- does not contain cycles that violate the ng-route requirements: for every pair of integers j_1 and j_2 such that $0 < j_1 < j_2 < k$ and $v_{j_1} = v_{j_2}$, there exists another integer j_3 such that $j_1 < j_3 < j_2$ and $v_{j_1} \notin NG_{v_{j_3}}$.

This path is elementary (i.e., it belongs to R) if $v_{j_1} \neq v_{j_2}$ for all pairs of integers j_1 and j_2 such that $0 < j_1 < j_2 < k$. The reduced cost of p is given by $\bar{c}_p = \sum_{j=0}^{k-1} \bar{c}_{v_j, v_{j+1}}$.

According to definition (6), the selective ng-SPPRC PP corresponds to finding a negative reduced cost feasible ng-route in G if there exists at least one negative reduced cost feasible elementary route in G, or proving that no such elementary route exists.

3.2 Labeling algorithm

We start by describing a (mono-directional) labeling algorithm for solving the ng-SPPRC without selective pricing. A labeling algorithm starts from an initial label at node o of G and extends labels forwardly in G using resource extension functions. Each label L is associated with a node $n(L)$ and represents a feasible ng-path $p(L)$ from o to $n(L)$. To avoid enumerating all feasible ng-paths, a dominance rule is applied to eliminate labels that cannot yield an optimal ng-path.

For the ng-SPPRC arising from the VRPTW, a label L contains the following components: $n(L)$, its resident node; $Z(L)$, the reduced cost of $p(L)$; $D(L)$, the load accumulated along $p(L)$; $T(L)$, the earliest service start time at $n(L)$ if reached using $p(L)$; and $M(L) \subseteq NG_{n(L)}$, the subset of nodes to which label L cannot be extended because it would be infeasible with regards to vehicle capacity, time windows or the ng-route constraints. This label writes as $L = (n(L), Z(L), D(L), T(L), M(L))$. The initial label at node o is

given by $L_o = (o, 0, 0, 0, \emptyset)$. A label L with $n(L) = i$ can be extended along an arc $(i, j) \in A$ only if $j \notin M(L)$. When performing this extension, the following resource extension functions are used to create a new label L':

$$\begin{aligned} n(L') &= j && (10)\\ Z(L') &= Z(L) + \bar{c}_{ij} && (11)\\ D(L') &= D(L) + d_j && (12)\\ T(L') &= \max\{e_j, T(L) + s_i + t_{ij}\} && (13)\\ M(L') &= \big(\{j\} \cup M(L) \cup U(n(L'), D(L'), T(L'))\big) \cap NG_j, && (14) \end{aligned}$$

where $U(n, D, T) = \{\nu \in N \cup \{\bar{o}\} \mid D + d_\nu > Q \text{ or } T + s_n + t_{n\nu} > \ell_\nu\}$ is the set of unreachable nodes from a label at node n associated with a load of D and an earliest service start time of T. In the above definition of $U(n, D, T)$, we assume that the service and travel time consumptions along the arcs satisfy the triangle inequality. If this is not the case, $s_n + t_{n\nu}$ must be replaced by the shortest path duration between n and ν. Finally, a label L is said to dominate a label L' if

$$\begin{aligned} n(L) &= n(L') && (15)\\ Z(L) &\leq Z(L') && (16)\\ D(L) &\leq D(L') && (17)\\ T(L) &\leq T(L') && (18)\\ M(L) &\subseteq M(L'). && (19) \end{aligned}$$

All dominated labels are discarded except when two labels L and L' dominate each other (i.e., all the previous relations hold at equality). In this case, one of these two labels is kept unless it is dominated by another label.

Note that (15)–(19) are sufficient conditions which guarantee that any feasible extension χ of $p(L')$ is also feasible for $p(L)$ and that the reduced cost of $p(L) \oplus \chi$ is less than or equal to that of $p(L') \oplus \chi$, where the symbol $\oplus$ denotes the concatenation of the two paths it links.

In the rest of this section, we propose to modify the above algorithm to perform selective pricing. In this new algorithm, we consider for each label L the set of paths $S(L)$ that contains $p(L)$ and all paths identified by the algorithm as dominated by $p(L)$. These paths include the extensions of the paths that were dominated by the subpaths of $p(L)$ and, recursively, the extensions of the paths dominated by them. For example, let $p(L) = (o, v_1, \ldots, v_k)$, where $v_j \in N \cup \{\bar{o}\}$, $j \in \{1, \ldots, k\}$. Denote by L_j, $j \in \{1, \ldots, k\}$, the label associated with subpath $p(L_j) = (o, v_1, \ldots, v_j)$. In the course of the algorithm, if L_j dominates another label L_m associated with path $p(L_m) = (o, w_1, w_2, \ldots, w_m = v_j)$, then the path $p(L_m) \oplus (v_j, v_{j+1}, \ldots, v_k)$ becomes part of $S(L)$. Furthermore, if the labels associated with the paths $(o, w_1, \ldots, w_q)$, $q \in \{1, 2, \ldots, m\}$, have dominated other labels, then the corresponding paths extended up to node $n(L)$ are also part of $S(L)$. Finally, path $p(L)$ is always added to $S(L)$.

Figure 1 gives an example of a set $S(L)$ for a label L associated with path $p(L) = (o, 1, 2, 3, 4)$. It is assumed that paths $(o, 1, 6)$, $(o, 8, 9)$, $(o, 9, 2)$, and $(o, 5, 6, 7, 3)$ were dominated by paths $(o, 5, 6)$, $(o, 9)$, $(o, 1, 2)$, and $(o, 1, 2, 3)$, respectively. Because $(o, 9, 2)$ was dominated by the subpath $(o, 1, 2)$ of $p(L)$, the path $(o, 9, 2) \oplus (2, 3, 4) = (o, 9, 2, 3, 4) \in S(L)$. Furthermore, because $(o, 8, 9)$ was dominated by the subpath $(o, 9)$ of $(o, 9, 2)$, the path $(o, 8, 9, 2, 3, 4)$ also belongs to $S(L)$. Three other paths, including $p(L)$, belong to $S(L)$ as listed in this figure.

In the new algorithm, we do not need to store the set $S(L)$ for each label L. Instead, we only need to retain the customer nodes $C(L)$ that are common to all paths in $S(L)$. Let $N(p)$ be the set of customer nodes visited along path p. Then $C(L) = \bigcap_{p \in S(L)} N(p)$. Note that $C(L)$ is never empty if $n(L) \in N$ because it always contains at least $n(L)$. Note also that the set $S(L)$ may continue to enlarge after extending label L. In this case, $C(L)$ may diminish but if L has been extended, it becomes difficult to extend the updated set forwardly. In consequence, we will ensure that set $C(L)$ remains valid after the extension of label L by further restricting the labels that L can dominate.

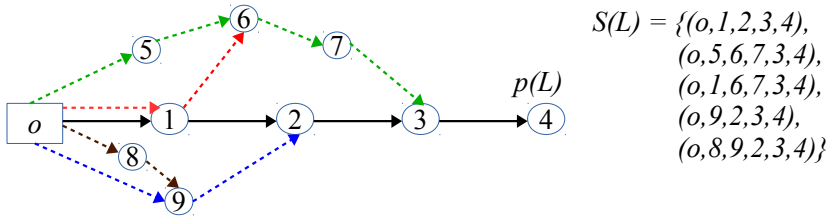

Figure 1: Example of a set $S(L)$

Observe that a label L does not have to be extended along an arc $(n(L), j)$ if $j \in C(L)$, even if $j \notin M(L)$. Indeed, such an extension would yield non-elementary paths for all paths in $S(L)$. Furthermore, in the dominance test, condition (19) can be relaxed to $M(L) \subseteq M(L') \cup C(L')$ because all extensions of L' that contain a node j in $C(L)$ can be considered infeasible and, therefore, it does not matter if j is in the memory of L.

In the modified algorithm, a label is defined as $L = (n(L), Z(L), D(L), T(L), M(L), C(L))$. In the initial label L_o, $C(L_o) = \emptyset$. A label L with $n(L) = i$ is extendable along an arc $(i, j) \in A$ only if $j \notin M(L) \cup C(L)$. When extending label L along this arc to create a new label L', its components $n(L')$, $Z(L')$, $D(L')$, $T(L')$, and $M(L')$ are computed using the extension functions (10)–(14). The computation of $C(L')$ is special. Indeed, it is initially computed when extending L along (i, j) as follows:

$$C(L') = C(L) \cup \{j\}. \tag{20}$$

Then, every time that L' dominates a label L'', $C(L')$ is updated as follows:

$$C(L') = C(L') \cap C(L''). \tag{21}$$

The dominance rule states as follows. We say that L dominates L' if

$$\begin{aligned}
n(L) &= n(L') && (22)\\
Z(L) &\leq Z(L') && (23)\\
D(L) &\leq D(L') && (24)\\
T(L) &\leq T(L') && (25)\\
M(L) &\subseteq M(L') \cup C(L') && (26)\\
C(L) &\subseteq C(L') \quad \text{if } L \text{ has been extended.} && (27)
\end{aligned}$$

Again, all dominated labels are discarded except when two labels dominate each other (one of them must be kept). Note that condition (27) ensures that set $C(L)$ and every set $C(\hat{L})$ associated with a label $\hat{L}$ resulting from an extension of L do not change due to newly dominated paths added to the corresponding sets $S(L)$ or $S(\hat{L})$ once L has been extended.

Dominance rule (22)–(27) is valid in the sense stated in the following proposition. In this proposition, we also associate with each label L the set $W(L) \subseteq N$ of the customer nodes visited along path $p(L)$.

Proposition 2 *Let $p = (v_0 = o, v_1, v_2, \ldots, v_k)$ be a feasible elementary path from o to a node $v_k \in V$ that would be associated with a label L if it was generated. The labeling algorithm with dominance rule (22)–(27) generates at least one feasible ng-path p' associated with a non-dominated label L' such that $n(L') = n(L)$, $p \in S(L')$, $D(L') \leq D(L)$, $T(L') \leq T(L)$, $M(L') \subseteq W(L)$, and $Z(L') \leq Z(L)$.*

Proof. Let $p_j = (v_0, v_1, \ldots, v_j)$ for $j = 0, 1, \ldots, k$, be the subpaths of p starting in v_0 and denote by L_j, $j = 0, 1, \ldots, k$, the associated labels if they were generated. The proof proceeds by induction on the value

of k. The case $k = 0$ is trivial because $p' = p = (o)$ is the only path generated that ends in node o. Now, assume that there exists an ng-path p'_{j-1} associated with a label L'_{j-1} such that

$$\begin{aligned} n(L'_{j-1}) &= n(L_{j-1}) && (28)\\ p_{j-1} &\in S(L'_{j-1}) && (29)\\ D(L'_{j-1}) &\leq D(L_{j-1}) && (30)\\ T(L'_{j-1}) &\leq T(L_{j-1}) && (31)\\ M(L'_{j-1}) &\subseteq W(L_{j-1}) && (32)\\ Z(L'_{j-1}) &\leq Z(L_{j-1}). && (33)\end{aligned}$$

Because $p_{j-1} \in S(L'_{j-1})$, we have $C(L'_{j-1}) \subseteq W(L_{j-1})$. Furthermore, $v_j \notin M(L'_{j-1}) \cup C(L'_{j-1})$ because p_j is feasible and elementary. Therefore, label L'_{j-1} can be extended along arc (v_{j-1}, v_j) to yield a label L'_j with

$$\begin{aligned} n(L'_j) &= n(L_j) && (34)\\ p_j &\in S(L'_j) && (35)\\ D(L'_j) &= D(L'_{j-1}) + d_{v_j} \leq D(L_{j-1}) + d_{v_j} = D(L_j) \leq Q && (36)\\ T(L'_j) &= \max\{e_{v_j}, T(L'_{j-1}) + s_{v_{j-1}} + t_{v_{j-1},v_j}\} \\ &\leq \max\{e_{v_j}, T(L_{j-1}) + s_{v_{j-1}} + t_{v_{j-1},v_j}\} = T(L_j) \leq \ell_{v_j} && (37)\\ M(L'_j) &\subseteq M(L'_{j-1}) \cup \{v_j\} \subseteq W(L_j) && (38)\\ Z(L'_j) &= Z(L'_{j-1}) + c_{ij} \leq Z(L_{j-1}) + c_{ij} = Z(L_j). && (39)\end{aligned}$$

Thus, L'_j is feasible and satisfies all conditions of the proposition if it is not dominated. In this case, $L' = L'_j$. Now, assume that L'_j is dominated by a feasible label L''_j. In this case,

$$\begin{aligned} n(L''_j) &= n(L'_j) = n(L_j) && (40)\\ p_j &\in S(L''_j) && (41)\\ D(L''_j) &\leq D(L'_j) \leq D(L_j) && (42)\\ T(L''_j) &\leq T(L'_j) \leq T(L_j) && (43)\\ Z(L''_j) &\leq Z(L'_j) \leq Z(L_j). && (44)\end{aligned}$$

Furthermore, because $p_j \in S(L'_j)$, observe that $C(L'_j) \subseteq W(L_j)$. Thus, $M(L''_j) \subseteq M(L'_j) \cup C(L'_j) \subseteq W(L_j)$. Consequently, L''_j has the same characteristics as L'_j. Repeating this process until obtaining a non-dominated label L proves the proposition. □

Corollary 1 *If there exists a feasible elementary path from o to $\bar{o}$ that has a negative reduced cost, then the algorithm generates at least one feasible ng-path from o to $\bar{o}$ that has a negative reduced cost.*

It ensues from this corollary that the proposed labeling algorithm solves the selective PP according to Definition 6.

Finally, note that the above algorithm can easily be converted into a backward labeling algorithm that would extend labels backwardly in G starting from an initial label at node $\bar{o}$. In consequence, a bidirectional labeling algorithm (see Righini and Salani, 2006) can also be devised to improve computational efficiency.

3.3 Heuristic labeling algorithm

Without condition (27) in the dominance rule, the new labeling algorithm would generate at most the same number of labels (and often less labels) than the standard algorithm for solving the ng-SPPRC. This condition is, however, necessary to ensure the exactness of the algorithm. We propose a heuristic version of the new labeling algorithm that omits condition (27) in the dominance rule and replace condition (26) by $M(L) \subseteq W(L')$. This latter condition is a relaxation of (26) because $M(L') \cup C(L') \subseteq W(L')$. The rationale behind this new condition when considered together with the other conditions (22)–(25), is that any feasible

extension χ of $p(L')$ that is not be feasible for $p(L)$ would yield a path $p(L') \oplus \chi$ that contains a cycle. Thus, such an extension does not need to be considered for $p(L')$. Nevertheless, Proposition 2 would not be true with this condition because there might exist a path $\tilde{p} \in S(L')$ for which χ is feasible, $\tilde{p} \oplus \chi$ does not contain a cycle, and $\tilde{p} \oplus \chi$ is not dominated by $p(L) \oplus \chi$.

As proposed by several authors (see Desaulniers et al., 2014), one or several heuristic algorithms can be called first at each column generation iteration. If one of them succeeds to find negative reduced cost columns, these columns are added to the RMP before starting a new iteration. Otherwise (i.e., if all heuristics fail to find such columns), the exact version of the labeling algorithm is invoked. The heuristic labeling algorithm presented above will be used in this way.

4 Computational results

In this section we provide computational evidence of the effectiveness of the selective pricing strategy when incorporated into a branch-price-and-cut algorithm, namely, the one developed by Pecin et al. (2016) for the VRPTW. This algorithm relies on an *ng*-SPPRC PP to generate negative reduced cost columns. Its labeling algorithm can thus be replaced by the new labeling algorithm introduced in Section 3. The algorithm of Pecin et al. (2016) uses three fast heuristic labeling algorithms to generate negative reduced cost columns as much as possible: the first heuristic keeps only the least reduced-cost label for each pair of time and load values. The second and third heuristics rely on a network containing a subset of arcs, ensuring that at least 7 arcs (or 12 arcs for the third heuristic) enter and exit each customer node as proposed by Desaulniers et al. (2008). For our tests, we combined the second and third heuristics with the one described in Section 3.3. We can thus compare the performance of two branch-price-and-cut algorithms, namely, the algorithm of Pecin et al. (2016) with the classical *ng*-SPPRC PP (hereafter labeled `StdP`) and the same algorithm but with the selective *ng*-SPPRC PP (labeled `SelP`). For both algorithms, the size of the neighborhoods was set to $\lambda = 15$. This is a compromise between the values used in Pecin et al. (2016) (20 for certain instances and 10 for others) although 15 is most probably not the best value for any of the tested instances. Furthermore, both algorithms use the same cuts and cut separation procedure as in Pecin et al. (2016). The cuts considered are the rounded capacity cuts (Laporte and Nobert, 1983) and some rank-1 Chvátal-Gomory cuts (see Jepsen et al., 2008; Petersen et al., 2008; Pecin et al., 2014, 2016). The former are qualified as *robust* cuts because handling their dual variables in the PP does not change the nature of the PP. The latter are said *non-robust* because they complexify the PP. In consequence, in the algorithms, non-robust cuts are not separated as long as robust cuts can be found. The algorithms are coded in C++ and rely on the CPLEX, version 12.6, linear programming solver.

Preliminary tests showed that selective pricing was not helpful (but neither harmful) for VRPTW instances that are not very hard to solve or for which cycles are not problematic. In consequence, we have decided to concentrate our experiments on the most difficult VRPTW instances from the dataset of Gehring and Homberger (2001) with 200 customers for which cycles can be problematic, namely, the seven instances in classes C2, RC2, and R2 that could not be solved by Pecin et al. (2016) and two additional instances in class RC2. Given that our goal was not to solve to optimality these instances but rather to illustrate the impact of applying selective pricing and the proposed labeling heuristic, we focused only on the root node results. All tests were executed on an Intel Xeon ES-2637 3.5GHz with 128GB RAM, running Linux Oracle Server 6.7.

We performed three sets of experiments to assess the effectiveness of the selective pricing strategy and the proposed labeling heuristic. In the first experiments, we compared the results of three algorithms, namely, `SelP`, `StdP`, and the algorithm in Pecin et al. (2016) (labeled `PCDU16`) which does not incorporate the new labeling heuristic. The experiments were conducted as follows. First, we solved the root node of each instance using the `SelP` algorithm. The algorithm was stopped when one of the following two conditions was met: 1) No more cuts were found; and 2) the time required by the labeling algorithm to solve the PP at a given iteration exceeded the minimum between 200 seconds and 3 times the time required to solve the last PP before adding non-robust cuts. This second condition indicates that solving the PP becomes too difficult and it is preferable to stop adding cuts and increase the difficulty of solving the PP. In any case, the lower bound achieved at the root node is equal to the value of the RMP at the last iteration when the PP was solved

using the exact labeling algorithm and no negative reduced cost columns were generated. Then, we ran the `StdP` algorithm and stopped it when it reached the lower bound obtained by `SelP`. For `PCDU16`, we simply collected from the Pecin et al. (2016) experimental results the time it took to achieve the same bound or the best bound if the same bound was not reached.

The results of these experiments are reported in Table 1. The columns *Time* give the total computational times required by the three algorithms. The columns *Gap* indicate the optimality gap (in percentage) between the computed lower bound and the upper bound corresponding to the cost of the best solution found by the hybrid genetic algorithm of Vidal et al. (2013). The last two columns specify the time reduction in percentage yielded by `SelP` with respect to the other two algorithms. When comparing `SelP` with `StdP`, we observe that `SelP` yields an average time gain of 16.3% for these instances. This average gain is modest but the results show that a substantial time reduction of 32.7% can be achieved for one instance while no time increase has been recorded. Next, when comparing `SelP` with `PCDU16`, we compute an average time reduction of 35.4% and an impressive maximum reduction of 59.9% for instance RC2-2-10. Note that the average reduction should be larger because `PCDU16` did not reach the lower bound achieved by `SelP` for instances R2-2-8 and RC2-2-8. In fact, for instance R2-2-8, the solution process was killed after running for 4.5 days. For RC2-2-8, the heuristic separation procedure could not find any violated cut. We observe that `SelP` yielded a time increase for only one instance, namely, RC2-2-3. This time increase is, however, very small. Consequently, we can say that `SelP` clearly outperforms `PCDU16`.

Table 1: Results of the three algorithms

	`SelP`		`StdP`		`PCDU16`		Time Gain (%)	
`Instance`	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	vs `StdP`	vs `PCDU16`
C2-2-3	19809	0.70	22633	0.70	24189	0.70	12.4	18.1
C2-2-4	51273	1.82	54735	1.82	110540	1.82	6.3	53.6
R2-2-4	90186	0.18	122627	0.18	141827	0.18	26.5	36.4
R2-2-8	224930	0.39	240499	0.39	396535	0.52	6.5	43.3
RC2-2-3	36316	0.02	42828	0.02	34425	0.02	15.2	-5.5
RC2-2-4	224861	0.26	333896	0.26	357851	0.26	32.7	37.2
RC2-2-8	42042	0	56466	0	83057	0.04	25.5	49.4
RC2-2-9	144597	0.89	174460	0.89	196346	0.89	17.1	26.4
RC2-210	133951	1.14	140123	1.14	334078	1.14	4.4	59.9
Average							16.3	35.4

The second set of experiments focused on the exact labeling algorithms with and without selective pricing. These tests were performed on the five instances for which `SelP` required less than 100,000 seconds in Table 1. For each of these instances, we ran again `SelP` and collected some statistics about the solution process of the PP when the exact labeling algorithm with selective pricing is executed at two specific iterations, namely, the last iteration before starting to generate rank-1 cuts (labeled `Last iteration 1`) and the last iteration of the whole solution process (labeled `Last iteration 2`). At these two iterations, we also execute the standard labeling algorithm on the same PP, but the (potentially) found columns are not inserted in the RMP and its computational time is stored separately. The results of these experiments are given in Table 2. In this table, we report: under columns labeled NL, the number of non-dominated labels generated by each labeling algorithm; under columns labeled T, the computational time (in seconds) spent by each algorithm; under columns labeled RC, the minimum reduced cost computed by the standard labeling algorithm; and under the column labeled C, the number of binding rank-1 cuts. The results show that, in all but two cases (C2-2-3 and C2-2-4 for `Last iteration 1`), the selective pricing strategy was effective to abort the execution of the algorithm, that would have continued otherwise if standard pricing had been used. We observe that, in some cases, the minimum reduced costs found by the standard labeling algorithm are non-negligible, indicating that several additional iterations would be required to complete convergence with the standard labeling algorithm. Looking at the number of generated non-dominated labels, we observe that less labels are generated by the selective pricing algorithm in six cases and more in the other four cases. The variations are, however, relatively small in general (between -11.0 and 7.6% except for the case RC2-2-3, `Last iteration 2` where the number of labels increased by more than 27%). In terms of computational time, there is no clear dominance between the two algorithms, however on cases with very large computational times induced by the handling of the rank-1 cut duals, selective pricing seems to take the lead by some interesting margins.

Table 2: Results for the last pricing

	Last iteration 1					Last iteration 2					
	SelP		StdP			SelP		StdP			
Instance	NL	T	NL	T	RC	NL	T	NL	T	RC	C
C2-2-3	20121	21.3	21426	21.9	0	400019	467.4	433132	532.9	-5.5	235
C2-2-4	86698	60.7	88215	60.0	0	669452	408.5	696329	430.0	-5.1	238
R2-2-4	36033	34.0	38194	34.3	-22.9	57191	13.9	51512	12.9	-112.1	125
RC2-2-3	20542	15.9	21875	16.6	-120.9	81406	15.1	63771	11.3	-131.5	222
RC2-2-8	69907	11.7	67706	12.9	-11.0	53288	3.8	48355	3.4	-40.5	127

Finally, the last experiments consisted of solving the root node relaxation considering only capacity cuts using four different algorithms: `SelP`, `StdP`, and these two algorithms without incorporating the new labeling heuristic. These tests were run on the nine instances considered in Table 1. For each of these instances, the four algorithms produced almost the same lower bound at the end of the solution process and, therefore, it is fair to compare their computational times. The results of these tests are synthesized in Table 3. For each algorithm and each instance, we report the computational time in seconds. Furthermore, we indicate in the fourth and seventh columns the relative gain in time obtained by using the new labeling heuristic in `SelP` and `StdP`, respectively. Finally, the last two columns give the time reduction in percentage yielded by `SelP` with respect to `StdP` with and without the new heuristic, respectively. From these results, we first observe that using the heuristic is always profitable except for one case (RC2-210 with `SelP`). When used in `SelP` (resp. `StdP`), this heuristic reduces the average computational time by 12.4% (resp. 22.1%) and the reduction can be quite large (up to 58%) for certain instances. When looking at the time gains given in the next-to-last column, we observe that, compared to the standard labeling algorithm, selective pricing deteriorates the average computational time by 8.9% on average. This shows that selective pricing can be helpful when the PP is very hard to solve, that is, when non-robust cuts are generated in our case (see the results in Table 1).

Table 3: Results with capacity cuts only

	SelP			StdP			Time Gain (%)	
							SelP vs StdP	
Instance	Time (s) with Heu	Time (s) no Heu	Time Gain (%)	Time (s) with Heu	Time (s) no Heu	Time Gain (%)	with Heu	no Heu
C2-2-3	8244	8356	1.3	6696	9113	26.5	-23.1	8.3
C2-2-4	28815	35990	19.9	25532	38651	33.9	-12.9	6.9
R2-2-4	29622	33818	12.4	33572	36664	8.4	11.8	7.8
R2-2-8	40815	68610	40.5	38635	65152	40.7	-5.6	-5.3
RC2-2-3	11972	12584	4.9	11707	13342	12.3	-2.3	5.7
RC2-2-4	54189	86622	37.4	43677	103835	58.0	-24.1	16.6
RC2-2-8	7381	7618	3.1	7470	8121	8.0	1.2	6.2
RC2-2-9	9570	11493	16.7	9400	9842	4.5	-1.8	-16.8
RC2-210	36286	29138	-24.5	29385	31573	6.9	-23.5	7.7
Average			12.4			22.1	-8.9	4.1

5 Conclusion

In this paper, we introduced a new paradigm, called selective pricing, that can be applied to improve the solution process of hard-to-solve VRPs by branch-price-and-cut when route relaxation is used to reduce the complexity of the PP. This new paradigm requires the development of an ad hoc labeling algorithm which can find a relaxed-feasible route with a negative reduced cost when at least one non-relaxed-feasible route with a negative reduced cost exists or prove that no such non-relaxed-feasible routes exist even if some relaxed-feasible routes have a negative reduced cost. To illustrate selective pricing, we applied it to a branch-price-and-cut algorithm for the VRPTW where the relaxed PP is an *ng*-SPPRC. We developed an ad hoc labeling algorithm and showed through computational experiments that it can yield substantial time reductions (up to 32%) to reach a good lower bound on certain very-hard-to-solve VRPTW instances with 200 customers. We also introduced a new labeling heuristic which is also efficient at reducing the overall

computational times. Combining selective pricing and this heuristic produced an average time reduction of at least 35% compared to a state-of-the-art branch-price-and-cut algorithm.

This new paradigm opens up several research avenues. In particular, new labeling algorithms based on the selective pricing strategy can be devised for various VRPs. For instance, one might think about various ways of relaxing the PP for the pickup and delivery problem and, consequently, of defining new labeling algorithms that might yield improved lower bounds if selective pricing is well-defined. Finally, although the computational results presented in this paper are promising, it is a first attempt at using selective pricing for improving the performance of a branch-price-and-cut algorithm. We believe that new ideas can be proposed to enhance our work. For example, it might be interesting to devise a column management procedure that would remove relaxed-feasible routes from the RMP in the hope of avoiding to generate them again due to selective pricing. This mechanism would help to obtain better lower bounds.

References

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

C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. Operations Research, 46(3):316–329, 1998.

M. Battarra, J.-F. Cordeau, and M. Iori. Pickup-and-delivery problems for goods transportation. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization, chapter 6, pages 161–191. SIAM, 2nd edition, 2014.

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

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

G. Desaulniers, O. Madsen, and S. Ropke. The vehicle routing problem with time windows. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization, chapter 5, pages 119–159. SIAM, 2nd edition, 2014.

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

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

K. Doerner and J. Salazar-González. Pickup-and-delivery problems for people transportation. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization, chapter 7, pages 193–212. SIAM, 2nd edition, 2014.

M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. Operations Research, 42:977–979, 1994.

H. Gehring and J. Homberger. A parallel two-phase metaheuristic for routing problems with time windows. Asia-Pacific Journal of Operational Research, 18:35–47, 2001.

M. Gendreau, O. Jabali, and W. Rei. Stochastic vehicle routing problems. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization, chapter 8, pages 213–239. SIAM, 2nd edition, 2014.

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

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

S. Irnich, M. Schneider, and D. Vigo. Four variants of the vehicle routing problem. In P. Toth and D. Vigo, editors, Vehicle Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization, chapter 9, pages 241–271. SIAM, 2nd edition, 2014.

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

G. Laporte and Y. Nobert. A branch and bound algorithm for the capacitated vehicle routing problem. Operations-Research-Spektrum, 5(2):77–85, 1983.

M. Lübbecke and J. Desrosiers. Selected topics in column generation. Operations Research, 53(6):1007–1023, 2005.

D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. In J. Lee and J. Vygen, editors, Integer Programming and Combinatorial Optimization: 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings, pages 393–403. Springer International Publishing, 2014.

D. Pecin, C. Contardo, G. Desaulniers, and E. Uchoa. New enhancements for the exact solution of the vehicle routing problem with time windows. Technical Report G-2016-13, Cahiers du GERAD, 2016.

B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In The Vehicle Routing Problem: Latest Advances and New Challenges, pages 397–419. Springer, 2008.

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

P. Toth and D. Vigo, editors. Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization. SIAM, Philadelphia, PA, 2nd edition, 2014.

T. Vidal, T. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. Computers and Operations Research, 40:475–489, 2013.