# Enhanced Branch and Price and Cut for Vehicle Routing with Split Deliveries and Time Windows

**Article** *in* Transportation Science · August 2011

DOI: 10.2307/23018528

CITATIONS
41

READS
453

**3 authors:**

C. Archetti
ESSEC Business School in Paris
**91** PUBLICATIONS **2,767** CITATIONS
SEE PROFILE

Mathieu Bouchard
Laval University
**28** PUBLICATIONS **234** CITATIONS
SEE PROFILE

Guy Desaulniers
Polytechnique Montréal
**126** PUBLICATIONS **4,143** CITATIONS
SEE PROFILE

# Enhanced Branch-and-Price-and-Cut for Vehicle Routing with Split Deliveries and Time Windows

C. Archetti[(1)], M. Bouchard[(2)], G. Desaulniers[(3)]

[(1)] *University of Brescia, Department of Quantitative Methods, Brescia, Italy*

[(2)] *Université Laval, Mechanical Engineering Department, Forac Research Consortium, Québec, Canada*

[(3)] *Ecole Polytechnique de Montréal and GERAD, Montréal, Canada*

archetti@eco.unibs.it, mathieu.bouchard@forac.ulaval.ca, guy.desaulniers@gerad.ca

September 19, 2010

**Abstract**

In this paper, we study the split delivery vehicle routing problem with time windows (SDVRPTW) that is a variant of the well-known vehicle routing problem with time windows (VRPTW) where each customer can be served by more than one vehicle. We propose enhancement procedures for the exact branch-and-price-and-cut algorithm recently developed by Desaulniers (2010) for the SDVRPTW. In particular, we introduce a tabu search algorithm to solve the column generation subproblem, extensions of several classes of valid inequalities, and a new separation algorithm for the $k$-path inequalities. Computational results show the effectiveness of the proposed enhancements.

**Keywords** Vehicle routing, time windows, split deliveries, branch-and-price, valid inequalities, tabu search column generator.

## Introduction

The vehicle routing problem (VRP) is one of the most famous combinatorial optimization problem and consists of finding a set of minimum cost vehicle routes that serve a set of customers such that the demand collected (or picked up) by each vehicle does not exceed the vehicle capacity (for a survey on VRP see Toth and Vigo (2002)). A classical assumption of the VRP is that each customer is served by exactly one vehicle. This has an obvious consequence: the demand of each customer cannot exceed the vehicle capacity. In the split delivery vehicle routing problem (SDVRP) this assumption is relaxed, thus each customer can be served by several vehicles. The SDVRP has been introduced by Dror and Trudeau (1989, 1990) who showed that allowing split deliveries can result in remarkable savings both in the

number of vehicles needed to serve the customers and in the total distance traveled. In fact, these savings can reach 50% as shown by Archetti et al. (2006b) and is strongly related to the configuration of the customer demands, as shown in Archetti et al. (2008b).

In the recent years the SDVRP has received a lot of attention and the research in the field is growing. Given the high complexity of the problem, the research has been mainly focused on heuristic solution approaches. The first heuristic approach is a local search algorithm proposed by Dror and Trudeau (1989, 1990). Then, more complex approaches, like metaheuristics or hybrid algorithms, have been proposed: three construction heuristics by Frizzell and Giffin (1992), two branch-and-price heuristics by Sierksma and Tijssen (1998) and Jin et al. (2008), two tabu search heuristics by Ho and Haugland (2004) and Archetti et al. (2006a), a hybrid record-to-record travel/integer programming method by Chen et al. (2007), and a hybrid tabu search/integer programming method by Archetti et al. (2008a). Only few papers have been devoted to exact approaches: Dror et al. (1994) and Belenguer et al. (2000) propose classes of valid inequalities and a lower bounding procedure, respectively, while Lee et al. (2006) and Jin et al. (2007) develop two exact approaches based on a dynamic programming method and on an iterative two-stage method, respectively. For a survey on the SDVRP the reader is referred to Archetti and Speranza (2008).

Very few papers have dealt with the SDVRPTW. Frizzell and Giffin (1995) and Mullaseril et al. (1997) developed construction and improvement heuristics. Gendreau et al. (2006) introduced an exact branch-and-price-and-cut method that succeeded to solve instances with up to 50 customers. Recently, Desaulniers (2010) proposed a new branch-and-price-and-cut algorithm which is significantly different from the one proposed by Gendreau et al. (2006), the main difference being that Gendreau et al. (2006) consider the delivery amounts within the column generation master problem whereas Desaulniers (2010) considers them within the subproblem. The method proved to be very effective being able to solve instances with up to 100 customers.

Branch-and-price has been the leading methodology for solving the VRPTW since the beginning of the 1990s. It consists of a branch-and-bound method where the lower bounds in the search tree are computed by a column generation algorithm. However, as noticed in Sierksma and Tijssen (1998), Jin et al. (2008) and Desaulniers (2010), the adaptation to the SDVRPTW of the classical column generation approaches used for the VRPTW is not straightforward because the delivered quantities are also decision variables. A solution to this issue is proposed in Gendreau et al. (2006) where the decisions on the delivered quantities

are taken in the master problem. This modeling approach, however, yields an exponential number of constraints in the master problem, one for each generated route. In Desaulniers (2010) a different decomposition is proposed that avoids this drawback by handling the delivered quantities in the column generation phase. In this case, the pricing subproblem corresponds to an elementary shortest path problem with resource constraints combined with the linear relaxation of a bounded knapsack problem.

The algorithm of Desaulniers (2010) is currently the leading method for solving the SDVRPTW to optimality. Moreover, as mentioned by the author himself, the performance of this algorithm can certainly be improved by applying different acceleration strategies. This is exactly the focus of the present paper. In particular, we introduce new procedures to speed up the solution of the subproblem, on one side, and to strengthen the lower bound, on the other side. For the subproblem we develop a fast heuristic algorithm, namely a tabu search algorithm. Such an algorithm has proved itself as a very effective procedure to find negative reduced cost columns for the VRPTW (see Desaulniers et al. 2008), reducing the need for expensive calls to the exact pricing algorithm. To improve the lower bound, we propose extensions of several classes of valid inequalities and a new heuristic algorithm to separate the $k$-path inequalities. We report computational results on 504 benchmark instances showing the efficiency of the proposed enhancement methods: while in Desaulniers (2010) 176 out of these 504 instances were solved to optimality in less than one hour of computational time, with the new methodologies we are able to increase this number to 262 instances. Moreover, the number of 100-customer instances solved to optimality increased from 1 to 8.

The paper is organized as follows. In Section 1 we give a formal description of the SDVRPTW and report some known properties. In Section 2 we give a mathematical formulation of the problem. In Section 3 we summarize the branch-and-price-and-cut algorithm of Desaulniers (2010). In Section 4 we focus on the new enhancement procedures including the tabu search column generator (Section 4.1), the heuristic separation algorithm for the $k$-path inequalities (Section 4.2) and the valid inequalities (Section 4.3). Computational results are reported in Section 5 before drawing some conclusions in Section 6.

# 1 Problem description and properties

The SDVRPTW is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ and $\mathcal{A}$ are the vertex and arc sets, respectively. The vertex set $\mathcal{V}$ is composed of the vertices 0 and $n+1$ representing the

starting and ending depot, respectively, and the set of vertices $\mathcal{N} = \{1, \ldots, n\}$ representing the $n$ customers. A positive demand $d_i$ is associated with each customer $i \in \mathcal{N}$. The service at customer $i$ must start within a prescribed time window $[e_i, l_i]$, meaning that a vehicle can arrive at customer $i$ prior to $e_i$ but, in this case, it has to wait until $e_i$ before serving customer $i$. A time window $[e_0, l_0] = [e_{n+1}, l_{n+1}]$, that may be unconstraining, is also associated with the depots. The set of arcs $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ define the feasible movements between the different vertices in $\mathcal{V}$. A nonnegative cost $c_{ij}$ and travel time $t_{ij}$ is associated with each arc $(i, j) \in \mathcal{A}$. We assume that both cost and travel time matrices satisfy the triangle inequality. The service time $s_i > 0$ at customer $i$ is included in the travel time $t_{ij}$ and is independent of the quantity delivered to $i$. For each pair of vertices $i, j \in \mathcal{V}$, $i \neq j$, except the pairs $(0, n+1)$ and $(n+1, 0)$, there exists an arc $(i, j) \in \mathcal{A}$ if $e_i + t_{ij} \leq l_j$.

An unlimited fleet of identical vehicles with capacity $Q$ is available to serve the customers and each vehicle can perform a single route defined as a path that starts at the starting depot 0, visits certain customers in $\mathcal{N}$ and ends at the ending depot $n + 1$. A route is feasible if the total demand delivered at the visited customers does not exceed the vehicle capacity and if it respects the time window of each visited vertex. Its cost is given by the sum of the corresponding path's arc costs. The objective of the SDVRPTW is to find a set of least-cost feasible routes that serve all customers in $\mathcal{N}$. Note that each customer can be served by several vehicles and that $d_i$ can be greater than $Q$.

Under the assumption that the costs and travel times fulfill the triangle inequality, the following properties can be useful to reduce the search space of the SDVRPTW.

**Property 1** *(Desaulniers 2010) There exists an optimal solution in which each edge $(i, j)$ linking two customers appears at most once, where an edge $(i, j)$ represents the arc $(i, j)$ and, if it exists, its reverse arc $(j, i)$.*

**Property 2** *(Gendreau et al. 2006) There exists an optimal solution in which each route visits each customer at most once.*

This last property allows us to consider only elementary paths from 0 to $n + 1$ in $\mathcal{G}$ and this turns out to strengthen the formulation proposed in the next section. It can be shown that there always exists an optimal solution satisfying both properties.

# 2 Problem formulation

Desaulniers (2010) derived an extended formulation for the SDVRPTW by applying the Dantzig-Wolfe decomposition principle (Dantzig and Wolfe 1960) to a compact arc-flow formulation. Since the focus of this paper is on enhancement procedures, we present in this section a slightly modified (but equivalent) version of this extended formulation that contains the essential features to understand these procedures. The reader is referred to Desaulniers (2010) for a comprehensive exposition of the decomposition process.

The extended formulation relies on two concepts: feasible routes and extremal delivery patterns compatible with the routes. A route, which corresponds to a path in $\mathcal{G}$ from 0 to $n+1$, is feasible if there exists a feasible schedule w.r.t. the time windows of the visited vertices. A delivery pattern for a route specifies the quantity delivered to each visited customer along the route. The sum of these quantities must not exceed the vehicle capacity $Q$. Such a pattern is qualified as extremal if every visited customer $i$ receives either a full delivery $d_i$ or a zero delivery, except possibly for one customer that can receive a partial or split delivery, that is, a positive quantity less than $d_i$.

Before presenting the extended formulation, let us introduce some notation.

- $\mathcal{R}$: set of all feasible routes, i.e., routes satisfying time window constraints.
- $c_r = \sum_{(i,j)\in r} c_{ij}$: cost of route $r$.
- $a_{ir}$: binary parameter equal to 1 if customer $i$ is visited in route $r$, and 0 otherwise.
- $b_{ijr}$: binary parameter equal to 1 if arc $(i,j)$ is used in route $r$, and 0 otherwise.
- $\mathcal{W}_r$: set of all feasible extremal delivery patterns compatible with route $r$, i.e., patterns in which only the customers visited in $r$ can receive a positive quantity and at most one of them receives a split delivery.
- $\delta_{iw}$: quantity delivered at customer $i$ in delivery pattern $w$.
- $k^C_{\mathcal{U}} = \lceil \sum_{i\in\mathcal{U}} \frac{d_i}{Q} \rceil$: minimum number of vehicles needed to serve all customers in a subset $\mathcal{U} \subseteq \mathcal{N}$ when considering vehicle capacity. When $\mathcal{U} = \{i\}$, we write $k^C_i$ instead of $k^C_{\mathcal{U}}$.
- $k^T_{\mathcal{U}}$: parameter equal to 2 if we can prove (by solving a resource-constrained elementary shortest path problem) that one vehicle is not sufficient to serve all customers in $\mathcal{U} \subseteq \mathcal{N}$ when considering the time window constraints, and 1 otherwise (for more details, see Desaulniers et al. 2008).

- $k_{\mathcal{U}} = \max\{k^C_{\mathcal{U}}, k^T_{\mathcal{U}}\}$.
- $\mathcal{P}(\mathcal{N})$: set of the subsets $\mathcal{U} \subseteq \mathcal{N}$ such that $|\mathcal{U}| \geq 2$ and $k_{\mathcal{U}} \geq 2$.
- $\mathcal{A}^-(\mathcal{U}) = \{(i,j) \in \mathcal{A} : i \in \mathcal{V} \setminus \mathcal{U}, j \in \mathcal{U}\}$: subset of the arcs entering vertex subset $\mathcal{U}$.

Let us now define the following variables.

- $\theta_{rw}$: nonnegative variable indicating the number of vehicles assigned to route $r \in \mathcal{R}$ and delivery pattern $w \in \mathcal{W}_r$.
- $\theta_r$: nonnegative integer variable indicating the number of vehicles assigned to route $r \in \mathcal{R}$.

With this notation, the SDVRPTW can be formulated as follows.

$$\textit{Minimize} \quad \sum_{r\in\mathcal{R}} \sum_{w\in\mathcal{W}_r} c_r \theta_{rw} \tag{1}$$

$$\text{subject to:} \quad \sum_{r\in\mathcal{R}} \sum_{w\in\mathcal{W}_r} \delta_{iw} \theta_{rw} \geq d_i, \qquad \forall\, i \in \mathcal{N} \tag{2}$$

$$\sum_{r\in\mathcal{R}} \sum_{w\in\mathcal{W}_r} a_{ir} \theta_{rw} \geq k^C_i, \qquad \forall\, i \in \mathcal{N} \tag{3}$$

$$\sum_{r\in\mathcal{R}} \sum_{w\in\mathcal{W}_r} \sum_{(i,j)\in\mathcal{A}^-(\mathcal{U})} b_{ijr} \theta_{rw} \geq k_{\mathcal{U}}, \qquad \forall\, \mathcal{U} \in \mathcal{P}(\mathcal{N}) \tag{4}$$

$$\theta_{rw} \geq 0, \qquad \forall\, r \in \mathcal{R}, w \in \mathcal{W}_r \tag{5}$$

$$\theta_r = \sum_{w\in\mathcal{W}_r} \theta_{rw}, \ \text{integer}, \qquad \forall\, r \in \mathcal{R}. \tag{6}$$

The objective function (1) aims at minimizing total costs. Constraints (2) ensure that the demand of each customer is satisfied, while constraints (3) establish a minimum number of visits to each customer. Constraints (4) are the $k$-path inequalities introduced by Kohl et al. (1999). Such an inequality forces the assignment of at least $k_{\mathcal{U}}$ vehicles to serve the vertices in $\mathcal{U}$. Constraints (5) impose nonnegativity requirements on the variables $\theta_{rw}$, while constraints (6) impose integrality requirements on the aggregated route variables $\theta_r$. Note that, by Property 1, the integrality requirements (6) can be restricted to binary requirements for any route visiting at least two customers. In this case, any feasible (not necessarily

extremal) delivery pattern for such a route can be obtained as a convex combination of its extremal delivery patterns. This is the reason why integrality is imposed on the $\theta_r$ variables, and not on the $\theta_{rw}$ variables. Note that constraints (3) and (4) are implied by (2) and (6). They are introduced to strengthen the formulation.

In Desaulniers (2010), the extended formulation also contains another set of inequalities, called the arc inequalities, that impose a maximum flow of one on each edge as stipulated in Property 1. In the algorithm proposed by Desaulniers (2010), this set of inequalities and inequalities (4) are initially relaxed and introduced as cutting planes when they are identified as violated. Computational tests (Gendreau et al. 2006, Desaulniers 2010) have shown that, while violated arc cuts are easy to identify, they are not very useful. Nevertheless, they will be used for our experiments, but we will not discuss them further for the sake of brevity. Violated $k$-path cuts (4) are much harder to identify and can be very useful to tighthen the linear relaxation. In Section 4.2, we propose a new separation algorithm for these inequalities.

Formulation (1)–(6) contains an exponential number of variables and thus cannot be solved explicitly. To overcome this difficulty, Desaulniers (2010) developed a branch-and-price-and-cut method which is summarized in the next section.

# 3 Branch-and-price-and-cut method

Desaulniers (2010) introduced a branch-and-price-and-cut method for solving model (1)–(6), which is called the master problem (MP) in this context. In such a branch-and-bound method (see, e.g., Desrosiers and Lübbecke 2005), the lower bounds in the search tree are computed by column generation and cutting planes are added to tighten these bounds. In the following, we denote by LMP the linear relaxation of MP. Note that constraints (6) and variables $\theta_r$ are omitted from LMP.

Starting with a subset of columns (variables $\theta_{rw}$) in the LMP, the column generation method solves at each iteration the LMP restricted to the subset of known columns and a subproblem to produce negative reduced cost columns based on the dual solution of the current LMP. These columns are then introduced into LMP and the procedure is iterated until no negative reduced cost columns can be found. At that point, the optimal solution of LMP has been obtained.

The goal of the subproblem is to find at each iteration negative reduced cost columns or to prove that none exists. Consider the following dual variables of LMP.

- $\pi_i$: nonnegative dual variable of constraint (2) for customer $i$.
- $\alpha_i$: nonnegative dual variable of constraint (3) for customer $i$. Let $\alpha_0 = 0$.
- $\lambda_{\mathcal{U}}$: nonnegative dual variable of constraint (4) for vertex subset $\mathcal{U}$.

The reduced cost $\bar{c}_{rw}$ of variable $\theta_{rw}$ is given by

$$\bar{c}_{rw} = \sum_{(i,j)\in r} \bar{c}_{ij} - \sum_{i\in\mathcal{N}} \delta_{iw}\pi_i, \tag{7}$$

where

$$\bar{c}_{ij} = c_{ij} - \alpha_i - \sum_{\mathcal{U}\in\mathcal{P}(\mathcal{N}):\,(i,j)\in\mathcal{A}^-(\mathcal{U})} \lambda_{\mathcal{U}}, \qquad \forall\,(i,j)\in\mathcal{A}. \tag{8}$$

The subproblem can thus be formulated as

$$\min_{r\in\mathcal{R},w\in\mathcal{W}_r} \bar{c}_{rw}. \tag{9}$$

If its optimal value is nonnegative, then the column generation algorithm stops. Otherwise, at least one negative reduced cost column is identified.

This subproblem corresponds to an elementary shortest path problem with resource constraints (ESPPRC) combined with the linear relaxation of a bounded knapsack problem (LP-BKP). It is a variant of the ESPPRC (see Irnich and Desaulniers (2005)) for which Desaulniers (2010) outlines a new label-setting algorithm that exploits the fact that finding the optimal delivered quantities along a given path (according to the $\pi_i$ dual values) reduces to solving an LP-BKP. In this algorithm, a feasible partial path from the source vertex 0 to any vertex $i \in \mathcal{V}$ is represented by a label that contains $6 + n$ components to represent the (reduced) cost of the partial path and its resource consumptions. The efficiency of the algorithm relies in the ability to remove dominated labels that cannot lead to an optimal path. Those labels are identified using an ad hoc dominance rule. The reader is referred to Desaulniers (2010) for a complete description of this algorithm.

Desaulniers (2010) use different acceleration techniques to speed up the subproblem solution. Furthermore, before calling the exact label-setting algorithm described above, a heuristic version of it is invoked. This heuristic version is obtained by relaxing the dominance criterion, yielding many more dominated labels that are discarded and, thus, much less generated labels overall. The exact label-setting algorithm is called only when this heuristic algorithm fails.

To strengthen LMP, $k$-path inequalities (4) that are identified as violated by the current LMP solution are added to LMP. Given that their identification is computationally expensive, such cuts are sought only in the first few nodes of the branch-and-bound tree. Finally, to derive integer solutions, Desaulniers (2010) applies four branching rules concerning the total number of vehicles used, the number of vehicles visiting a customer, the flow on an arc, and the flow on two consecutive arcs.

# 4 Enhancement procedures

In this section we describe the enhancement procedures proposed to improve the performance of the branch-and-price-and-cut algorithm described in the previous section. In particular, the focus has been put on accelerating the solution of the subproblem and on improving the lower bounds. In Section 4.1 we present a tabu search algorithm that solves the subproblem in a heuristic but very efficient way. In Section 4.2 we describe heuristic algorithms to separate the $k$-path inequalities (4): in particular, two known algorithms have been implemented and a new separation algorithm is proposed. Finally, in Section 4.3, we discuss valid inequalities that yield a strengthened formulation.

## 4.1 Tabu search column generator

The column generation subproblem, which combines an ESPPRC with an LP-BKP, is an NP-hard problem. However, it is not necessary to solve it to optimality as long as negative reduced cost columns are found. To solve it heuristically and be able to find negative reduced cost columns with little computational time, we develop a tabu search algorithm similar to the one proposed by Desaulniers et al. (2008) for the VRPTW. The input of this algorithm is the set of routes associated with the basic columns in the computed solution of the current LMP. Each of these columns has a zero reduced cost and is thus a good starting point to find a negative reduced cost column. Let us denote with $R^B$ the set of routes associated with these basic columns. The tabu search algorithm works independently on each of these routes.

We now introduce some notation needed to describe the tabu search algorithm. Let $maxCol$ be a parameter specifying the maximum number of negative reduced cost columns that may be generated by the tabu search algorithm at a given column generation iteration. Let $maxIter$ be another parameter indicating the maximum number of tabu search iterations

to perform on each route $r \in R^B$ (i.e., after $maxIter$ iterations the tabu search algorithm starts from a new route $r' \in R^B$). Finally, we define $S^N$ to be the set of negative reduced cost columns found by the algorithm and $iter$ to be the number of iterations performed for a given starting route $r \in R^B$.

In the tabu search algorithm (see Algorithm 1), a solution $(r, w)$ is defined by a route $r$ starting at 0, visiting a set of vertices $\mathcal{N}_r \subseteq \mathcal{N}$ and ending at vertex $n+1$, and a corresponding delivery pattern $w$. The value of solution $(r, w)$ is given by $\bar{c}_{rw}$, the reduced cost of the corresponding variable $\theta_{rw}$ as defined by (7). The neighborhood of a solution $(r, w)$ is defined by the following two types of moves.

1. *Remove*: Consider a vertex $i \in \mathcal{N}_r$ and remove it from $r$. The removal is made by joining the predecessor of $i$ with the successor of $i$. The route obtained by this removal is denoted $r - i$.

2. *Insertion*: Consider a vertex $i \in \mathcal{N} \setminus \mathcal{N}_r$ and insert $i$ into $r$ using the cheapest insertion that is feasible, i.e., there is no violation of the time window constraints due to the insertion. Because the insertion position of $i$ does not influence the (optimal) quantity delivered at $i$ and, therefore, the term $-\sum_{i \in \mathcal{N}} \delta_{iw}\pi_i$ in the reduced cost (7) of a variable, the cheapest insertion is determined only according to the arc reduced costs $\bar{c}_{ij}$ defined by (8). The route derived from this insertion is denoted $r + i$.

For both types of moves, the optimal delivery pattern $w$ associated with the resulting route $r'$ (equal to $r-i$ or $r+i$) is computed using a procedure, called $OptimizeQuantities(r')$. This procedure simply finds the optimal solution of LP-BKP for the customers in $r'$ and works as follows. First, sort the vertices $i \in \mathcal{N}_{r'}$ in decreasing order of their corresponding dual values $\pi_i$. Denote by $i_1, i_2, \ldots, i_{|\mathcal{N}_{r'}|}$ the indices of the ordered vertices and let $\bar{d}_i = min\{d_i, Q\}$ be the maximum quantity that a single vehicle can deliver to customer $i$. Then, the optimal delivery pattern $w$ is given by

$$\delta_{i_1,w} = \bar{d}_{i_1} \tag{10}$$

$$\delta_{i_j,w} = \min\{\bar{d}_{i_j}, Q - \sum_{\ell=1}^{j-1} \delta_{i_\ell,w}\}, \qquad j = 2, \ldots, |\mathcal{N}_{r'}|, \tag{11}$$

that is, deliver as much as possible (i.e., the minimum between $\bar{d}_{i_j}$ and the residual capacity of the vehicle) to each vertex $i_j$ while respecting the proposed order. Note that procedure

**Algorithm 1** Tabu search column generator

$S^N \leftarrow \emptyset$
**for** each route $r \in R^B$ **do**
  $w \leftarrow OptimizeQuantities(r)$
  **if** $\bar{c}_{rw} < 0$ **then**
    $S^N \leftarrow S^N \cup \{(r, w)\}$
    **if** $|S^N| = maxCol$ **then**
      Return $S^N$
  $TL_{remove} \leftarrow \emptyset$, $TL_{insert} \leftarrow \emptyset$, $iter \leftarrow 1$
  **while** $iter \leq maxIter$ **do**
    $r_{best} \leftarrow NIL$, $w_{best} \leftarrow NIL$, $i_{best} \leftarrow NIL$
    **for** each vertex $i \in \mathcal{N}_r$ **do**
      **if** $i \notin TL_{remove}$ **then**
        $w \leftarrow OptimizeQuantities(r - i)$
        **if** $\bar{c}_{r-i,w} < 0$ **then**
          $S^N \leftarrow S^N \cup \{(r - i, w)\}$
          **if** $|S^N| = maxCol$ **then**
            Return $S^N$
        **if** $r_{best} = NIL$ or $\bar{c}_{r-i,w} < \bar{c}_{r_{best},w_{best}}$ **then**
          $r_{best} \leftarrow r - i$, $w_{best} \leftarrow w$, $i_{best} \leftarrow i$
    **for** each vertex $i \in \mathcal{N} \setminus \mathcal{N}_r$ **do**
      **if** $i \notin TL_{insert}$ **then**
        $w \leftarrow OptimizeQuantities(r + i)$
        **if** $\bar{c}_{r+i,w} < 0$ **then**
          $S^N \leftarrow S^N \cup \{(r + i, w)\}$
          **if** $|S^N| = maxCol$ **then**
            Return $S^N$
        **if** $r_{best} = NIL$ or $\bar{c}_{r+i,w} < \bar{c}_{r_{best},w_{best}}$ **then**
          $r_{best} \leftarrow r + i$, $w_{best} \leftarrow w$, $i_{best} \leftarrow i$
    **if** $r_{best}$ **then**
      **if** $i_{best} \in \mathcal{N}_r$ **then**
        $TL_{remove} \leftarrow TL_{remove} \cup \{i_{best}\}$
      **else**
        $TL_{insert} \leftarrow TL_{insert} \cup \{i_{best}\}$
      $r \leftarrow r_{best}$
    $iter \leftarrow iter + 1$
    Update $TL_{remove}$ and $TL_{insert}$
Return $S^N$

*OptimizeQuantities*$(r)$ is also applied for each route $r \in R^B$ in order to check if $r$ and its corresponding optimal delivery pattern $w$ forms a negative reduced cost solution.

Each evaluated solution $(r', w)$ with a negative reduced cost is added to the set $S^N$. In each iteration of the tabu search algorithm, the vertex $i$ yielding the best move is stored as $i_{best}$ and the resulting route and delivery pattern as $r_{best}$ and $w_{best}$, respectively. If $i_{best}$ is removed from $r$, then $i_{best}$ is added to a tabu list $TL_{insert}$ and the insertion of $i_{best}$ into the current route becomes tabu. Similarly, if $i_{best}$ is inserted into $r$, then $i_{best}$ is added to a tabu list $TL_{remove}$ and its removal from the current route becomes tabu. Each vertex in both tabu lists remains in the list for a number of iterations equal to $minTabu + \lfloor \tau\, U(\sqrt{n}) \rfloor$, where $minTabu$ and $\tau$ are predefined constants, and $U(x)$ is a uniform number selected in the interval $[0, x]$. At the end of each iteration, the tabu lists are updated, that is, customers that have reached their number of iterations in their tabu list are removed from it.

At each column generation, the tabu search algorithm is invoked first for solving the subproblem. If it fails, i.e., it is unable to find at least one negative reduced cost column, then the heuristic label-setting algorithm is called, followed if needed by the exact label-setting algorithm.

For our computational experiments, the values of the tabu search algorithm parameters were set as follows: $maxCol = 300$, $maxIter = 10$, $minTabu = 5$, and $\tau = 0.5$. These values (and the other parameter values given in the subsequent sections) were chosen following preliminary computational tests. In particular, parameters $minTabu$ and $\tau$ have been set to values such that cycling is avoided but also a sufficiently large set of non tabu solutions is available at each iteration in order to let the algorithm move around the solution space. Moreover, preliminary tests showed that using a variable length of the tabu list which is related to the size of the instance $(n)$ tends to give better results with respect to using a fixed length.

## 4.2 Heuristic separation algorithms for $k$-path inequalities

As already mentioned, inequalities (4) are first relaxed and introduced in LMP only when they are identified as violated. In order to find such violated inequalities, Desaulniers (2010) applied a separation algorithm based on a partial enumeration of subsets $\mathcal{U} \subseteq \mathcal{N}$. This algorithm is heuristic in the sense that not all subsets $\mathcal{U} \subseteq \mathcal{N}$ are enumerated. Even with a partial enumeration, this algorithm can be very time consuming especially for certain classes of instances. Thus, we propose different separation algorithms in order to speed up

the separation process and be able to find a larger number of violated inequalities. All the algorithms we are going to describe only consider the vehicle capacity constraint. Therefore, for each subset $\mathcal{U} \subseteq \mathcal{N}$, they check if the number of vehicles serving the customers in $\mathcal{U}$ is greater than or equal to $k^C_{\mathcal{U}}$.

We first implemented two heuristic separation algorithms proposed by Ralphs et al. (2003) for the classical VRP. The first heuristic is called the *extended shrinking heuristic* and works as follows. Let $\tilde{\theta}$ be the computed solution of the current LMP and $\tilde{y}_{ij} = \sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} b_{ijr} \tilde{\theta}_{rw}$ be the total flow on arc $(i,j) \in \mathcal{A}$. Let us define the *support graph* $\widehat{G} = (\widehat{\mathcal{V}}, \widehat{\mathcal{A}})$ of this solution where $\widehat{\mathcal{V}} = \mathcal{V}$ is the vertex set and $\widehat{\mathcal{A}} = \{(i,j) \in \mathcal{A} : \tilde{y}_{ij} > 0\}$ the arc set. The heuristic is iterative. At each iteration, it selects the arc $(i,j) \in \widehat{\mathcal{A}}$ such that $i \neq 0$, $j \neq n+1$, and $y_{ij} + y_{ji}$ (only $y_{ij}$ if the reverse arc $(j,i)$ does not exist) is maximum. If the subset $\mathcal{U}$ corresponding to the customers associated with $i$ and $j$ induces a violated inequality (4), then this inequality is added to LMP. Otherwise, the arc is *shrinked* into a super-vertex (grouping all customers associated with $i$ and $j$) and the procedure is iterated. The procedure stops when there are no more shrinkable arcs or when the maximum value of $y_{ij} + y_{ji}$ is less than a predetermined threshold (equal to 0.5 for our tests).

The second heuristic algorithm is the *connected component heuristic*. This heuristic considers sequentially the connected components of $\widehat{G}$ once the depot is removed. Let $\mathcal{U}$ be the vertices of one such connected component. If inequality (4) is violated for $\mathcal{U}$, then the inequality is added to LMP. Otherwise, a vertex $v \in \mathcal{U}$ is removed from $\mathcal{U}$ and the procedure is iterated. The procedure stops if there is no vertex $v$ such that $k^C_{\mathcal{U}} = k^C_{\mathcal{U} \setminus \{v\}}$ and the flow entering into the set $\mathcal{U} \setminus \{v\}$ (that is, the left-hand side of the corresponding inequality (4)) is less than that entering into the set $\mathcal{U}$.

Observe that, in the connected component heuristic, no violated inequality can be found by removing from the current subset $\mathcal{U}$ a single vertex $v$ unless there exists at least one route $r$ such that $\mathcal{N}_r \cap \mathcal{U} = \{v\}$ (i.e., $v$ is the only customer of $\mathcal{U}$ visited in $r$) and $\tilde{\theta}_{rw} > 0$ for a certain delivery pattern $w$. Otherwise, the value of the left-hand side of (4) does not diminish when replacing $\mathcal{U}$ by $\mathcal{U} \setminus \{v\}$, whereas $k^C_{\mathcal{U} \setminus \{v\}} \leq k^C_{\mathcal{U}}$. On the other hand, the value of the left-hand side necessarily decreases when removing from $\mathcal{U}$ all the vertices in $\mathcal{N}_r \cap \mathcal{U}$ for a route $r$ such that $\tilde{\theta}_{rw} > 0$ for a delivery pattern $w$.

We propose a variant of the connected component heuristic that exploits this observation. This variant is called the *route-based algorithm*. It also treats one connected component at

a time. Starting with $\mathcal{U}$, the vertex set of a connected component, it considers all positive-valued columns that visit at least one vertex in $\mathcal{U}$. These columns, indexed by $(r, w)$, are sorted in decreasing order of $\tilde{\theta}_{rw} - \sum_{i \in \mathcal{U} \cap \mathcal{N}_r} \frac{d_i}{Q}$. Then, starting from the first column $(r, w)$, remove from the set $\mathcal{U}$ all customers in $\mathcal{N}_r$. If the new set identifies a violated inequality, this inequality is added to LMP. Otherwise, iterate the procedure recursively on the other routes if the number of vertices currently removed from the original set does not exceed $maxRem$, a predefined number that was set to 5 for our tests. Note that the sorting criterion favors the construction of subsets $\mathcal{U}$ such that the number of routes entering $\mathcal{U}$ (the left-hand side of (4)) is minimized and the value of $k^C_{\mathcal{U}}$ (and indirectly the value of $k_{\mathcal{U}}$, the right-hand side of (4)) is kept as high as possible.

Preliminary tests (see Section 5.2) showed that the route-based heuristic is much more efficient for finding violated inequalities than the connected component heuristic. Therefore, we decided to use only the former one, together with the extended shrinking heuristic, and the enumeration heuristic of Desaulniers (2010). Given that the enumeration heuristic can be quite expensive computationally, it is invoked only when the other two algorithms fail to find a violated inequality.

## 4.3 Valid inequalities

In this section, we propose three new classes of valid inequalities for the SDVRPTW. The inequalities in Section 4.3.1 are extensions of known inequalities for the VRPTW, those in Section 4.3.2 are new inequalities, whereas the ones in Section 4.3.3 correspond to a strenghtening of the $k$-path cuts. In all cases, the inequalities are directly defined on the MP variables and, thus, require additional resources (label components) to be treated in the subproblem by the label-setting algorithm described in Section 3 (see Desaulniers et al. 2009). Below, we discuss the treatment of each new inequality class individually.

### 4.3.1 Subset row inequalities

Subset row inequalities were introduced by Jepsen et al. (2008) for the VRPTW and later applied with success by Desaulniers et al. (2008), also for the VRPTW. They correspond to a subset of the Chvátal-Gomory inequalities of rank one that are defined on the set packing constraints associated with each customer that must be visited exactly once in the VRPTW. Using enumeration to find the violated inequalities, Jepsen et al. (2008) considered only the

inequalities involving three rows (customers), which coincide with clique inequalities. In this case, the subset row inequalities can be expressed as follows:

$$\sum_{r \in \mathcal{R}} I_r^{\mathcal{U}} \theta_r \leq 1, \qquad \forall \mathcal{U} \subseteq \mathcal{N}, |\mathcal{U}| = 3 \tag{12}$$

where $I_r^{\mathcal{U}} = 1$ if route $r$ visits at least two customers in subset $\mathcal{U}$ and 0 otherwise.

For the SDVRPTW, these inequalities are no longer valid since each customer can be visited more than once. However, each customer can receive at most one full delivery. Thus, inequalities (12) can be adapted to the SDVRPTW in the following way:

$$\sum_{r \in \mathcal{R}} I_{rw}^{\mathcal{U}} \theta_{rw} \leq 1, \qquad \forall \mathcal{U} \subseteq \mathcal{N}, |\mathcal{U}| = 3 \tag{13}$$

where $I_{rw}^{\mathcal{U}} = 1$ if route $r$ visits at least two customers in $\mathcal{U}$ to perform full deliveries according to pattern $w$, and 0 otherwise.

Let $\sigma_{\mathcal{U}}$ be the nonpositive dual variables associated with inequalities (13). The reduced cost $\bar{c}_{rw}$ of variable $\theta_{rw}$ becomes

$$\bar{c}_{rw} = \sum_{(i,j) \in r} \bar{c}_{ij} - \sum_{i \in \mathcal{N}} \delta_{iw} \pi_i - \sum_{\mathcal{U} \subseteq \mathcal{N} \,:\, |\mathcal{U}| = 3} I_{rw}^{\mathcal{U}} \sigma_{\mathcal{U}}. \tag{14}$$

Therefore, these new dual variables destroy the subproblem structure. In particular, the computation of an optimal extremal delivery pattern for a given route does not correspond anymore to solving an LP-BKP. Nevertheless, the label-setting algorithm proposed by Desaulniers (2010) can be modified to take these dual variables into account. First, a label contains an additional component for each generated inequality (13) such that $\sigma_{\mathcal{U}} \neq 0$. This component computes the number of customers of subset $\mathcal{U}$ that receive a full delivery in the partial path associated with the label. When this component is set at two for the first time along a path, the corresponding dual variable $\sigma_{\mathcal{U}}$ is subtracted from the label reduced cost component. The dominance rule also needs to be modified. The required modifications are similar to those proposed by Jepsen et al. (2008) and detailed in Archetti et al. (2009b).

In the tabu search column generator (see Algorithm 1), the dual variables $\sigma_{\mathcal{U}}$ are simply not taken into account when computing the delivery pattern associated with a route using the *OptimizeQuantities*() function. The reduced cost of the column associated with this route and the computed delivery pattern is, however, computed exactly. Given that the label-setting algorithm is invoked when this heuristic column generator fails to find a negative reduced cost column, this omission does not change the exactitude of the overall algorithm.

### 4.3.2 Strong minimum number of vehicles inequalities

Let $\mathcal{N}^-$ be the set of customers $i \in \mathcal{N}$ such that $d_i \leq Q$, i.e., $k_i^C = 1$. For these customers, constraints (3) can be stated as

$$\sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} a_{ir} \theta_{rw} \geq 1, \qquad \forall\, i \in \mathcal{N}^-.$$

These inequalities can be strengthened thanks to the following observation: if customer $i$ receives a full delivery, then there is no need to visit this customer again. The strong inequalities are

$$\sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} (2a_{irw}^F + a_{irw}^{SZ}) \theta_{rw} \geq 2, \qquad \forall\, i \in \mathcal{N}^-, \tag{15}$$

where the binary parameter $a_{irw}^F = 1$ if a full delivery at customer $i$ is performed in pattern $w$ for route $r$, and the binary parameter $a_{irw}^{SZ} = 1$ if a split or zero delivery at customer $i$ is performed in pattern $w$ for route $r$. These inequalities can be generalized to subsets $\mathcal{U}$ of customers such that $k_{\mathcal{U}}^C = 1$, yielding the inequalities

$$\sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} (2a_{\mathcal{U}rw}^F + a_{\mathcal{U}rw}^{SZ}) \theta_{rw} \geq 2, \qquad \forall\, \mathcal{U} \subseteq \mathcal{N}, k_{\mathcal{U}}^C = 1, \tag{16}$$

where the binary parameter $a_{\mathcal{U}rw}^F = 1$ if all customers in $\mathcal{U}$ are visited in route $r$ and receive full deliveries in pattern $w$, and the binary parameter $a_{\mathcal{U}rw}^{SZ} = 1$ if at least one customer in $\mathcal{U}$ is visited in $r$ and not all customers in $\mathcal{U}$ receive a full delivery in pattern $w$. Note that this set of inequalities includes all inequalities (15), that is, those for which $|\mathcal{U}| = 1$.

We propose to replace constraints (3) by constraints (16). Let $\gamma_{\mathcal{U}}$ be the nonnegative dual variables associated with these constraints. The reduced cost $\bar{c}_{rw}$ of variable $\theta_{rw}$ becomes

$$\bar{c}_{rw} = \sum_{(i,j) \in r} \bar{c}_{ij} - \sum_{i \in \mathcal{N}} \delta_{iw} \pi_i - \sum_{\mathcal{U} \subseteq \mathcal{N} \,:\, k_{\mathcal{U}}^C = 1} (2a_{\mathcal{U}rw}^F + a_{\mathcal{U}rw}^{SZ}) \gamma_{\mathcal{U}}, \tag{17}$$

where the dual variables $\alpha_i$ associated with (3) are removed from the expression (8) of the arc reduced costs $\bar{c}_{ij}$.

Because inequalities (16) with $|\mathcal{U}| = 1$ can be easily handled by the subproblem solution algorithms, they are all added a priori. The others with $|\mathcal{U}| \geq 2$ are generated dynamically as cutting planes. For these inequalities, the considerations made for the subset row cuts are also valid, that is, the label-setting algorithm needs to be modified. In this case, a label contains an additional component for each generated inequality (16) such that $k_{\mathcal{U}}^C = 1$,

$|\mathcal{U}| \geq 2$ and $\gamma_{\mathcal{U}} \neq 0$. This component computes the number of customers in $\mathcal{U}$ receiving a full delivery in the partial path associated with the label. When a partial path visits a first customer in such a subset $\mathcal{U}$ (no matter the delivery type), the corresponding dual variable $\gamma_{\mathcal{U}}$ is subtracted from the label reduced cost component. Furthermore, when the new component is set at $|\mathcal{U}|$ for the first time along a path, the dual variable is subtracted again. Modifications to the dominance rule, involving these new label components, are also necessary to handle these cuts. For details, see Archetti et al. (2009b).

As for the subset row cuts, the *OptimizeQuantities*() function in the tabu search column generator (see Algorithm 1) does not consider the dual variables $\sigma_{\mathcal{U}}$ even for the subsets $\mathcal{U}$ with $|\mathcal{U}| = 1$.

Preliminary tests showed that the strong minimum number of vehicles inequalities for subsets $|\mathcal{U}|$ with $|\mathcal{U}| \geq 4$ are rarely violated. Therefore, we consider only the inequalities (16) for subsets $\mathcal{U}$ with $|\mathcal{U}| \leq 3$. In this case, the separation of these inequalities is done by enumeration.

### 4.3.3 Strong $k$-path inequalities

The $k$-path inequalities (4) can be strengthened as follows:

$$\sum_{r \in \mathcal{R}} \sum_{w \in \mathcal{W}_r} b_{\mathcal{U}r} \theta_{rw} \geq k_{\mathcal{U}}, \qquad \forall \mathcal{U} \in \mathcal{P}(\mathcal{N}) \tag{18}$$

where the binary parameter $b_{\mathcal{U}r} = 1$ if route $r$ enters subset $\mathcal{U}$. It is easy to see that inequalities (18) dominate inequalities (4) since the left-hand side of (4) counts how many times each route enters subset $\mathcal{U}$.

As discussed below, we propose to use the two versions (4) and (18) of the $k$-path inequalities. Let $\zeta_{\mathcal{U}}$ be the nonnegative dual variables associated with constraints (18). The reduced cost $\bar{c}_{rw}$ of variable $\theta_{rw}$ becomes

$$\bar{c}_{rw} = \sum_{(i,j) \in r} \bar{c}_{ij} - \sum_{i \in \mathcal{N}} \delta_{iw} \pi_i - \sum_{\mathcal{U} \in \mathcal{P}(\mathcal{N})} b_{\mathcal{U}r} \zeta_{\mathcal{U}}. \tag{19}$$

Again, handling the new dual variables $\zeta_{\mathcal{U}}$ in the label-setting algorithm requires modifications. A label must contain an additional component for each generated inequality (18) such that $\mathcal{U} \in \mathcal{P}(\mathcal{N})$ and $\zeta_{\mathcal{U}} \neq 0$. This binary component indicates if at least one customer in $\mathcal{U}$ is visited along the partial path associated with the label. When a partial path visits a first customer in this subset $\mathcal{U}$, the corresponding dual variable $\zeta_{\mathcal{U}}$ is subtracted from the label reduced cost component. The dominance rule is also modified (see Archetti et al. 2009b).

As opposed to the two previous types of inequalities, the dual variables of the strong $k$-path cuts do not influence the computation of the optimal extremal delivery pattern for a given route. Nevertheless, the tabu search algorithm is slightly modified to compute the reduced cost of a column appropriately.

The strong $k$-path cuts are separated using the separation algorithms for the weak $k$-path cuts, except for the computation of the flow entering into a subset $\mathcal{U}$. Given the additional complexity of dealing with the dual variables of the strong path cuts, we use both versions (weak and strong) of the cuts in our branch-and-price-and-cut algorithm, giving priority to the weak cuts. When searching for violated $k$-path inequalities, we used the separation algorithms as follows. First, the extended shrinking and the route-based separation algorithms search for violated inequalities of both types. If weak cuts are found, they are added to LMP and all strong cuts found are dropped. Otherwise, if only strong $k$-path cuts are found, they are added to LMP. If no cuts are found by these two heuristics, then we verify if the strong versions of the weak $k$-path cuts present in the current LMP are violated. If so, the weak versions of the violated cuts are replaced by their strong versions. Finally, if no weak cuts are converted into strong cuts, the enumeration separation algorithm of Desaulniers (2010) is invoked. Again, if weak cuts are found, they are the only ones added to LMP. Otherwise, strong cuts, if any, are added to LMP. The rationale behind this procedure is to avoid as much as possible the use of the enumeration algorithm that can be quite expensive computationally for certain instances and to favor weak cuts over strong cuts because the latter ones can considerably slow down the label-setting algorithm.

### 4.3.4 Cutting strategy

In the proposed branch-and-price-and-cut algorithm, four types of cutting planes can be added to LMP during the solution process, namely, the subset row (SR) cuts, the strong minimum number of vehicles (SMV) cuts, the weak $k$-path (WP) cuts, and the strong $k$-path (SP) cuts. As in Desaulniers (2010), these cuts are added only in the first few nodes of the branch-and-bound tree (for our tests, only for nodes whose depth is less than or equal to two). In each of these nodes, we execute the following cutting strategy. Cuts of the same type are added at once. Violated inequalities are sought, at first, in the following order: WP, SP, SR, and SMV. This order favors the identification of the WP cuts that are quite efficient and not computationally costly to handle. Separating them might, however, require a significant computational time when the enumeration heuristic is used. Therefore, when

no WP or SP cuts are found and cuts of the other types are identified in a node, the order is changed by moving the search for WP and SP at the end.

# 5 Computational experiments

The branch-and-price-and-cut method described in the previous section was tested on the same SDVRPTW instances as those used in Gendreau et al. (2006) and Desaulniers (2010). These instances are derived from the well-known benchmark VRPTW instances of Solomon (1987) by simply allowing split deliveries. They are divided into six classes: R1, C1, RC1, R2, C2, and RC2. In the R1 and R2 classes, the customers are located randomly in a $100 \times 100$ square. In the C1 and C2 classes, they are clustered. The RC1 and RC2 instances contain a mix of random and clustered customer locations. In general, the time constraints of the R2, C2, and RC2 instances are less restrictive than those of the R1, C1, and RC1 instances, increasing the number of feasible paths and the difficulty to solve these instances. There are between 8 and 12 instances in each class, for a total of 56. From these 100-customer instances, 25- and 50-customer instances are derived by considering only the first 25 and 50 customers, respectively. Furthermore, for each of these instances, we consider three vehicle capacities: $Q = 30$, 50, and 100. Consequently, the overall number of instances is 504, that is, $56 \times 3$ (sizes) $\times 3$ (capacities).

All tests were performed on a Linux PC equipped with a Pentium D processor clocked at 2.8 GHz. The solution method was implemented using the Gencol library (version 4.5) that is commercialized by Kronos Inc. All restricted master problems were solved using the CPLEX solver, version 10.1.1. This is exactly the same experimental setup as the one used by Desaulniers (2010). As in Desaulniers (2010), a maximum running time of 1 hour per instance was allowed for all our tests.

As the focus of this work is to propose enhancements for the branch-and-price-and-cut algorithm of Desaulniers (2010), we present results obtained by our method and that of Desaulniers (2010), so that we can evaluate the performance of the proposed enhancements. We also report results for evaluating the efficiency of different separation algorithms for the $k$-path inequalities and the effectiveness of the different types of inequalities.

## 5.1 Linear relaxation results

We first conducted experiments to verify the efficiency of our method for solving LMP at the root node of the search tree. In particular, for the two cases in which LMP is solved

with and without the tabu search column generator (Section 4.1), we provide the number of instances for which LMP is solved to optimality and the average computational time to get the optimal solution. Data are reported in Table 1 by group of instances, where a group contains the instances with the same number of customers $n$, the same capacity $Q$, and belonging to the same instance class. In particular, the times reported refer to the solution of LMP at the root node of the branch-and-bound tree, without the introduction of any violated valid inequality.

As can be seen from Table 1, the tabu search algorithm is helpful especially for the R2 and RC2 instances with a high number of customers (50 and 100), which are the hardest instances. In general, the benefits of using this algorithm increase with the value of $Q$ as the length of the feasible paths increases. A detailed analysis of the effect of the introduction of the tabu search algorithm shows that it is very powerful in finding, in a very short amount of time, a large number of negative reduced cost columns in the first pricing iterations, when the dual variables have relatively high values and many reduced cost columns exist. In this phase tabu search yields a great speed up of the solution process as it is much faster than the exact and heuristic label-setting algorithms and it avoids to call them. When the values of the dual variables start to decrease, then tabu search still produces more columns than the heuristic label-setting procedure in a shorter time. Only in the final phase of the pricing process, when the LMP solution is close to optimality and there are only few negative reduced cost columns, tabu search may fail to find such columns and the heuristic label-setting algorithm is called. In general, when the tabu search algorithm is used, the exact pricing algorithm is invoked only once or twice for each solution of LMP. On the other hand, the number of pricing iterations for solving LMP is similar with and without tabu search. This behavior is analogous to the one observed by Desaulniers et al. (2008) for the VRPTW.

## 5.2 Separation algorithm results

We conducted a second series of tests to evaluate the performance of the different $k$-path cut separation algorithms described in Section 4.2. These tests, which considered only the weak $k$-path cuts, were conducted on a subset of twelve instances (four instances for each capacity value $Q = 30$, 50 and 100) that were selected because they had relatively large integrality gaps, they yielded large branch-and-bound trees in Desaulniers (2010), and their optimal values (UB) are known, enabling us to evaluate the gap closed with the introduction of the new separation procedures. The results of these experiments are reported in Table

Table 1: Root node linear relaxation results with and without the tabu search column generator

| | | | $Q = 30$ | | | | $Q = 50$ | | | | $Q = 100$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | w/o tabu | | with tabu | | w/o tabu | | with tabu | | w/o tabu | | with tabu | |
| $n$ | class | Nb inst | Nb solved | time (s) | Nb solved | time (s) | Nb solved | time (s) | Nb solved | time (s) | Nb solved | time (s) | Nb solved | time (s) |
| 25 | R1 | 12 | 12 | <1 | 12 | <1 | 12 | 1 | 12 | <1 | 12 | 1 | 12 | <1 |
| | C1 | 9 | 9 | <1 | 9 | <1 | 9 | 1 | 9 | <1 | 9 | 2 | 9 | 2 |
| | RC1 | 8 | 8 | <1 | 8 | <1 | 8 | <1 | 8 | <1 | 8 | 1 | 8 | <1 |
| 50 | R1 | 12 | 12 | 2 | 12 | 1 | 12 | 7 | 12 | 4 | 12 | 15 | 12 | 8 |
| | C1 | 9 | 9 | 2 | 9 | 1 | 9 | 5 | 9 | 3 | 9 | 21 | 9 | 9 |
| | RC1 | 8 | 8 | 1 | 8 | <1 | 8 | 2 | 8 | 2 | 8 | 10 | 8 | 6 |
| 100 | R1 | 12 | 12 | 31 | 12 | 20 | 12 | 118 | 12 | 78 | 12 | 597 | 12 | 384 |
| | C1 | 9 | 9 | 9 | 9 | 7 | 9 | 33 | 9 | 25 | 9 | 130 | 9 | 70 |
| | RC1 | 8 | 8 | 11 | 8 | 10 | 8 | 48 | 8 | 39 | 8 | 325 | 8 | 215 |
| 25 | R2 | 11 | 11 | 2 | 11 | <1 | 11 | 9 | 11 | 1 | 11 | 75 | 11 | 2 |
| | C2 | 8 | 8 | 2 | 8 | <1 | 8 | 2 | 8 | 1 | 8 | 9 | 8 | 4 |
| | RC2 | 8 | 8 | 1 | 8 | <1 | 8 | 1 | 8 | <1 | 8 | 14 | 8 | 1 |
| 50 | R2 | 11 | 11 | 51 | 11 | 4 | 11 | 107 | 11 | 15 | 11 | 896 | 11 | 64 |
| | C2 | 8 | 8 | 7 | 8 | 2 | 8 | 17 | 8 | 9 | 8 | 161 | 8 | 46 |
| | RC2 | 8 | 8 | 9 | 8 | 1 | 8 | 8 | 8 | 5 | 8 | 245 | 8 | 24 |
| 100 | R2 | 11 | 8 | 1044 | 11 | 54 | 6 | 1045 | 11 | 245 | 1 | 189 | 7 | 869 |
| | C2 | 8 | 8 | 44 | 8 | 17 | 8 | 127 | 8 | 91 | 8 | 812 | 8 | 382 |
| | RC2 | 8 | 8 | 264 | 8 | 36 | 8 | 418 | 8 | 156 | 3 | 1037 | 8 | 1130 |

*Nb solved*: number of instances for which the root node LMP is solved to optimality within 3600 seconds.
*time*: Average total computational time to get the optimal LMP solution over the solved instances.

2. For each instance, we provide the lower bound achieved at the root node of the search tree when using different combinations of the separation algorithms. Column E (only the enumeration algorithm) corresponds to the strategy used by Desaulniers (2010). The last three rows in this table give average results over the twelve instances. In particular, the average gap closed row specifies the average gap closed with respect to the gap yielded by the enumeration algorithm only, that is, $(lb_x - lb_E)/(UB - lb_E)$ where $lb_x$ is the lower bound for combination $x$. We also report the average number of cuts generated and the average computational time for computing the lower bound, including the cut separation time. Bold values indicate that the lower bound is equal to the upper bound.

From these results we make the following observations. When combined individually with the enumeration algorithm (columns 6 to 8), the best of the three new separation algorithms is the route-based algorithm that succeeds to close 48% of the gap on average. When combining two new separation algorithms with the enumeration algorithm (columns 9 to 11), the best combination is yielded by the extended shrinking and the route-based algorithms. In this case, the average gap closed increases to 50%. This is not a large improvement, but it is not computationally costly since the extended shrinking heuristic is extremely fast. Adding the connected component algorithm to the route-based algorithm achieves the same average gap closed as without the connected component heuristic (the lower bound slightly improves for only one instance). Therefore, we decided to leave out the connected component heuristic from the overall branch-and-price-and-cut algorithm. The last column, which considers the case with the extended shrinking and the route-based algorithms but without the enumeration algorithm, clearly shows the importance of using the enumeration algorithm. This algorithm is complementary to the other two as it considers all subsets of customers with a small cardinality. Finally, notice that the average computational time does not vary much with the algorithm combination used. Nevertheless, we remark a small increase when the recursive route-based algorithm is used.

## 5.3 Valid inequality results

We then conducted a third series of experiments to check how the new inequalities presented in Section 4.3 helped in improving the value of the lower bound. These experiments were performed on the subset of instances used in the previous section. The results are reported in Table 3, which gives the lower bound obtained for each instance and for the various inequalities. Column WP (weak $k$-path cuts with the enumeration separation algorithm

Table 2: Lower bounds for the various weak $k$-path cut separation algorithms

| inst | $n$ | $Q$ | UB | E | E +SH | E +CC | E +RB | E+SH +CC | E+SH +RB | E+CC +RB | SH +RB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C105 | 25 | 30 | 821.1 | 803.2 | 818.3 | 806.7 | **821.1** | 820.7 | **821.1** | **821.1** | **821.1** |
| C206 | 25 | 30 | 909.9 | 907.3 | 907.6 | 907.3 | 909.8 | 907.8 | 909.8 | 909.8 | 909.8 |
| R111 | 25 | 30 | 747.5 | 739.2 | 739.2 | 739.2 | 739.2 | 739.2 | 739.2 | 739.2 | 738.3 |
| R202 | 25 | 30 | 744.9 | 739.1 | 739.0 | 739.1 | 739.1 | 739.0 | 739.1 | 739.1 | 738.3 |
| C103 | 50 | 50 | 1012.3 | 1008.5 | 1008.9 | 1008.6 | 1010.8 | 1009.3 | 1011.2 | 1011.0 | 1010.6 |
| C205 | 50 | 50 | 1157.1 | 1154.8 | 1154.9 | 1155.1 | 1155.5 | 1154.9 | 1155.5 | 1155.5 | 1153.5 |
| R102 | 50 | 50 | 1114.2 | 1102.9 | 1103.4 | 1104.3 | 1106.0 | 1106.1 | 1106.1 | 1106.0 | 1105.4 |
| RC102 | 25 | 50 | 933.5 | 909.3 | **933.5** | 923.5 | **933.5** | **933.5** | **933.5** | **933.5** | **933.5** |
| C103 | 25 | 100 | 291.5 | 287.8 | 291.1 | 291.1 | 291.2 | 291.1 | 291.2 | 291.2 | 291.2 |
| R109 | 50 | 100 | 804.2 | 789.3 | 789.3 | 789.3 | 789.3 | 789.3 | 789.3 | 789.3 | 787.7 |
| R205 | 50 | 100 | 758.9 | 753.4 | 753.4 | 754.2 | 754.9 | 754.3 | 754.9 | 754.9 | 754.8 |
| R206 | 25 | 100 | 404.1 | 393.5 | 397.5 | 395.0 | 397.6 | 399.1 | 399.1 | 397.6 | 398.9 |
| Averages over all instances | | | | | | | | | | | |
| Gap closed (%) | | | | | 25 | 19 | 48 | 36 | 50 | 48 | 37 |
| Number cuts | | | | 33 | 34 | 33 | 43 | 35 | 48 | 45 | 39 |
| Total time (s) | | | | 12 | 13 | 13 | 15 | 13 | 16 | 16 | 12 |

*UB*: optimal value; *E*: enumeration algorithm; *SH*: extended shrinking algorithm;
*CC*: connected component algorithm; *RB*: route-based algorithm;

only) corresponds to the setup of Desaulniers (2010). The average gap closed for each case is reported per group of instances and for all the instances, where a group contains all the instances with the same vehicle capacity. Furthermore, we also report the average number of cuts generated (weak $k$-path cuts plus others) and the average computational time. Again, bold values indicate that the lower bound is equal to the upper bound.

These results show that the use of the new separation algorithms for the $k$-path cuts (WP+NSA) has the most significant impact with an average gap closed of 50%. Among the newly proposed valid inequalities, the SMV inequalities are the most useful with an average gap closed of 28%. Observe that all inequalities are more effective for the instances with $Q = 100$. For the SP cuts, this can be explained by the fact that they become useful (compared to the WP cuts) when the routes are relatively long, increasing the possibility of entering into a subset $\mathcal{U}$ more than once. A similar reason (long routes) applies for the SR cuts, which involve routes visiting at least two customers of three-customer subsets $\mathcal{U}$. For the SMV inequalities, the increased effectiveness for $Q = 100$ can be due to an increase number of SMV inequalities, that is, there are more subsets $\mathcal{U}$ such that $k^C_{\mathcal{U}} = 1$. Even though some of these inequalities do not seem very useful, using them all in combination with the new separation algorithms has a significant impact on the average gap closed for all groups of instances (see column ALL). In fact, additional computational experiments (whose results are not reported in this paper) showed that removing any individual type of inequality reduces the number of instances that can be solved to optimality within the 1-hour time limit. Therefore, we have decided to keep all these inequalities in the proposed branch-and-price-and-cut algorithm.

## 5.4 Integer solution results

Finally, we conducted a final set of experiments that are aimed at solving to optimality all SDVRPTW instances. Table 4 presents a summary of the results. For each instance group (in this section, a group is identified by its number of customers $n$, class, and capacity $Q$) for which at least one instance could be solved to proven optimality within one hour of computational time, it provides the number of instances in this group (*Nb inst*), the number of instances solved (*Nb solved*) and the following averages over the solved instances: the numbers of vehicles (*veh*) and split customers (*splits*) in the computed optimal solution; the gaps in percentage between the optimal IP and LP values (*gap* LP) and between the optimal IP value and the optimal LP value after adding cuts at the root node (*gap* LP + *cuts*); the

Table 3: Lower bounds for the various inequalities

| inst | $n$ | $Q$ | UB | WP | WP +SP | WP +SMV | WP +SR | WP +NSA | ALL |
|---|---|---|---|---|---|---|---|---|---|
| C105 | 25 | 30 | 821.1 | 803.2 | 803.5 | 803.2 | 803.4 | **821.1** | **821.1** |
| C206 | 25 | 30 | 909.9 | 907.3 | 907.4 | 907.4 | 907.3 | 909.8 | **909.9** |
| R111 | 25 | 30 | 747.5 | 739.2 | 739.7 | 741.5 | 739.2 | 739.2 | 741.5 |
| R202 | 25 | 30 | 744.9 | 739.1 | 739.0 | 741.1 | 739.1 | 739.1 | 741.1 |
| Avg. gap closed (%) | | | | | 2 | 17 | 0 | 49 | 66 |
| C103 | 50 | 50 | 1012.3 | 1008.5 | 1008.8 | 1008.7 | 1009.0 | 1011.2 | **1012.3** |
| C205 | 50 | 50 | 1157.1 | 1154.8 | 1154.9 | 1155.2 | 1155.3 | 1155.5 | 1156.6 |
| R102 | 50 | 50 | 1114.2 | 1102.9 | 1102.9 | 1104.0 | 1102.9 | 1106.1 | 1108.7 |
| RC102 | 25 | 50 | 933.5 | 909.3 | 909.9 | 909.5 | 909.3 | **933.5** | **933.5** |
| Avg. gap closed (%) | | | | | 4 | 8 | 9 | 57 | 82 |
| C103 | 25 | 100 | 291.5 | 287.8 | 288.7 | 288.8 | 288.1 | 291.2 | **291.5** |
| R109 | 50 | 100 | 804.2 | 789.3 | 790.9 | 799.0 | 792.1 | 789.3 | 800.8 |
| R205 | 50 | 100 | 758.9 | 753.4 | 755.9 | **758.9** | 755.1 | 754.9 | **758.9** |
| R206 | 25 | 100 | 404.1 | 393.5 | 394.1 | 398.5 | 394.8 | 399.1 | **404.1** |
| Avg. gap closed (%) | | | | | 22 | 59 | 20 | 43 | 94 |
| Averages over all instances | | | | | | | | | |
| Gap closed (%) | | | | | 9 | 28 | 10 | 50 | 81 |
| Number cuts | | | | 33 | 27+17 | 33+30 | 33+27 | 48 | 48+26 |
| Total time (s) | | | | 12 | 23 | 43 | 25 | 16 | 38 |

*UB*: optimal value; *WP*: weak path cuts (enumeration separation algorithm only);
*SP*: strong path cuts; *SMV*: strong constraints on minimum number of vehicles;
*SR*: subset row cuts; *NSA*: new separation algorithms; *ALL*: all cuts.

total numbers of cuts generated and branch-and-bound nodes explored (including the root node); and the times in seconds for solving the first linear relaxation (before adding cuts), for generating cuts, and for solving the whole problem (*total*). Note that the results differ from the ones reported in Table 1 since here we are solving SDVRPTW to optimality and not solely LMP. Thus, not all instances reported in Table 1 are reported here because some of them could not be solved within the one-hour time limit. Also, since the computational times given in Table 1 are averages over the instances solved (which differ in the two tables), they can be different from the LP computational times reported in Table 4.

As a comparison with the results obtained by Desaulniers (2010), we now solve to optimality 262 of the 504 instances against the 176 solved by Desaulniers (2010). In particular, we can solve all 25-customer instances while he could solve 135 of these 168 instances. Moreover, we can solve 8 instances with 100 customers, while he solved only one. The success of our method is certainly due to the proposed valid inequalities that considerably help reducing the integrality gaps. Indeed, one can remark that the average integrality gap after adding cuts is for most groups of solved instances smaller than 0.1% and often equal to 0. The optimal values of the 262 instances solved to optimality can be found on the web page *www.gerad.ca/∼guyd/sdvrptw.html*.

# 6 Conclusions

In this paper, we proposed new enhancement procedures to solve the SDVRPTW through a branch-and-price-and-cut algorithm. In particular, we developed a tabu search algorithm for speeding up the solution of the subproblem. Furthermore, in order to improve the value of the lower bounds computed in the search tree, we introduced extensions of several classes of valid inequalities together with a new heuristic separation algorithm for the $k$-path inequalities.

The computational results demonstrate that our procedures are very effective in improving the performance of the branch-and-price-and-cut algorithm of Desaulniers (2010): within a one-hour time limit, we were able to solve to optimality 86 additional instances, including 7 instances with 100 customers.

# References

Archetti, C., N. Bianchessi, M.G. Speranza. 2009a. A Column Generation Approach for the Split Delivery Vehicle Routing Problem. Forthcoming in *Networks*.

Table 4: Summary of the integer solution results

| | | | Nb | numbers | | | gap (%) | | numbers | | times (s) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | class | $Q$ | inst | solved | veh | splits | LP | LP+cuts | cuts | nodes | LP | cuts | total |
| 25 | R1 | 30 | 12 | 12 | 12.0 | 3.8 | 1.9 | 0.2 | 50.0 | 77.9 | <1 | 4 | 48 |
| 25 | R1 | 50 | 12 | 12 | 7.2 | 1.1 | 1.2 | <0.1 | 22.7 | 1.2 | <1 | <1 | 5 |
| 25 | R1 | 100 | 12 | 12 | 4.9 | 0.1 | 0.4 | 0 | 10.7 | 1.0 | <1 | <1 | 2 |
| 25 | C1 | 30 | 9 | 9 | 16.0 | 5.2 | 2.8 | 0 | 96.2 | 6.1 | <1 | 2 | 9 |
| 25 | C1 | 50 | 9 | 9 | 10.0 | 2.1 | 1.7 | 0 | 27.4 | 1.0 | <1 | <1 | 5 |
| 25 | C1 | 100 | 9 | 9 | 5.0 | 0 | 1.9 | <0.1 | 26.3 | 1.2 | 2 | <1 | 15 |
| 25 | RC1 | 30 | 8 | 8 | 17.8 | 7.1 | 2.4 | <0.1 | 81.2 | 4.5 | <1 | 2 | 5 |
| 25 | RC1 | 50 | 8 | 8 | 10.6 | 1.8 | 6.9 | 0 | 86.8 | 1.0 | <1 | <1 | 5 |
| 25 | RC1 | 100 | 8 | 8 | 6.0 | 0.4 | 0.8 | 0 | 12.6 | 1.0 | <1 | 0 | 2 |
| 25 | R2 | 30 | 11 | 11 | 12.0 | 3.9 | 2.3 | 0.2 | 61.5 | 218.2 | <1 | 5 | 165 |
| 25 | R2 | 50 | 11 | 11 | 7.0 | 1.1 | 1.3 | <0.1 | 26.5 | 1.2 | 2 | 2 | 15 |
| 25 | R2 | 100 | 11 | 11 | 3.8 | 0.1 | 1.6 | 0 | 38.9 | 1.0 | 3 | 3 | 24 |
| 25 | C2 | 30 | 8 | 8 | 16.0 | 6.4 | 1.4 | 0 | 57.8 | 2.9 | <1 | <1 | 9 |
| 25 | C2 | 50 | 8 | 8 | 10.0 | 2.5 | 1.3 | <0.1 | 25.0 | 5.2 | 1 | <1 | 11 |
| 25 | C2 | 100 | 8 | 8 | 5.0 | 1.0 | 0.7 | <0.1 | 12.0 | 1.2 | 4 | <1 | 19 |
| 25 | RC2 | 30 | 8 | 8 | 18.0 | 6.6 | 2.4 | 0 | 79.5 | 3.8 | <1 | 2 | 9 |
| 25 | RC2 | 50 | 8 | 8 | 10.8 | 1.8 | 6.9 | <0.1 | 91.4 | 1.2 | <1 | 1 | 12 |
| 25 | RC2 | 100 | 8 | 8 | 6.0 | 0.4 | 0.8 | 0 | 12.5 | 1.0 | 1 | 0 | 4 |
| 50 | R1 | 50 | 12 | 2 | 15.0 | 4.0 | 1.9 | 0.3 | 84.0 | 314.0 | 1 | 16 | 533 |
| 50 | R1 | 100 | 12 | 6 | 9.7 | 0.8 | 1.1 | 0.2 | 129.0 | 7.0 | 3 | 37 | 553 |
| 50 | C1 | 30 | 9 | 3 | 29.0 | 10.7 | 1.7 | <0.1 | 214.0 | 56.7 | 3 | 25 | 219 |
| 50 | C1 | 50 | 9 | 9 | 18.0 | 4.3 | 1.5 | <0.1 | 113.6 | 10.8 | 5 | 12 | 114 |
| 50 | C1 | 100 | 9 | 8 | 8.8 | 1.0 | 4.7 | <0.1 | 173.6 | 2.2 | 5 | 43 | 353 |
| 50 | RC1 | 30 | 8 | 8 | 33.0 | 8.9 | 1.6 | 0 | 287.9 | 1.0 | <1 | 9 | 50 |
| 50 | RC1 | 50 | 8 | 8 | 20.0 | 4.4 | 0.5 | 0 | 29.0 | 1.0 | 2 | <1 | 11 |
| 50 | RC1 | 100 | 8 | 8 | 10.0 | 1.0 | 0.8 | 0 | 28.6 | 1.0 | 7 | <1 | 22 |
| 50 | R2 | 100 | 11 | 1 | 8.0 | 1.0 | 0.7 | 0 | 54.0 | 1.0 | 23 | 44 | 134 |
| 50 | C2 | 50 | 8 | 7 | 18.0 | 7.1 | 1.2 | <0.1 | 120.0 | 35.6 | 9 | 22 | 395 |
| 50 | C2 | 100 | 8 | 2 | 9.0 | 3.0 | 1.4 | 0.2 | 165.0 | 7.0 | 19 | 66 | 1314 |
| 50 | RC2 | 30 | 8 | 8 | 33.0 | 9.2 | 1.6 | 0 | 333.5 | 1.0 | 2 | 8 | 161 |
| 50 | RC2 | 50 | 8 | 8 | 20.0 | 4.8 | 0.5 | 0 | 30.9 | 1.0 | 5 | <1 | 30 |
| 50 | RC2 | 100 | 8 | 8 | 10.0 | 0.9 | 0.9 | 0 | 27.8 | 1.0 | 29 | <1 | 94 |
| 100 | R1 | 100 | 12 | 1 | 20.0 | 0 | 0.1 | 0 | 16.0 | 1.0 | 2 | <1 | 5 |
| 100 | C1 | 100 | 9 | 5 | 19.0 | 2.4 | 3.6 | <0.1 | 319.8 | 1.8 | 31 | 99 | 1667 |
| 100 | C2 | 100 | 8 | 2 | 19.0 | 5.5 | 1.7 | 0 | 139.0 | 1.0 | 152 | 14 | 1407 |

Archetti, C., M. Bouchard, G. Desaulniers. 2009b. Enhanced Branch-and-Price-and-Cut for Vehicle Routing with Split Deliveries and Time Windows. Technical Report G-2009-81, Les Cahiers du GERAD, HEC Montréal, Montréal, Canada.

Archetti, C., A. Hertz, M.G. Speranza. 2006a. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science* **40** 64–73.

Archetti, C., M.W.P. Savelsbergh, M.G. Speranza. 2006b. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science* **40**(2) 226–234.

Archetti, C., M.W.P. Savelsbergh, M.G. Speranza. 2008a. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science* **42**(1) 22–31.

Archetti, C., M.W.P. Savelsbergh, M.G. Speranza. 2008b. To split or not to split: That is the question. *Transportation Research Part E* **44** 114–123.

Archetti, C., M.G. Speranza. 2008. The split delivery vehicle routing problem: a survey. B. Golden, R. Raghavan, E. Wasil, eds., *Vehicle Routing Problem: Latest Advances and New Challenges*, chap. 5. Springer, 103–122.

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

Chen, S., B. Golden, E. Wasil. 2007. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks* **49**(4) 318–329.

Dantzig, G.B., P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8** 101–111.

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

Desaulniers, G., J. Desrosiers, S. Spoorendonk. 2009. *Cutting planes for branch-and-price algorithms*. Technical report, Les Cahiers du Gerad G-2009-52, HEC Montreal, Canada.

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

Desrosiers, J., M.E. Lübbecke. 2005. A primer in column generation. G. Desaulniers, J. Desrosiers, M.M. Solomon, eds., *Column Generation*, chap. 1, Springer, New-York, NY, 1–32.

Dror, M., G. Laporte, P. Trudeau. 1994. Vehicle routing with split deliveries. *Discrete Applied Mathematics* **50** 239–254.

Dror, M., P. Trudeau. 1989. Savings by split delivery routing. *Transportation Science* **23** 141–145.

Dror, M., P. Trudeau. 1990. Split delivery routing. *Naval Research Logistics* **37** 383–402.

Frizzell, P.W., J.W. Giffin. 1992. The bounded split delivery vehicle routing problem with grid network distances. *Asia-Pacific Journal of Operational Research* **9**(1) 101–116.

Frizzell, P.W., J.W. Giffin. 1995. The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers and Operations Research* **22**(6) 655–667.

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

Ho, S.C., D. Haugland. 2004. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers and Operations Research* **31** 1947–1964.

Irnich, S., G. Desaulniers. 2005. Shortest path problems with resource constraints. G. Desaulniers, J. Desrosiers, M.M. Solomon, eds., *Column Generation*, chap. 2, Springer, New-York, NY, 33–65.

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

Jin, M., K. Liu, R.O. Bowden. 2007. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Journal of Production Economics* **105** 228–242.

Jin, M., K. Liu, B. Eksioglu. 2008. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters* **36** 265–270.

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

Lee, C.G., M.A. Epelman, C.C. White III, Y.A. Bozer. 2006. A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research Part B* **40** 265–284.

Mullaseril, P.A., M. Dror, P. Trudeau. 1997. Split-delivery routing heuristics in livestock feed distribution. *Journal of the Operational Research Society* **48** 107–116.

Ralphs, T.K., L. Kopman, W.R. Pulleyblank, L.E. Trotter. 2003. On the capacitated vehicle routing problem. *Mathematical Programming* **94** 343–359.

Sierksma, G., G.A. Tijssen. 1998. Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research* **76** 261–286.

Solomon, M.M. 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research* **35**(2) 254–265.

Toth, P., D. Vigo, eds. 2002. *The vehicle routing problem.* SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA.