Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Branch-Price-and-Cut Algorithm for the Inventory-Routing Problem

Guy Desaulniers, Jørgen G. Rakke, Leandro C. Coelho

To cite this article:
Guy Desaulniers, Jørgen G. Rakke, Leandro C. Coelho (2016) A Branch-Price-and-Cut Algorithm for the Inventory-Routing Problem. Transportation Science 50(3):1060-1076. https://doi.org/10.1287/trsc.2015.0635

TRANSPORTATION SCIENCE
Vol. 50, No. 3, August 2016, pp. 1060–1076
ISSN 0041-1655 (print) | ISSN 1526-5447 (online)

http://dx.doi.org/10.1287/trsc.2015.0635

A Branch-Price-and-Cut Algorithm for the Inventory-Routing Problem

Guy Desaulniers
Department of Mathematics and Industrial Engineering, École Polytechnique and GERAD, Montréal, Québec H3C 2A7, Canada, guy.desaulniers@gerad.ca

Jørgen G. Rakke
Department of Marine Technology, Norwegian University of Science and Technology, 7491 Trondheim, Norway, jorgen.rakke@ntnu.no

Leandro C. Coelho
Faculté des sciences de l'administration, Université Laval and CIRRELT, Québec, Québec G1V 0A6, Canada, leandro.coelho@cirrelt.ca

The inventory-routing problem (IRP) integrates two well-studied problems, namely, inventory management and vehicle routing. Given a set of customers to service over a multiperiod horizon, the IRP consists of determining when to visit each customer, which quantity to deliver in each visit, and how to combine the visits in each period into feasible routes such that the total routing and inventory costs are minimized. In this paper, we propose an innovative mathematical formulation for the IRP and develop a state-of-the-art branch-price-and-cut algorithm for solving it. This algorithm incorporates known and new families of valid inequalities, including an adaptation of the well-known capacity inequalities, as well as an ad hoc labeling algorithm for solving the column generation subproblems. Through extensive computational experiments on a widely used set of 640 benchmark instances involving between two and five vehicles, we show that our branch-price-and-cut algorithm clearly outperforms a state-of-the-art branch-and-cut algorithm on the instances with four and five vehicles. In this instance set, 238 were still open before this work and we proved optimality for 54 of them.

Keywords: inventory routing; branch price and cut; capacity inequalities; labeling algorithm; exact algorithm
History: Received: April 2014; revision received: February 2015; accepted: May 2015. Published online in *Articles in Advance* October 26, 2015.

1. Introduction

Vendor-managed inventory (VMI) systems are examples of successful business practices that were made possible by new technologies. The strategy behind these systems is based on the cooperation between a supplier and its customers in which demand and inventory information from the customers are shared with the supplier. Under this paradigm, the supplier is then responsible for controlling the inventory level of the customers, deciding when and how much to deliver to each customer. This is a win-win situation: customers employ less resources to control their inventory and to place replenishment orders, while the supplier can better integrate the visits to several customers and thus smooth its production, inventory, and distribution efforts (Adulyasak, Cordeau, and Jans 2015; Andersson et al. 2010).

To operate such an integrated strategy, the supplier has to solve an inventory-routing problem (IRP), which combines two well-known problems over a multiperiod horizon: inventory management and vehicle routing. In the IRP, the supplier has to make three types of decisions simultaneously: in which period(s) each customer must be visited, how much to deliver in each visit, and how to combine customer visits into feasible routes in each period. The aim is to find the optimal trade-off between vehicle routing and inventory holding costs, such that the total distribution and inventory costs are minimized. For a detailed account of applications and industrial aspects of the IRP, see Andersson et al. (2010).

The IRP has been studied considering different replenishment policies that impose rules on the quantity that can be delivered in each delivery to a customer. The two most popular are the order-up-to-level (OU) and the maximum-level (ML) policies. In the OU policy each delivery must fill the inventory to its maximum capacity, effectively linking two of the decisions: once one decides to visit a customer, the quantity to be delivered is simply the difference between its maximum capacity and its current inventory level. This policy has been proposed by Dror, Ball, and Golden (1985) as a way to simplify the search for good solutions and has since been widely studied (e.g., Bertazzi, Paletta, and Speranza 2002; Archetti et al. 2007; Solyalı and Süral 2011, and Coelho, Cordeau, and Laporte 2012b).

In the ML policy, any quantity can be delivered as long as the maximum capacity is not exceeded. The ML policy clearly encompasses the OU one and is more flexible, but also more difficult to solve given the extra set of decision variables. For this reason, in this paper we focus the algorithmic effort on the ML policy.

Different methods have been proposed for the solution of single-vehicle and multivehicle IRPs under various replenishment policies. For the single-vehicle version of the problem, heuristic algorithms include the fast local search of Bertazzi, Paletta, and Speranza (2002), the hybrid of mathematical programming and tabu search of Archetti et al. (2012), and the adaptive large neighborhood search (ALNS) of Coelho, Cordeau, and Laporte (2012a). The first exact algorithm for this version of the problem is the branch and cut developed by Archetti et al. (2007). Algorithms capable of solving multivehicle instances are more recent, but the literature is quickly expanding. These include the ALNS heuristic of Coelho, Cordeau, and Laporte (2012b), and the exact branch-and-cut algorithms of Adulyasak, Cordeau, and Jans (2014) and Coelho and Laporte (2013a, 2014). All of these papers provide an algorithm to solve the problem under the same assumptions, and are tested on the same set of benchmark instances (also used in this paper), thus allowing for a clear comparison of the performance of each algorithm. For a recent review on the algorithmic aspects of the IRP, see Coelho, Cordeau, and Laporte (2014).

Some other papers working on different assumptions deserve to be mentioned here because they also use algorithms based on branch and price. Engineer et al. (2012) study a different problem arising in maritime transportation in which the storage capacity, production, and consumption rates change over time at each location. The resulting column generation subproblem is solved by dynamic programming. Hewitt et al. (2013) solve a series of constrained mixed-integer linear programs to obtain heuristic solutions for another maritime application. The choice of the restrictions applied to the original problem is done through a pricing procedure. Grønhaug et al. (2010) solve a gas distribution problem in the maritime sector through branch and price, in which the master problem handles the inventory management and the customer capacity constraints, and the subproblems generate the ship routes. Bard and Nananukul (2010) use branch and price to solve the production-inventory-routing problem, an extended variant of the IRP in which one also optimizes production variables. In their solution approach, the column generation subproblems generate delivery schedules where a delivery schedule includes a set of vehicle routes for a given period and the quantity delivered in each visit. These subproblems are formulated as mixed-integer programs and are solved by a mixed-integer programming solver. This exact branch-and-price algorithm is able to solve small-sized instances involving up to 10 customers and two periods in less than 30 minutes of computational time. To solve larger instances, the authors modified their algorithm to yield branch-and-price heuristics.

In this paper we introduce a state-of-the-art branch-price-and-cut algorithm for the IRP, that is, a column generation algorithm embedded within a branch-and-cut framework. We first propose an innovative mathematical formulation for the problem, and tighten it with the inclusion of known and new families of valid inequalities. In particular, we show how the well-known capacity inequalities (Laporte, Nobert, and Desrochers 1985) can be applied to the IRP. In the column generation algorithm, there is one subproblem per period that allows to generate individual vehicle routes together with the quantity delivered to each visited customer. Each subproblem corresponds to an elementary shortest path problem with resource constraints combined with the linear relaxation of a knapsack problem. To solve it, we develop an ad-hoc labeling algorithm derived from that for the split-delivery vehicle routing problem (Desaulniers 2010). We also incorporate several acceleration techniques to enhance the performance of the overall algorithm, which is assessed on a set of benchmark instances involving between two and five vehicles. The computational results indicate that the proposed branch-price-and-cut algorithm clearly outperforms a state-of-the-art branch-and-cut algorithm on the largest instances containing four and five vehicles.

The remainder of this paper is organized as follows. In §2 we provide a formal definition and a mathematical model for the problem under study. In §3 we describe the proposed branch-price-and-cut algorithm. The results of extensive computational experiments are detailed in §4, followed by conclusions in §5.

2. Problem Definition and Mathematical Model

In the IRP, a single supplier, denoted 0, produces a known quantity d_0^p of a single commodity at each period p of a finite planning horizon $P = \{1, 2, \ldots, \rho\}$. To handle end inventories, a fictitious period $\rho + 1$ is also considered. Using a homogeneous fleet of K vehicles with capacity Q, the supplier serves a set N of customers where each customer $i \in N$ consumes a known quantity d_i^p in period $p \in P$ (these quantities are referred to as the customer demands). Each customer $i \in N$ (the supplier 0) has an inventory capacity C_i (C_0), an initial inventory $I_i^0 \le C_i$ ($I_0^0 \le C_0$), and a unit holding cost h_i (h_0). The IRP consists of building feasible delivery vehicle routes in each period such that no stockouts occur, while respecting inventory capacities. A route is deemed feasible if it satisfies vehicle capacity. Furthermore, each customer can be serviced at most

once per period. The objective of the IRP is to minimize the sum of the vehicle traveling costs and the inventory holding costs at the supplier and at the customers. These holding costs are charged on the inventory at the end of each period. In this paper, we consider an ML inventory replenishment policy. We assume the following sequences of operations at each period: the supplier performs production before making any deliveries, and customers receive their deliveries at the beginning of the period and can use them to fulfill their demand in that period.

We model the IRP as a mixed-integer program that exploits some key features of two existing formulations. First, our model involves continuous variables associated with a route and a route delivery pattern (RDP) as in the model of Desaulniers (2010) for the split-delivery vehicle routing problem with time windows. Spanning a single period, a route starts at the supplier and visits a sequence of customers before returning to the depot. An RDP specifies the quantity delivered to each customer along the corresponding route. As in Desaulniers (2010), we consider only extreme RDPs (defined below) and use their convex combinations to generate any other RDP. Second, as in the facility location-based formulation of the lot sizing problem introduced by Krarup and Bilde (1977) (see also Brahimi et al. 2006), our model indicates the detailed usage of each delivery, that is, the demands that it will cover (fully or partially) or the quantity that will remain in the end inventory. Consequently, each delivery can be seen as a set of subdeliveries, one for each demand it can cover or for the end inventory. To limit the number of potential subdeliveries to a customer, we exploit the fact that there always exists an optimal solution to the IRP in which the delivered quantities are consumed following a first-in, first-out (FIFO) rule. Given this rule, the initial inventory at each customer $i \in N$ can be used to fulfill its demands of the first periods, yielding residual demands. Let $I_i^{0,s} = \max\{0, I_i^0 - \sum_{\ell=1}^{s} d_i^\ell\}$ be the quantity remaining from the initial inventory at customer $i \in N$ at the end of period $s \in P$. Then, the residual demands at customer i correspond to

$$\bar{d}_i^s = \begin{cases} \max\{0, d_i^1 - I_i^0\} & \text{if } s = 1 \\ \max\{0, d_i^s - I_i^{0,s-1}\} & \text{otherwise,} \end{cases} \qquad \forall s \in P.$$

Moreover, given a period $p \in P$, a customer $i \in N$, its holding capacity C_i, and its demands d_i^s and residual demands $\bar{d}_i^s$, $s \in P$, one can determine, under the FIFO rule, the set of periods $P_{ip}^+ = \{s \in \{p, p+1, \ldots, \rho+1\} \mid (s \in P, \bar{d}_i^s > 0 \text{ and } (s = p \text{ or } \sum_{\ell=p}^{s-1} d_i^\ell < C_i)) \text{ or } (s = \rho+1 \text{ and } \sum_{\ell=p}^{s-1} d_i^\ell < C_i)\}$ associated with the subdeliveries of a delivery to customer i in period p. For reasons of conciseness, we assume that $P_{ip}^+ \neq \varnothing$. An upper bound u_{ip}^s on the quantity dedicated to each period $s \in P_{ip}^+$ can also be computed as

$$u_{ip}^s = \begin{cases} \min\{\bar{d}_i^s, C_i - I_i^{0,s-1}\} & \text{if } s = p \\ C_i - \sum_{\ell=p}^{s-1} d_i^\ell - I_i^{0,s-1} & \text{if } s = \rho+1 \\ \min\{\bar{d}_i^s, C_i - \sum_{\ell=p}^{s-1} d_i^\ell - I_i^{0,s-1}\} & \text{otherwise.} \end{cases}$$

Let R be the set of feasible routes, N_r the set of customers visited in route $r \in R$, and A_r the set of pairs of locations i and j (in $N \cup \{0\}$) visited consecutively along route r. For each route r, we define binary parameters a_{ri}, $i \in N$, indicating whether route r visits customer i ($a_{ri} = 1$) or not ($a_{ri} = 0$) and associate a set of extreme RDPs W_r^p when r is used in period $p \in P$. An RDP $w \in W_r^p$ specifies the quantity $q_{wi}^s \in [0, u_{ip}^s]$ delivered to each customer $i \in N_r$ and dedicated to each period $s \in P_{ip}^+$. We say that q_{wi}^s corresponds to a *zero subdelivery* if $q_{wi}^s = 0$, a *full subdelivery* if $q_{wi}^s = u_{wi}^s$, and a *partial subdelivery* otherwise. RDP w is said to be an *extreme RDP* if it contains at most one partial subdelivery. Let $q_w = \sum_{i \in N_r} \sum_{s \in P_{ip}^+} q_{wi}^s$ be the total quantity delivered in RDP w and, therefore, loaded at the supplier.

Given a route $r \in R$ and an extreme RDP $w \in W_r^p$, we can identify the quantity b_{wi}^s delivered to customer $i \in N_r$ that will be in inventory at the end of period $s \in P_{ip}^+$. Let $c_{rw} = \sum_{(i,j) \in A_r} c_{ij} + \sum_{i \in N_r} \sum_{s \in P_{ip}^+} h_i b_{wi}^s$ be the cost associated with route r and RDP w. In this expression, the first term corresponds to the traveling costs along route r, where c_{ij} is the traveling cost between locations i and j. The second term gives the total holding costs at the visited customers incurred by the deliveries in RDP w. Finally, denote by $P_{is}^- = \{p \in P \mid s \in P_{ip}^+\}$ the set of periods at which a subdelivery can be made to fulfill the demand of customer $i \in N$ at period $s \in P$.

The proposed mathematical model involves two types of variables: the continuous variable y_{rw}^p with value in $[0, 1]$ provides the proportion of the route $r \in R$ operated with RDP $w \in W_r^p$ in period $p \in P$, and the nonnegative variable I_0^p indicates the inventory at the supplier at the end of period $p \in P$.

Using this notation, we can formulate the IRP as the following mixed-integer program:

$$\min \left\{ \sum_{p \in P} \sum_{r \in R} \sum_{w \in W_r^p} c_{rw} y_{rw}^p + \sum_{p \in P} h_0 I_0^p \right\} \tag{1}$$

$$\text{s.t. } I_0^{p-1} + d_0^p - \sum_{r \in R} \sum_{w \in W_r^p} q_w y_{rw}^p = I_0^p, \quad \forall p \in P, \tag{2}$$

$$\sum_{p \in P_{is}^-} \sum_{r \in R} \sum_{w \in W_r^p} q_{wi}^s y_{rw}^p = \bar{d}_i^s, \quad \forall i \in N, s \in P \text{ such that } \bar{d}_i^s > 0, \tag{3}$$

$$I_i^{0,s} + \sum_{p\in P_{is}^-}\sum_{r\in R}\sum_{w\in W_r^p} b_{wi}^s y_{rw}^p + d_i^s \le C_i, \quad \forall i\in N,\ s\in P, \tag{4}$$

$$\sum_{r\in R}\sum_{w\in W_r^p} a_{ri} y_{rw}^p \le 1, \quad \forall i\in N,\ p\in P, \tag{5}$$

$$\sum_{r\in R}\sum_{w\in W_r^p} y_{rw}^p \le K, \quad \forall p\in P, \tag{6}$$

$$0 \le I_0^p \le C_0, \quad \forall p\in P, \tag{7}$$

$$y_{rw}^p \ge 0, \quad \forall p\in P,\ r\in R,\ w\in W_r^p, \tag{8}$$

$$\sum_{w\in W_r^p} y_{rw}^p \in \{0,1\}, \quad \forall p\in P,\ r\in R. \tag{9}$$

The objective function (1) minimizes the total traveling and holding costs. Constraints (2) balance the inventory at the supplier from one period to the next. Constraints (3) ensure that the demand of each customer is met in each period. Constraints (4) impose the holding capacity at each customer in each period. Recall that the maximum inventory in a period s (the left-hand side term of (4)) is reached just before consumption after a possible delivery. It is thus equal to the inventory at the end of period s, arising from the initial inventory ($I_i^{0,s}$) or from past deliveries ($\sum_{p\in P_{is}^-}\sum_{r\in R}\sum_{w\in W_r^p} b_{wi}^s y_{rw}^p$), plus the demand ($d_i^s$) in this period. Constraints (5) and (6) ensure that, in each period, every customer is visited at most once and no more than K vehicles are used. Nonnegativity requirements on the variables are given by (7) and (8), together with the maximum inventory at the supplier. Binary requirements are not imposed directly on the y_{rw}^p variables, but rather on the routes themselves, allowing convex combinations of extreme RDPs (as in Desaulniers 2010).

As stated in the following proposition, depending on the specific IRP instance at hand, certain holding capacity constraints (4) can be redundant with constraints (3) and (5), and can, thus, be removed from the formulation to yield a more compact one. This can occur, for example, when the demand of a customer is the same for each period and its holding capacity is a multiple of this demand. The following notation is required for the proposition. For each customer $i\in N$ and each period $p\in P$, denote by σ_{ip} the latest period in the set P_{ip}^+ that we assumed to be nonempty. Note that $\sigma_{i,p-1} \le \sigma_{ip}$ for all customers $i\in N$ and periods $p\in P\backslash\{1\}$.

Proposition 2.1. *The capacity constraint* (4) *for customer* $i\in N$ *and period* $s\in P$ *is redundant with constraints* (3) *and* (5) *if* $s=1$ *or if* $s>1$ *and* $\sigma_{i,s-1}<\sigma_{is}$.

Proof. The inventory at customer i and period s, denoted I_i^s, can come from the initial inventory or from deliveries made in period s or before, that is, $I_i^s = I_i^{0,s} + \sum_{p\in P_{is}^-}\sum_{r\in R}\sum_{w\in W_r^p} b_i^s y_{rw}^p$. First, observe that if $\sigma_{is}=s$, then I_i^s must be equal to 0 and the inventory constraint (4) for i and s is not necessary. Thus, let us assume for the rest of the proof that $\sigma_{is}>s$.

The inventory I_i^s is dedicated to fulfill the demands (or the end inventory) of the periods in $\mathcal{L} = \{\ell\in P \mid s<\ell\le\sigma_{is}\}$. Let q_ℓ be the quantity dedicated to period $\ell\in\mathcal{L}$. Consequently $I_i^s = \sum_{\ell\in\mathcal{L}} q_\ell$. For $\ell\in\mathcal{L}\backslash\{\sigma_{is}\}$, we get that

$$\begin{aligned} q_\ell &= d_i^\ell && \text{if } I^{0,\ell}>0,\\ q_\ell &\le \bar{d}_i^\ell = d_i^\ell && \text{if } I^{0,\ell-1}=0,\\ q_\ell &\le I_i^{0,\ell-1} + \bar{d}_i^\ell = d_i^\ell && \text{otherwise.}\end{aligned}$$

The inequalities in the second and third cases ensue from the demand constraint (3) for customer i and period ℓ. For $\ell=\sigma_{is}\in\mathcal{L}$, observe first that $I^{0,\ell}=0$ given the assumption $P_{is}^+\neq\varnothing$. Furthermore, when $s=1$ or when $s>1$ and $\sigma_{i,s-1}<\sigma_{is}$, no quantity is dedicated to period σ_{is} unless it is delivered in period s. Given constraint (5) that imposes a maximum of one visit per customer and period, the maximum quantity delivered in this period and dedicated to period σ_{is} is less than or equal to $u_{is}^{\sigma_{is}}$, which in turn is less than or equal to $C_i - \sum_{j=s}^{\sigma_{is}-1} d_i^j - I_i^{0,\sigma_{is}-1}$. Consequently,

$$q_{\sigma_{is}} \le I_i^{0,\sigma_{is}-1} + C_i - \sum_{j=s}^{\sigma_{is}-1} d_i^j - I_i^{0,\sigma_{is}-1} = C_i - \sum_{j=s}^{\sigma_{is}-1} d_i^j.$$

The results deduced above can be gathered to derive the following inequality:

$$I_i^s = \sum_{\ell\in\mathcal{L}\backslash\{\sigma_{is}\}} q_\ell + q_{\sigma_{is}} \le \sum_{\ell\in\mathcal{L}\backslash\{\sigma_{is}\}} d_i^\ell + C_i - \sum_{j=s}^{\sigma_{is}-1} d_i^j = C_i - d_i^s,$$

which completes the proof. □

We also studied a different model that did not involve subdeliveries but contained inventory balance constraints for the customers (i.e., like the depot inventory balance constraints (2)) as used by, e.g., Grønhaug et al. (2010) and Engineer et al. (2012). As we report in §4, this model yields much weaker lower bounds than model (1)–(9) because it allows to fulfill the demand of a customer for a given period using a fraction of a visit in which a quantity larger than the demand is delivered. This is not possible with the proposed model because the quantity dedicated to each subdelivery cannot exceed the corresponding demand.

3. Branch-Price-and-Cut Algorithm

To solve model (1)–(9), we propose a branch-price-and-cut algorithm (see Barnhart et al. 1998; Desaulniers, Desrosiers, and Solomon 2005; Lübbecke and Desrosiers 2005), that is, a branch-and-bound algorithm where the lower bounds are computed using column generation and cutting planes are added dynamically to tighten

the linear relaxations. This section describes the column generation procedure, the cutting strategy, and the branching process, before discussing how the model and algorithm can be adapted to other IRP variants.

3.1. Column Generation

Column generation is used to solve the linear relaxations encountered in the branch-and-bound search tree. At the root node, the first linear relaxation corresponds to the linear relaxation of model (1)–(9) and is thus defined by (1)–(8). Note that the upper bounds imposed by the linear relaxation of (9) are redundant with (5) and can therefore be omitted. The other linear relaxations in the search tree are obtained by adding cutting planes and branching decisions. In this section, we focus on the first linear relaxation. In §§3.2 and 3.3, we indicate how column generation is adapted to handle cuts and branching decisions.

Consider the linear relaxation (1)–(8), which is called the master problem. Column generation is an iterative procedure that solves at each iteration the master problem restricted to a subset of its variables, referred to as the restricted master problem (RMP), and several subproblems (one per period $p \in P$). It starts with an RMP that contains a small number of variables y^p_{rw}. The routes and delivery patterns associated with these variables are computed using a greedy algorithm that builds for each period a set of routes visiting each customer once to deliver its demand for the corresponding period. Because these routes might not yield a feasible solution (due to, e.g., vehicle availability), artificial variables with a very large cost are also added to the RMP to ensure that a dual solution can be obtained. Once the RMP is solved by the simplex algorithm (or any other linear programming algorithm), its dual solution is used to define the objective function of the subproblems. The subproblem for period $p \in P$ aims at identifying negative reduced cost variables y^p_{rw} with respect to this dual solution. When negative reduced cost columns (variables) are found, they are added to the RMP, which is solved again to start a new iteration. Otherwise, when no subproblems can provide such columns, the column generation process stops and the computed primal solution to the current RMP is also optimal for the master problem.

In §§3.1.1–3.1.3, we define the subproblems, describe how they are solved, and discuss acceleration techniques used to speed up the column generation process.

3.1.1. Subproblem Definition. A feasible solution to the subproblem for period $p \in P$ corresponds to a feasible route $r \in R$ to be operated in period p together with a feasible extreme RDP $w \in W^p_r$. The cost of this solution must be equal to the reduced cost $\bar{c}^p_{rw}$ of the corresponding variable y^p_{rw}. Denoting by π^2_p, π^3_{is}, π^4_{is}, π^5_{ip}, and π^6_p the dual variables associated with constraints (2)–(6), respectively, this reduced cost is given by

$$\bar{c}^p_{rw} = c_{rw} + q_w \pi^2_p - \sum_{i \in N_r} \sum_{s \in P^+_{ip}} q^s_{wi} \pi^3_{is} - \sum_{i \in N_r} \sum_{s \in P^+_{ip}} b^s_{wi} \pi^4_{is} - \sum_{i \in N_r} \pi^5_{ip} - \pi^6_p. \tag{10}$$

The subproblem for period p can thus be defined as finding a route $r^* \in R$ and an extreme RDP $w^* \in W^p_{r^*}$ such that

$$(r^*, w^*) \in \arg\min_{r \in R,\, w \in W^p_r} \bar{c}^p_{rw}.$$

The routing part of this subproblem can be modeled on a directed network $G^p = (V^p, A^p)$, where V^p and A^p are the set of vertices and arcs, respectively. Set V^p comprises a source vertex v^S and a sink vertex v^E representing the depot at the start and the end of period p, respectively, as well as a vertex v^i for each customer $i \in N$. Set A^p is composed of all arcs $(i, j) \in N \times N$, $i \neq j$, all arcs (v^S, i), $i \in N$, and all arcs (i, v^E), $i \in N$. The "reduced" cost $\bar{c}_{ij}$ of arc $(i, j) \in A^p$ is given by

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi^6_p & \text{if } i = v^S \\ c_{ij} - \pi^5_{ip} & \text{otherwise,} \end{cases} \tag{11}$$

where c_{ij} is the travel cost associated with arc (i, j). Every route $r \in R$ corresponds to a path from v^S to v^E in G^p, and vice versa.

The delivery part of the subproblem can be modeled by considering subdelivery variables for each customer vertex. More precisely, with each customer $i \in N$ and each period $s \in P^+_{ip}$, we associate a variable ξ^s_i specifying the quantity delivered to customer i that is dedicated to fulfill the demand of period s if $s \in P$ or to stock the end inventory if $s = \rho + 1$. This variable must take a value in $[0, u^s_{ip}]$. For a route $r \in R$ visiting the customers in N_r, all variables ξ^s_i, $s \in P^+_{ip}$, for customers $i \in N \backslash N_r$ must take value 0. Furthermore, $\sum_{i \in N_r} \sum_{s \in P^+_{ip}} \xi^s_i \leq Q$. When respecting the above conditions, the vector $\xi = (\xi^s_i)_{i \in N, s \in P^+_{ip}}$ defines an RDP w associated with route r. In this case, the reduced cost $\bar{c}^p_{rw}$ introduced in (10) can be rewritten as

$$\bar{c}^p_{rw} = \sum_{(i,j) \in A_r} \bar{c}_{ij} + \sum_{i \in N_r} \sum_{s \in P^+_{ip}} \xi^s_i \left(\pi^2_p - \pi^3_{is} + \sum_{t \in P^+_{ip} | p \leq t < s} (h_i - \pi^4_{it}) \right), \tag{12}$$

where $A_r \subset A^p$ denotes the set of arcs defining route r in G^p.

Given a route $r \in R$, observe that one can find an RDP w (or equivalently, the corresponding values of the ξ^s_i variables) that yields the least reduced cost by solving the linear relaxation of a knapsack problem. In this case, at most one variable ξ^s_i can take a value in the open interval $]0, u^s_{ip}[$ (i.e., can correspond to a

partial subdelivery), yielding an extreme RDP in W_r^p. Consequently, the subproblem for period p can be seen as an elementary shortest path problem with resource constraints (ESPPRC) combined with the linear relaxation of a knapsack problem. This subproblem is similar to the one introduced by Desaulniers (2010) for the split delivery vehicle routing problem with time windows and can be solved using the labeling algorithm described in §3.1.2.

3.1.2. Labeling Algorithm. In a labeling algorithm for an ESPPRC defined on network $G^p = (V^p, A^p)$, labels are used to represent partial paths (routes) starting at the source vertex v^S. Labels are extended forwardly in G^p using extension functions and a dominance rule is applied to discard labels that cannot yield an optimal source-to-sink path. For the ESPPRC combined with the linear relaxation of a knapsack problem, a label does not only represent a partial path but also an associated extreme RDP. However, for a partial path associated with a vertex $i \neq v^E$ and an RDP containing a partial subdelivery, the quantity delivered in this subdelivery is unknown. It is only revealed when a path reaches the sink vertex v^E. Information about the subdelivery is, however, kept to properly compute its quantity once reaching the sink vertex v^E and the reduced cost of the resulting path/RDP. Below, we describe the label components, the extension functions, and the dominance rule used in our algorithm.

Let r be a partial path in G^p from v^S to a vertex $i \in V^p$ and $w \in W_p^r$ an associated RDP. The label, denoted $E_i = (T_i^{cost}, T_i^{loadF}, (T_i^{cust_k})_{k\in N}, T_i^{part}, T_i^{ratePiP}, T_i^{maxP})$, representing this feasible path/RDP contains the following components:

- T_i^{cost}: The reduced cost of the path/RDP (r, w). If $i \neq v^E$, this cost omits the dual contribution from the partial subdelivery if any.
- T_i^{loadF}: The total quantity delivered along the path r according to RDP w. If $i \neq v^E$, this quantity includes only the full subdeliveries.
- $T_i^{cust_k}$: A binary value that indicates whether or not customer $k \in N$ has been visited along path r.
- T_i^{part}: A binary value that indicates whether or not RDP w contains a partial subdelivery.
- $T_i^{ratePiP}$: The unit rate of contribution to the reduced cost associated with the partial subdelivery if any.
- T_i^{maxP}: The maximum quantity that can be delivered in the partial subdelivery if any.

In the labeling algorithm, an extreme RDP is seen as a sequence of customer delivery patterns (CDPs), one for each visited customer. A *CDP* specifies a combination of subdelivery types associated with a customer $i \in N$ in period p. More precisely, for each period $s \in P_{ip}^+$, it indicates if the subdelivery associated with s is a zero (Z), full (F), or partial subdelivery (P). The latter type is admissible only if $u_{ip}^s > 1$. A CDP can contain at most one partial subdelivery and the total quantity delivered in its full subdeliveries cannot exceed Q. With each customer vertex $i \in N$, we associate a list Γ_{ip} of CDPs. For instance, if P_{ip}^+ contains two periods, then this list is composed of the following CDPs: FF, FP, PF, FZ, ZF, PZ, ZP, and ZZ, where, e.g., FP means that a full subdelivery occurs for the first period in P_{ip}^+ and a partial one for its second period. To reduce the size of the CDP lists, we can again apply the FIFO rule. Under this rule, various CDPs (for example, FPF, FZF, PZF if $|P_{ip}^+| = 3$) can be discarded.

For each CDP $\gamma \in \Gamma_{ip}$ and each period $s \in P_{ip}^+$, we define the binary parameter f_γ^s (respectively, g_γ^s) that takes value 1 if CDP γ contains a full (respectively, partial) subdelivery for period s. With each CDP $\gamma \in \Gamma_{ip}$, we associate the following values:

- $\tau_\gamma^{cost} = \sum_{s\in P_{ip}^+} f_\gamma^s u_{ip}^s(\pi_p^2 - \pi_{is}^3 + \sum_{t\in P_{ip}^+|p\le t<s}(h_i - \pi_{it}^4))$: The contribution to the path/RDP reduced cost (12).
- $\tau_\gamma^{loadF} = \sum_{s\in P_{ip}^+} f_\gamma^s u_{ip}^s$: The total quantity delivered in the full subdeliveries.
- $\tau_\gamma^{part} = \sum_{s\in P_{ip}^+} g_\gamma^s$: The number of partial deliveries (0 or 1).
- $\tau_\gamma^{ratePiP} = \sum_{s\in P_{ip}^+} g_\gamma^s(\pi_p^2 - \pi_{is}^3 + \sum_{t\in P_{ip}^+|p\le t<s}(h_i - \pi_{it}^4))$: The rate of contribution to the path/RDP reduced cost (12) for each unit delivered in the partial delivery if any.
- $\tau_\gamma^{maxP} = \sum_{s\in P_{ip}^+} g_\gamma^s(u_{ip}^s - 1)$: The maximum quantity that can be delivered in the partial delivery if any.

Note that we can discard any CDP γ that contains a partial delivery such that $\tau_\gamma^{ratePiP} \le 0$. Indeed, in this case, replacing the partial delivery by a zero delivery can never yield a worse RDP. Thus, we assume in the following that $\tau_\gamma^{ratePiP} > 0$.

To generate new labels, the labeling algorithm applies the following extension functions. Let $E_i = (T_i^{cost}, T_i^{loadF}, (T_i^{cust_k})_{k\in N}, T_i^{part}, T_i^{ratePiP}, T^{maxP})$ be a label associated with vertex $i \in V^p\backslash\{v^E\}$. This label is extended along an arc $(i, j) \in A^p$, $j \neq v^E$, as many times as there are CDPs in the list Γ_{jp}. Let γ be one of these CDPs and denote by $E_j = (T_j^{cost}, T_j^{loadF}, (T_j^{cust_k})_{k\in N}, T_j^{part}, T_j^{ratePiP}, T_j^{maxP})$ the label computed using the following extension functions:

$$T_j^{cost} = T_i^{cost} + \bar{c}_{ij} + \tau_\gamma^{cost}, \tag{13}$$

$$T_j^{loadF} = T_i^{loadF} + \tau_\gamma^{loadF}, \tag{14}$$

$$T_j^{cust_k} = \begin{cases} T_i^{cust_k} + 1 & \text{if } j = k \\ T_i^{cust_k} & \text{otherwise,} \end{cases} \quad \forall k \in N, \tag{15}$$

$$T_j^{part} = T_i^{part} + \tau_\gamma^{part}, \tag{16}$$

$$T_j^{ratePiP} = T_i^{ratePiP} + \tau_\gamma^{ratePiP}, \tag{17}$$

$$T_j^{maxP} = \begin{cases} \min\{\tau_\gamma^{maxP}, Q - T_i^{loadF} - \tau_\gamma^{loadF}\} \\ \qquad \text{if } \tau_\gamma^{part} = 1 \\ \min\{T_i^{maxP}, Q - T_i^{loadF} - \tau_\gamma^{loadF}\} \\ \qquad \text{otherwise.} \end{cases} \tag{18}$$

The resulting label E_j is declared feasible if $T_j^{loadF} \le Q$, $T_j^{cust_k} \le 1$ for all $k \in N$, and $T_j^{part} \le 1$.

When label E_i is extended along an arc (i, j) with $j = v^E$, the only label component of E_j that needs to be computed is T_j^{cost}. It is computed as

$$T_j^{cost} = \begin{cases} T_i^{cost} + \bar{c}_{ij} + T_i^{maxP} T_i^{ratePiP} & \text{if } T_i^{ratePiP} < 0 \\ T_i^{cost} + \bar{c}_{ij} & \text{otherwise.} \end{cases} \tag{19}$$

To eliminate nonpromising labels, a dominance rule is applied. Given that the quantity to be delivered in a partial subdelivery remains unknown until reaching the sink vertex v^E, this rule must take into account all possible reduced costs that can be achieved with this subdelivery, if any. The reduced cost of a label can thus be seen as a function of the quantity that can be delivered in this subdelivery. More precisely, if $T_i^{part} = 1$ in a label $E_i = (T_i^{cost}, T_i^{loadF}, (T_i^{cust_k})_{k \in N}, T_i^{part}, T_i^{ratePiP}, T_i^{maxP})$, then the reduced cost $\bar{C}_i(\xi^{part})$ of this label in function of the quantity ξ^{part} that can be delivered in the partial subdelivery is

$$\bar{C}_i(\xi^{part}) = T_i^{cost} + \xi^{part} T_i^{ratePiP}, \quad \forall\, \xi^{part} \in [0, T_i^{maxP}].$$

This function corresponds to a line segment and the dominance rule must, therefore, allow the comparison of line segments. The dominance rule that we use was introduced by Desaulniers (2010) (except for the time component that is not present in our case) and states as follows.

Definition 3.1. *A label* $E_1 = (T_1^{cost}, T_1^{loadF}, (T_1^{cust_k})_{k \in N}, T_1^{part}, T_1^{ratePiP}, T_1^{maxP})$ *is said to dominate a label* $E_2 = (T_2^{cost}, T_2^{loadF}, (T_2^{cust_k})_{k \in N}, T_2^{part}, T_2^{ratePiP}, T_2^{maxP})$ *if both labels* E_1 *and* E_2 *are associated with the same vertex and the following conditions are satisfied*:

(a) $T_1^{loadF} \le T_2^{loadF}$;
(b) $T_1^{cust_k} \le T_2^{cust_k}$, $\forall k \in N$;
(c) $T_1^{part} \le T_2^{part}$;
(d) $T_1^{cost} - T_1^{maxP} T_1^{ratePiP} \le T_2^{cost} - T_2^{maxP} T_2^{ratePiP}$;
(e) $T_1^{cost} - (T_2^{loadF} - T_1^{loadF}) T_1^{ratePiP} \le T_2^{cost}$;
(f) $T_1^{cost} - (T_2^{loadF} + T_2^{maxP} - T_1^{loadF}) T_1^{ratePiP} \le T_2^{cost} - T_2^{maxP} T_2^{ratePiP}$.

Conditions (d)–(f) allow to compare the reduced cost functions associated with labels E_1 and E_2 (see Desaulniers 2010 for explanations). Dominated labels according to this dominance rule can be discarded except when both labels E_1 and E_2 dominate each other (i.e., when all conditions (a)–(f) hold at equality). In the latter case, one of the two labels must be kept.

3.1.3. Acceleration Techniques. To speed up the column generation procedure, we apply the following acceleration techniques.

CDP handling. A list Γ_{ip} of CDPs is associated with each customer $i \in N$ and each period $p \in P$. This list is established once at the beginning of the solution process but the values τ_γ^{cost} and $\tau_\gamma^{ratePiP}$ must be updated at the beginning of each column generation iteration according to the current RMP dual solution. Before executing the labeling algorithm, the list Γ_{ip} can be filtered to remove dominated CDPs. To compare CDPs, we apply the dominance rule of Definition 3.1 (except for condition (b), which is not considered) in which all T values are replaced by the corresponding τ values associated with the CDPs.

ng-path relaxation. Given the complexity of the subproblems, it might be advantageous to relax them by allowing the generation of paths containing cycles. We use the *ng*-path relaxation introduced by Baldacci, Mingozzi, and Roberti (2011). Given the network $G^p = (V^p, A^p)$ associated with period $p \in P$, a neighborhood $\mathcal{N}_v$ is defined for each vertex $v \in V^p$. It contains v and the vertices that are the closest to v in terms of distance such that $|\mathcal{N}_v| = \lambda$, a predefined parameter value (set to 5 in our computational tests). An *ng*-path can contain a cycle $v_1 - v_2 - \cdots - v_h$ with $v_1 = v_h$ only if there exists $\ell \in \{2, 3, \ldots, h-1\}$ such that $v_1 \notin \mathcal{N}_{v_\ell}$. To generate *ng*-paths, the labeling algorithm can easily be adapted as explained in Desaulniers, Madsen, and Røpke (2014).

Bidirectional labeling. To further speed up the solution of the subproblems, we apply bidirectional labeling (Righini and Salani 2006) that consists of, first, extending labels forwardly until reaching half of the vehicle capacity, second, extending labels backwardly until reaching half of the vehicle capacity, and, finally, merging forward and backward labels associated with the same vertex.

Heuristic column generators. The following two heuristic column generators are used to attempt to generate negative reduced cost columns within short computational times.

The first heuristic is a multistart tabu search algorithm similar to the one proposed in Archetti, Bouchard, and Desaulniers (2011). Given an initial route associated with a given period, it starts by optimizing the quantities delivered along this route, that is, it computes an optimal RDP by solving the linear relaxation of a knapsack problem. Then, at each iteration of the tabu search algorithm, it applies a move of one of the two following types: customer insertion and customer removal. To determine which move to apply, all non tabu moves are evaluated with respect to their impact on the route/RDP reduced cost. The move yielding the smallest reduced cost is retained and the inverse move is made tabu for a certain number of iterations (five

for our tests). Tabu search is stopped when reaching a maximum number of iterations (15 for our tests). As proposed in Desaulniers, Lessard, and Hadjar (2008), the tabu search algorithm is executed on each route associated with a basic variable in the current RMP solution.

The second heuristic column generator corresponds to the labeling algorithm applied on networks that contain only a subset of their arcs. The arcs are selected using the procedure of Desaulniers, Lessard, and Hadjar (2008) that removes every arc whose reduced cost is too large to be part of the κ least reduced cost arcs out of its tail node and into its head node. In our case, we add to the reduced cost of an arc the average reduced cost of the CDPs associated with each of its end customers (assuming that no quantity is delivered in the partial deliveries). For our tests, the value of κ is dynamic. It starts at 1 and increases by 3 after every column generation iteration where this column generator is invoked and cannot generate columns in every subproblem.

Three algorithms are, thus, available to generate columns: tabu search, heuristic labeling, and exact labeling. These column generators are used as proposed in Desaulniers, Lessard, and Hadjar (2008). At each column generation iteration, the multistart tabu search heuristic is first executed. If negative reduced cost columns are found, they are added to the RMP, which is reoptimized to start a new iteration. Otherwise, the heuristic labeling algorithm is invoked to solve each subproblem. Here again, if new columns are generated, a new iteration is started. Otherwise, the exact algorithm is applied to solve each subproblem.

Constraint relaxation. Given that the constraints (5) limiting to one the number of visits to each customer in each period are numerous and may often be inactive, these constraints are initially relaxed from the formulation and reintroduced whenever they are violated as in a branch-and-cut fashion.

3.2. Cutting

When the computed optimal solution to a linear relaxation does not satisfy the integrality requirements (9) and the corresponding node is not pruned, we search for violated valid inequalities to strengthen this linear relaxation. In §§3.2.1–3.2.4, we present the four families of valid inequalities considered and discuss separately how the column generation algorithm is modified to handle them.

3.2.1. Inequalities on the Minimum Number of Visits per Customer. Given the vehicle capacity Q, the holding capacity at a customer $i \in N$, and the residual demands $\bar{d}_i^s$, $s \in P$, of this customer, it is possible to compute a lower bound on the number of times that this customer must be visited in periods 1 to ℓ for every $\ell \in P$. This lower bound is given by $lb_{i\ell}^V = \lceil \sum_{s=1}^{\ell} \bar{d}_i^s / \min\{Q, C_i\} \rceil$. Therefore, the following inequalities are valid:

$$\sum_{p=1}^{\ell} \sum_{r \in R} \sum_{w \in W_r^p} a_{ri} y_{rw}^p \geq lb_{i\ell}^V, \quad \forall i \in N,\ \ell \in P. \tag{20}$$

For an arc flow model, these inequalities were introduced by Archetti et al. (2007) for the single-vehicle case and generalized by Coelho and Laporte (2014) for the multiple-vehicle case.

Violated inequalities are found by enumeration and added to the master problem. Let $\pi_{i\ell}^{20}$, $i \in N$, $\ell \in P$, be the dual variables associated with these inequalities. In the subproblem for period $p \in P$, these dual variables modify the arc reduced costs $\bar{c}_{ij}$ as follows:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi_p^6 & \text{if } i = v^S \\ c_{ij} - \pi_{ip}^5 - \sum_{\ell=p}^{\rho} \pi_{i\ell}^{20} & \text{otherwise,} \end{cases} \quad \forall (i,j) \in A^p. \tag{21}$$

Remark that, for a given customer $i \in N$, an inequality (20) for a period $\ell \in P \backslash \{1\}$ is dominated by the inequality (20) for period $\ell - 1$ if $lb_{i\ell}^V = lb_{i,\ell-1}^V$. In this case, there is no need to generate the former inequality.

3.2.2. Inequalities on the Minimum Number of Routes per Time Interval. Similar to the reasoning of the previous section, one can sum the residual demands of all customers in a period for every period and determine lower bounds on the minimum number of routes required to service all demands in a time interval starting at the beginning of the planning horizon. In particular, a lower bound on the number of routes required to service all residual demands arising in periods 1 to $\ell \in P$ is given by $lb_\ell^R = \lceil \sum_{i \in N} \sum_{s=1}^{\ell} \bar{d}_i^s / Q \rceil$. From these bounds, we deduce the following valid inequalities:

$$\sum_{p=1}^{\ell} \sum_{r \in R} \sum_{w \in W_r^p} y_{rw}^p \geq lb_\ell^R, \quad \forall \ell \in P. \tag{22}$$

Again, violated inequalities are found by enumeration and added to the master problem. In the subproblem associated with period $p \in P$, the corresponding dual variables, denoted π_ℓ^{22}, $\ell \in P$, modify the arc reduced costs $\bar{c}_{ij}$ as follows:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi_p^6 - \sum_{\ell=p}^{\rho} \pi_\ell^{22} & \text{if } i = v^S \\ c_{ij} - \pi_{ip}^5 & \text{otherwise,} \end{cases} \quad \forall (i,j) \in A^p. \tag{23}$$

3.2.3. Inequalities on the Minimum Number of Subdeliveries per Demand. Inequalities on the minimum number of (sub-)deliveries per demand (MNSD) were introduced in Desaulniers (2010). For a given

customer $i \in N$ and its residual demand $\bar{d}_i^s$ for a period $s \in P$, this inequality stipulates that this demand can be satisfied in two different ways, namely, either by performing a subdelivery of $\bar{d}_i^s$ in a period $p \in P_{is}^-$ or by performing at least two subdeliveries in different periods $p \in P_{is}^-$. More precisely, the MNSD inequalities are as follows:

$$\sum_{p \in P_{is}^-} \sum_{r \in R} \sum_{w \in W_r^p} (2a_{iw}^S + a_{iw}^M) y_{rw}^p \geq 2,$$

$$\forall i \in N, s \in P \text{ such that } \bar{d}_i^s > 0, \quad (24)$$

where a_{iw}^S (respectively, a_{iw}^M) is a binary parameter equal to 1 if $a_{ir} = 1$ and $\bar{d}_i^s$ (respectively, less than $\bar{d}_i^s$) units is delivered in the subdelivery for customer i and period s in the RDP w and 0 otherwise.

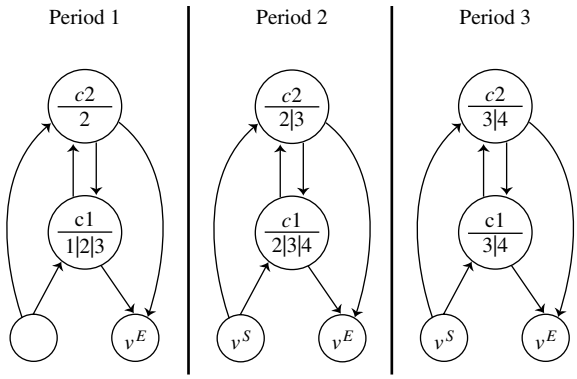
These inequalities are separated by enumeration and violated ones are added to the master problem. Let π_{is}^{24}, $i \in N$, $s \in P$, be the dual variables associated with the inequalities (24). To take these dual values into account in the subproblem for period $p \in P$, we modify the definition of parameters τ_γ^{cost}, $\gamma \in \Gamma_{ip}$, $i \in N$, as follows:

$$\tau_\gamma^{cost} = \sum_{s \in P_{ip}^+} \left[f_\gamma^s u_{ip}^s \left(\pi_p^2 - \pi_{is}^3 - \sum_{t \in P_{ip}^+ \mid p \leq t < s} (h_i - \pi_{it}^4) \right) - (1 + f_\gamma^s) \pi_{is}^{24} \right].$$

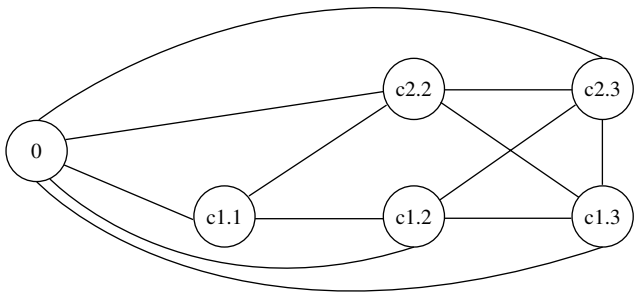
3.2.4. Capacity Inequalities. Capacity inequalities have been introduced by Laporte, Nobert, and Desrochers (1985) for the capacitated vehicle routing problem (CVRP) and later used by many authors for several variants of the vehicle routing problem. In this context, these inequalities can be stated as follows. Given a subset $U \subseteq N$ of customers, each with a known demand, and a lower bound $\kappa(U)$ on the number of vehicles required to service these customers according to vehicle capacity, the total flow of vehicles incident to the subset U must be greater than or equal to $2\kappa(U)$. To our knowledge, these inequalities have not yet been adapted for the IRP. This adaptation is not straightforward because, in the IRP, each customer has several demands and each demand can be covered using subdeliveries in several periods. Here, we provide such an adaptation.

Instead of using the customers to define the capacity inequalities as in the CVRP, we use the positive residual demands of the customers. Let $RD = \{(i, s) \in N \times P \mid \bar{d}_i^s > 0\}$ be the set of these demands and let $G^* = (V^*, E^*)$ be an auxiliary graph that allows to represent the flow between consecutive residual demands (or the depot and the residual demands) assuming that the subdeliveries associated with a customer vertex are performed in chronological order. The vertex set V^* contains a depot vertex 0 and one vertex for each demand in RD. In the edge set E^*, there is an edge linking the depot 0 to each demand vertex $(i, s) \in RD$. Furthermore, a demand vertex (i, s) is linked to its successor demand vertex $(i, s+1)$ if this successor exists. Finally, a demand vertex (i, s) is linked to another demand vertex (i', s'), $i \neq i'$, if there exists a period $p \in P$ such that s is the latest period in $P_{ip}^+ \cap P$ and s' is the earliest period in $P_{i'p}^+ \cap P$.

To illustrate the structure of an auxiliary graph (see Figure 1), consider an example for which $P = \{1, 2, 3\}$ and $N = \{c1, c2\}$. There is a positive residual demand in all periods for customer $c1$ but only in periods 2 and 3 for customer $c2$. Figure 1(a) shows the networks $G^p = (V^p, A^p)$ associated with each period $p \in P$. For each customer vertex $i \in V^p$, the periods in set P_{ip}^+ are given in the circle below the customer identification. For instance, $P_{c2,1}^+ = \{2\}$ while $P_{c1,2}^+ = \{2, 3, 4\}$ where $4 = \rho + 1$ is the fictitious period associated with the end inventory. The corresponding auxiliary graph $G^* = (V^*, E^*)$ is drawn in Figure 1(b). In G^*, the demand vertex associated with customer i and period p is denoted $i.p$ (for instance, $c2.3$). For example, there is an edge between $c2.3$ and $c1.2$ because the latest period in $P_{c2,2}^+ \cap P$ is 3, and the earliest period in $P_{c1,2}^+$ is 2. This edge is used to represent the direct flow between these demands and can be computed as the flow on the arc $(c2, c1) \in A^2$. In certain cases, the flow on an



(a) Networks G^1, G^2, and G^3



(b) Auxiliary graph G^*

Figure 1 Example of Networks and Their Corresponding Auxiliary Graph

edge is computed as the total flow on multiple arcs (e.g., the flow on the edge linking 0 and $c2.2$ is equal to the total flow on arcs $(v^S, c2) \in G^1$, $(c2, v^E) \in G^1$, and $(v^S, c2) \in G^2$) or as the total flow into one or several vertices (e.g., the flow on the edge linking $c1.2$ and $c1.3$ is equal to the total flow into vertices $c1 \in N^1$ and $c1 \in N^2$).

For all $e \in E^*$, denote by $A^p_e \subset A^p$ and $V^p_e \subset V^p$, $p \in P$, the subsets of arcs and vertices in G^p associated with edge e, respectively. Let $U \subseteq RD$ be a subset of the residual demands, $\delta(U) \subseteq E^*$ the subset of edges with one vertex in U and the other in $V^* \backslash U$, and $\kappa(U) = \lceil \sum_{(i,s)\in U} \bar{d}^s_i / Q \rceil$ a lower bound on the number of deliveries required to cover the demands in U. For each edge $e \in E^*$, we define a nonnegative edge flow variable x_e that is computed as the total direct flow between the residual demands (or the depot) linked by edge e

$$x_e = \sum_{p\in P} \sum_{r\in R} \sum_{w\in W^p_r} \left(\sum_{(i,j)\in A^p_e} a_{rij} + \sum_{i\in V^p_e} a_{ri} \right) y^p_{rw},$$

where a_{rij} is a binary parameter indicating whether arc $(i, j) \in A^p$ is traversed or not in route $r \in R$. Given this notation, the capacity inequalities can be written as follows:

$$\sum_{e\in\delta(U)} x_e = \sum_{e\in\delta(U)} \sum_{p\in P} \sum_{r\in R} \sum_{w\in W^p_r} \left(\sum_{(i,j)\in A^p_e} a_{rij} + \sum_{i\in V^p_e} a_{ri} \right) y^p_{rw} \geq 2\kappa(U), \quad \forall U \subseteq RD. \tag{25}$$

Generated capacity cuts (25) are added to the master problem. Let π^{25}_U be the dual variable associated with the cut for subset $U \subseteq RD$. In the subproblem for period $p \in P$, these dual values modify the arc reduced costs $\bar{c}_{ij}$ as follows:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi^6_p - \sum_{U\subseteq RD} m^U_{ijp} \pi^{25}_U & \text{if } i = v^S \\ c_{ij} - \pi^5_{ip} - \sum_{U\subseteq RD} (m^U_{ijp} + m^U_{ip}) \pi^{25}_U, & \text{otherwise,} \end{cases} \quad \forall (i, j) \in A^p, \tag{26}$$

where m^U_{ijp} takes value 1 if the edge associated with arc $(i, j) \in A^p$ is in $\delta(U)$ and 0 otherwise, and m^U_{ip} indicates the number of edges in $\delta(U)$ associated with vertex $i \in V^p$.

The separation of the capacity inequalities is known to be strongly $\mathcal{NP}$-hard. Therefore, we apply various heuristics to search for violated capacity inequalities. First, we invoke the separation routines of the CVRPSEP package that were developed by Lysgaard, Letchford, and Eglese (2004). These routines are applied on the auxiliary graph G^*. Because these routines were designed for the CVRP in which the flow through each demand (customer) vertex is equal to one, they might identify inequalities that are not violated for the IRP in which the flow through a demand vertex might exceed one. In consequence, we filter out all proposed inequalities that are not violated.

Second, we inspect the routes/RDPs (r, w) associated with a variable y^p_{rw} taking a positive fractional value in the current linear relaxation solution and for which w contains exactly one partial subdelivery. In this case, the sum of the demands associated with the partial and the full subdeliveries in the RDP w exceeds the vehicle capacity (otherwise, the partial subdelivery would not be partial), showing that, for this subset U of demands, $\kappa(U) \geq 2$. For each subset U built in this way, we verify if the corresponding capacity inequality (25) is violated.

Finally, we apply a route-based connected component heuristic similar to the one proposed by Archetti, Bouchard, and Desaulniers (2011) for the split delivery vehicle routing problem with time windows. This heuristic proceeds as follows. First, the flow on each edge of the auxiliary graph G^* is computed. Edges with no flow are removed and the connected components of the remaining graph (excluding the depot vertex) are identified. Let $\mathscr{C}$ be the set of connected components. Each connected component $\phi \in \mathscr{C}$ defines a subset $U_\phi \subseteq RD$ for which we verify if the corresponding inequality (25) is violated. Then, for each component $\phi \in \mathscr{C}$ and each period $p \in P$, we determine the subset $R^p_\phi \subset R$ of routes operated in period $p \in P$ that flow into component ϕ. For each route $r \in R^p_\phi$, we compute its total flow $y^p_r = \sum_{w\in W^p_r} y^p_{rw}$. Each route with no flow is removed from R^p_ϕ. Let $R_\phi = \cup_{p\in P} R^p_\phi$ be the set of all routes with a positive flow associated with ϕ. Then for each connected component $\phi \in \mathscr{C}$, the heuristic applies a recursive procedure that starts with $U = U_\phi$ and, at each iteration, removes from the current set U the subset $RD_{r'p'}$ of residual demands covered by a route (r', p') in R_ϕ. To determine which route (r', p') to select, we sort the routes (r, p) in R_ϕ in decreasing order of their value $\mu_{rp}(U) = \sum_{e\in\delta(U)} (\sum_{(i,j)\in A^p_e} a_{rij} + \sum_{i\in N^p_e} a_{ri}) y^p_r - \sum_{i\in N_r} \sum_{s\in P^+_{ip}|(i,s)\in U} \bar{d}^s_i$ and select them in this order. The first part of $\mu_{rp}(U)$ is equal to the decrease in the left-hand side of the capacity inequality (25) if the demands in RD_{rp} are removed from U, and its second part positively influences a decrease in its right-hand side. As a consequence, large values of $\mu_{rp}(U)$ favor a high decrease in the left-hand side and a low decrease in the right-hand side, thus increasing the chance of finding a violated inequality. The values $\mu_{rp}(U)$ are computed for every U encountered during the search. The recursive search is limited in depth and in width by two parameters. For our tests, we used a maximum depth of 5 (that is, the residual demands of at most five routes can be removed from the initial set U_ϕ) and a maximum width of 8 (for each depth at most eight routes are selected for removal).

3.3. Branching

In our algorithm, the following four types of branching decisions can be imposed when the computed linear relaxation solution is fractional. They are defined on the following variables:

1. The total number of routes over all periods ($\sum_{p\in P}\sum_{r\in R}\sum_{w\in W_r^p} y_{rw}^p$).
2. The number of routes in each period $p \in P$ ($\sum_{r\in R}\sum_{w\in W_r^p} y_{rw}^p$).
3. The flow through each customer vertex $i \in N$ in each period $p \in P$ ($\sum_{r\in R}\sum_{w\in W_r^p} a_{ri} y_{rw}^p$).
4. The flow on each edge $< i, j >$ in each period $p \in P$ where the flow on an edge $< i, j >$ in period p is equal to the sum of the flows on the arcs (i, j) and (j, i) in A^p ($\sum_{r\in R}\sum_{w\in W_r^p}(a_{rij} + a_{rji}) y_{rw}^p$).

Also used by Archetti et al. (2007), the third branching decision type focuses on determining in which periods each customer must be visited. Such decisions are very effective (see §4.3) as they highly constrain the quantity that can be delivered in each customer visit and substantially increase the lower bounds in many branch-and-bound nodes. On the other hand, the latter branching decision type is sufficient to guarantee an optimal integer solution. Indeed, it can be proven that the integrality requirements (9) are equivalent to integrality requirements on the arc flows in the networks G^p, $p \in P$. Furthermore, observe that, for every route $r = (v^S, v_1, v_2, \ldots, v^E) \in R$ and every period $p \in P$, the reverse route $r' = (v^E, \ldots, v_2, v_1, v^S)$ exists, has the same cost, and can be associated with the same RDP. Consequently, traversing a route in one direction or the other is equivalent and the integrality requirements on the arc flows can be relaxed to integrality requirements on the edge flows.

All branching decisions are imposed by adding a constraint in the master problem, except when the flow on an edge $< i, j >$ in a period p must be set to 0 (a decision of type 4). In this case, both arcs (i, j) and (j, i) are removed from A^p. When a constraint is added to the master problem, the corresponding dual variable must be subtracted from the reduced cost of certain arcs (for reasons of conciseness, we omit the details here).

To determine which decisions are imposed when a fractional linear relaxation is obtained (and the branch-and-bound node is not pruned), we proceed as follows. For each type of decisions 1 to 4, we compute the value of each candidate variable (i.e., the total number of routes for type 1, the number of routes for each period for type 2, the flow through each vertex for type 3, and the flow on each edge for type 4). For each type, we select the candidate variable whose fractional value is closest to 0.5. If one of the variables selected for type 3 and type 4 has a fractional value in the interval $[0.25, 0.75]$, then we branch on one of these two variables, favoring the one with the value closest to 0.5 (at equality, we select the decision of type 3). Otherwise, we choose the variable whose value is closest to 0.5 among all decision types.

The branch-and-bound tree is explored using a combination of best-first and depth-first search that we call *local depth-first* search. It tries to exploit strengths of these two exploration strategies: best-first minimizes the number of explored branch-and-bound nodes whereas depth-first allows a fast average reoptimization time from one linear relaxation to the next. Under the local depth-first policy, a node is first chosen according to the best-first rule and then a subtree of its progeny is explored using a limited depth-first search. The nodes explored in this subtree are limited by a tolerance on the gap between the best available lower bound and the lower bound associated with a node (this tolerance was set to 10 for our tests). More precisely, when locally exploring a subtree using depth-first search, a node is not evaluated if the gap between the best available lower bound and the lower bound of its father node exceeds the given tolerance. The evaluation of this node is then postponed until it is selected by the best-first rule. In fact, once the exploration of a subtree is completed, the next node to evaluate is chosen according to the best-first criterion. The process then repeats by exploring a subtree of its progeny.

3.4. Adaptations for Other Problem Variants

The model and algorithm proposed for the IRP can easily be adapted to handle some variants of the IRP, namely, with customer time windows, split deliveries, a heterogeneous fleet, or multiple products. Let us briefly discuss the adaptations required for each case.

Customer time windows restrict the time at which service can occur at each customer taking into account travel and service times. They restrict the set of feasible routes R and must be handled in the subproblems to ensure that every generated route satisfies them. To do so, a time component must be added to the labels. Given that route feasibility becomes direction dependent in this case, it is not sufficient anymore to branch on the edge flows. One must ensure that arc flows are integer.

A split delivery occurs when a customer is visited by more than one vehicle in the same period. To handle split deliveries, constraints (5) must be removed from model (1)–(9), together with their dual variables from the subproblems. Given that the flow through a vertex $i \in V^p$, $p \in P$, can exceed one, the branching rules defined in §3.3 might not be sufficient to reach integrality. To complement these rules, one can also branch, for instance, on the flow on each pair of arcs traversed consecutively in a route (for details, see Desaulniers 2010).

To consider a heterogeneous fleet of vehicles (with different capacities or traveling costs, or housed in

multiple depots), a subproblem per period and vehicle type (or depot) according to the vehicle type characteristics must be defined. In model (1)–(9), each vehicle availability constraint (6) needs to be replaced by a set of similar constraints, one per vehicle type. Their corresponding dual variables must be taken into account in their respective subproblems.

Finally, the variant with multiple products (see Coelho and Laporte 2013b) can also be handled in theory. In models (1)–(9), copies of the inventory variables and the constraints (2)–(4) for each product would be required, assuming that the inventory capacity is separated by product. CDPs would then specify the type (F, Z, or P) of each of the possible subdeliveries for each product. Because the products delivered along a route share the same vehicle capacity, the constraint specifying that at most one partial subdelivery can occur in a route remains valid. Nevertheless, in practice, the increase in the number of CDPs at each customer might prevent solving even small-sized instances.

4. Computational Experiments

The branch-price-and-cut algorithm described in §3 was implemented using C and the Gencol library (version 4.5) with modifications. All restricted master problems were solved using CPLEX 12.2. This algorithm is compared to the branch-and-cut algorithm of Coelho and Laporte (2014) that was implemented in C++ using CPLEX 12.2. All tests were performed on an Intel Core i7-2600 processor clocked at 3.4 GHz with 8 cores and 16 GB RAM. For our tests, only a single core was used for both algorithms.

To evaluate our algorithm, we used the instances created by Archetti et al. (2007). The original instance set is composed of 160 instances involving five to 50 customers. They are divided into four classes based on inventory holding cost and planning horizon length: H3, H6, L3, and L6. The H3 and H6 (L3 and L6) classes contain instances with high (low) holding costs, whereas the H3/L3 and H6/L6 classes have three and six time periods, respectively. All instances involve a single vehicle, but they have been used to evaluate multivehicle algorithms by simply dividing the original vehicle capacity by the number of desired vehicles. In our tests we ran each instance with two to five vehicles, totaling 640 different instances. Optimal solutions are known only for a limited number of instances, and typically the existing branch-and-cut algorithms can consistently solve within reasonable times instances with up to 25 customers, three periods, and three vehicles. For the remaining instances, the gaps are rather large, and for many instances there are no known feasible solutions. In what follows, we provide comparisons of our branch-price-and-cut algorithm with the branch-and-cut algorithm of Coelho and Laporte (2014). For both algorithms, a maximum time limit of two hours was imposed for all tests. Detailed results on all instances are available online at http://www.leandro-coelho.com.

For the main tests (§§4.1 and 4.2), we used the branch-price-and-cut algorithm described above for all instances except that the MNSD inequalities (24) are not applied for the H3 and L3 instances. Preliminary tests showed that they were not useful for instances with three periods. In §4.3, we present a sensitivity analysis on the usage of some of the main components of our algorithm.

4.1. Linear Relaxation Results

First, we provide an analysis of the computational results obtained when solving the linear relaxation of (1)–(9), with or without the cuts presented in §3.2. We compare these results with those computed by the algorithm of Coelho and Laporte (2014) on the linear relaxation of their arc-flow model when all cuts are enabled. We also compare them with those obtained by a branch-and-price algorithm applied to a model with customer inventory balance constraints and no subdeliveries (see §2). The results are presented by groups of instances, where a group contains the instances of the same class with the same number of vehicles (denoted by K).

For each instance group, Table 1 reports the average best upper bound on the integrality gap in percentage obtained by the branch-and-cut algorithm (B&C) of Coelho and Laporte (2014), by a branch-and-price algorithm with customer inventory balance constraints (B&P with CIBC), our branch-and-price algorithm without cuts (B&P), and our branch-price-and-cut algorithm (BP&C). For each instance, the best upper bound on the integrality gap (hereafter called the integrality gap) is computed using the formula $(\bar{z} - \underline{z})/\bar{z}$, where $\underline{z}$ is the lower bound computed at the root node of the search tree (with or without cuts) and $\bar{z}$ is the optimal value or, if unknown, the best upper bound computed by any of the algorithms. The results are also summarized for each number of vehicles in Figure 2. From these results, we observe that model (1)–(9) yields tighter bounds than the model with customer inventory balance constraints. Indeed, average reductions of the average integrality gap (in percentage) by group ranging between 0.37 and 1.27 are achieved using subdeliveries and customer constraints (3) and (4). Given these average gap differences, the model with customer inventory balance constraints underperformed in preliminary tests against the proposed model (1)–(9) and was not included in the following computational experiments. On the other hand, the branch-price-and-cut algorithm, even without cuts, obtains much tighter lower bounds (i.e., smaller integrality gaps) than the branch-and-cut algorithm. It actually obtains stronger lower bounds

Table 1 Average Integrality Gap (in Percentage) by Group

	B&C				B&P with CIBC				B&P				BP&C			
Instance class	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$
H3	3.28	5.95	8.85	10.88	1.60	2.24	3.53	4.15	1.23	1.81	3.06	3.68	0.49	0.80	1.64	2.17
L3	8.18	14.10	20.22	23.64	3.83	5.60	8.65	9.28	2.99	4.62	7.66	8.35	1.21	2.23	4.56	5.25
H6	6.29	10.59	15.05	20.50	4.37	3.48	6.32	10.32	3.43	2.47	5.05	9.45	3.30	2.12	4.61	8.85
L6	11.10	19.25	24.57	29.94	3.04	5.94	8.82	13.02	2.37	5.05	7.85	12.25	2.12	4.54	7.22	11.35
Average	7.21	12.47	17.17	21.24	3.21	4.31	6.83	9.19	2.50	3.49	5.91	8.43	1.78	2.42	4.51	6.91

for all groups of instances. This clearly shows that the new formulation is tighter than the arc-flow model used by the branch-and-cut algorithm. Nevertheless, the times required to compute the lower bounds (not reported here) are larger for the branch-price-and-cut algorithm, especially when the number of vehicles is low. Consequently, better bounds do not necessarily yield optimal solutions in less time. From these results, we also observe that, for all algorithms, the average gap increases with the number of vehicles and with the number of periods, showing that the size of the search tree to explore increases, in general, with the number of vehicles and the number of periods. Note also that the average gap is larger for the L3 and L6 instances compared to the H3 and H6 instances. This is due to the fact that the holding costs are 10 times higher in the latter instances, accounting for 40% to 75% of the total cost instead of 5% to 10%. In fact, we observe that using low or high holding costs for a given instance results in approximately the same absolute integrality gap.

To assess the effectiveness of the cuts, we compute for each instance group the average integrality gap (in percentage) closed by the cuts in the branch-price-and-cut algorithm, which is equal to $(\underline{z}^c - \underline{z})/(\bar{z} - \underline{z})$, where $\underline{z}^c$ and $\underline{z}$ are the lower bounds at the root node with and without the cuts, respectively. These averages (not presented here) show that, for the H3 and L3 instances, the cuts have a significant impact on the lower bounds as they close on average near 50% of the gap. For the H6 and L6 instances, only around 10% of the gap is closed by the cuts. We believe that this statistic is biased by the fact that several upper bounds used for these instances do not correspond to optimal values and may, thus, yield largely overestimated optimality gaps. To investigate this hypothesis, we analyzed the effect of the cuts only on the instances for which the optimal value is known. The results from this analysis are given in Figure 3. They show that the effect of the cuts is much more consistent, in particular for the instances with four and five vehicles (for several large instances with two and three vehicles, the algorithm was stopped at the time limit before generating all possible cuts in the root node). On average, the cuts reduce the optimality gap by 44%, 56%, 70%, and 83% for the instances with two, three, four, and five vehicles, respectively.

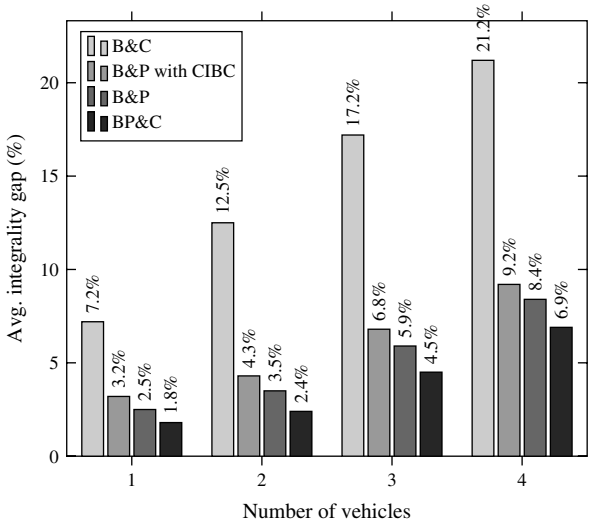


Figure 2 Average Integrality Gap by Number of Vehicles

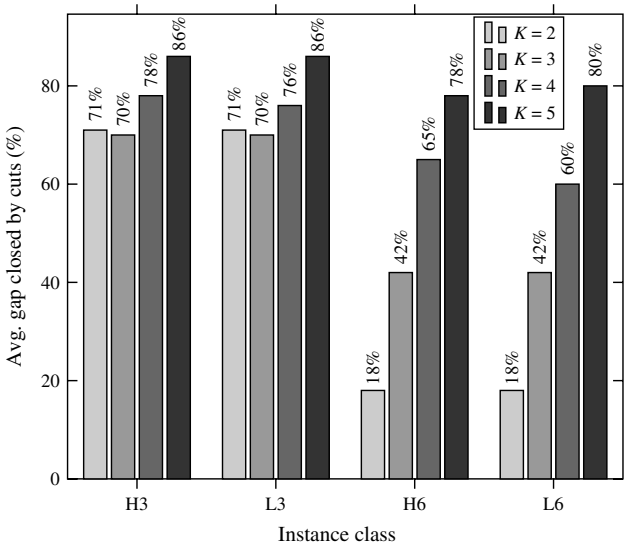


Figure 3 Average Integrality Gap Closed by the Cuts in the Branch-Price-and-Cut Algorithm (Instances with Known Optimal Value)

4.2. Integer Solution Results

With our branch-price-and-cut algorithm and the branch-and-cut algorithm of Coelho and Laporte (2014), we tried to solve to optimality all instances considering the integrality requirements and a two-hour time limit. The results of these experiments are shown by instance group in Table 2. From these results, we observe that the branch-price-and-cut algorithm is less effective than the branch-and-cut algorithm for solving instances with a small number of vehicles. In these instances, the vehicle capacity is large and routes can contain a large number of customer visits. Thus, the subproblems are hard to solve because the number of labels generated in the labeling algorithm is huge. On the other hand, the branch-price-and-cut algorithm provides optimal solutions for more than half of the instances with four and five vehicles, significantly outperforming the branch-and-cut algorithm. These results clearly highlight the negative impact that an increase in the number of vehicles has on the performance of the branch-and-cut algorithm. This impact is slightly positive for the branch-price-and-cut algorithm. We also observe that the performance of the branch-price-and-cut algorithm (in terms of the number of instances solved) does not depend on the magnitude of the holding costs despite the fact that the relative integrality gaps are much larger for the H3 and H6 instances than for the L3 and L6 instances. As mentioned above, the absolute integrality gaps are similar for both types of instances and this statistic seems to be the most determining factor. Finally, we remark that, for both algorithms, the instances with six periods are much more difficult to solve than those with three periods.

Next, we compare the computational times of both algorithms. To get a fair comparison, we analyze the results only for the instances solved to optimality by both of them. Table 3 reports the average computational time per instance group and Figure 4 depicts the average by number of vehicles. Here, we observe that the branch-and-cut algorithm is significantly faster on the two-vehicle instances, and the branch-price-and-cut algorithm provides the least average computational times for the instances with four and five vehicles. For the three-vehicle instances, the branch-and-cut algorithm is on average only 12% faster than the branch-price-and-cut algorithm. These results are consistent with the observations made when examining the number of instances solved to optimality by the different methods. The time spent solving the subproblems is the main reason for the poor performance of the branch-price-and-cut algorithm on the two-vehicle instances, while the performance of the branch-and-cut algorithm deteriorates with the number of vehicles because of a decrease in the quality of the lower bounds and an increase in the number of variables.

In Table 4, we provide additional statistics concerning the solution process of the branch-price-and-cut algorithm. These statistics correspond to averages by instance group computed over the instances solved to optimality within the time limit. They are the percentage of the total time spent solving the subproblems (ST), the number of nodes explored in the branch-and-bound search tree (BBn), the number of MNSD cuts (24) generated (C24), and the number of capacity cuts (25) generated (C25). These results indicate that the proportion of the time dedicated to solving the subproblems is much larger for the short-horizon H3 and L3 instances. On one hand, doubling the horizon length also doubles the number of constraints in the RMP and, thus, the time for solving it. On the other hand, increasing the number of periods and, consequently, the number of subproblems also increases the subproblem

Table 2 Number of Instances Solved to Optimality by Instance Group

	B&C					BP&C				
Instance class	$K=2$	$K=3$	$K=4$	$K=5$	Total	$K=2$	$K=3$	$K=4$	$K=5$	Total
H3	48	37	22	19	126	29	34	33	40	136
L3	48	38	22	18	126	28	28	34	37	127
H6	21	8	5	4	38	9	8	9	7	33
L6	21	8	5	4	38	9	7	8	6	30
Total	138	91	54	45	328	75	77	84	90	326

Table 3 Average Computational Time by Instance Group (Instances Solved to Optimality by Both Algorithms)

	B&C				BP&C			
Instance class	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$
H3	32.1	714.8	801.9	1,271.2	667.7	1,061.6	67.2	135.9
L3	29.1	482.2	1,052.1	1,098.4	450.7	455.9	222.4	308.2
H6	68.0	329.7	37.6	100.0	451.8	307.5	110.2	79.6
L6	149.4	773.9	128.4	158.2	1,229.5	776.6	527.3	72.5
Average	69.7	575.1	505.0	657.0	699.9	650.4	231.8	149.0

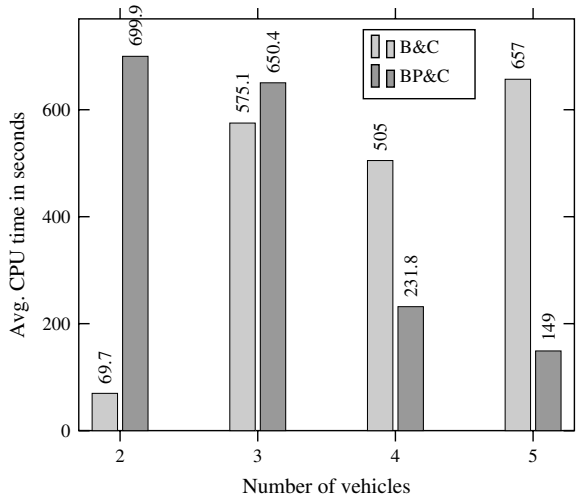


Figure 4 **Average Computational Time by Number of Vehicles (Instances Solved to Optimality by Both Algorithms)**

time. However, given that only small-sized H6 and L6 instances can be solved to optimality, the latter increase is not as important as the increase of the RMP time. We also observe that the percentage of time devoted to the subproblems slightly decreases with the number of vehicles. This decrease is due to the subproblems that are easier to solve for the instances with a larger number of vehicles because vehicle capacity and, thus, the number of customers per route decrease with the number of vehicles in the benchmark instances.

The results in Table 4 also show that the number of branch-and-bound nodes increases with the horizon length because of larger gaps. Although it decreases on average with the number of vehicles starting from three vehicles, this observation is biased by the results for the few L6 instances solved and we cannot establish a general tendency. The number of capacity cuts generated is positively correlated with the number of branch-and-bound nodes. However, the ratio of the number of generated cuts over the number of branch-and-bound nodes increases with the number of vehicles. Finally, the results show that, for all H6 and L6 instances, the average number of MNSD cuts generated is small compared to the average number of capacity cuts generated.

To conclude this section, we report in Table 5 the number of instances for which we proved optimality for the first time. Prior to this paper, a total of 238 instances (out of 640) were still unsolved to optimality. We succeeded to close 54 of them, all for instances with four or five vehicles. This is quite impressive given that, in previous works (Adulyasak, Cordeau, and Jans 2015; Coelho and Laporte 2013a, 2014), parallel branch-and-cut implementations using six or eight threads were used and, in some cases, higher time limits (6 or 12 hours) were applied.

Table 5 **Number of New Optimal Solutions Found by the Branch-Price-and-Cut Algorithm**

	$K=2$	$K=3$	$K=4$	$K=5$	Total
Open	5	46	80	107	238
Closed	0	0	16	38	54

4.3. Sensitivity Analysis Results

In this section, we provide a sensitivity analysis of the performance of the branch-price-and-cut algorithm with respect to several of its components. These components and their acronyms are the following:

MnsdI: Inequalities on the minimum number of subdeliveries per demand (24) (see §3.2.3).

CapI: Capacity inequalities (25) (see §3.2.4).

CapRBS: Route-based separation heuristic for the capacity inequalities (see §3.2.4).

Tabu: Tabu search column generator (see §3.1.3).

Relax: Relaxation of constraints (5) (see §3.1.3).

CPfix: Branching on the flow through each customer vertex in each period (see §3.3).

Ldepth: Local depth-first strategy to explore the search tree (see §3.3).

The other components have not been considered in this analysis either because their effectiveness is obvious or already established (this is the case for the first three acceleration techniques described in §3.1.3) or because they are not often called and, therefore, have limited impact (this is the case for the inequalities in §§3.2.1 and 3.2.2 or the first two types of branching decisions presented in §3.3).

To perform this sensitivity analysis, we ran computational tests on a selected subset of instances that are not trivial to solve with the proposed branch-price-and-cut algorithm (computational times varying between 578 and 5,718 seconds). This subset contains 18 instances in

Table 4 **Additional Statistics by Instance Group for the Branch-Price-and-Cut Algorithm**

	$K=2$				$K=3$				$K=4$				$K=5$			
Instance class	ST (%)	BBn	C24	C25	ST (%)	BBn	C24	C25	ST (%)	BBn	C24	C25	ST (%)	BBn	C24	C25
H3	91.2	23.0	—	16.8	90.3	268.6	—	78.1	82.7	279.0	—	106.1	79.1	387.4	—	159.0
L3	88.3	25.9	—	26.8	86.7	455.7	—	96.9	79.8	260.7	—	144.7	80.7	804.4	—	271.2
H6	68.9	1,470.6	20.1	150.8	46.9	26,115.8	31.5	3,105.4	47.9	10,150.6	23.0	1,855.6	35.7	18,821.9	21.4	5,870.0
L6	64.1	3,600.6	17.8	294.7	41.9	51,202.7	26.3	5,855.7	35.7	43,416.9	23.0	8,328.5	32.1	24,365.8	14.3	8,596.7
Average	78.1	1,280.0	18.9	122.3	64.4	19,510.7	28.9	2,284.0	61.5	13,526.8	23.0	2,608.7	56.9	11,094.9	17.9	3,724.2

Table 6 Analysis of the Effectiveness of Different Algorithm Components (Time in Seconds)

Instance	Base time	+MnsdI time	−MnsdI time	−CapI time	−CapRBS time	−Tabu time	−Relax time	−CPfix time	−Ldepth time
H3_4n20_k3	3,803.4	2,569.2	N/A	7,200.0	3,782.9	7,200.0	3,801.3	7,200.0	6,060.9
H3_2n30_k3	3,326.1	4,758.6	N/A	7,200.0	4,020.8	7,200.0	3,458.2	2,205.5	5,980.2
H3_3n40_k3	769.0	1,676.5	N/A	7,200.0	799.0	5,120.2	737.4	683.7	1,644.5
L3_5n15_k3	2,572.4	3,741.9	N/A	3,334.6	1,565.3	5,044.2	2,519.2	7,200.0	2,876.1
L3_3n35_k3	920.8	1,881.4	N/A	7,200.0	7,200.0	7,200.0	2,260.0	902.4	1,563.5
H3_5n15_k5	1,784.0	1,895.3	N/A	7,200.0	1,273.5	1,614.0	1,920.2	1,996.5	2,836.4
H3_3n35_k5	1,284.0	1,874.8	N/A	7,200.0	1,036.4	4,649.0	1,614.2	1,040.5	1,467.7
L3_4n20_k5	1,881.9	3,965.0	N/A	7,200.0	1,699.3	2,131.1	2,177.9	7,200.0	2,382.6
L3_5n30_k5	904.6	637.6	N/A	7,200.0	650.4	2,513.8	1,016.0	999.9	1,047.4
L3_3n45_k5	1,428.5	1,599.5	N/A	7,200.0	1,834.6	7,200.0	1,581.8	1,097.0	3,114.1
H6_2n5_k3	1,683.1	N/A	2,086.8	2,053.7	1,289.7	1,991.2	2,517.6	7,200.0	2,842.3
H6_1n10_k3	5,718.5	N/A	6,455.4	7,200.0	3,744.6	7,200.0	3,754.1	7,200.0	7,200.0
L6_5n5_k3	749.2	N/A	785.5	920.6	576.2	764.2	788.4	1,844.2	1,051.9
L6_3n10_k3	578.3	N/A	524.8	525.8	599.1	1,060.9	715.2	751.5	924.3
H6_4n10_k5	4,532.8	N/A	5,696.7	7,200.0	7,200.0	5,971.3	7,200.0	7,200.0	7,200.0
H6_5n10_k5	2,059.6	N/A	1,824.3	2,379.1	1,708.9	2,086.4	2,793.1	7,200.0	3,859.3
L6_3n10_k5	3,796.4	N/A	4,269.9	7,200.0	7,200.0	3,402.9	4,364.6	2,732.6	7,200.0
L6_5n10_k5	5,257.8	N/A	5,566.4	7,200.0	4,919.4	5,755.7	7,200.0	7,200.0	7,200.0
Average	2,391.7	2,720.9	2,549.1	>5,711.9	>2,838.9	>4,339.2	>2,801.1	>3,991.8	>3,691.8

total. With respect to the number of vehicles, the holding cost magnitude, and the number of periods, they are distributed as follows: nine instances with three vehicles and nine with five vehicles; nine instances with high holding costs and nine with low holding costs; 10 with three periods and eight with six periods. The name of an instance has the form CC_SnX_kY, where CC indicates the class, S is a seed number (from one to five), X is the number of customers, and Y is the number of vehicles. For example, H3_4n20_k3 is the name of the fourth instance with 20 customers and three vehicles belonging to class H3.

Table 6 reports the computational times in seconds for all instances and all algorithm variants tested. The column "Base" corresponds to the algorithm used to obtain the results discussed in the previous section. Recall that it applies the inequalities on the minimum number of subdeliveries per demand (24) only for the instances with six periods. The remaining columns show the computational times when removing (−) or adding (+) a given feature to the base algorithm. An N/A entry in columns +MnsdI and −MnsdI indicates that this component is already considered or not considered in the base algorithm for the corresponding instance. The last row of the table gives the average time for each method. For the columns +MnsdI and −MnsdI, the averages are computed by replacing every N/A entry with the corresponding one from the base algorithm.

From the results, we observe that all features have a positive impact on the average computational time for the tested instances, that is, the base algorithm can solve all of the instances within the 7,200-second time limit and produces the least average computational time (2,391.7 seconds). The most important feature in this comparison is the capacity inequalities (CapI). Without them, 13 of the 18 instances cannot be solved within the time limit and the computational time increases for four of the five other instances. The second most important feature in terms of number of solved instances is branching on the flow through each customer vertex in each period (CPfix). Without this component, the time limit is reached for eight instances. In terms of average computational time, the tabu search column generator (Tabu) is the second most important feature. On the other hand, using or not using the MnsdI component for all instances does not change the results. In fact, all instances can be solved to optimality within the time limit and the average computational time increases by only 13.7% and 6.6%, respectively. For all of the other components, we observe that some instances cannot be solved within two hours of computational time.

5. Conclusion

We have introduced an innovative formulation for the IRP that specifies for each delivery in which time period(s) the delivered quantity should be consumed. We have also developed a state-of-the-art branch-price-and-cut algorithm that incorporates known and new families of valid inequalities, an ad hoc labeling algorithm for solving the column generation subproblems, and several acceleration techniques. In particular, we proposed an adaptation of the well-known capacity inequalities that proved to be a very efficient component of our algorithm.

The reported computational results show that our algorithm outperforms existing exact algorithms for instances with more than three vehicles, and that our formulation provides much smaller integrality gaps.

We solved to optimality 54 previously open instances from a large instance set widely used, even though we use less computational resources than other papers. We have shown that the proposed valid inequalities, branching decisions, and other speed up strategies are effective, and in most cases necessary to solve some instances.

For future works, we suggest three different avenues. First, to improve the proposed branch-price-and-cut algorithm and solve larger instances, new families of valid inequalities that relate the quantity delivered to a customer in a period with the minimum number of visits required in this period and the subsequent ones could yield improved lower bounds. Such inequalities have been proposed by Archetti et al. (2007), but stronger ones involving only binary coefficients may be devised in a column generation framework where the subproblems determine the quantity delivered to each visited customer. Second, as discussed in §3.4, one can adapt our branch-price-and-cut algorithm to tackle several IRP variants, the most challenging one being the multiproduct IRP. Finally, it might be interesting to try to integrate the proposed capacity inequalities into a branch-and-cut algorithm.

Acknowledgments

The authors thank two anonymous reviewers whose valuable comments helped improve this paper. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada [Grants 157935-2012 and 2014-05764], and the Research Council of Norway through the MARFLIX project. This support is greatly appreciated.

References

Adulyasak Y, Cordeau J-F, Jans R (2014) Formulations and branch-and-cut algorithms for multi-vehicle production and inventory routing problems. *INFORMS J. Comput.* 26(1):103–120.

Adulyasak Y, Cordeau J-F, Jans R (2015) The production routing problem: A review of formulations and solution algorithms. *Comput. Oper. Res.* 55:141–152.

Andersson H, Hoff A, Christiansen M, Hasle G, Løkketangen A (2010) Industrial aspects and literature survey: Combined inventory management and routing. *Comput. Oper. Res.* 37(9): 1515–1536.

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

Archetti C, Bertazzi L, Hertz A, Speranza MG (2012) A hybrid heuristic for an inventory routing problem. *INFORMS J. Comput.* 24(1):101–116.

Archetti C, Bertazzi L, Laporte G, Speranza MG (2007) A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Sci.* 41(3):382–391.

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

Bard JF, Nananukul N (2010) A branch-and-price algorithm for an integrated production and inventory routing problem. *Comput. Oper. Res.* 37(12):2202–2217.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 46(3):316–329.

Bertazzi L, Paletta G, Speranza MG (2002) Deterministic order-up-to level policies in an inventory routing problem. *Transportation Sci.* 36(1):119–132.

Brahimi N, Dauzere-Peres S, Najid NM, Nordli A (2006) Single item lot sizing problems. *Eur. J. Oper. Res.* 168(1):1–16.

Coelho LC, Laporte G (2013a) The exact solution of several classes of inventory-routing problems. *Comput. Oper. Res.* 40(2):558–565.

Coelho LC, Laporte G (2013b) A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *Internat. J. Production Res.* 51(23–24):7156–7169.

Coelho LC, Laporte G (2014) Improved solutions for inventory-routing problems through valid inequalities and input ordering. *Internat. J. Production Econom.* 155:391–397.

Coelho LC, Cordeau J-F, Laporte G (2012a) The inventory-routing problem with transshipment. *Comput. Oper. Res.* 39(11): 2537–2548.

Coelho LC, Cordeau J-F, Laporte G (2012b) Consistency in multi-vehicle inventory-routing. *Transportation Res. Part C: Emerging Tech.* 24(1):270–287.

Coelho LC, Cordeau J-F, Laporte G (2014) Thirty years of inventory-routing. *Transportation Sci.* 48(1):1–19.

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

Desaulniers G, Desrosiers J, Solomon MM (2005) *Column Generation* (Springer, New York).

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

Desaulniers G, Madsen OBG, Røpke S (2014) Vehicle routing problems with time windows. Toth P, Vigo D, eds. *Vehicle routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization (SIAM, Philadelphia), 119–159.

Dror M, Ball MO, Golden BL (1985) A computational comparison of algorithms for the inventory routing problem. *Ann. Oper. Res.* 4(1):3–23.

Engineer FG, Furman KC, Nemhauser GL, Savelsbergh MWP, Song J-H (2012) A branch-and-price-and-cut algorithm for single-product maritime inventory routing. *Oper. Res.* 60(1): 106–122.

Grønhaug R, Christiansen M, Desaulniers G, Desrosiers J (2010) A branch-and-price method for a liquefied natural gas inventory routing problem. *Transportation Sci.* 44(3):400–415.

Hewitt M, Nemhauser GL, Savelsbergh MWP, Song J-H (2013) A branch-and-price guided search approach to maritime inventory routing. *Comput. Oper. Res.* 40(5):1410–1419.

Krarup J, Bilde O (1977) Plant location, set covering, and economic lot size: An O(*mn*)-algorithm for structured problems. Collatz L, Meinardus G, Wetterling W, eds. *Numerische Methoden bei Optimierungsverfahren, Band 3*, Internat. Series Numerical Math., Vol. 36 (Birkhäuser Verlag, Basel, Switzerland), 155–180.

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

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper. Res.* 53(6):1007–1023.

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

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

Solyalı O, Süral H (2011) A branch-and-cut algorithm using a strong formulation and an a priori tour based heuristic for an inventory-routing problem. *Transportation Sci.* 45(3):335–345.