Les Cahiers du GERAD ISSN: 0711–2440

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

G. Desaulniers, J.G. Rakke,
L.C. Coelho

G–2014–19

April 2014

Les textes publiés dans la série des rapports de recherche HEC n'engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d'une subvention du Fonds de recherche du Québec – Nature et technologies.

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

Guy Desaulniers

GERAD & Department of Mathematics and Industrial Engineering
Polytechnique Montréal
Montréal (Québec) Canada, H3C 3A7

guy.desaulniers@gerad.ca

Jørgen G. Rakke

Department of Marine Technology
The Norwegian University of Science and Technology
Trondheim, Norway

jorgen.rakke@ntnu.no

Leandro C. Coelho

CIRRELT and Faculté des sciences de l'administration
Université Laval
Québec (Québec) Canada, G1V 0A6

leandro.coelho@cirrelt.ca

April 2014

Les Cahiers du GERAD

G–2014–19

Copyright © 2014 GERAD

Abstract: 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 multi-period 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 49 of them.

Key Words: Inventory-routing, branch-price-and-cut, capacity inequalities, labeling algorithm, exact algorithm.

Acknowledgments: This work was partly supported by the Natural Sciences and Engineering Research Council of Canada and the Research Council of Norway through the MARFLIX project. This support was greatly appreciated.

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 et al. 2014b, Andersson et al. 2010).

In order to operate such an integrated strategy, the supplier has to solve an inventory-routing problem (IRP), which combines two well-known problems over a multi-period 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 et al. (1985) as a way to simplify the search for good solutions and has since been widely studied (e.g., in Bertazzi et al. (2002), Archetti et al. (2007), Solyalı and Süral (2011), and Coelho et al. (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- and multi-vehicle IRPs under various replenishment policies. For the single-vehicle version of the problem, heuristic algorithms include the fast local search of Bertazzi et al. (2002), the hybrid of mathematical programming and tabu search of Archetti et al. (2012), and the adaptive large neighborhood search (ALNS) of Coelho et al. (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 multi-vehicle instances are more recent, but the literature is quickly expanding. These include the ALNS heuristic of Coelho et al. (2012b), and the exact branch-and-cut algorithms of Adulyasak et al. (2014a) and Coelho and Laporte (2013, 2014). All 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 et al. (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, the production and the 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, while 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 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 et al. 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 Section 2 we provide a formal definition and a mathematical model for the problem under study. In Section 3 we describe the proposed branch-price-and-cut algorithm. The results of extensive computational experiments are detailed in Section 4, followed by conclusions in Section 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 $I_i^0 \leq C_i$ ($I_0^0 \leq C_0$), 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 sub-deliveries, one for each demand it can cover or for the end inventory. To limit the number of potential sub-deliveries to a customer, we exploit the fact that there always exists an optimal solution 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 $P_{ip}^+ = \{s \in \{p+1, p+2, \ldots, \rho+1\} \mid (s \in P, \bar{d}_i^s > 0 \text{ and } \sum_{\ell=p}^{s} d_i^\ell \leq C_i) \text{ or } (s = \rho+1 \text{ and } \sum_{\ell=p}^{s} d_i^\ell < C_i)\}$ associated with the sub-deliveries of a delivery to customer i in period p. For reasons of conciseness, we assume that $P_{ip}^+ \neq \emptyset$. 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 and N_r the set of customers visited in route $r \in R$. 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 sub-delivery* if $q_{wi}^s = 0$, a *full sub-delivery* if $q_{wi}^s = u_{wi}^s$, and a *partial sub-delivery* otherwise. RDP w is said to be an *extreme RDP* if it contains at most one partial sub-delivery. 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 $s \in P_{ip}^+$. Consequently, we can also evaluate the total holding cost incurred by the deliveries in this RDP. Let c_{rw} be the sum of the travel costs and holding costs associated with route r and RDP w. The travel costs are computed as the sum of the costs c_{ij} incurred by traveling between each pair of locations i and j (in $N \cup \{0\}$) visited consecutively along route r. Finally, denote by $P_{is}^- = \{p \in P \mid s \in P_{ip}^+\}$ the set of periods at which a sub-delivery 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 non-negative 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:

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

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

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

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

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

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

$$0 \leq I_0^p \leq C_0, \qquad \forall p \in P, \tag{7}$$

$$y^p_{rw} \geq 0, \qquad \forall p \in P, r \in R, w \in W^p_r, \tag{8}$$

$$\sum_{w \in W^p_r} y^p_{rw} \in \{0,1\}, \qquad \forall p \in P, r \in R. \tag{9}$$

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 is reached just before consumption after a possible delivery, justifying the subtraction of d^s_i in the right-hand side term. Constraints (5) and (6) ensure that, in each period, every customer is visited at most once and no more than K vehicles are used. Non-negativity 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^p_{rw} 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 non-empty. Note that $\sigma_{i,p-1} \leq \sigma_{ip}$ for all customers $i \in N$ and periods $p \in P \setminus \{1\}$.

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^s_i, can come from the initial inventory or from deliveries made in period s or before, that is, $I^s_i = I^{0,s}_i + \sum_{p \in P^-_{is}} \sum_{r \in R} \sum_{w \in W^p_r} b^s_i y^p_{rw}$. First, observe that if $\sigma_{is} = s$, then I^s_i 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^s_i is dedicated to fulfill the demands (or the end inventory) of the periods in $\mathcal{L} = \{\ell \in P \mid s < \ell \leq \sigma_{is}\}$. Let q_ℓ be the quantity dedicated to period $\ell \in \mathcal{L}$. Consequently $I^s_i = \sum_{\ell \in \mathcal{L}} q_\ell$. For $\ell \in \mathcal{L} \setminus \{\sigma_{is}\}$, we get that

$$\begin{array}{ll} q_\ell = d^\ell_i & \text{if } I^{0,\ell} > 0 \\ q_\ell \leq \bar{d}^\ell_i = d^\ell_i & \text{if } I^{0,\ell-1} = 0 \\ q_\ell \leq I^{0,\ell-1}_i + \bar{d}^\ell_i = d^\ell_i & \text{otherwise.} \end{array}$$

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 $P^+_{is} \neq \emptyset$. 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 $u^{\sigma_{is}}_{is}$, which in turn is less than or equal to $C_i - \sum_{j=s}^{\sigma_{is}-1} d^j_i - I^{0,\sigma_{is}-1}_i$. Consequently, $q_{\sigma_{is}} \leq I^{0,\sigma_{is}-1}_i + C_i - \sum_{j=s}^{\sigma_{is}-1} d^j_i - I^{0,\sigma_{is}-1}_i = C_i - \sum_{j=s}^{\sigma_{is}-1} d^j_i$.

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

$$I^s_i = \sum_{\ell \in \mathcal{L} \setminus \{\sigma_{is}\}} q_\ell + q_{\sigma_{is}} \leq \sum_{\ell \in \mathcal{L} \setminus \{\sigma_{is}\}} d^\ell_i + C_i - \sum_{j=s}^{\sigma_{is}-1} d^j_i = C_i - d^s_i,$$

which completes the proof. □

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 et al. (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.

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 Sections 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.

Below, 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:

$$\bar{c}^p_{rw} = c_{rw} + q_w \pi^2_p - \sum_{i \in N_r} \sum_{s \in P \mid \bar{d}^s_i > 0} q^s_{wi} \pi^3_{is} - \sum_{i \in N_r} \sum_{s \in P} b^s_{wi} \pi^4_{is} - \sum_{i \in N_r} \pi^5_{ip} - \pi^6_p.$$

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

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi^6_p & \text{if } i = v^S \\ c_{ij} - \pi^5_{ip} & \text{otherwise,} \end{cases} \tag{10}$$

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 sub-delivery 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 $i \in N \setminus N_r$ 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:

$$\bar{c}^p_{rw} = \sum_{(i,j) \in A_r} \bar{c}_{ij} + \sum_{i \in N_r} \sum_{s \in P^+_{ip}} \xi^s_i \big(\pi^2_p - \pi^3_{is} - \sum_{t \in P^+_{ip} \mid p < t < s} \pi^4_{it}\big), \tag{11}$$

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 sub-delivery), yielding an extreme RDP in W^p_r. 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 next.

3.1.2 Labeling algorithm

Consider first an ESPPRC defined on network $G^p = (V^p, A^p)$. In a labeling algorithm for this problem, a partial path (route) from the source vertex v^S to a vertex in V^p is represented by a vector called a *label*. Starting from an initial label at the source vertex v^S, this algorithm enumerates partial paths by extending this label forwardly in G^p. The extension of a label E_i representing a path ending in vertex $i \in V^p$ along an arc $(i, j) \in A^p$ yields a new label E_j representing a path ending in vertex j that may or may not be feasible. This extension is performed using extension functions that allow to compute each component of label E_j. To avoid enumerating all feasible paths, a dominance rule is applied to compare the labels associated with the same vertex, discarding labels for which it can be proven that they 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 sub-delivery, the quantity delivered in this sub-delivery is unknown. It is only revealed when a complete path reaching the sink vertex v^E is reached. Information about the sub-delivery 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 $i \in N^p$ and $w \in W^r_p$ an associated RDP. The label, denoted $E_i = (T^{cost}_i, T^{loadF}_i, (T^{cust_k}_i)_{k \in N}, T^{part}_i, T^{ratePiP}_i, T^{maxP}_i)$, representing this feasible path/RDP contains the following components:

- T^{cost}_i: The reduced cost of the path/RDP (r, w). If $i \neq v^E$, this cost omits the dual contribution from the partial sub-delivery if any.
- T^{loadF}_i: The total quantity delivered along the path r according to RDP w. If $i \neq v^E$, this quantity includes only the full sub-deliveries.
- $T^{cust_k}_i$: A binary value that indicates whether or not customer $k \in N$ has been visited along path r.
- T^{part}_i: A binary value that indicates whether or not RDP w contains a partial sub-delivery.
- $T^{ratePiP}_i$: The unit rate of contribution to the reduced cost associated with the partial sub-delivery if any.
- T^{maxP}_i: The maximum quantity that can be delivered in the partial sub-delivery 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 sub-delivery types associated with a customer $i \in N$ in period p. More precisely, for each period $s \in P_{ip}^+$, it indicates if the sub-delivery associated with s is a zero (Z), a full (F), or a partial sub-delivery (P). The latter type is admissible only if $u_{ip}^s > 1$. A CDP can contain at most one partial sub-delivery and the total quantity delivered in its full sub-deliveries 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 sub-delivery 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, ZPF 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 (resp. g_γ^s) that takes value 1 if CDP γ contains a full (resp. partial) sub-delivery for period s. With each CDP $\gamma \in \Gamma_{ip}$, we associate the following values:

$\tau_\gamma^{cost} = \sum_{s \in P_{ip}^+} f_\gamma^s u_{ip}^s \big(\pi_p^2 - \pi_{is}^3 - \sum_{t \in P_{ip}^+ \,|\, p<t<s} \pi_{it}^4\big)$ **:** The contribution to the path/RDP reduced cost (11).

$\tau_\gamma^{loadF} = \sum_{s \in P_{ip}^+} f_\gamma^s u_{ip}^s$ **:** The total quantity delivered in the full sub-deliveries.

$\tau_\gamma^{part} = \sum_{s \in P_{ip}^+} g_\gamma^s$ **:** The number of partial delivery (0 or 1).

$\tau_\gamma^{ratePiP} = \sum_{s \in P_{ip}^+} g_\gamma^s \big(\pi_p^2 - \pi_{is}^3 - \sum_{t \in P_{ip}^+ \,|\, p<t<s} \pi_{it}^4\big)$ **:** The rate of contribution to the path/RDP reduced cost (11) 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.

Starting from a null initial label at the source vertex v^S, the labeling algorithm applies extension functions to generate new labels. 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 $i \in N^p$. 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:

$$T_j^{cost} = T_i^{cost} + \bar{c}_{ij} + \tau_\gamma^{cost} \tag{12}$$

$$T_j^{loadF} = T_i^{loadF} + \tau_\gamma^{loadF} \tag{13}$$

$$T_j^{cust_k} = \begin{cases} T_i^{cust_k} + 1 & \text{if } j = k \\ T_i^{cust_k} & \text{otherwise,} \end{cases} \qquad \forall k \in N \tag{14}$$

$$T_j^{part} = T_i^{part} + \tau_\gamma^{part} \tag{15}$$

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

$$T_j^{maxP} = \begin{cases} \min\{\tau_\gamma^{maxP}, Q - T_i^{loadF} - \tau_\gamma^{loadF}\} & \text{if } \tau_\gamma^{part} = 1 \\ \min\{T_i^{maxP}, Q - T_i^{loadF} - \tau_\gamma^{loadF}\} & \text{otherwise.} \end{cases} \tag{17}$$

The resulting label E_j is declared feasible if $T_j^{loadF} \leq Q$, $T_j^{cust_k} \leq 1$ for all $k \in N$, and $T_j^{part} \leq 1$. It is discarded when not feasible.

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:

$$T_j^{cost} = \begin{cases} T_i^{cost} + \bar{c}_{ij} + T_i^{maxP}\, T_i^{ratePiP} & \text{if } T_i^{ratePiP} < 0 \\ T_i^{cost} + \bar{c}_{ij} & \text{otherwise.} \end{cases} \tag{18}$$

The path/RDP represented by a label associated with the sink vertex v^E can be retrieved by keeping in every label an additional component that indicates the label used to generate it.

To eliminate non-promising labels, a dominance rule is applied to compare labels associated with the same vertex. Given that the quantity to be delivered in a partial sub-delivery remains unknown until

reaching the sink vertex v^E, this dominance rule must take into account all possible reduced cost that can be achieved with this sub-delivery if any. The reduced cost of a label can thus be seen as a function of the quantity that can be delivered in this sub-delivery. 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 sub-delivery is:

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

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:*

1. $T_1^{loadF} \leq T_2^{loadF}$;
2. $T_1^{cust_k} \leq T_2^{cust_k}$, $\forall k \in N$;
3. $T_1^{part} \leq T_2^{part}$;
4. $T_1^{cost} - T_1^{maxP} T_1^{ratePiP} \leq T_2^{cost} - T_2^{maxP} T_2^{ratePiP}$;
5. $T_1^{cost} - (T_2^{loadF} - T_1^{loadF}) T_1^{ratePiP} \leq T_2^{cost}$;
6. $T_1^{cost} - (T_2^{loadF} + T_2^{maxP} - T_1^{loadF}) T_1^{ratePiP} \leq T_2^{cost} - T_2^{maxP} T_2^{ratePiP}$.

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 1 to 6 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 the condition 2 that 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. These paths, if generated, are eliminated when cuts or branching decisions are applied. On the one hand, solving relaxed subproblems requires less computational effort. On the other hand, generating cyclic paths weakens the lower bounds, resulting in a larger search tree. Therefore, a compromise between the strength of the relaxation and the difficulty of solving the relaxed subproblems must be achieved.

Recently, Baldacci et al. (2011) introduced the *ng*-path relaxation that allows to reach such a compromise. An *ng*-path is a path that may contain some cycles if these cycles respect a condition defined with respect to neighborhoods. 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 $v_1 - v_2 - \ldots - v_h$ 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 et al. (2014).

Bidirectional labeling: To further speed up the solution of the subproblems, Righini and Salani (2006) proposed to use bidirectional labeling. This algorithm proceeds in three steps. First, starting with an initial label at the source node of a network, labels are extended forwardly until either reaching the sink node or delivering half of the vehicle capacity (without taking into account the quantity delivered in partial deliveries). Second, starting with an initial label at the sink node, labels are extended backwardly until either reaching the source node or delivering half of the vehicle capacity. Third, forward and backward labels associated with the same node are merged to yield complete source-to-sink paths which must be checked to ensure their feasibility.

Heuristic column generators: There is no need to solve to optimality the subproblems at each column generation iteration. Indeed, one or several (fast) heuristics can be used to find negative reduced cost columns and, when none can be found, then the exact labeling algorithm must be invoked to verify that none exists. In this spirit, we use two following heuristic column generators.

The first heuristic is a multi-start tabu search algorithm similar to the one proposed in Archetti et al. (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 to 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 (5 for our tests). Tabu search is stopped when reaching a maximum number of iterations (15 for our tests). As proposed in Desaulniers et al. (2008), the tabu search algorithm is executed on each route associated with a basic variable in the current RMP solution. These routes are good starting points because their associated variables have a zero reduced cost.

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 et al. (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.

Consequently, three algorithms are available to generate columns: tabu search, heuristic labeling, and exact labeling. These column generators are used as proposed in Desaulniers et al. (2008). At each column generation iteration, the multi-start 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 re-introduced 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. Four families of valid inequalities are considered: minimum number of visits per customer, minimum number of routes per time interval, minimum number of sub-deliveries per demand, and capacity inequalities. Below, we present these four families 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:

$$\sum_{p=1}^{\ell} \sum_{r \in R} \sum_{w \in W_r^p} a_{ri} y_{rw}^p \geq lb_{i\ell}^V, \qquad \forall i \in N, \ell \in P. \tag{19}$$

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 $\pi_{i\ell}^{19}$, $i \in N$, $\ell \in P$, 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:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi_p^6 & \text{if } i = v^S \\ c_{ij} - \pi_{ip}^5 - \sum_{\ell=p}^{\rho} \pi_{i\ell}^{19} & \text{otherwise,} \end{cases} \qquad \forall (i,j) \in A^p. \tag{20}$$

Remark that, for a given customer $i \in N$, an inequality (19) for a period $\ell \in P \setminus \{1\}$ is dominated by the inequality (19) 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

Similarly 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:

$$\sum_{p=1}^{\ell} \sum_{r \in R} \sum_{w \in W_r^p} y_{rw}^p \geq lb_\ell^R, \qquad \forall \ell \in P. \tag{21}$$

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^{21}, $\ell \in P$, modify the arc reduced costs $\bar{c}_{ij}$ as follows:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi_p^6 - \pi_p^{21} & \text{if } i = v^S \\ c_{ij} - \pi_{ip}^5 & \text{otherwise,} \end{cases} \qquad \forall (i,j) \in A^p. \tag{22}$$

3.2.3 Inequalities on the minimum number of sub-deliveries per demand

Inequalities on the minimum number of deliveries per demand 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 sub-delivery of $\bar{d}_i^s$ in a period $p \in P_{is}^-$ or by performing two sub-deliveries in two periods $p_1, p_2 \in P_{is}^-$. More precisely, these inequalities write as follows:

$$\sum_{p \in P_{is}^-} \sum_{r \in R} \sum_{w \in W_r^p} (2a_{iw}^S + a_{iw}^M) y_{rw}^p \geq 2, \qquad \forall i \in N, s \in P \text{ such that } \bar{d}_i^s > 0, \tag{23}$$

where a_{iw}^S (resp. a_{iw}^M) is a binary parameter equal to 1 if $a_{ir} = 1$ and $\bar{d}_i^s$ (resp. less than $\bar{d}_i^s$) units is delivered in the sub-delivery 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^{23}_{is}, $i \in N$, $s \in P$, be the dual variables associated with the inequalities (23). To take these dual values into account in the subproblem for period $p \in P$, we modify the definition of parameters τ^{cost}_{γ}, $\gamma \in \Gamma_{ip}$, $i \in N$, as follows:

$$\tau^{cost}_{\gamma} = \sum_{s \in P^+_{ip}} \Big[f^s_{\gamma} u^s_{ip} \big(\pi^2_p - \pi^3_{is} - \sum_{t \in P^+_{ip} \mid p<t<s} \pi^4_{it} \big) - (2a^S_{is\gamma} + a^M_{is\gamma}) \pi^{23}_{is} \Big],$$

where $a^S_{is\gamma}$ (resp. $a^M_{is\gamma}$) is a binary parameter equal to 1 if $\bar{d}^s_i$ (resp. less than $\bar{d}^s_i$) units are delivered in the sub-delivery for customer i and period s in the CDP γ, and 0 otherwise.

3.2.4 Capacity inequalities

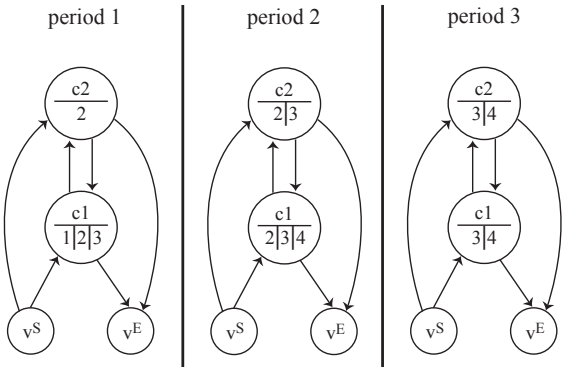
Capacity inequalities have been introduced by Laporte et al. (1985) for the capacitated vehicle routing problem (CVRP) and later used by many authors for several variants of the vehicle routing problem. In this context, these inequalities can be stated as follows. Given a subset $U \subseteq N$ of customers, 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 the best of 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 sub-deliveries 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 $RD = \{(i,s) \in N \times P \mid \bar{d}^s_i > 0\}$ 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 sub-deliveries 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 $P^+_{i'p} \cap P$.

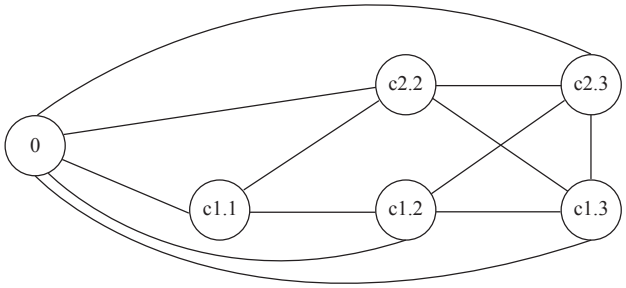
To illustrate the structure of an auxiliary graph (see Figure 1), consider an example for which $P = \{1,2,3\}$ and $N = \{c1, c2\}$. There is a positive residual demand in all periods for customer $c1$ but only in periods 2 and 3 for customer $c2$. Part (a) of Figure 1 shows the networks $G^p = (V^p, A^p)$ 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 part (b) of this figure. 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 edge is computed as the total flow on multiple arcs (e.g., the flow on the edge linking $c1.3$ and $c2.2$ is equal to the total flow on the arcs $(c1, c2) \in G^1$ and $(c1, c2) \in G^2$) or as the total flow into one or several vertices (e.g., the flow on the edge linking $c1.2$ and $c2.2$ is equal to the total flow into node $c1 \in N^1$ 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 $V^* \setminus U$, 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:

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



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



(b) Auxiliary graph G^*

Figure 1: Example of networks and their corresponding auxiliary graph

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:

$$\sum_{e\in\delta(U)} x_e = \sum_{e\in\delta(U)} \sum_{p\in P} \sum_{r\in R} \sum_{w\in W_r^p} \Big(\sum_{(i,j)\in A_e^p} a_{rij} + \sum_{i\in N_e^p} a_{ri} \Big) y_{rw}^p \geq 2\kappa(U), \qquad \forall U \subseteq RD. \tag{24}$$

Generated capacity cuts (24) are added to the master problem. Let π_U^{24} 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 $\bar{c}_{ij}$ and the CDP reduced cost contributions τ_γ^{cost} as follows:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi_p^6 - \sum\limits_{U\subseteq RD} m_{ijp}^U \pi_U^{24} & \text{if } i = v^S \\ c_{ij} - \pi_{ip}^5 - \sum\limits_{U\subseteq RD} m_{ijp}^U \pi_U^{24} & \text{otherwise,} \end{cases} \qquad \forall (i,j) \in A^p \tag{25}$$

$$\tau_\gamma^{cost} = \sum_{s\in P_{ip}^+} \Big[f_\gamma^s u_{ip}^s \big(\pi_p^2 - \pi_{is}^3 - \sum_{t\in P_{ip}^+ \,|\, p<t<s} \pi_{it}^4 \big) \Big] - \sum_{U\subseteq RD} m_{ip}^U \pi_U^{24}, \qquad \forall i \in N, \gamma \in \Gamma_{ip} \tag{26}$$

where m_{ijp}^U and m_{ip}^U are the numbers of edges in $\delta(U)$ associated with arc $(i,j) \in A^p$ and vertex $i \in N^p$, respectively.

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 et al. (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 sub-delivery. In this case, the sum of the demands associated with the partial and the full sub-deliveries in the RDP w exceeds the vehicle capacity (otherwise, the partial sub-delivery 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 (24) is violated.

Finally, we apply a route-based connected component heuristic similar to the one proposed by Archetti et al. (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 $\mathcal{C}$ be the set of connected components. Each connected component $\phi \in \mathcal{C}$ defines a subset $U_\phi \subseteq RD$ for which we verify if the corresponding inequality (24) is violated. Then, for each component $\phi \in \mathcal{C}$ 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 $\phi \in \mathcal{C}$, 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 $\mu_{rp}(U) = \sum_{e \in \delta(U)} (\sum_{(i,j) \in A^p_e} a_{rij} + \sum_{i \in N^p_e} a_{ri}) y^p_r - \sum_{i \in N_r} \sum_{s \in P^+_{ip} \mid (i,s) \in U} \bar{d}^s_i$ 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 (24) if the demands in RD_{rp} are removed from U while its second part positively influences a decrease in its right-hand side. In consequence, large values of $\mu_{rp}(U)$ favor a high decrease in the left-hand side and a low decrease in the right-hand, 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 5 routes can be removed from the initial set U_ϕ) and a maximum width of 8 (for each depth at most 8 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;
2. The number of routes in each period $p \in P$;
3. The flow through each customer vertex $i \in N$ in each period $p \in P$;
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.

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 while 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.

4 Computational experiments

The branch-price-and-cut algorithm described in the previous section 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 Concert Technology version 12.2. All tests were performed on a Intel(R) Core(TM) i7-2600 processor clocked at 3.40GHz with 8 cores and 16GB of 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 5 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, while 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 multi-vehicle 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 to 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 (Sections 4.1 and 4.2), we used the branch-price-and-cut algorithm described above for all instances except that the inequalities on the minimum number of sub-deliveries per demand (23) are not applied for the H3 and L3 instances. Preliminary tests showed that they were not useful for instances with three periods. In Section 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 Section 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. For the latter

algorithm, all cuts are enabled. 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), 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 (called hereafter 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 even without the cuts, the branch-price-and-cut algorithm obtains much tighter lower bounds (i.e., smaller integrality gaps) than the branch-and-cut algorithm. It actually obtains stronger lower bounds 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.

Figure 3 presents the average integrality gap (in percentage) closed by the cuts in the branch-price-and-cut algorithm for each instance group. These results 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

Table 1: Average integrality gap (in percentage) by group

Instance class	B&C				B&P				BP&C			
	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$	$K=2$	$K=3$	$K=4$	$K=5$
H3	3.23	5.95	8.65	10.86	1.23	1.81	3.14	4.45	0.49	0.80	1.74	2.97
L3	8.07	13.89	19.54	23.23	2.99	6.38	7.72	10.09	1.21	4.04	4.70	7.09
H6	6.29	10.59	15.05	19.82	3.43	2.47	5.05	9.13	3.30	2.12	4.61	8.55
L6	11.10	19.25	24.57	31.28	2.37	5.05	7.85	12.64	2.12	4.54	7.22	11.80
Average	7.17	12.42	16.95	21.30	2.50	3.93	5.94	9.08	1.78	2.87	4.57	7.60

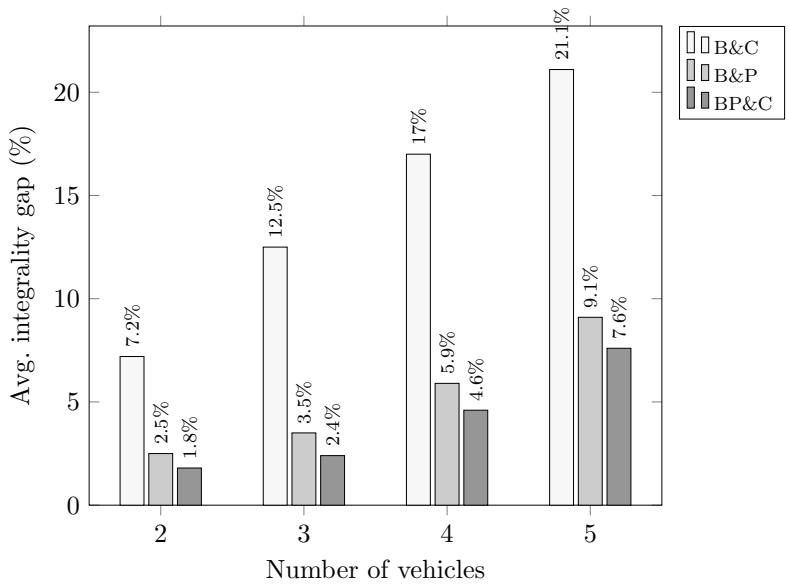


Figure 2: Average integrality gap by number of vehicles

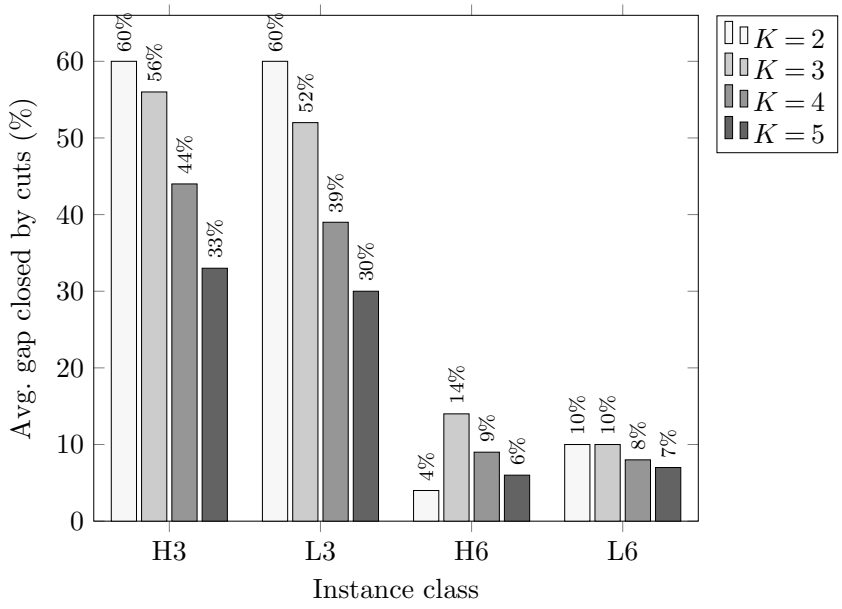


Figure 3: Average integrality gap closed by the cuts in the branch-price-and-cut algorithm (all instances)

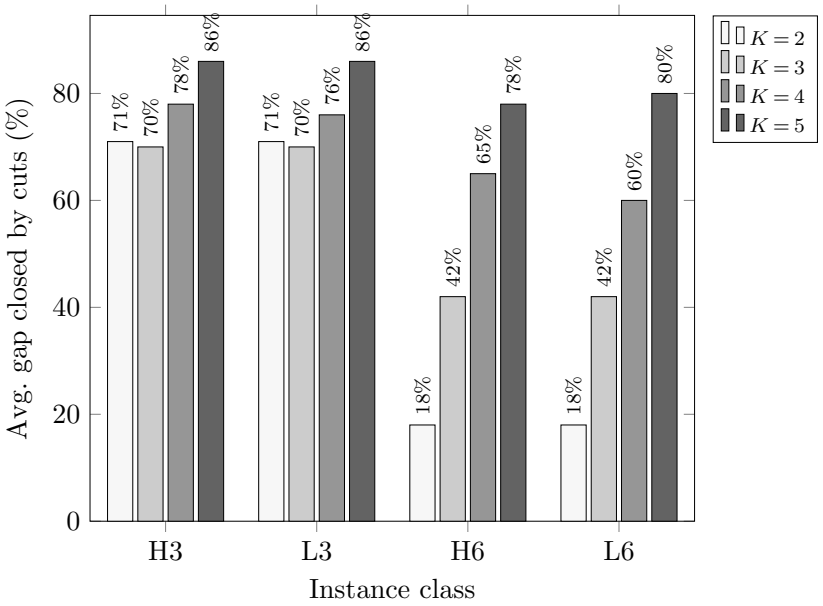


Figure 4: Average integrality gap closed by the cuts in the branch-price-and-cut algorithm (instances with known optimal value)

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. In order 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 4. 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.

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 2-hour time limit. The results of these experiments are shown by instance group in Table 2 and summarized by number of vehicles in Figure 5. 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. Figure 5 clearly highlights 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. Finally, we remark (see Table 2) 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 while Figure 6 depicts the average by number of vehicles. Here, we observe that the branch-and-cut algorithm is significantly faster on the 2-vehicle instances, while the branch-price-and-cut algorithm provides the least average computational times for the instances with four and five vehicles. For the 3-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 2-vehicle instances, while the

Table 2: Number of instances solved to optimality by instance group

Instance class	B&C					BP&C				
	$K=2$	$K=3$	$K=4$	$K=5$	Total	$K=2$	$K=3$	$K=4$	$K=5$	Total
H3	48	37	22	19	126	29	34	34	38	135
L3	48	38	22	18	126	28	28	34	35	125
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	85	86	323

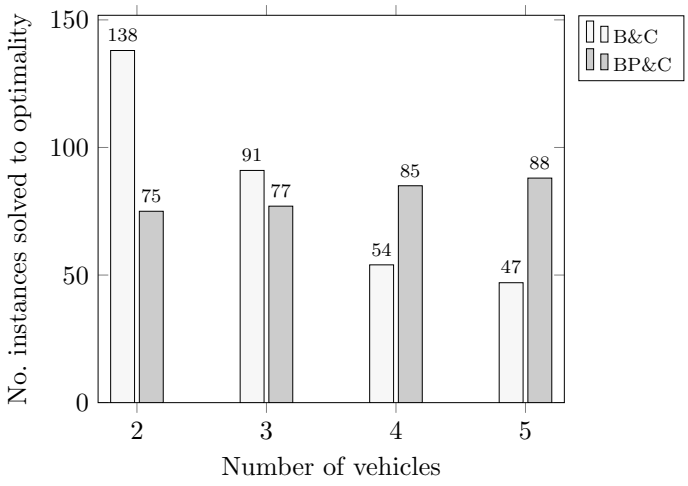


Figure 5: Number of instances solved to optimality by number of vehicles

Table 3: Average computational time by instance group (instances solved to optimality by both algorithms)

Instance class	B&C				BP&C			
	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
H3	32.1	714.8	801.9	1271.2	667.7	1061.6	67.2	135.9
L3	29.1	482.2	1052.1	1098.4	450.7	455.9	222.4	308.2
H6	68.0	329.7	37.6	80.0	451.8	307.5	110.2	63.7
L6	149.4	773.9	128.4	126.6	1229.5	776.6	527.3	58.0
Average	69.7	575.1	505.0	644.0	699.9	650.4	231.8	141.4

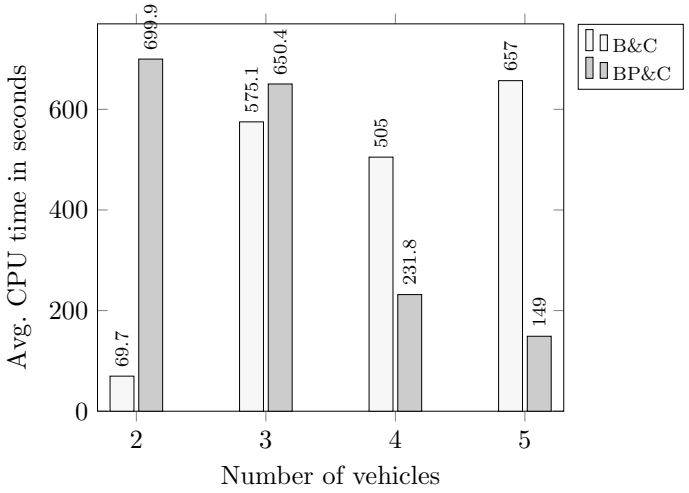


Figure 6: Average computational time by number of vehicles (instances solved to optimality by both algorithms)

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.

To conclude this section, we report in Table 4 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) where still unsolved to optimality. We succeeded to close 49 of them, all for instances with 4 or 5 vehicles. This is quite impressive given that, in previous works (Adulyasak et al. 2014b, Coelho and Laporte 2013, 2014), parallel branch-and-cut implementations using 6 or 8 threads were used and, in some cases, higher time limits (6 or 12 hours) were applied.

Table 4: Number of new optimal solutions found by the branch-and-price algorithm

	$K = 2$	$K = 3$	$K = 4$	$K = 5$	Total
Open	5	46	80	107	238
Closed	0	0	15	34	49

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:

MnsdI: Inequalities on the minimum number of sub-deliveries per demand (23) (see Section 3.2.3);

CapI: Capacity inequalities (24) (see Section 3.2.4);

CapRBS: Route-based separation heuristic for the capacity inequalities (see Section 3.2.4);

Tabu: Tabu search column generator (see Section 3.1.3);

Relax: Relaxation of constraints (5) (see Section 3.1.3);

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

Ldepth: Local depth-first strategy to explore the search tree (see Section 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 Section 3.1.3) or because they are not often called and, therefore, have limited impact (this is the case for the inequalities in Sections 3.2.1 and 3.2.2 or the first two types of branching decisions presented in Section 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 5718 seconds). This subset contains 18 instances in 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 1 to 5), X is the number of customers, and Y the number of vehicles. For example, `H3_4n20_k3` is the name of the fourth instance with 20 customers and 3 vehicles belonging to class H3.

Table 5 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 sub-deliveries per demand (23) 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 the instances within the 7200-second time limit and produces the least average computational time (2391.7 seconds). The most important feature in

Table 5: 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	3803.4	2569.2	N/A	7200.0	3782.9	7200.0	3801.3	7200.0	6060.9
H3_2n30_k3	3326.1	4758.6	N/A	7200.0	4020.8	7200.0	3458.2	2205.5	5980.2
H3_3n40_k3	769.0	1676.5	N/A	7200.0	799.0	5120.2	737.4	683.7	1644.5
L3_5n15_k3	2572.4	3741.9	N/A	3334.6	1565.3	5044.2	2519.2	7200.0	2876.1
L3_3n35_k3	920.8	1881.4	N/A	7200.0	7200.0	7200.0	2260.0	902.4	1563.5
H3_5n15_k5	1784.0	1895.3	N/A	7200.0	1273.5	1614.0	1920.2	1996.5	2836.4
H3_3n35_k5	1284.0	1874.8	N/A	7200.0	1036.4	4649.0	1614.2	1040.5	1467.7
L3_4n20_k5	1881.9	3965.0	N/A	7200.0	1699.3	2131.1	2177.9	7200.0	2382.6
L3_5n30_k5	904.6	637.6	N/A	7200.0	650.4	2513.8	1016.0	999.9	1047.4
L3_3n45_k5	1428.5	1599.5	N/A	7200.0	1834.6	7200.0	1581.8	1097.0	3114.1
H6_2n5_k3	1683.1	N/A	2086.8	2053.7	1289.7	1991.2	2517.6	7200.0	2842.3
H6_1n10_k3	5718.5	N/A	6455.4	7200.0	3744.6	7200.0	3754.1	7200.0	7200.0
L6_5n5_k3	749.2	N/A	785.5	920.6	576.2	764.2	788.4	1844.2	1051.9
L6_3n10_k3	578.3	N/A	524.8	525.8	599.1	1060.9	715.2	751.5	924.3
H6_4n10_k5	4532.8	N/A	5696.7	7200.0	7200.0	5971.3	7200.0	7200.0	7200.0
H6_5n10_k5	2059.6	N/A	1824.3	2379.1	1708.9	2086.4	2793.1	7200.0	3859.3
L6_3n10_k5	3796.4	N/A	4269.9	7200.0	7200.0	3402.9	4364.6	2732.6	7200.0
L6_5n10_k5	5257.8	N/A	5566.4	7200.0	4919.4	5755.7	7200.0	7200.0	7200.0
Average	2391.7	2720.9	2549.1	> 5711.9	> 2838.9	> 4339.2	> 2801.1	> 3991.8	> 3691.8

this comparison is the capacity inequalities (CapI). Without them, 13 of the 18 instances cannot be solved within the time limit. 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 8 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 the MnsdI component for all instances does not change much 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 the other components, we observe that some instances cannot be solved within 2 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 tighter integrality gaps. We solved to optimality 49 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.

References

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

Y. Adulyasak, J.-F. Cordeau, and R. Jans. The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, forthcoming, 2014b. doi: http://dx.doi.org/10.1016/j.cor.2014.01.011.

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

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

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

C. Archetti, L. Bertazzi, A. Hertz, and M.G. Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012.

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

J.F. Bard and N. Nananukul. A branch-and-price algorithm for an integrated production and inventory routing problem. *Computers & Operations Research*, 37(12):2202–2217, 2010.

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

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

N. Brahimi, S. Dauzere-Peres, N.M. Najid, and A. Nordli. Single item lot sizing problems. *European Journal of Operational Research*, 168(1):1–16, 2006.

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

L.C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558–565, 2013.

L.C. Coelho and G. Laporte. Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, forthcoming, 2014. doi: 10.1016/j.ijpe.2013.11.019.

L.C. Coelho, J.-F. Cordeau, and G. Laporte. The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548, 2012a.

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

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

G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column generation*. Springer, New York, NY, 2005.

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

G. Desaulniers, O.B.G. Madsen, and S. Røpke. Vehicle routing problems with time windows. In P. Toth and D. Vigo, editors, *Vehicle routing: Problems, methods, and applications*, pages 0–0. MOS-SIAM Series on Optimization, SIAM, Philadelphia, PA, 2014.

M. Dror, M. O. Ball, and B. L. Golden. A computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research*, 4:3–23, 1985.

F.G. Engineer, K.C. Furman, G. L. Nemhauser, M.W.P. Savelsbergh, and J.-H. Song. A branch-and-price-and-cut algorithm for single-product maritime inventory routing. *Operations Research*, 60(1):106–122, 2012.

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

M. Hewitt, G.L. Nemhauser, M.W.P. Savelsbergh, and J.-H. Song. A branch-and-price guided search approach to maritime inventory routing. *Computers & Operations Research*, 40(5):1410–1419, 2013.

J. Krarup and O. Bilde. Plant location, set covering, and economic lot size: an o(mn)-algorithm for structured problems. In L. Collatz, G. Meinardus, and W. W. E. Wetterling, editors, *Optimierung bei graphentheoretischen und ganzzahligen Probleme*, volume 3 of *Numerische Methoden bei Optimierungsverfahren*, pages 155–180. Birkhäuser Verlag, Basel, 1977.

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

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

J. Lysgaard, A.N. Letchford, and R.W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.

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

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