European Journal of Operational Research 302 (2022) 1189–1201

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

An improved formulation for the inventory routing problem with time-varying demands

Jørgen Skålnes[a,*], Henrik Andersson[a], Guy Desaulniers[b], Magnus Stålhane[a]

[a] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz veg 3, 7491 Trondheim, Norway
[b] Department of Mathematics and Industrial Engineering & GERAD, Polytechnique Montreal, Montreal, Canada

ARTICLE INFO

Article history:
Received 4 May 2021
Accepted 3 February 2022
Available online 9 February 2022

Keywords:
Routing
Inventory routing
Branch-and-cut
Dantzig-Wolfe reformulation
Valid inequalities

ABSTRACT

The Inventory Routing Problem (IRP) is a broad class of complex routing problems where the quantities of delivered products must also be determined. In this paper, we consider the classic IRP where a single supplier must determine when to visit its customers, how much to deliver and how to combine the customer visits in each period into routes. We propose a branch-and-cut algorithm based on a new mathematical formulation for the IRP, improving the average lower bound obtained from algorithms based on the branch-and-cut methodology. The new formulation substitutes parts of the original formulation with a convex combination of extreme points. We call these extreme points customer schedules and for each customer they contain information about delivery periods and corresponding delivered quantities. We show that this algorithm outperforms a state-of-the-art branch-and-cut algorithm on instances with time-varying demands. The customer schedule-based algorithm obtains better lower bounds, which improves the average optimality gap by 29% and 15% on two new sets of instances with time-varying demands.

1. Introduction

Advances in both operations research and supply chain management have led to increasing use of vendor-managed inventory (VMI), a business practice where a supplier makes the replenishment decisions for products delivered to customers based on specific inventory and supply chain policies (Coelho, Cordeau, & Laporte, 2014). This practice results in a situation where vendors save distribution and production costs, because they can coordinate shipments made to different customers, and the customers benefit by not allocating efforts to inventory control.

In the context of VMI, the inventory routing problem (IRP) arises. Here the supplier has to make three simultaneous decisions over a planning horizon consisting of discrete time periods to minimize transportation and inventory holding costs. The supplier has to create a schedule determining (1) when to serve a given customer (once or multiple times), (2) how much to deliver at each customer visit, and (3) how to combine customer visits into routes. A route starts and ends at the supplier in a given time period. Versions of the IRP can be identified both for land-based and maritime transportation, making it a problem with a wide range of applications. For road-based transportation, the application areas described in the literature are the transportation of products such as heating oil, beer, soft drinks, industrial gases, and groceries (Andersson, Hoff, Christiansen, Hasle, & Løkketangen, 2010).

The IRP was first proposed by Bell et al. (1983), and has later attracted significant attention from the research community (Coelho et al., 2014). The last decades' research has improved both exact methods and heuristics. The first exact solution method for this problem was proposed by Archetti, Bertazzi, Laporte, & Speranza (2007) for the IRP with a single supplier, a fixed time horizon divided into discrete periods, a single vehicle, and no stock-outs. The method presented is based on branch-and-cut (B&C), and is used to compare the effect of two inventory policies, referred to as the order-up-to level policy (OU) where the delivered quantity has to fill up the inventory, and the maximum level inventory policy (ML) where any quantity may be delivered as long as it does not violate the inventory limits. The authors showed that the ML inventory policy could drastically reduce both transportation and inventory costs. In addition, they also published a set of benchmark instances for the single-vehicle IRP.

* Corresponding author.
E-mail address: jorgen.skalnes@ntnu.no (J. Skålnes).

https://doi.org/10.1016/j.ejor.2022.02.011

Since then, improved exact solution methods for the IRP have been presented by Solyal & Süral (2011) and Avella, Boccia, & Wolsey (2015) for the OU policy (IRP-OU), and by Coelho, Cordeau, & Laporte (2012a), Coelho & Laporte (2014), Adulyasak, Cordeau, & Jans (2014), Avella et al. (2015), Desaulniers, Rakke, & Coelho (2016), Avella, Boccia, & Wolsey (2018), Guimarães, Schenekemberg, Coelho, Scarpin, & Pécora (2020) and Manousakis, Repoussis, Zachariadis, & Tarantilis (2021) for the ML policy (IRP-ML). Apart from the paper by Desaulniers et al. (2016), which presents a branch-price-and-cut method, all papers referenced above present solution methods based on the B&C methodology. The benchmark instances typically used for the IRP is often divided into a small and a large set of instances, originally published for the single-vehicle IRP by Archetti et al. (2007) and Archetti, Bertazzi, Hertz, & Speranza (2012), respectively. These instances were later modified by Coelho et al. (2012a) to also accommodate the multi-vehicle IRP, resulting in a total of 798 small instances and 300 large instances.The authors presented a three-index arc-flow formulation for the multi-vehicle IRP-ML, where subtour elimination constraints were added dynamically. This method was improved by Coelho & Laporte (2014), who presented three new families of valid inequalities that improved the results both for the single-vehicle IRP and the multi-vehicle IRP.

Adulyasak et al. (2014) compared different formulations of both the multi-vehicle production routing problem (also considering the quantity to produce at the supplier in each period as a decision) and the multi-vehicle IRP, solving them by combining a heuristic with a B&C algorithm. In addition to the three-index formulation proposed by Coelho & Laporte (2014), they proposed a two-index arc-flow formulation where capacitated subtour elimination constraints were added dynamically. They showed that the algorithm based on the three-index formulation is superior in finding optimal solutions, but that the algorithm based on the two-index formulation obtains tighter dual bounds on the largest instances.

Avella et al. (2015) presented a new two-index arc-flow formulation for the single-vehicle IRP that is valid in cases where the inventory capacity at every customer is an integer multiple of demand, and derived new valid inequalities for this formulation, referred to as the single item lot-sizing inequalities. Their B&C algorithm based on the new formulation obtained new best-known results for several of the benchmark instances both for the OU and ML inventory policies. This reformulation was also used for the multi-vehicle IRP studied by Avella et al. (2018), where they presented a new family of valid inequalities, the disjoint route inequalities. By adding two special cases of these valid inequalities to their two-index arc-flow formulation they improved the optimality gap of the benchmark instances with 50 customers and three time periods, and 30 customers and six time periods.

The latest improvements in exact solution methods further improved the results on the benchmark instances by integrating primal heuristics and feasibility mechanisms within the B&C framework. Guimarães et al. (2020) presented two feasibility mechanisms and improvement routines to enhance a B&C scheme for the multi-vehicle IRP. Using these mechanisms they were able to find several new best-known solutions and proved optimality for new instances. Manousakis et al. (2021) presented a two-commodity flow formulation for the IRP and a new set of valid inequalities for this formulation. The formulation exploits the interaction between the flow of goods and the flow of empty space (inverse flow). Together with an efficient primal heuristic, they obtained new best-known solutions on some of the benchmark instances.

In addition to the development of exact methods, there has been a considerable effort on developing heuristic solution methods for the IRP. Most of these heuristics can in fact be classified as matheuristics (Archetti et al., 2012; Archetti, Boland, & Speranza, 2017; Chitsaz, Cordeau, & Jans, 2019; Diniz, Martinelli, & Poggi, 2020; Vadseth, Andersson, & Stålhane, 2021), i.e., "heuristic algorithms made by the interoperation of metaheuristics and mathematical programming techniques" (Boschetti, Maniezzo, Roffilli, & Bolufé Röhler, 2009).

Archetti et al. (2012) proposed a heuristic that combines a tabu search scheme with ad hoc designed mixed-integer linear programming models for the single-vehicle IRP, obtaining optimal or near-optimal solutions in much shorter solution time than the exact B&C method proposed by Archetti et al. (2007). This matheuristic was later extended to also handle the IRP with multiple vehicles (Archetti et al., 2017), obtaining new best-known solutions on 92% of the large benchmark instances. These results were further improved by Chitsaz et al. (2019) when they found 194 new best-known solutions with a three-phase decomposition method, originally developed for the assembly routing problem, on the 300 large IRP benchmark instances. Diniz et al. (2020) presented a matheuristic using an iterative local search method with a randomized variable neighbourhood search and the solution of network flow problems. They found 113 new best-known solutions on the small benchmark set. Vadseth et al. (2021) described an iterative matheuristic for the IRP, where they used a giant tour and simple operators to heuristically create routes that are included in a path-flow formulation. The matheuristic then iterates between solving this path-flow model and updating the set of routes based on the optimal solution of the path-flow model from the previous iteration. Running this algorithm they found 179 new best-known solutions out of 240 of the large multi-vehicle instances.

The IRP also forms the basis for richer problems, such as the IRP with perishable products (Alvarez, Cordeau, Jans, Munari, & Morabito, 2021) and the IRP with pickups and deliveries (Archetti, Christiansen, & Grazia Speranza, 2018; Archetti, Speranza, Boccia, Sforza, & Sterle, 2020). Alvarez et al. (2021) proposed a new formulation for the IRP with perishable products and developed a hybrid heuristic, combining an iterated local search metaheuristic and two mathematical programming components. The proposed matheuristic was also modified and tested on the standard IRP benchmark instances, finding high-quality solutions compared with the state-of-the-art methods. Archetti et al. (2018) presented the single-vehicle IRP with pickups and deliveries along with a model formulation and a B&C algorithm tailored to solve this problem. They solved 487 out of 640 instances, with up to 50 customers and three time periods and up to 30 customers and six time periods. Archetti et al. (2020) designed a B&C algorithm for the multi-vehicle IRP with pickups and deliveries, while also proposing a new set of valid inequalities, called interval inequalities. They solved 946 out of 1280 instances to optimality, and outperformed the state-of-the-art method for the single-vehicle version by finding 133 new best-known solutions. There also exist other variants of the IRP, such as the IRP with transshipments (Coelho, Cordeau, & Laporte, 2012b), the IRP with demand moves (Baller, Dabia, Desaulniers, & Dullaert, 2021) and the two-echelon multi-vehicle IRP (Guimarães, Coelho, Schenekemberg, & Scarpin, 2019).

In this paper, we present a new innovative formulation of the IRP, based on a Dantzig-Wolfe reformulation of a two-index arc-flow model. We refer to this reformulation as the *customer schedule formulation* (CSF), in which we introduce a set of variables containing information about which periods a customer is visited and the quantity delivered in each of these periods. This reformulation allows us to significantly reduce the number of constraints, but at the cost of adding more variables. We prove that several of the valid inequalities proposed by Coelho & Laporte (2014) and Avella et al. (2015) are not useful for the CSF, and that it is more general than the reformulation proposed by Avella et al. (2015) since it does not require an integer inventory capacity to demand ratio at the customers. Further, we modify the capacity inequalities proposed by Desaulniers et al. (2016) to be applicable to arc-flow for-

mulations, and propose a new separation algorithm for these valid inequalities.

We have implemented a B&C algorithm based on the CSF including the valid inequalities of Desaulniers et al. (2016) and Avella et al. (2018). This algorithm is compared with a reimplementation of a state-of-the-art B&C algorithm based on the formulation proposed by Avella et al. (2018) and including the valid inequalities of Coelho & Laporte (2014) and Desaulniers et al. (2016). We test these two algorithms on the benchmark instances proposed by Archetti et al. (2007) and Coelho et al. (2012a), and on modified versions of these instances with time-varying demands. The results show that the performance of the CSF-based algorithm is similar to the state-of-the-art reimplementation for the existing benchmark instances, but significantly better for instances with time-varying demands.

It is worth pointing out that an improvement of the dual bound at the termination of a B&C method can be achieved in two ways, either by using a stronger formulation yielding a better linear relaxation or by quickly finding good primal solutions. A good primal solution makes it possible to perform variable fixing by reduced cost or allows for faster pruning of the branch-and-bound (B&B) tree. The latter is not true if using a best-first strategy, but most commercial solvers embed plunging strategies and a good primal solution allows for faster pruning whenever the solver performs a plunge, i.e., swapping from a best-first to a depth-first strategy. Our work is focused on improving the dual bound at the root node, and is the reason why we chose to compare our results with those of Avella et al. (2018) as they represent the strongest known arc-flow formulation for the IRP.

The remainder of this paper is organized as follows. Section 2 presents the mathematical model along with the valid inequalities used in our reimplementation of a state-of-the-art B&C algorithm. Section 3 explains the concept of customer schedules, how they can be used to reformulate the problem, and a thorough analysis of the properties of this reformulation. The B&C algorithm is presented in Section 4 before reporting our computational results in Section 5. Finally, conclusions are drawn in Section 6.

2. Problem definition and mathematical model

In this section, the two-index arc-flow formulation for the single-depot multi-vehicle IRP is presented along with the valid inequalities of Coelho & Laporte (2014), the capacity inequalities of Desaulniers et al. (2016) and the disjoint route inequalities of Avella et al. (2018). In Section 2.1, the problem is formally stated, before presenting the mathematical model in Section 2.2 and the valid inequalities in Section 2.3.

2.1. Problem definition and notation

In the single-depot multi-vehicle IRP a single supplier, denoted 0, produces a single product which is delivered to a set of customers $\mathcal{N}^C = \{1, \dots, n\}$ over a planning horizon that is divided into a set $\mathcal{T}$ of discrete time periods. In each time period, $t \in \mathcal{T}$, S_t units of the product are produced at the supplier, while D_{it} units of the same product are consumed at each customer $i \in \mathcal{N}^C$. A customer can be visited at most once in each time period. Both the supplier, 0, and each customer, i, have a storage with given upper and lower inventory capacities, $\bar{L}_i$ and $\underline{L}_i$, respectively, an initial inventory level I_i at the beginning of the planning horizon, and a unit holding cost H_{it} for each time period t. In addition, we introduce $I_{it} = \max\{I_i - \sum_{s=1}^{t} D_{is}, 0\}$ to denote the remaining units of the product from the initial inventory at customer i in period t, given the first-in, first-out principle.

To keep all storages within their inventory limits, a fleet of K homogeneous vehicles, each with a capacity to hold Q units of the product, is used to transport the product from the supplier to the customers. The problem of designing routes for these vehicles may be defined on a graph $G = (\mathcal{N}, \mathcal{A})$, where the set of nodes $\mathcal{N} = \{0, \dots, n\}$ consists of one supply node (0) and one node for each customer, and the set of arcs $\mathcal{A} = \{(i, j) \in \{\mathcal{N} \times \mathcal{N}\} \mid i \neq j\}$ connects the nodes. The route driven by a vehicle can be seen as a simple cycle in the graph starting and ending at node 0, where C_{ij} represents the cost of driving directly from node i to node j. The goal is to create at most one route for each vehicle in each time period, delivering units of the product to each customer along that route, so that the inventory limits at all nodes are satisfied in each time period and the sum of the inventory holding costs and the transportation costs is minimized.

2.2. Model

To model this problem we introduce the following variables. Let x_{ijt} be equal to 1 if a vehicle traverses arc $(i, j) \in \mathcal{A}$ in period $t \in \mathcal{T}$, and 0 otherwise. In time period $t \in \mathcal{T}$, let δ_{it} be 1 if customer $i \in \mathcal{N}^C$ is visited, 0 otherwise, and let δ_{0t} be the number of vehicles leaving the supplier. For each node $i \in \mathcal{N}$ and period $t \in \mathcal{T}$, we define a non-negative variable s_{it} that represents the inventory level at node i at the end of period t. In addition, for each node $i \in \mathcal{N}$ we denote the initial inventory s_{i0}. Further, we let q_{it} denote the quantity delivered to customer $i \in \mathcal{N}^C$ in period $t \in \mathcal{T}$.

Using this notation, we can now present the formulation for the single-depot multi-vehicle IRP as the following mixed-integer linear program (MILP):

$$\min \sum_{(i,j)\in\mathcal{A}} \sum_{t\in\mathcal{T}} C_{ij} x_{ijt} + \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} H_{it} s_{it} \tag{1}$$

$$s_{0t} = S_t - \sum_{i\in\mathcal{N}^C} q_{it} + s_{0(t-1)}, \qquad \forall\, t \in \mathcal{T}, \tag{2}$$

$$s_{it} = q_{it} - D_{it} + s_{i(t-1)}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{3}$$

$$\underline{L}_0 \leq s_{0t} \leq \bar{L}_0, \qquad \forall\, t \in \mathcal{T}, \tag{4}$$

$$\underline{L}_i \leq s_{it} \leq \bar{L}_i, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{5}$$

$$\sum_{i\in\mathcal{N}^C} q_{it} \leq Q\delta_{0t}, \qquad \forall\, t \in \mathcal{T}, \tag{6}$$

$$q_{it} \leq \bar{L}_i - s_{i(t-1)}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{7}$$

$$q_{it} \leq \min\{\bar{L}_i - I_{it}, Q\}\delta_{it}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{8}$$

$$\sum_{j\in\mathcal{N}\setminus\{i\}} x_{ijt} = \delta_{it}, \qquad \forall\, i \in \mathcal{N}, t \in \mathcal{T}, \tag{9}$$

$$\sum_{j\in\mathcal{N}\setminus\{i\}} x_{ijt} = \sum_{j\in\mathcal{N}\setminus\{i\}} x_{jit}, \qquad \forall\, i \in \mathcal{N}, t \in \mathcal{T}, \tag{10}$$

$$\sum_{(i,j)\in(\mathcal{S}:\mathcal{S})} x_{ijt} \leq \sum_{i\in\mathcal{S}} \delta_{it} - \delta_{mt}, \qquad \forall\, \mathcal{S} \subset \mathcal{N}^C, |\mathcal{S}| \geq 2, t \in \mathcal{T}, m \in \mathcal{S}, \tag{11}$$

$$\sum_{(i,j)\in(\mathcal{S}:\mathcal{N}\setminus\mathcal{S})} Qx_{ijt} \geq \sum_{i\in\mathcal{S}} q_{it}, \qquad \forall\, \mathcal{S} \subset \mathcal{N}^C, |\mathcal{S}| \geq 2, t \in \mathcal{T}, \tag{12}$$

$$q_{it} \geq 0, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{13}$$

$$\delta_{it} \in \{0,1\}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{14}$$

$$\delta_{0t} \in \{0,1,\ldots,K\}, \qquad \forall\, t \in \mathcal{T}, \tag{15}$$

$$x_{ijt} \in \{0,1\}, \qquad \forall\, (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{16}$$

where $s_{00} = I_0$ and $s_{i0} = I_i$, $\forall i \in \mathcal{N}^C$. The objective function (1) minimizes the sum of the transportation cost and the inventory holding cost. Constraints (2) and (3) balance the inventory at the supplier and at the customers, respectively. Constraints (4) and (5) make sure that the inventory levels always stay between their lower and upper limits at the supplier and at the customers, respectively. Constraints (6) state that the fleet of vehicles leaving the supplier never delivers more than its capacity to the customers in a given period. Constraints (7) enforce the ML inventory policy. Constraints (8) ensure that a customer can only receive a delivery if visited. Constraints (9) are the degree constraints and constraints (10) ensure a correct flow of vehicles between nodes. The subtour and capacitated subtour elimination constraints (SECs and CSECs, respectively) are stated in constraints (11) and (12), respectively, where $(\mathcal{E} : \mathcal{F}) = \{(i,j) : i \in \mathcal{E}, j \in \mathcal{F} \setminus \{i\}\}$ denotes the set of arcs going from a node in set $\mathcal{E}$ to a node in set $\mathcal{F}$. Finally, constraints (13)–(16) impose integrality and non-negativity of the variables.

2.3. Valid inequalities

To strengthen the linear relaxation of the model defined by (1)–(16), we apply three sets of valid inequalities in our B&C algorithm.

2.3.1. Valid inequalities from Coelho & Laporte (2014)

The following inequalities were proposed by Coelho & Laporte (2014):

$$x_{ijt} \leq \delta_{it}, \qquad \forall\, (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{17}$$

$$\delta_{it} \leq \delta_{0t}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{18}$$

$$\sum_{t'=1}^{t} \delta_{it'} \geq \left\lceil \left(\sum_{t'=1}^{t} D_{it'} - I_i \right) \Big/ \min\{\bar{L}_i, Q\} \right\rceil, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{19}$$

$$\sum_{t'=t_1}^{t_2} \delta_{it'} \geq \left\lceil \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) \Big/ \min\{\bar{L}_i, Q\} \right\rceil, \qquad \forall\, i \in \mathcal{N}^C, t_1 \in \mathcal{T} \setminus \{1\}, t_2 \in \mathcal{T}, t_2 \geq t_1, \tag{20}$$

$$\sum_{t'=t_1}^{t_2} \delta_{it'} \geq \left(\sum_{t'=t_1}^{t_2} D_{it'} - s_{i(t_1-1)} \right) \Big/ \min\{\bar{L}_i, Q, \sum_{t'=t_1}^{t_2} D_{it'}\}, \qquad \forall\, i \in \mathcal{N}^C, t_1, t_2 \in \mathcal{T}, t_2 \geq t_1. \tag{21}$$

Constraints (17) were originally developed for an edge formulation, but can be improved for an arc-flow formulation. We know that if arc x_{ijt} is used, arc x_{jit} cannot be used, and the use of the arcs out from node j has to be greater than or equal to the use of arc x_{ijt}.

$$x_{ijt} \leq \sum_{k \in \mathcal{N} \setminus \{i\}} x_{jkt}, \qquad \forall\, (i,j) \in (\mathcal{N}^C : \mathcal{N}^C), t \in \mathcal{T}. \tag{22}$$

Constraints (18) give a tighter relationship between the visiting variables δ_{it} for the customers and the visiting variables δ_{0t} for the supplier. Constraints (19) and (20) compute a lower bound on how many times a customer i needs to be visited in a given time interval. Constraints (21) compute a lower bound given the actual inventory level at customer i at the beginning of a given time interval. To maintain the linear characteristics we cannot round up this expression, but it gives a tighter relationship between the δ_{it} and s_{it} variables.

2.3.2. Capacity inequalities from Desaulniers et al. (2016)

We also adapt the capacity inequalities of Desaulniers et al. (2016) to be applicable in an arc-flow formulation. The capacity inequalities ensue from the same idea as the rounded capacity inequalities for the capacitated vehicle routing problem (Laporte & Nobert, 1983). Since we can decide the delivered quantity in each period we do not know for sure that we will deliver a quantity equal to the demand in each period. However, we do know that the residual demand for a given period has to be delivered before this given period. We define residual demand, $\bar{D}_{it} = \max\{D_{it} - I_{it}, 0\}$, to be the demand that cannot be covered by the initial inventory. Thus, assuming a first-in, first-out policy for the units in the inventory we can calculate a lower bound on the number of vehicles needed to serve a given subset of residual demands.

Let $\mathcal{RD}$ be the set of all positive residual demands across all customers and time periods, $\mathcal{RD} = \{(i,t) \in \mathcal{N}^C \times \mathcal{T} \mid \bar{D}_{it} > 0\}$, and let $\mathcal{U} \subseteq \mathcal{RD}$ be a subset of all positive residual demands. A positive residual demand $\bar{D}_{jt}$ can be served by a vehicle traversing an arc (i,j) in period t or previously as long as the inventory capacity is not violated.

Thus, we introduce $\mathcal{P}^-_{jt} = \{(i,j,m) \in \mathcal{N} \times \mathcal{N}^C \times \mathcal{T} \mid (\bar{D}_{jt} > 0 \wedge m \leq t \wedge (\sum_{l=m}^{t} \bar{D}_{jl} + I_{jm} \leq \bar{L}_j))\}$ to be the set of arcs (i,j) in period m, denoted (i,j,m), that can serve residual demand $\bar{D}_{jt}$. Let $\mathcal{A}_{\mathcal{U}} = \{(i,j,m) \in P^-_{jt} \mid (j,t) \in \mathcal{U}\}$ be the set of arcs that can serve the residual demands in $\mathcal{U}$. It is worth pointing out that the same arc (i,j,m) can potentially serve several consecutive residual demands at customer j, but appears at most once in $\mathcal{A}_{\mathcal{U}}$. Then we get the following capacity inequalities:

$$\sum_{(i,j,t) \in \mathcal{A}_{\mathcal{U}}} x_{ijt} \geq \left\lceil \sum_{(i,t) \in \mathcal{U}} \bar{D}_{it} \Big/ Q \right\rceil, \qquad \forall\, \mathcal{U} \subseteq \mathcal{RD}. \tag{23}$$

2.3.3. Disjoint route inequalities from Avella et al. (2018)

The disjoint route (DR) inequalities were introduced by Avella et al. (2018). Let $\mathcal{R}$ be the set of arcs in a route and let $\mathcal{V}(\mathcal{R})$ be the set of nodes in that route. For a given period $t \in \mathcal{T}$, a subset $\mathcal{S} \subseteq \mathcal{N}^C$, a partition $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ of $\mathcal{S}$, and the coefficients $\mu_{ij} \geq 0$ for all $(i,j) \in \mathcal{A}$, a valid DR inequality can be formulated as:

$$\sum_{(i,j) \in \mathcal{A}} \mu_{ij} x_{ijt} \geq \sum_{j \in \mathcal{S}_1} q_{jt} + \sum_{j \in \mathcal{S}_2} (q_{jt} - s_{j(t+1)}) + \sum_{j \in \mathcal{S}_3} (q_{jt} - s_{jt}) + \sum_{j \in \mathcal{S}_4} (s_{jt} - (\bar{L}_j - \sum_{m=t-1}^{t} D_{jm})) \tag{24}$$

if

$$\sum_{(i,j) \in \mathcal{R}} \mu_{ij} \geq \min\{Q, \sum_{i \in \mathcal{V}(\mathcal{R}) \cap \mathcal{S}_1} \bar{L}_i + \sum_{i \in \mathcal{V}(\mathcal{R}) \cap \mathcal{S}_2} (D_{it} + D_{i(t+1)}) + \sum_{i \in \mathcal{V}(\mathcal{R}) \cap \mathcal{S}_3} D_{it} + \sum_{i \in \mathcal{V}(\mathcal{R}) \cap \mathcal{S}_4} D_{i(t-1)}\}, \tag{25}$$

for every route starting and ending at the supplier.

We include the two special cases of the DR inequalities proposed by Avella et al. (2018), namely the simple DR inequalities and the h-DR inequalities (Avella et al., 2018). The technical details regarding these valid inequalities can be found in Avella et al. (2018).

3. Customer schedules

We first present a new reformulation of the problem in Section 3.1 based on a Dantzig-Wolfe decomposition, before stating and proving several properties of this new formulation in Section 3.2.

3.1. Reformulation

Let us define a bounded set of feasible solutions P_i described by the subset of constraints (3), (5), (7)–(8), (13) and (14) for each customer $i \in \mathcal{N}^C$. Let Ω_i be the set of extreme points of the convex hull of P_i, and let $Q_{it\omega}, S_{it\omega}$ and $A_{it\omega}$ be the values of q_{it}, s_{it} and δ_{it}, respectively, for each extreme point $\omega \in \Omega_i$. Like in a Dantzig & Wolfe (1960) decomposition, we may express all points in P_i as convex combinations of these extreme points. To do this, we introduce a set of weighting variables $\lambda_{i\omega}$ and express the set of feasible solutions as follows:

$$q_{it} = \sum_{\omega \in \Omega_i} Q_{it\omega}\lambda_{i\omega}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{26}$$

$$s_{it} = \sum_{\omega \in \Omega_i} S_{it\omega}\lambda_{i\omega}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{27}$$

$$\delta_{it} = \sum_{\omega \in \Omega_i} A_{it\omega}\lambda_{i\omega}, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{28}$$

$$\sum_{\omega \in \Omega_i} \lambda_{i\omega} = 1, \qquad \forall\, i \in \mathcal{N}^C, \tag{29}$$

$$\lambda_{i\omega} \geq 0, \qquad \forall\, i \in \mathcal{N}^C, \omega \in \Omega_i. \tag{30}$$

In the following, we refer to each extreme point $\omega \in \Omega_i$ as a *customer schedule* for customer $i \in \mathcal{N}^C$. A customer schedule can be interpreted as a set of days the customer is visited ($\{t : t \in \mathcal{T} \wedge A_{it\omega} = 1\}$) and the quantity delivered ($Q_{it\omega}$) on each of these days. The delivered quantities also implicitly decide the inventory levels ($S_{it\omega}$) at the customers throughout the planning period due to constraints (3). By substituting for q_{it}, s_{it} and δ_{it} in the mathematical formulation (1)–(16), we state the CSF of the problem as follows:

$$\min \sum_{(i,j)\in\mathcal{A}} \sum_{t\in\mathcal{T}} C_{ij}x_{ijt} + \sum_{i\in\mathcal{N}^C} \sum_{t\in\mathcal{T}} \sum_{\omega\in\Omega_i} H_{it}S_{it\omega}\lambda_{i\omega} + \sum_{t\in\mathcal{T}} H_{0t}s_{0t} \tag{31}$$

$$s_{0t} = S_t - \sum_{i\in\mathcal{N}^C} \sum_{\omega\in\Omega_i} Q_{it\omega}\lambda_{i\omega} + s_{0(t-1)}, \qquad \forall\, t \in \mathcal{T}, \tag{32}$$

$$\underline{L}_0 \leq s_{0t} \leq \bar{L}_0, \qquad \forall\, t \in \mathcal{T}, \tag{33}$$

$$\sum_{i\in\mathcal{N}^C} \sum_{\omega\in\Omega_i} Q_{it\omega}\lambda_{i\omega} \leq Q\delta_{0t}, \qquad \forall\, t \in \mathcal{T}, \tag{34}$$

$$\sum_{j\in\mathcal{N}\setminus\{i\}} x_{ijt} = \sum_{\omega\in\Omega_i} A_{it\omega}\lambda_{i\omega}, \qquad \forall\, i \in \mathcal{N}, t \in \mathcal{T}, \tag{35}$$

$$\sum_{j\in\mathcal{N}\setminus\{i\}} x_{ijt} = \sum_{j\in\mathcal{N}\setminus\{i\}} x_{jit}, \qquad \forall\, i \in \mathcal{N}, t \in \mathcal{T}, \tag{36}$$

$$\sum_{(i,j)\in(\mathcal{S}:\mathcal{S})} x_{ijt} \leq \sum_{i\in\mathcal{S}} \sum_{\omega\in\Omega_i} A_{it\omega}\lambda_{i\omega} - \sum_{\omega\in\Omega_m} A_{mt\omega}\lambda_{m\omega}, \qquad \forall\, \mathcal{S} \subset \mathcal{N}^C, |\mathcal{S}| \geq 2, t \in \mathcal{T}, m \in \mathcal{S}, \tag{37}$$

$$\sum_{(i,j)\in(\mathcal{S}:\mathcal{N}\setminus\mathcal{S})} Qx_{ijt} \geq \sum_{i\in\mathcal{S}} \sum_{\omega\in\Omega_i} Q_{it\omega}\lambda_{i\omega}, \qquad \forall\, \mathcal{S} \subset \mathcal{N}^C, |\mathcal{S}| \geq 2, t \in \mathcal{T}, \tag{38}$$

$$\sum_{\omega\in\Omega_i} \lambda_{i\omega} = 1, \qquad \forall\, i \in \mathcal{N}^C, \tag{39}$$

$$\lambda_{i\omega} \geq 0, \qquad \forall\, i \in \mathcal{N}^C, \omega \in \Omega_i, \tag{40}$$

$$x_{ijt} \leq \sum_{k\in\mathcal{N}\setminus\{i\}} x_{jkt}, \qquad \forall\, (i,j) \in (\mathcal{N}^C : \mathcal{N}^C), t \in \mathcal{T}, \tag{41}$$

$$\delta_{0t} \in \{0, 1, \ldots, K\}, \qquad \forall\, t \in \mathcal{T}, \tag{42}$$

$$x_{ijt} \in \{0, 1\}, \qquad \forall\, (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{43}$$

The objective function (31) is equivalent to (1), constraints (32) and (33) balance the inventory and keep it within the inventory capacity at the supplier, respectively. The constraints (34)–(38) express the same as constraints (6), (9)–(12) found in the original formulation, while constraints (39) ensure that only a convex combination of customer schedules may be selected. Constraints (41) are the valid inequalities (22) repeated here for the sake of readability. Finally, the non-negativity requirements for the weighting variables are defined by constraints (40), and constraints (42)-(43) are the same as constraints (15)-(16).

3.2. Properties of the reformulation

Let us analyze the properties of the CSF. Propositions 1–3 show that the new formulation is at least as strong as the original formulation including the valid inequalities (19)–(21) and the single item lot-sizing inequalities (Avella et al., 2015). The proofs of Propositions 2 and 3 can be found in Appendix A. Let R^{CSF} and R^O be the polytopes defined by the linear relaxations of the CSF (32)–(43) and the original formulation (2)–(16), respectively.

Proposition 1. $R^{CSF} \subseteq R^O$.

This proposition is true because the CSF is a Dantzig-Wolfe reformulation of (3)–(16).

The next proposition indicates that the valid inequalities (19)–(21) are not useful for the CSF.

Proposition 2. *The valid inequalities (19)–(21) are satisfied by all solutions in* R^{CSF}.

The following proposition shows that another set of valid inequalities is not useful when employing the CSF.

Proposition 3. *All solutions in* R^{CSF} *satisfy the valid inequalities defined per customer based on the single item lot-sizing reformulation proposed by Avella et al. (2015).*

We conclude this section by showing a numerical example where R^{CSF} is a proper subset of R^O. Assume we have a problem with only one customer i and two time periods, where demand is 50 in each period, and both the vehicle capacity and the inventory capacity at this customer is 100. Let us also assume that the inventory cost is higher at the customer than at the supplier and that the initial inventory at the customer is 0. For this problem, we can easily observe that R^O contains a solution χ such that $\delta_{i1} = \delta_{i2} = 0.5$ and $q_{i1} = q_{i2} = 50$. Let us show that this solution does not belong to R^{CSF}. Indeed, for this example, there are only two customer schedules:

$$Q_{i11} = 100, Q_{i21} = 0, \qquad A_{i11} = 1, A_{i21} = 0.$$
$$Q_{i12} = 50, Q_{i22} = 50, \qquad A_{i12} = 1, A_{i22} = 1.$$

In this case, constraints (28) and (29) write as follows:

$$\delta_{i1} = \lambda_{i1} + \lambda_{i2},$$
$$\delta_{i2} = 0\lambda_{i1} + \lambda_{i2},$$
$$\lambda_{i1} + \lambda_{i2} = 1.$$

Therefore, $\delta_{i1} = 1$ for all solutions in R^{CSF} and $\chi \notin R^{CSF}$. Thus, R^{CSF} can be a proper subset of R^{O}.

4. Branch-and-cut algorithm

In this paper, we compare the two formulations for the IRP previously presented, and their performance when put into a B&C framework. In the remainder of this paper, we refer to the formulation defined by (1)–(16) and the valid inequalities (18)–(22), as the *state-of-the-art formulation* (SOTAF). In addition, we add the single item lot-sizing inequalities proposed by Avella et al. (2015) a priori for the test instances where they are applicable. The B&C algorithm based on this formulation is referred to as SOTAF-BC.

The B&C algorithm based on the CSF, is denoted CSF-BC. The a priori generation of customer schedules is done by a labeling algorithm enumerating all possible combinations of visits and corresponding quantities. It is worth noting that a simple forward propagation alone is not enough to enumerate all schedules due to the inter-dependencies between delivered quantities across different periods. Section 4.1 therefore gives a detailed description of the labeling algorithm used to enumerate the customer schedules. Section 4.2 gives an overview of the proposed B&C algorithms, and Section 4.3 presents a detailed description of the separation of the capacity inequalities.

4.1. A priori generation of customer schedules

The customer schedules are generated a priori using a labeling algorithm (Algorithm 1) for each customer $i \in \mathcal{N}^C$. A label is used

Algorithm 1 Customer schedules for customer i.

1: $U = L_0$
2: **while** $U \neq \emptyset$ **do**
3: $L = remove(U)$
4: **if** $\forall t' \in \mathcal{T} : \underline{s}_{it'}(L) = \bar{s}_{it'}(L)$ **then**
5: $\mathcal{P} = \mathcal{P} \bigcup \{L\}$
6: **end if**
7: **for** $t \in \mathcal{T} : \underline{s}_{it}(L) < \bar{s}_{it}(L)$ **do**
8: create labels L_1, L_2
9: $\forall t' \in \mathcal{T} : \bar{s}_{it'}(L_1) = f^U_{tt'}(\underline{s}_{it}(L), \bar{s}_{it'}(L))$
10: $\forall t' \in \mathcal{T} : \underline{s}_{it'}(L_1) = f^L_{tt'}(\underline{s}_{it}(L), \underline{s}_{it'}(L))$
11: $U = U \bigcup \{L_1\}$
12: $\forall t' \in \mathcal{T} : \bar{s}_{it'}(L_2) = f^U_{tt'}(\bar{s}_{it}(L), \bar{s}_{it'}(L))$
13: $\forall t' \in \mathcal{T} : \underline{s}_{it'}(L_2) = f^L_{tt'}(\bar{s}_{it}(L), \underline{s}_{it'}(L))$
14: $U = U \bigcup \{L_2\}$
15: **end for**
16: **end while**

to store lower and upper bounds on the inventory levels for each time period, and the values of these bounds for label L are referred to as $\underline{s}_{it}(L)$ and $\bar{s}_{it}(L)$, respectively. The labeling algorithm starts by initiating a set of unprocessed labels, U, initially only consisting of the label L_0 (line 1). For L_0 the initial values of the inventory bounds are set as follows:

$$\underline{s}_{it}(L) = \max\{I_i - \sum_{t'=1}^{t} D_{it'}, 0\}, \qquad t \in \mathcal{T}, \qquad (44)$$

$$\bar{s}_{it}(L) = \min\{I_i + \sum_{t'=1}^{t} (\min\{\bar{L}_i, Q\} - D_{it'}), \bar{L}_i\} \qquad t \in \mathcal{T}, \qquad (45)$$

which represents the minimum and maximum possible inventory bounds for any given period. The algorithm then selects one label from the set of unprocessed labels (line 3). If the lower and upper inventory bounds are equal for each time period, then the label represents a complete customer schedule and is added to the set of completed labels $\mathcal{P}$ (line 5). Otherwise, the algorithm iterates over the time periods where the upper and lower inventory bounds are different, and creates two new labels, L_1 and L_2, for each selected time period (line 8). For the selected time period t the upper inventory bound in L_1 is decreased to the lower inventory bound (line 9) and the lower inventory bound in L_2 is increased to the upper inventory bound (line 13). The upper and lower inventory bounds for the remaining periods are then updated according to the functions f^U and f^L (lines 9–10 and lines 12–13), respectively, which is defined as:

$$f^L_{t_1t_2}(a,b) = \begin{cases} \max\{a - \sum_{t=t_1}^{t_2} D_{it}, b\} & \text{if } t_1 < t_2, \\ a, & \text{if } t_1 = t_2, \\ \max\{a + \sum_{t=t_2}^{t_1} (D_{it} - \min\{\bar{L}_i, Q\}), b\}, & \text{otherwise.} \end{cases} \qquad (46)$$

$$f^U_{t_1t_2}(a,b) = \begin{cases} \min\{a + \sum_{t=t_1}^{t_2} (\min\{\bar{L}_i, Q\} - D_{it}), b\}, & \text{if } t_1 < t_2, \\ a, & \text{if } t_1 = t_2, \\ \min\{a + \sum_{t=t_2}^{t_1} D_{it}, b\}, & \text{otherwise.} \end{cases} \qquad (47)$$

where a is the fixed value in time period t_1 and b is the old value of the lower/upper inventory bound in time period t_2. Once the inventory bounds are updated a new period t where $\bar{s}_{it} \neq \underline{s}_{it}$ is selected.

If there is any inconsistent bounds for any of the time periods (i.e. $\underline{s}_{it}(L) > \bar{s}_{it}(L)$) the label is discarded, otherwise it is added to the set of unprocessed labels. Once all labels have been processed, the corresponding delivered quantities can be calculated and at which periods customer i is visited can be identified. To satisfy the equality in constraints (35), for every customer schedule with a non-delivery period we must duplicate the given customer schedule so that we have one customer schedule with a visit and one without a visit. This means that for a customer $i \in \mathcal{N}^C$ in time period $t \in \mathcal{T}$ with customer schedule $\omega_1 \in \Omega$ where $Q_{it\omega_1} = 0$, we must create customer schedule ω_2 where $A_{it\omega_1} = 1$ and $A_{it\omega_2} = 0$, or vice versa, and where the remaining information remains the same.

From Algorithm 1 we see that we have a time complexity of $O(|\mathcal{T}|2^{|\mathcal{T}|})$ for each customer. This means that the time complexity grows linearly with the number of customers, but exponentially with the number of time periods. However, for the benchmark instances used in the literature and in this paper the time horizon is short and the computational time used to enumerate the customer schedules a priori is negligible.

4.2. Overview of the branch-and-cut algorithms

A B&C algorithm is an extension of the well-known branch-and-bound (B&B) algorithm where each node in the B&B tree is solved multiple times, adding cutting planes to the formulation in each iteration. These cutting planes are identified by solving one (or more) separation problem(s) for each family of valid inequalities. Adding these cutting planes strengthens the dual bound of the node, hopefully leading to a significantly smaller B&B tree.

For both the SOTAF-BC and the CSF-BC, the SECs, CSECs, and valid inequalities presented in Sections 2.3.2 and 2.3.3 are added dynamically to the linear relaxations.

The separation problems are solved in the following order in the root node:

(i) SECs (11)

(ii) CSECs (12)
(iii) Capacity inequalities (23)
(iv) Simple DR inequalities (special case of (24))
(v) h-DR inequalities (special case of (24))

The separation algorithm moves on to the next separation problem when the dual bound improvement from the previous iteration falls below a given threshold. Due to the computational complexity of the last three, they are only solved in the root node. For the remaining part of the B&B tree, the SECs and CSECs are separated once in each node of the tree if the solution of the linear relaxation is fractional.

We separate the SECs by first identifying strongly connected components, which was shown by Drexl (2013) to work well for the elementary shortest path problem. However, since this separation algorithm is exact only for integral solutions, we also identify min-cuts between the supplier and each customer ($(0:i)\ \forall\ i \in \mathcal{N}^C$) which are solvable in polynomial time (Picard & Ratliff, 1973), to find additional cuts when the cuts defined by the strongly connected components do not improve the dual bound significantly. The separation problem for the CSECs is solved similarly, but where the first part is solved by finding the connected components in the graph. The separation of the two families of DR-inequalities are done in the same manner as described by Avella et al. (2018). Below we present the separation algorithm for the capacity inequalities.

Vanderbei (2014) shows that sparse constraint matrices can be exploited in the simplex method to solve linear programs more efficiently. Thus, for potentially faster re-optimization of each linear program solved repeatedly in a B&C method, we choose the sparsest form of the SECs based on the size of their set $\mathcal{S}$. If $|\mathcal{S}| \leq \frac{|\mathcal{N}^C|}{2}$ we use the inequalities (11). If not, we use the following form:

$$\sum_{(i,j)\in(\overline{\mathcal{S}}:\overline{\mathcal{S}})} x_{ijt} + \frac{1}{2}\sum_{i\in\overline{\mathcal{S}}}(x_{0it}+x_{i0t}) - \frac{1}{2}\sum_{i\in\mathcal{S}}(x_{0it}+x_{i0t}) \leq \sum_{i\in\overline{\mathcal{S}}}\delta_{it} - \delta_{mt},$$
$$\forall\ \mathcal{S}\subset\mathcal{N}^C, |\mathcal{S}|\geq 2, \overline{\mathcal{S}} = \mathcal{N}^C\setminus\mathcal{S}, t\in\mathcal{T}, m\in\mathcal{S}. \quad (48)$$

For the CSECs we use the following form if $|\mathcal{S}| \leq \frac{|\mathcal{N}^C|}{2}$ (Adulyasak et al., 2014):

$$\min\{Q, \sum_{i\in\mathcal{S}}(\bar{L}_i - I_{it})\} \sum_{(i,j)\in(\mathcal{S}:\mathcal{S})} x_{ijt} \leq \min\{Q, \sum_{i\in\mathcal{S}}(\bar{L}_i - I_{it})\}\sum_{i\in\mathcal{S}}\delta_{it} - \sum_{i\in\mathcal{S}} q_{it},$$
$$\forall\ \mathcal{S}\subset\mathcal{N}^C, |\mathcal{S}|\geq 2, t\in\mathcal{T}. \quad (49)$$

If not, we use the complement of $\mathcal{S}$:

$$\min\{Q, \sum_{i\in\mathcal{S}}(\bar{L}_i - I_{it}\})(\sum_{(i,j)\in(\overline{\mathcal{S}}:\overline{\mathcal{S}})} x_{ijt} + \frac{1}{2}\sum_{i\in\overline{\mathcal{S}}}(x_{0it}+x_{i0t}) - \frac{1}{2}\sum_{i\in\mathcal{S}}(x_{0it}+x_{i0t})) \leq \min\{Q, \sum_{i\in\mathcal{S}}(\bar{L}_i - I_{it})\}\sum_{i\in\overline{\mathcal{S}}}\delta_{it} - \sum_{i\in\mathcal{S}} q_{it},$$
$$\forall\ \mathcal{S}\subset\mathcal{N}^C, |\mathcal{S}|\geq 2, \overline{\mathcal{S}} = \mathcal{N}^C\setminus\mathcal{S}, t\in\mathcal{T}. \quad (50)$$

4.3. Separation of the capacity inequalities

Desaulniers et al. (2016) presented three separation heuristics for the capacity inequalities. They used a modified version of the heuristic for the rounded capacity inequalities (Lysgaard, Letchford, & Eglese, 2004), in addition to two heuristics exploiting the fact that each variable in their formulation represents a vehicle route. As the latter two do not apply to the arc-flow formulations presented in this paper, we propose an alternative separation procedure based on solving a MILP, maximizing the violation of the cut given a fractional solution.

First, we introduce the variable σ_{ijt} which is equal to 1 if the arc (i,j) can be used in period t to serve some future residual demand $\overline{D}_{is}$ in period s, 0 otherwise. Let γ_{it} be equal to 1 if residual demand $\overline{D}_{it}$ is included in $\mathcal{U}$, 0 otherwise, and let α_{it} be an auxiliary variable ensuring that all arcs that can enter the set of residual demands $\mathcal{U}$ are included in the cut. Let w be a variable between 0 and $1-\epsilon$, where ϵ is an arbitrary small value, accounting for the rounding up of the right hand side. In addition, we define the set $\mathcal{T}_{is}^-$ to be the set of periods when residual demand $\overline{D}_{is}$ can be delivered. Let x^*_{ijt} be the value of the corresponding x_{ijt} variable in the optimal solution of the linear relaxation. Thus, the MILP consists of maximizing the violation of the cut, i.e. minimizing its left-hand side minus its right-hand side:

$$\min \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}^C}\sum_{t\in\mathcal{T}} x^*_{ijt}\sigma_{ijt} - c \quad (51)$$

$$c = \sum_{i\in\mathcal{N}^C}\sum_{t\in\mathcal{T}}\frac{\overline{D}_{it}}{Q}\gamma_{it} + w, \quad (52)$$

$$\gamma_{it} \leq \alpha_{is}, \qquad \forall\ i\in\mathcal{N}^C, t\in\mathcal{T}, s\in\mathcal{T}_{it}^-, \quad (53)$$

$$\sigma_{ijt} \geq \alpha_{jt} - \alpha_{it}, \qquad \forall\ i\in\mathcal{N}, j\in\mathcal{N}^C, t\in\mathcal{T}, \quad (54)$$

$$0 \leq w \leq 1-\epsilon, \quad (55)$$

$$c \geq 1, \text{integer}, \quad (56)$$

$$\sigma_{ijt} \geq 0, \qquad \forall\ (i,j)\in\mathcal{A}, t\in\mathcal{T}, \quad (57)$$

$$\alpha_{0t} = 0, \alpha_{it}\in\{0,1\}, \qquad \forall\ i\in\mathcal{N}, t\in\mathcal{T}. \quad (58)$$

Constraint (52) together with (56) represent the right-hand side of the cut. Constraints (53) ensure that if a residual demand $\overline{D}_{is}$ is included in the right-hand side of the cut $\mathcal{U}$, then all arcs able to serve this residual demand can potentially be included in the left-hand side of the cut. Constraints (54) make sure that we do not include arcs between the residual demands included in the set. Constraints (55)–(58) define the domains of the variables.

5. Computational study

In this section, we compare the computational performance of the B&C algorithms based on the SOTAF and the CSF, as well as the cut separation procedure proposed in this paper. In Section 5.1 we present the test instances used, while Section 5.2 compares exact and heuristic separation of the capacity inequalities (23), and the impact of excluding one family of valid inequalities at a time. Finally, Section 5.3 compares the SOTAF-BC with the CSF-BC on an extensive set of instances of the IRP.

The B&C algorithms are implemented in C++ using the commercial MILP solver Gurobi 9.0.2 with default settings, except for using a single thread and disabling the internal Gurobi cutting plane procedures. Graph algorithms from the Boost Graph Library (BGL) are

used to identify strongly connected components, connected components, and min-cut sets in the cut separation procedures. All tests were run on a 12 core Intel E5-2670v3 processor clocked at 2.3 GHz and 64 GB RAM. The computational time limit was set to 7200 s. To obtain primal bounds on the new instances we also set a time limit for solving the root node to 3600 s, still with a total computational time limit of 7200 s. It is clearly stated otherwise for the tests where this does not apply.

5.1. Test instances

We base all tests in this computational study on the benchmark instances first introduced by Archetti et al. (2007) for the single-vehicle case and later modified by Coelho et al. (2012a) to include the multi-vehicle version of the problem. The instances have either three or six time periods and involve from 5 to 50 and from 5 to 30 customers, respectively. They can further be divided into two sets having either high or low inventory costs. The vehicle fleet of the instances ranges from one to five vehicles, but the capacity of the vehicle fleet is constant, resulting in diminishing capacity for each vehicle when the number of vehicles increases.

One limitation of the benchmark instances described above is that they have constant demand over time for each customer and that the inventory capacity at each customer is an integer multiple of demand. This special structure is not a property of the IRP itself, and therefore we have modified the benchmark instances to create two new sets of instances. We do this by defining two sets of weights, $W_3 = \{0.85, 0.95, 1.2\}$ and $W_6 = \{0.7, 0.8, 0.9, 1.1, 1.2, 1.3\}$ used to modify the instances with three and six time periods, respectively. The sum of the weights is equal to the cardinality to make sure that the total demand over the planning horizon is no larger than in the original benchmark instances. In addition, the weights are chosen such that the inventory capacity of the customers are no longer an integer multiple of demand.

For each instance in the original benchmark set we define two cases, called *correlated demands* and *uncorrelated demands*, respectively. In the first case, we randomly draw one weight, F_t, for each time period t, from the set W_3 or W_6 (depending on the number of time periods in the instance) without replacement, and modify the demands as follows:

$$D_{it} = \lfloor F_t D_i \rfloor, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T} \text{ (Correlated demands)}.$$

In the second case, we again draw without replacement one weight randomly for each time period from the set W_3 or W_6, but repeat the procedure for each customer i, to obtain the weights F_{it}. These weights are then used to modify the demands as follows:

$$D_{it} = \lfloor F_{it} D_i \rfloor, \qquad \forall\, i \in \mathcal{N}^C, t \in \mathcal{T} \text{ (Uncorrelated demands)}.$$

5.2. Separation of capacity inequalities and impact of the valid inequalities

In this section, we investigate the effect on the time used, and the dual bound obtained, in the root node when separating the capacity inequalities (23) as proposed by Desaulniers et al. (2016) using the separation routine from the CVRPSEP package of Lysgaard et al. (2004), and using the separation routine described in Section 4.3, respectively. In addition, we give an overview of the impact of the valid inequalities presented in this paper, when excluding one family of valid inequalities at a time.

Table 1 compares the computational performance of solving the root node of the B&B tree using the model defined by (1)–(21) ('Standard'), with the addition of the capacity inequalities separated using the CVRPSEP library ('+CVRPSEP') or the exact MILP formulation ('+MILP'), respectively.

The results are aggregated over all the original benchmark instances per number of periods and vehicles (first two columns). For each of the three algorithm versions, we report the average relative optimality gap ('Gap') and the average computational time in seconds spent at the root node ('Seconds'). The optimality gap is calculated using the best-known upper bound from the exact methods presented in the literature (UB) and the final lower bound obtained in the root node (LB): $\text{Gap} = (UB - LB)/UB$. Finally the columns 'Imp.' show how much of the gap is closed by separating the capacity inequalities (23) by the CVRPSEP library and the exact separation algorithm: $\text{Imp.} = (\text{Gap}^{\text{standard}} - \text{Gap}^{j})/\text{Gap}^{\text{standard}}$ for $j =$ +CVRPSEP or +MILP.

Table 1 shows that the exact separation of the capacity inequalities gives a considerably better dual bound in the root node compared with separating them heuristically. Using only the CVRPSEP heuristics improve the root node gap by less than 2% on average and no more than 3.21% for the subset of six periods and five vehicles where it is most effective. On the other hand, using the MILP to separate the inequalities, improves the root node gap by more than 66% on average, and almost 80% for the subsets where it has the largest effect. It seems like these cuts are better for the three-period instances compared with the six-period instances. A possible explanation for this is that the majority of the customers requires a single visit in an optimal solution of the three-period instances whereas multiple visits are necessary for the six-period instances. Given that each residual demand can be covered by different potential visits to the corresponding customer, the capacity inequalities have less chances to be violated when more visits to each customer are required at optimality. For instance, any set of customers with positive residual demands impose at least one visit to this set, which is a much stronger bound when the optimal solution requires a single visit than when it requires multiple visits.

Another interesting observation from Table 1 is that separating these cuts using the CVRPSEP library seems to be slower for the three-period instances. However, this is only due to a few of the 50-customer instances, where the heuristic separation finds marginally improving cuts, leading to a large number of iterations. On the other hand, the exact separation algorithm uses few iterations because it finds better cuts in each of them. Thus, the total computational time becomes larger for the heuristic, even though it is significantly faster per iteration. For the six-period instances, however, the total time spent to solve the root node is $10 - 20$ times longer using the exact separation procedure compared with the CVRPSEP library. This final observation makes it clear that there is a big potential for speed-up when it comes to separating these cuts.

To the best of our knowledge, this is the first work including exact separation routines for both the capacity inequalities and the DR inequalities. Therefore, in Table 2 we give an overview of the impact of each family of valid inequalities by solving the root node, excluding a given family of valid inequalities one by one. These results are aggregated per number of vehicles for all the original benchmark instances. We present results for the CSF-BC without the capacity inequalities (- Cap), without the h-DR inequalities (- h-DR) and without both subfamilies of the DR inequalities (- (DR + h-DR)). They are compared with the CSF-BC and we can see that they all contribute significantly to the average gap at the root node. However, when excluding the capacity inequalities, the average gap at the root node increases from 1.36% to 3.34% indicating that they are the largest contributor to the dual bound obtained by the CSF-BC. When excluding the h-DR inequalities the increase of the average gap is much smaller, but still almost 17% worse than when including them. This indicates that they cut off additional solutions to the linear relaxation. Excluding also the simple DR inequalities the average gap is further increased by 8% compared with the CSF-BC.

In addition, it is interesting to see how the computational time increases when omitting the capacity inequalities. The observed

Table 1
Comparison between heuristic and exact separation of the capacity inequalities (23).

		Standard		+CVRPSEP			+MILP		
Periods	Vehicles	Gap (%)	Seconds	Gap (%)	Imp. (%)	Seconds	Gap (%)	Imp. (%)	Seconds
3	2	4.65	3	4.52	2.69	18	0.97	79.06	14
3	3	6.81	4	6.76	0.81	42	1.49	78.15	26
3	4	8.69	4	8.61	0.92	71	2.51	71.09	40
3	5	9.54	6	9.32	2.35	121	3.02	68.33	61
Average 3 periods		**7.43**	**4**	**7.30**	**1.63**	**63**	**2.00**	**73.07**	**35**
6	2	5.17	4	5.15	0.34	6	2.43	52.91	116
6	3	6.62	7	6.57	0.69	7	3.07	53.55	121
6	4	6.55	9	6.47	1.23	9	2.89	55.82	126
6	5	6.43	11	6.23	3.21	11	2.86	55.54	160
Average 6 periods		**6.19**	**8**	**6.10**	**1.40**	**8**	**2.82**	**54.52**	**130**
Average all		6.96	5	6.86	1.55	42	2.30	66.92	71

Table 2
Comparative results on the original benchmark instances.

	CSF-BC		- Cap		- h-DR		- (DR + h-DR)	
Vehicles	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds	Gap (%)	Seconds
1	**0.40**	306	0.71	299	0.43	10	0.45	**6**
2	**0.97**	1 386	2.58	1 869	1.22	80	1.32	**36**
3	**1.58**	1 528	3.91	1 942	1.87	126	1.99	**53**
4	**1.94**	1 564	4.73	1 788	2.26	180	2.42	**64**
5	**1.91**	1 539	4.76	1 804	2.18	249	2.35	**81**
Average all	**1.36**	1 264	3.34	1 540	1.59	129	1.70	**48**

Table 3
Comparative results on the six-period benchmark instances tested by Avella et al. (2018).

	Avella et al. (2018)			SOTAF-BC			CSF-BC		
Vehicles	RG (%)	OG (%)	Seconds	RG (%)	OG (%)	Seconds	RG (%)	OG (%)	Seconds
2	1.59	0.90	**4 114**	1.33	0.74	4 336	**1.32**	**0.69**	4 174
3	2.26	1.77	**5 050**	1.96	1.54	5 168	**1.95**	**1.51**	5 156
4	2.20	1.87	**5 124**	1.95	1.73	5 330	**1.93**	**1.70**	5 296
5	2.37	2.10	**5 190**	2.08	1.87	5 400	**2.07**	**1.84**	5 358
Average all	2.10	1.66	**4 869**	1.83	1.47	5 058	**1.82**	**1.44**	4 996

average time increase of 21.8% is solely due to generating more h-DR inequalities. Excluding the capacity inequalities results in adding 39% more h-DR inequalities, on average. On the other hand, the average number of added simple DR inequalities actually decreases by 5%, but with a negligible impact on the computational times. This suggests that the h-DR inequalities are the most computationally expensive inequalities to separate, while the separation routine for the simple DR inequalities performs similar to that of the capacity inequalities when considering computational times.

5.3. Computational results

In this section, we present the computational results comparing the CSF-BC proposed in this paper with the SOTAF-BC. First, we do a comparison on the benchmark instances proposed by Coelho et al. (2012a) in Section 5.3.1, before comparing the performance on the new sets of instances with uncorrelated and correlated time-varying demands in Sections 5.3.2 and 5.3.3, respectively. Detailed results can be found on the following webpage: http://axiomresearchproject.com/publications.

5.3.1. Comparing the SOTAF-BC with the CSF-BC on the original benchmark instances

Table 3 contains computational results comparing the results obtained with the SOTAF-BC and the CSF-BC with those of Avella et al. (2018). We limit the comparison to the six-period instances from the original set of benchmark instances also tested by Avella et al. (2018), namely the instances with 15, 20, 25 and 30 customers. Avella et al. (2018) ran their algorithm in two parts resulting in the total time limit being in the interval [3 600, 5 400]. Thus, we use a total time limit of 5 400 s for the tests presented in this section.

For each algorithm, we report the average gap at the root node (RG) and the average optimality gap at the end of the B&C (OG). These gaps are calculated as $(UB - LB)/UB$, where UB is the value of the best-known primal solution of the existing exact methods, and LB is either the lower bound achieved at the root node for RG or the best dual bound at the termination of the algorithm for OG. The table also reports the computational time in seconds. The best results for each measure across these three algorithms are highlighted in bold. It is worth noting that the time used to enumerate the customer schedules was less than 1.84 s for every instance and therefore not reported explicitly.

To get a fairer comparison of the computational times from Avella et al. (2018), they can be adjusted with the single thread rating, from https://www.cpubenchmark.net, of the different CPUs used between the two computational studies. The single thread rating for the CPU used by Avella et al. (2018) is 1459 and the rating for the CPU used in this paper is 1670. Thus, the computational times reported by Avella et al. (2018) can be adjusted by a factor of $1459/1670 = 0.8737$. From Table 3 it is clear that the SOTAF-BC yields better dual bounds than the algorithm proposed by Avella et al. (2018), both at the root node and at termination. The main difference between these two formulations and corresponding algorithms is the inclusion of the capacity inequalities (Desaulniers et al., 2016). Including these inequalities helps our implementation of the SOTAF-BC to obtain on average about 13% and 11% larger gaps at the root node and upon termination,

Table 4
Comparative results on the instances with time-varying and uncorrelated demands.

		SOTAF*-BC		CSF-BC		Improvement	
Periods	Vehicles	RG (%)	OG (%)	RG (%)	OG (%)	RG (%)	OG (%)
3	1	0.12	0.00	0.22	0.00	-80.8	0.00
3	2	0.13	0.00	0.26	0.00	-102.8	0.00
3	3	0.39	0.00	0.22	0.00	45.1	35.6
3	4	0.73	0.05	0.40	0.02	45.1	49.2
3	5	0.94	0.11	0.52	0.09	44.9	17.1
Average 3 periods		**0.46**	**0.03**	**0.32**	**0.02**	**30.2**	**27.2**
6	1	1.53	0.00	0.05	0.00	97.0	0.0
6	2	2.53	0.06	0.82	0.03	67.5	41.1
6	3	2.97	0.81	1.41	0.43	52.7	47.6
6	4	3.39	1.63	2.04	1.16	39.9	28.8
6	5	3.43	2.26	2.37	1.74	30.9	22.8
Average 6 periods		**2.77**	**0.95**	**1.34**	**0.67**	**51.8**	**29.3**
Average all		1.33	0.38	0.70	0.27	47.1	29.2

Table 5
Comparative computational times for the 671 instances with time-varying and uncorrelated demands solved to optimality by at least one of the algorithms.

		SOTAF*-BC		CSF-BC		Improvement		
Periods	Vehicles	RT	ET	RT	ET	RT (%)	ET (%)	# Optimal
3	1	380	396	944	983	-148.6	-148.5	100
3	2	1 603	1 627	1 134	1 173	29.3	27.9	100
3	3	1 562	1 702	1 070	1 191	31.5	30.0	100
3	4	1 636	2 047	1 144	1 540	30.1	24.8	96
3	5	1 562	2 203	975	1 475	37.5	33.0	84
Average 3 periods		**1 339**	**1 571**	**1 055**	**1 263**	**21.2**	**19.6**	**480/500**
6	1	1 144	1 188	280	284	75.5	76.1	60
6	2	2 193	2 791	1 015	1 475	53.7	47.2	58
6	3	1 314	3 172	631	1 571	51.9	50.5	34
6	4	312	2 318	58	1 000	81.4	56.8	22
6	5	18	907	8	580	55.2	36.0	17
Average 6 periods		**1 297**	**2 133**	**516**	**984**	**60.2**	**53.9**	**191/300**
Average all		1 327	1 731	902	1 184	32.0	31.6	671/800

respectively, than the B&C algorithm proposed by Avella et al. (2018). Due to the single item lot-sizing inequalities (that relies on constant demands and inventory capacity at each customer being an integer multiple of demand), it is not expected that the CSF-BC outperforms the SOTAF-BC, because the SOTAF-BC already represents a strong formulation tailored for the benchmark instances. However, it is still interesting to see that the more general CSF-BC performs slightly better than the SOTAF-BC, both concerning gaps and computational times. This supports the properties of the CSF discussed in Section 3.2.

The computational times reported by Avella et al. (2018) are slightly lower than our two implementations, mainly due to the implementation details making their total time limit to be in the interval [3 600, 5 400], potentially resulting in a lower time limit than the 5 400 s we use for our tests. However, this potential difference in the time limits does not seem to be crucial for the results, even when adjusting the times with the CPU ratings, especially looking at the 5 vehicle instances where we obtain lower root node gaps than Avella et al. (2018) report at the termination of their B&C algorithm. Thus, it is safe to claim that we have a state-of-the-art implementation of the strongest arc-flow formulation found in the literature, i.e. the SOTAF-BC does not perform worse than the algorithm proposed by Avella et al. (2018). This forms an important base to properly evaluate the performance of the CSF-BC on the modified instances.

5.3.2. Comparing the CSF-BC with the SOTAF*-BC on the instances with uncorrelated demands

In this section we evaluate the performance of the CSF-BC on the instances with time-varying and uncorrelated demands. Note that the single item lot-sizing inequalities (Avella et al., 2015) are not valid for the instances with time-varying demand. Thus, Table 4 compares the algorithm based on the SOTAF without these inequalities, denoted SOTAF*-BC, with the CSF-BC. Again, we report the root node gap (RG) and the optimality gap at termination (OG). In addition we report the relative improvement of the CSF-BC to the SOTAF*-BC for both gaps: $(\text{Gap}^{\text{SOTAF*}} - \text{Gap}^{\text{CSF}})/\text{Gap}^{\text{SOTAF*}}$.

From Table 4 it is clear that exploiting customer schedules has a positive impact on the dual bound both in the root node and when terminating the B&C. The improvement in the root node is especially good, with an average of 47.1%. We can also observe that the advantage of obtaining good dual bounds in the root node is still present at the termination of the B&C. The exceptions are the three-period instances with one and two vehicles, where the average dual bound in the root node obtained by the CSF-BC is significantly worse than the SOTAF*-BC. Out of 800 instances there are in total 55 instances where the SOTAF*-BC obtained a strictly better dual bound in the root node than the CSF-BC, and where 53 of them appear in the three-period instances. Despite having a worse dual bound on these 55 instances, the CSF-BC still solved all of them to optimality within the two-hour time limit.

We have identified two possible reasons why SOTAF*-BC sometimes obtains a better dual bound in the root node. Since the CSF-BC has a larger initial dual bound (the dual bound of the linear relaxation before starting the separation routine), the relative dual bound improvement is potentially lower than that of the SOTAF*-BC. Therefore, the CSF-BC might trigger the termination criterion prematurely making it start branching with a worse dual bound than the SOTAF*-BC. The use of the relative dual bound improvement to determine which separation algorithm to run also makes the CSF-BC call the costly *h*-DR separation algorithm more frequently. This is especially costly for the instances with few vehicles, because the route enumeration part of the algorithm cannot truncate routes as quickly when the vehicle capacity is high. There are 34 instances contributing to the SOTAF*-BC having a better average dual bound in the root node for the three-period instances with one and two vehicles, where for 26 of them the CSF-BC reached the time limit in the root node due to the separation of the *h*-DR inequalities. Nevertheless, on average the CSF-BC clearly obtains larger dual bounds than the SOTAF*-BC, especially for the more difficult six-period instances.

Furthermore, for the six-period instances the CSF-BC seems to induce the largest dual bounds in the root node when the vehicle capacity is large relative to the inventory capacity, which for these instances are when the number of vehicles is low. For instance, if the vehicle capacity Q would be equal to the demand we could deliver Q at every single period losing a lot of the benefit from the CSF. Thus, when the vehicle capacity decreases with respect to the inventory capacity we observe that the CSF becomes less effective, but there is still a significant gain by using this new formulation on these instances. However, we do not observe the same for the three-period instances due to the effects discussed above of using the dual bound improvement as the termination criterion and for guiding the separation routine.

We also compare the impact of using the CSF-BC on the computational time. Table 5 presents these results aggregated per number of periods and number of vehicles. Here we compare the computational time after solving the root node ('RT') and at termination ('ET'), while the improvement is calculated like the gaps presented in Table 4. The column '# Optimal' reports the number of instances solved to optimality by at least one of the algorithms out of the total number of instances. Note that the instances where both algorithms reached the time limit are left out of the comparison to not skew the results.

Table 6
Comparative results on the instances with time-varying and correlated demands.

		SOTAF*-BC		CSF-BC		Improvement	
Periods	Vehicles	RG (%)	OG (%)	RG (%)	OG (%)	RG (%)	OG (%)
3	1	0.41	0.00	0.12	0.00	70.0	0.00
3	2	0.36	0.01	0.24	0.01	35.0	27.7
3	3	0.49	0.02	0.32	0.01	35.1	39.3
3	4	0.87	0.25	0.62	0.24	29.4	3.1
3	5	1.07	0.47	0.80	0.45	24.8	5.3
Average 3 periods		**0.64**	**0.15**	**0.42**	**0.14**	**34.5**	**5.7**
6	1	2.92	0.00	0.27	0.00	90.8	0.00
6	2	3.52	0.33	1.25	0.26	64.5	20.8
6	3	3.77	1.60	2.08	1.21	44.8	24.5
6	4	4.44	2.83	3.05	2.36	31.4	16.4
6	5	4.56	3.37	3.37	2.93	26.1	13.1
Average 6 periods		**3.84**	**1.62**	**2.00**	**1.35**	**47.9**	**16.9**
Average all		1.84	0.70	1.01	0.59	45.0	15.4

Table 7
Comparative computational times for the 634 instances with time-varying and correlated demands solved to optimality by at least one of the algorithms.

		SOTAF*-BC		CSF-BC		Improvement		
Periods	Vehicles	RT	ET	RT	ET	RT (%)	ET (%)	# Optimal
3	1	232	258	356	379	-53.4	-47.1	100
3	2	1 242	1 342	729	848	41.3	36.8	98
3	3	1 381	1 639	1 133	1 404	18.0	14.3	99
3	4	1 241	1 715	888	1 325	28.4	22.7	87
3	5	1 083	1 752	845	1 465	22.0	16.3	79
Average 3 periods		**1 026**	**1 311**	**785**	**1 061**	**23.6**	**19.1**	**463/500**
6	1	1 209	1 263	205	225	83.1	82.2	60
6	2	1 834	2 623	722	1 197	60.6	54.4	48
6	3	766	1 666	285	1 186	62.8	28.8	26
6	4	312	2 262	104	1 390	66.5	38.6	20
6	5	190	1 630	41	1 365	78.7	16.3	17
Average 6 periods		**1 111**	**1 859**	**334**	**893**	**69.9**	**52.0**	**171/299**
Average all		1 049	1 459	663	1 016	36.8	30.4	634/799

Here we can see that on average the CSF-BC solves the root node and terminates about 30% faster than the SOTAF*-BC. However, we can observe that in addition to obtaining a worse average dual bound in the root node for the three-period instances with one vehicle, the CSF-BC spends too much time separating valid inequalities without sufficient contribution to the dual bound improvement, which ultimately results in a much larger average computational time than the SOTAF*-BC. On the other hand, it is interesting to see that despite having a worse average dual bound in the root node, the CSF-BC makes up for it with close to 28% reduction in the average computational time on the three-period instances with two vehicles. Once the complexity increases either by considering more vehicles or more periods, we observe a considerable reduction in computational times. Especially for the six-period instances, we can observe that the time spent in the B&B tree (ET - RT) is shorter for the CSF-BC than the SOTAF*-BC, indicating that a larger dual bound in the root node speeds up the B&B part of the B&C algorithm.

5.3.3. Comparing the CSF-BC and the SOTAF-BC for the instances with correlated demands*

The tables in this section have the same format as the ones in Section 5.3.2. Table 6 compares the gaps of the SOTAF*-BC with the CSF-BC on the set of instances with time-varying and correlated demands. Also for these instances, we observe a substantial improvement in the dual bound at both the root node and at termination. However, at termination, we can see that the benefit of using a stronger formulation (CSF) is much larger for the six-period instances than for the three-period instances. For the three-period instances with one, three and four vehicles, it seems like the benefit of having a larger dual bound in the root node by using the CSF-BC is more or less lost by the faster node processing of the SOTAF*-BC. It is therefore positive to see that this effect is maintained throughout the B&B search for the more complex six-period instances, which is an indication that the CSF-BC is better suited for more complex instances.

For the same reasons as mentioned in Section 5.3.2 we point out that there are, in fact, 68 three-period and 6 six-period instances where the SOTAF*-BC obtained larger dual bounds in the root node. However, the CSF-BC never obtained a worse dual bound at termination of the B&C for any of the instances. Here we can also observe that for both the three-period and the six-period instances the average dual bound in the root node for the CSF-BC is larger for a low number of vehicles, i.e. $Q > \bar{L}_i$, compared to a high number of vehicles. The reason we observe this for the three-period correlated demands instances, but not for the three-period uncorrelated demands instances, is that the CSF-BC only reached the one-hour time limit on 12 of the 68 mentioned instances where SOTAF*-BC obtained a better dual bound in the root node. Thus, the main reason why the SOTAF*-BC obtained a better dual bound in the root node on these instances is that the CSF-BC prematurely started branching. Comparing the average dual bound for the three-period correlated demands instances with the three-period uncorrelated demands instances, we can observe that the largest impact on the average dual bound in the root node seems to be when the CSF-BC reaches the one-hour time limit by spending too much time separating h-DR inequalities.

Table 7 compares the average computational times obtained by the two algorithms. Here we can identify a similar trend as for the uncorrelated case. We observe a significant speed-up on average by using the CSF-BC, but where the SOTAF*-BC is much faster on the relatively easy three-period and one-vehicle instances. Here we also see that the larger dual bound in the root node speeds up the B&B search for the six-period instances, just like for the uncorrelated case.

In addition, it is worth noting that the correlated demands instances (634 instances solved to optimality) seem to be more difficult to solve than the uncorrelated demands instances (671 instances solved to optimality). Note also that one of the 800 correlated demands instances is infeasible and thus not considered in the tables.

6. Concluding remarks

In this work, we have proposed a new Dantzig-Wolfe reformulation (customer schedule formulation) for the single-depot multi-vehicle inventory routing problem utilizing the concept of customer schedules. We show that the polytope defined by the linear relaxation of the customer schedule formulation is a proper subset of the polytope defined by the linear relaxation of the original formulation in some cases, but never the opposite. In addition, we provide a thorough computational study demonstrating the strength of the customer schedule formulation in practice. We have implemented a state-of-the-art B&C algorithm used to evaluate the performance of the B&C algorithm based on the customer schedule formulation. Comparing the customer schedule-based B&C algorithm to the state-of-the-art algorithm we observe that the average dual bound is considerably improved, giving a substantial reduction in the average computational time. Regarding the state-of-the-art B&C algorithm we have also gained additional insight into the separation routines showing the potential for speed-up when it comes to separating the capacity inequalities. Since the focus of this paper is to investigate the strength of the customer schedule formulation, improving the separation routine for the capacity inequalities are better left to future research, but we find it an inter-

esting observation worth pointing out. We believe this also applies to the DR inequalities and could form a basis for interesting future research.

Acknowledgements

This work was partly funded by the Research Council of Norway through the AXIOM project [grant number: 263031]. This support is gratefully acknowledged. The authors also thank Professor Pasquale Avella for providing his B&C code which was very helpful to efficiently reimplement the separation algorithms for the DR inequalities. Finally, the authors thank the anonymous reviewers for their valuable feedback that helped improve the presentation of this work.

Appendix A. Proof of propositions 2 and 3

This section gives the proofs of Propositions 2 and 3.

A1. Proof of Proposition 2

Proof. (by contradiction) Let us first present the proof for the valid inequalities (20). For these inequalities, let us assume that the convex combination of customer schedules can give a lower bound on the number of visits in a given time interval $[t_1, t_2]$ strictly worse than the known valid lower bound given by the right-hand side of (20), i.e.,

$$\left\lceil \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) \Big/ \min\{\bar{L}_i, Q\} \right\rceil > \sum_{t'=t_1}^{t_2} \sum_{\omega \in \Omega_i} A_{it'\omega} \lambda_{i\omega}. \tag{A.1}$$

Then there exists at least one customer schedule that gives a lower bound strictly worse than the right-hand side of (20).

$$\exists\ \omega \in \Omega_i : \left\lceil \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) \Big/ \min\{\bar{L}_i, Q\} \right\rceil > \sum_{t'=t_1}^{t_2} A_{it'\omega}. \tag{A.2}$$

However, this customer schedule ω must then be as follows:

$$\Rightarrow \left\lceil \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) \Big/ \min\{\bar{L}_i, Q\} \right\rceil \geq \sum_{t'=t_1}^{t_2} A_{it'\omega} + 1, \tag{A.3}$$

since $A_{it'\omega} \in \{0, 1\}$,

$$\Rightarrow \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) \Big/ \min\{\bar{L}_i, Q\} > \sum_{t'=t_1}^{t_2} A_{it'\omega} \tag{A.4}$$

$$\Rightarrow \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) > \sum_{t'=t_1}^{t_2} \min\{\bar{L}_i, Q\} A_{it'\omega} \tag{A.5}$$

$$\Rightarrow \left(\sum_{t'=t_1}^{t_2} D_{it'} - \bar{L}_i \right) > \sum_{t'=t_1}^{t_2} Q_{it'\omega}, \qquad \text{since } Q_{it'\omega} \leq \min\{\bar{L}_i, Q\}, \tag{A.6}$$

$$\Rightarrow \sum_{t'=t_1}^{t_2} D_{it'} - \sum_{t'=t_1}^{t_2} Q_{it'\omega} > \bar{L}_i. \tag{A.7}$$

However, this means that the difference between the amount of product consumed and delivered is strictly greater than the upper limit of the inventory, something that would lead to $s_{it_2} < 0$. Since by definition $s_{it} \geq 0$, such a customer schedule cannot exist. Thus, the CSF must induce a lower bound on the number of visits in a time interval at least as large as that imposed by the valid inequalities (20). We can use the same proof scheme for valid inequalities (19) if we substitute $\bar{L}_i$ with I_i, and t_1 with the first time period.

Similarly, a solution in R^{CSF} may violate an inequality (21) only if there exists at least one customer schedule giving a lower bound on the number of visits strictly worse than the right-hand side of (21):

$$\exists\ \omega \in \Omega_i : \left(\sum_{t'=t_1}^{t_2} D_{it'} - s_{i(t_1-1)} \right) \Big/ \min\{\bar{L}_i, Q, \sum_{t'=t_1}^{t_2} D_{it'}\} > \sum_{t'=t_1}^{t_2} A_{it'\omega}. \tag{A.8}$$

Let us now differentiate between the case (i) where $\min\{\bar{L}_i, Q, \sum_{t'=t_1}^{t_2} D_{it'}\} = \min\{\bar{L}_i, Q\}$, and the case (ii) where $\min\{\bar{L}_i, Q, \sum_{t'=t_1}^{t_2} D_{it'}\} = \sum_{t'=t_1}^{t_2} D_{it'}$. Following the same argumentation as in inequalities (A.4)–(A.7) for case (i), we obtain:

$$\sum_{t'=t_1}^{t_2} D_{it'} - \sum_{t'=t_1}^{t_2} Q_{it'\omega} > s_{i(t_1-1)}.$$

Such a customer schedule is infeasible and cannot exist, as it would lead to $s_{it_2} < 0$. For case (ii) the right-hand side of (21) lies in the interval (0,1] (we can ignore non-positive values for the right-hand side as they never become binding), because the nominator is always smaller than or equal to the denominator. Since $A_{it'\omega} \in \{0, 1\}$, the only way for the statement (A.8) to be true for case (ii) is if $\sum_{t'=t_1}^{t_2} A_{it'\omega} = 0$. However, whenever $\sum_{t'=t_1}^{t_2} D_{it'} - s_{i(t_1-1)} > 0$ such a customer schedule cannot exist as it would lead to $s_{it_2} < 0$ and thus an infeasible solution. Therefore, the CSF must have a lower bound on the number of visits in a given time interval at least as large as the one given by the inequality (21). □

A2. Proof of Proposition 3

Proof. The single-item reformulation used by Avella et al. (2015) to propose a set of valid inequalities for a given customer i writes as follows:

$$s'_{t-1} + x_t = 1 + s'_t, \qquad \forall\ t \in \mathcal{T} \setminus \{1\}, \tag{A.9}$$

$$0 \leq s'_t \leq V - 1, \qquad \forall\ t \in \mathcal{T}, \tag{A.10}$$

$$x_t \leq V \delta_t, \qquad \forall\ t \in \mathcal{T}, \tag{A.11}$$

$$x_t \geq 0, \qquad \forall\ t \in \mathcal{T}, \tag{A.12}$$

$$\delta_t \in \{0, 1\}, \qquad \forall\ t \in \mathcal{T}, \tag{A.13}$$

where the customer index is dropped. This reformulation is valid for the IRP if and only if $\frac{\bar{L}_i}{D_i} = V$ for some integer V. If we substitute for V in the formulation, and then multiply both sides of all constraints with D_i, and replace $D_i s'_t = s_t$ and $D_i x_t = q_t$, we get:

$$s_{t-1} + q_t = D_i + s_t, \qquad \forall\ t \in \mathcal{T} \setminus \{1\}, \tag{A.14}$$

$$0 \leq s_t \leq \bar{L}_i - D_i, \qquad \forall\ t \in \mathcal{T}, \tag{A.15}$$

$$q_t \leq \bar{L}_i \delta_t, \qquad \forall\ t \in \mathcal{T}, \tag{A.16}$$

$$q_t \geq 0, \qquad \forall\ t \in \mathcal{T}, \tag{A.17}$$

$$\delta_t \in \{0, 1\}, \qquad \forall\ t \in \mathcal{T}. \tag{A.18}$$

Let P_i^2 be the set of feasible solutions defined by constraints (A.14)–(A.18) for some customer i. We see that constraints (A.14) and (A.15) are the same as constraints (3) and (5), constraints (A.16) is a relaxed version of constraints (8), and constraints (A.17) and (A.18) are the same as constraints (13) and (14). Thus, $P_i \subseteq P_i^2$, and $Conv(P_i) \subseteq Conv(P_i^2)$. Let R_i and R_i^2 be the linear relaxations of P_i and P_i^2, respectively. Since the Dantzig-Wolfe reformulation creates a representation of the convex hull of P_i, then $R_i = Conv(P_i)$, and consequently $R_i \subseteq Conv(P_i^2)$. Thus, there cannot exist any valid inequalities for $Conv(P_i^2)$ that are violated by a solution in R_i. □

References

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

Alvarez, A., Cordeau, J.-F., Jans, R., Munari, P., & Morabito, R. (2021). Inventory routing under stochastic supply and demand. *Omega, 102*, 102304.

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

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

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

Archetti, C., Boland, N., & Speranza, M. G. (2017). A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing, 29*(3), 377–387.

Archetti, C., Christiansen, M., & Grazia Speranza, M. (2018). Inventory routing with pickups and deliveries. *European Journal of Operational Research, 268*(1), 314–324.

Archetti, C., Speranza, M. G., Boccia, M., Sforza, A., & Sterle, C. (2020). A branch-and–cut algorithm for the inventory routing problem with pickups and deliveries. *European Journal of Operational Research, 282*(3), 886–895.

Avella, P., Boccia, M., & Wolsey, L. A. (2015). Single-item reformulations for a vendor managed inventory routing problem: Computational experience with benchmark instances. *Networks, 65*(2), 129–138.

Avella, P., Boccia, M., & Wolsey, L. A. (2018). Single-period cutting planes for inventory routing problems. *Transportation Science, 52*(3), 497–508.

Baller, A. C., Dabia, S., Desaulniers, G., & Dullaert, W. E. H. (2021). The inventory routing problem with demand moves. In *SN operations research forum: vol. 2* (pp. 1–61). Springer.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., & Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces, 13*(6), 4–23.

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control. In M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *Hybrid metaheuristics* (pp. 171–177). Berlin, Heidelberg: Springer Berlin Heidelberg.

Chitsaz, M., Cordeau, J.-F., & Jans, R. (2019). A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing, 31*(1), 134–152.

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

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

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

Coelho, L. C., & Laporte, G. (2014). Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics, 155*(Supplement C), 391–397.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research, 8*(1), 101–111.

Desaulniers, G., Rakke, J. G., & Coelho, L. C. (2016). A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science, 50*(3), 1060–1076.

Diniz, P., Martinelli, R., & Poggi, M. (2020). An efficient matheuristic for the inventory routing problem. In M. Baïou, B. Gendron, O. Günlük, & A. R. Mahjoub (Eds.), *Combinatorial optimization* (pp. 273–285). Cham: Springer International Publishing.

Drexl, M. (2013). A note on the separation of subtour elimination constraints in elementary shortest path problems. *European Journal of Operational Research, 229*(3), 595–598.

Guimarães, T. A., Schenekemberg, C. M., Coelho, L. C., Scarpin, C. T., & Pécora, J. E., Jr (2020). Mechanisms for feasibility and improvement for inventory-routing problems. *Research report, CIRRELT-2020-12*. Université de Montréal, Canada. https://www.cirrelt.ca/documentstravail/cirrelt-2020-12.pdf

Guimarães, T. A., Coelho, L. C., Schenekemberg, C. M., & Scarpin, C. T. (2019). The two-echelon multi-depot inventory-routing problem. *Computers & Operations Research, 101*, 220–233.

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

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, 100*(2), 423–445.

Manousakis, E., Repoussis, P., Zachariadis, E., & Tarantilis, C. (2021). Improved branch-and-cut for the inventory routing problem based on a two-commodity flow formulation. *European Journal of Operational Research, 290*(3), 870–885.

Picard, J.-C., & Ratliff, H. D. (1973). A graph-theoretic equivalence for integer programs. *Operations Research, 21*(1), 261–269.

Solyal, 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 Science, 45*(3), 335–345.

Vadseth, S. T., Andersson, H., & Stålhane, M. (2021). An iterative matheuristic for the inventory routing problem. *Computers & Operations Research, 131*, 105262.

Vanderbei, R. J. (2014). Linear programming, foundations and extensions. *International Series in Operations Research & Management Science*: vol. 196 ((4th ed.)). New York: Springer US.