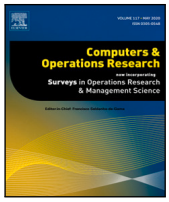
Computers & Operations Research 159 (2023) 106353

Contents lists available at ScienceDirect

Computers and Operations Research

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



# A branch-and-cut embedded matheuristic for the inventory routing problem

Jørgen Skålnes *, Simen T. Vadseth, Henrik Andersson, Magnus Stålhane

*Norwegian University of Science and Technology, Norway*

ARTICLE INFO

Dataset link: axiomresearchproject.com

*Keywords:*
Inventory routing problem
Matheuristic
Branch-and-cut
DIMACS Implementation Challenge

ABSTRACT

This paper presents an improved version of the solution method that won the inventory routing problem track of the 12th DIMACS Implementation Challenge. The solution method is a branch-and-cut embedded matheuristic where a matheuristic is called every time a new primal solution is found in a branch-and-cut method. The matheuristic consists of a construction heuristic and an improvement heuristic. The construction heuristic uses a giant tour method and a shifting assignments method to generate a set of promising routes which, in turn, are combined into a feasible solution to the problem by solving a route-based mathematical program. The improvement heuristic then solves a series of extended route-based mathematical programs where clusters of customers may be inserted and/or removed from the routes of the initial feasible solution. We have, to the best of our knowledge, gathered all detailed results from previously published methods for the inventory routing problem and made this overview available online. Compared with these results, the proposed method found the best-known solution for 741 out of 878 multi-vehicle inventory routing instances, where 247 of them are strictly better than the previously best-known solutions. Furthermore, we prove optimality for 458 of these solutions. The proposed method is also able to find the best-known solution for 116 out of 226 benchmark instances for the split delivery vehicle routing problem, and improve three best-known solutions from the CVRPlib for the capacitated vehicle routing problem.

## 1. Introduction

In this paper, we present an improved version of the solution method that won the inventory routing problem (IRP) track of the 12th DIMACS Implementation Challenge (DIMACS, 2022). The method was first developed for the IRP, but we show that it can be generalized and that it produces high-quality solutions also for the split delivery vehicle routing problem (SDVRP) and that the improvement heuristic part of the methodology finds new best-known solutions for instances of the capacitated vehicle routing problem (CVRP).

The IRP arises within the business practice of vendor-managed inventory, and the proposed algorithm is tailored to tackle the single-depot multi-vehicle IRP version. In this problem, a single depot has to manage the inventory of a set of customers so that each customer can meet its demand for a single product over a set of discrete time periods. A fleet of vehicles is used to deliver products to the customers. The decision maker must simultaneously decide (1) when to visit each customer, (2) how much to deliver to each customer with each vehicle, and (3) how to route the available vehicles to minimize the sum of the transportation cost and the inventory holding cost.

The CVRP and SDVRP both consist of a single depot that must deliver a given demand to a set of customers in a single time period using a fleet of vehicles. The only difference is that the CVRP limits each customer to be served by one vehicle, while the SDVRP allows multiple visits to each customer. The SDVRP can be seen as a special case of the IRP, where decision (2) and (3) must be made, but where decision (1) is predetermined as there is only a single time period. Therefore, we mainly focus on the IRP, which is the more general variant of the problem.

Bell et al. (1983) were the first to describe the IRP when studying it in the context of industrial gas distribution. The IRP has later received much attention, and there have been numerous advances in both exact and heuristic methods in the last two decades. The exact methods can be categorized to belong within a branch-and-cut (B&C) framework (Archetti et al., 2007; Solyalı and Süral, 2011; Coelho and Laporte, 2014; Adulyasak et al., 2014; Avella et al., 2015, 2018; Manousakis et al., 2021; Guimarães et al., 2023; Skålnes et al., 2022) or a branch-price-and-cut (BP&C) framework (Desaulniers et al., 2016). Archetti et al. (2007) proposed a B&C method for the single-vehicle IRP and showed the benefit of using the maximum level (ML) inventory policy compared with the order-up-to level inventory policy. They also published the first set of benchmark instances for the single-vehicle IRP, consisting of 5 to 50 customers, and three and six time periods.

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

https://doi.org/10.1016/j.cor.2023.106353
Received 3 April 2023; Received in revised form 28 June 2023; Accepted 11 July 2023
Available online 17 July 2023

The B&C methods of Avella et al. (2018), Manousakis et al. (2021) and Skålnes et al. (2022) are all based on two-index vehicle-flow formulations. Avella et al. (2018) proposed a new set of valid inequalities, the disjoint route (DR) inequalities, and implemented and tested two subfamilies of these. Manousakis et al. (2021) extended a two-vehicle flow formulation to a new two-commodity flow formulation inspired by the flow formulation in Baldacci et al. (2004). Skålnes et al. (2022), on the other hand, used a customer schedule reformulation, i.e., a Dantzig–Wolfe reformulation of the polyhedron connecting delivered quantities to customer visits, and adapted the capacity inequalities of Desaulniers et al. (2016) to be used within the context of B&C. Guimarães et al. (2023) proposed new mechanisms for feasibility and improvement of IRP solutions that can be embedded within both exact methods and heuristics. Archetti and Ljubić (2022) did a thorough comparison of the strength of two- and three-index vehicle flow formulations, showing that the optimal value of the linear relaxation is not improved by using a three-index vehicle flow formulation. The smaller number of variables and constraints in the two-index vehicle flow formulation is then a clear advantage, and with the results of Avella et al. (2018), Manousakis et al. (2021), and Skålnes et al. (2022) it is clear that a two-index vehicle flow formulation seems to be better than a three-index vehicle flow formulation in a B&C method. Even though the B&C method based on the two-commodity flow formulation of Manousakis et al. (2021) obtained better dual bounds than that of Avella et al. (2018) at termination, it did not obtain better dual bounds at the root node of the B&B tree. Thus, the formulation of Avella et al. (2018) seems to be the stronger of the two. Another method that obtained good dual bounds is the BP&C method of Desaulniers et al. (2016). On average, this method has produced the best dual bounds, and it outperformed the B&C methods on the four and five vehicle instances.

The heuristic methods for the IRP can mainly be divided into matheuristics (Archetti et al., 2012; Adulyasak et al., 2014; Archetti et al., 2017; Chitsaz et al., 2019; Diniz et al., 2020; Alvarez et al., 2020; Archetti et al., 2021; Vadseth et al., 2021; Solyalı and Süral, 2022; Achamrah et al., 2022; Vadseth et al., 2023) or metaheuristics (Alvarez et al., 2018; Sakhri et al., 2022). Archetti et al. (2012) focused on the single-vehicle IRP and proposed a hybrid heuristic that combines a tabu search scheme with an improvement phase that solved mixed integer linear programs (MILPs) to explore the neighborhood of the current incumbent solution. They also proposed a set of large benchmark instances consisting of 50, 100, and 200 customers and six time periods. Archetti et al. (2017) presented a matheuristic designed to tackle the multi-vehicle IRP. Here, they extended the tabu search scheme originally developed for the single-vehicle version and combined it with two MILPs to produce an initial solution or improve the current incumbent solution. Chitsaz et al. (2019) introduced a decomposition matheuristic originally designed for the assembly routing problem and demonstrated that it also works well for the IRP. Diniz et al. (2020) proposed a matheuristic that used an iterated local search algorithm, with a randomized variable neighborhood descent, to find the routes in each time period. Then, the matheuristic moved on to solve a network flow problem to determine the delivered quantities by using an enhanced network simplex method. Alvarez et al. (2020) presented a hybrid heuristic based on the combination of an iterated local search metaheuristic and two mathematical programming components to solve the IRP with perishable products. The authors also showed that the method worked well on the standard IRP. Archetti et al. (2021) proposed a kernel search matheuristic, that used information gathered by a tabu search to create a sequence of MILPs, which produced high-quality solutions. Vadseth et al. (2021) introduced a matheuristic that iteratively solves a route-based MILP, where the set of routes is altered between each iteration. The initial route set is created from a giant tour using a split algorithm. This is currently the solution method with the best average performance on the set of large benchmark instances, which are the set of instances where heuristics outperform exact methods.

Solyalı and Süral (2022) presented a matheuristic where three different MILPs are solved sequentially. The first two are used to construct a feasible solution, while the third one improves the solution by finding the best feasible routes within different giant tours created from the current solution. Achamrah et al. (2022) proposed a two-phase matheuristic that combines mathematical programming with a genetic algorithm and simulated annealing. The method was developed for the IRP with transshipments, but the authors also tested it on the standard IRP. Vadseth et al. (2023) developed a multi-start matheuristic for the production routing problem (PRP) that uses one MILP to construct solutions and another to improve them. The first MILP is modified in each restart, and the method proved efficient for the IRP as well. Metaheuristics for the IRP have been presented by Alvarez et al. (2018), and Sakhri et al. (2022). The former introduced a simulated annealing algorithm and an iterated local search algorithm (the same algorithm used in Alvarez et al. (2020)) and were able to produce good solutions for the IRP in a very short time. The latter proposed a memetic algorithm based on a genetic algorithm and a variable neighborhood search method.

The SDVRP was originally proposed by Dror and Trudeau (1989, 1990), as a relaxation of the CVRP. It was further shown by Archetti et al. (2006a) that the theoretical potential savings made by this relaxation are as large as 50%. The problem has received significant attention in the literature, with several exact and heuristic solution methods proposed. Like the IRP, exact methods for the SDVRP have mainly been based on B&C, with contributions from Dror et al. (1994), Belenguer et al. (2000), Archetti et al. (2014), Ozbaygin et al. (2018) and Munari and Savelsbergh (2022), while the only branch-and-price (B&P) approach for the SDVRP is proposed by Archetti et al. (2011). The results show that exact methods are able to solve benchmark instances of up to 80 customers to optimality, with one 100-customer instance solved by Archetti et al. (2014).

Due to the limited success of exact methods on larger instances, a large number of heuristics have been proposed. Most approaches are either local search-based metaheuristics such as (iterated) local search (Derigs et al., 2010; Silva et al., 2015), tabu search (Archetti et al., 2006b), and variable neighborhood search (Aleman et al., 2010), or evolutionary algorithms such as scatter search (Campos et al., 2008), genetic algorithms (Wilck and Cavalier, 2012), particle swarm algorithms (Shi et al., 2018), and memetic algorithms (Boudia et al., 2007; He and Hao, 2022). Only a few attempts have been made to develop matheuristics for this problem (Chen et al., 2007; Archetti et al., 2008; Jin et al., 2008), and none of them have produced any of the currently best-known solutions to the well-established benchmark instances for the SDVRP (He and Hao, 2022).

As seen in the above review of the literature, there are many solution methods for the IRP and the SDVRP. Exact solution methods for both problems are dominated by B&C approaches, while the best-performing heuristic approaches differ. For the IRP, matheuristics have been prevalent in recent years, while for the SDVRP, the recent approaches are mainly based on evolutionary algorithms, inspired by those used for the CVRP. A possible explanation is that changes to an IRP solution are seldom local, but may include changing the decisions in many time periods. This makes the evaluation of a local search operator very complex and time-consuming, which discourages the use of local-search based methods for the IRP.

In this paper we present a general B&C embedded matheuristic to solve the IRP and the SDVRP. Our main contributions are the following:

- We design a new matheuristic that extends the work of Vadseth et al. (2021) and Vadseth et al. (2023).
- We formulate a new improvement MILP within the improvement heuristic, allowing for clusters of customers to be removed from, or inserted into, routes of a solution.
- We develop a new route generating heuristic as part of a construction heuristic.

- We propose tightened versions of the valid inequalities presented by Coelho and Laporte (2014).
- We adapt the B&C method by Skålnes et al. (2022) to better handle large instances.
- We create, to the best of our knowledge, an overview of all known detailed results of methods that solve the multi-vehicle benchmark instances of the IRP.

The proposed method has found the best-known solution for 741 of the 878 multi-vehicle benchmark instances for the IRP, where 247 of them are strictly better than the previously best-known solutions found in the literature. Furthermore, 458 of the 741 solutions are proved to be optimal. This clearly outperforms all other solution methods for the IRP by a large margin and establishes the proposed method as state-of-the-art. We have also demonstrated that the proposed matheuristic is significantly better than the previous version, which won the IRP track of the DIMACS Implementation Challenge. Furthermore, we have collected, to the best of our knowledge, all published results from every published paper on the standard IRP and made this overview easily available online. We believe that this contribution is of great value to anyone interested in doing further research on the IRP. In addition, the proposed method produces results for the SDVRP that are competitive with the state-of-the-art and finds the best-known solution for 116 out of 226 benchmark instances, where 14 of them are strictly better than the previously best-known solutions found in the literature. Finally, the improvement heuristic is able to improve the best-known solution for three of the ten large benchmark instances for the CVRP released by Arnold et al. (2019).

The remainder of the paper is organized as follows. Section 2 defines the IRP and presents a mathematical formulation of the problem. The proposed method and each of its components are described in detail in Section 3, while our computational analyses for the IRP are reported in Section 4. The computational results for the SDVRP and CVRP are presented in Section 5. Lastly, our concluding remarks are presented in Section 6.

## 2. Problem description and mathematical model

In this section, we provide a detailed description of the IRP and present a mathematical formulation that both serves as a clear definition of the problem and forms the base of the B&C method presented in Section 3.3. To limit the length of the manuscript, we do not provide a full description of the SDVRP or the CVRP. Instead, we describe the modifications of the presented model necessary to define the other routing problems in the relevant sections.

The IRP version considered in this paper consists of a single depot, denoted 0, that manages the inventories of a set of customers $\mathcal{N}^C$ over a set of discrete time periods $\mathcal{T}$ so that each customer $i \in \mathcal{N}^C$ in each time period $t \in \mathcal{T}$ can satisfy its demand $D_{it}$ of a single product. The depot produces $S_t$ units of this product in time period $t$ and has to deliver the necessary quantities to the customers. To accommodate the deliveries, the depot must route a fleet of $V$ homogeneous vehicles, each with a capacity to hold $Q$ units of the product, such that the sum of transportation costs and inventory holding costs is minimized. Each customer and the depot $i \in \mathcal{N}^C \cup \{0\}$ have a lower and upper inventory level, $L_i$ and $U_i$, respectively. The depot and each customer $i \in \mathcal{N}^C \cup \{0\}$ have an initial inventory level $I_i$ at the beginning of the first time period. Let $I_{it} = \max\{I_i - \sum_{s=0}^{t} D_{is}, 0\}$ be the inventory that is left from the initial inventory at customer $i \in \mathcal{N}^C$ in time period $t \in \mathcal{T}$.

This problem can be modeled on a graph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \mathcal{N}^C \cup \{0\}$ is the set of nodes and $\mathcal{A} = \{(i,j) \in \{\mathcal{N} \times \mathcal{N}\} \mid i \neq j\}$ is the set of arcs connecting the nodes in the graph. Let $C_{ij}$ be the cost of traversing arc $(i,j)$ and let $C_i^H$ be the unit inventory holding cost of node $i \in \mathcal{N}$. Let $x_{ijt}$ be 1 if arc $(i,j) \in \mathcal{A}$ is traversed in time period $t \in \mathcal{T}$, and 0 otherwise. Let $s_{it}$ be the inventory level at the depot or a customer $i \in \mathcal{N}$ at the end of time period $t \in \mathcal{T}$ and let $\delta_{it}$ be 1 if customer $i \in \mathcal{N}^C$ is visited in time period $t \in \mathcal{T}$, and 0 otherwise. Moreover, let $\delta_{0t}$ be the number of vehicles that leave the depot in time period $t \in \mathcal{T}$, and let $q_{it}$ be the delivered quantity to customer $i \in \mathcal{N}^C$ in time period $t \in \mathcal{T}$. Using this notation, we can formulate the IRP as the following MILP:

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

$$s_{00} = I_0, \tag{2}$$

$$s_{i0} = I_i, \qquad i \in \mathcal{N}^C, \tag{3}$$

$$s_{0t} = S_t - \sum_{i\in\mathcal{N}^C} q_{it} + s_{0(t-1)}, \qquad t \in \mathcal{T}, \tag{4}$$

$$s_{it} = q_{it} - D_{it} + s_{i(t-1)}, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{5}$$

$$L_0 \leq s_{0t} \leq U_0, \qquad t \in \mathcal{T}, \tag{6}$$

$$L_i \leq s_{it} \leq U_i, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{7}$$

$$q_{it} \leq U_i - s_{i(t-1)}, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{8}$$

$$\sum_{i\in\mathcal{N}^C} q_{it} \leq Q\delta_{0t}, \qquad t \in \mathcal{T}, \tag{9}$$

$$q_{it} \leq \min\{U_i - I_{it}, Q\}\delta_{it}, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{10}$$

$$\sum_{j\in\mathcal{N}\setminus\{i\}} x_{ijt} = \delta_{it}, \qquad i \in \mathcal{N}, t \in \mathcal{T}, \tag{11}$$

$$\sum_{j\in\mathcal{N}\setminus\{i\}} x_{ijt} = \sum_{j\in\mathcal{N}\setminus\{i\}} x_{jit}, \qquad i \in \mathcal{N}, t \in \mathcal{T}, \tag{12}$$

$$\sum_{(i,j)\in(S:S)} x_{ijt} \leq \sum_{i\in S} \delta_{it} - \delta_{mt}, \qquad S \subset \mathcal{N}^C, |S| \geq 2, t \in \mathcal{T}, m \in S. \tag{13}$$

$$\sum_{(i,j)\in(S:\mathcal{N}\setminus S)} Q x_{ijt} \geq \sum_{i\in S} q_{it}, \qquad S \subset \mathcal{N}^C, |S| \geq 2, t \in \mathcal{T}, \tag{14}$$

$$q_{it} \geq 0, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{15}$$

$$\delta_{it} \in \{0,1\}, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{16}$$

$$\delta_{0t} \in \{0,1,\dots,V\}, \qquad t \in \mathcal{T}, \tag{17}$$

$$x_{ijt} \in \{0,1\}, \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{18}$$

The sum of the transportation cost and the inventory holding cost is minimized in objective function (1). The inventory balance at the depot and at the customers are enforced by constraints (4) and (5), respectively. Constraints (6) and (7) ensure that the inventory levels at the depot and at the customers always stay between their lower and upper limits. The ML inventory policy is enforced by constraints (8), while constraints (9) state that the depot never delivers more than the fleet capacity to the customers in a given time period. Constraints (10) ensure that a delivery to a given customer only can occur if the customer is visited. Constraints (11) are the degree constraints, and correct flow of vehicles between nodes is ensured by constraints (12). Constraints (13) and (14) impose the standard and capacitated subtour elimination constraints, 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 $\mathcal{F}$. Note that constraints (13) are not necessary for defining the problem, but preliminary testing indicated that they enhance performance. Finally, constraints (15)–(18) define the variable domains.

## 3. The branch-and-cut embedded matheuristic

The branch-and-cut embedded matheuristic presented in this paper includes a construction heuristic, an improvement heuristic, and a B&C method. The solution method is illustrated in Fig. 1 and begins by constructing a feasible solution using a construction heuristic followed by an improvement heuristic. This solution is then used as an initial primal solution for the B&C method. Whenever the B&C method encounters a new incumbent solution, it is fed to the improvement heuristic, and

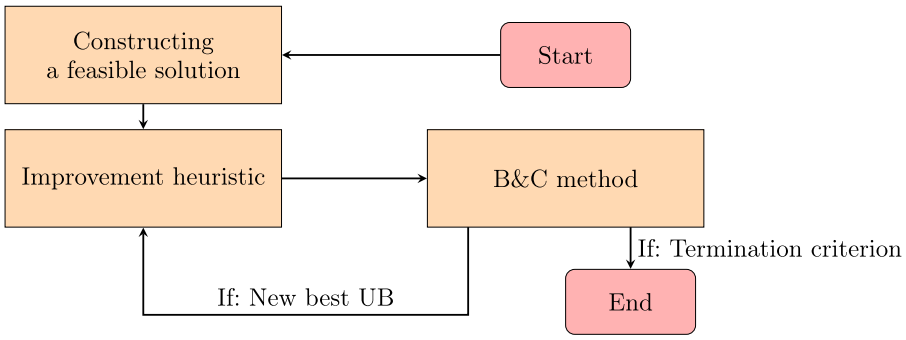



**Fig. 1.** An overview of the proposed solution method.

any improved solution found by the improvement heuristic is passed back to the B&C method. This cycle between the B&C method and the improvement heuristic continues until optimality is proven or a time limit is reached.

The rest of this section is organized as follows: In Sections 3.1–3.3 we present each part of the solution method designed for the IRP. First, the construction heuristic is described in detail in Section 3.1, before Section 3.2 introduces the improvement heuristic. The B&C method is presented in Section 3.3. Finally, we describe the small modifications needed to apply the solution method to the SDVRP in Section 3.4.

### 3.1. Construction heuristic

The construction heuristic is an extension of the construction heuristic presented in Vadseth et al. (2021). The goal of the construction heuristic is to construct a (good) feasible solution to the IRP. This is done by creating a small set of promising routes $\hat{\mathcal{R}}$ and then solving a MILP to select a subset of these routes that form a feasible solution to the IRP.

The route-based MILP is a Dantzig–Wolfe reformulation of the mathematical model given in Section 2, where each route $r \in \hat{\mathcal{R}}$ corresponds to an extreme point of the polytope defined by constraints (12), (13) and the linear relaxation of (18), describing the set of Hamiltonian cycles through a subset of the nodes including the depot. Note that this formulation is equivalent to the formulation given in Section 2 if $\hat{\mathcal{R}}$ includes all feasible routes.

The parameter $A_{ijr}$ is 1 if route $r$ traverses the arc $(i,j) \in \mathcal{A}$, and 0 otherwise. Hence, the cost of route $r \in \hat{\mathcal{R}}$ can be defined as $C_r^T = \sum_{(i,j)\in\mathcal{A}} C_{ij} A_{ijr}$. Further, if route $r \in \hat{\mathcal{R}}$ is used by a vehicle in time period $t \in \mathcal{T}$ then variable $\lambda_{rt}$ is 1, and 0 otherwise. In addition, the quantity product loaded onboard a vehicle traversing arc $(i,j) \in \mathcal{A}$ in time period $t \in \mathcal{T}$ is denoted $l_{ijt}$. With this notation, the model can be formulated as follows:

$$\min \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} C_i^H s_{it} + \sum_{r\in\hat{\mathcal{R}}} \sum_{t\in\mathcal{T}} C_r^T \lambda_{rt} \tag{19}$$

Constraints (2)–(8),

$$\sum_{j\in\mathcal{N}} l_{jit} - q_{it} - \sum_{j\in\mathcal{N}} l_{ijt} = 0, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{20}$$

$$l_{ijt} - Q \sum_{r\in\hat{\mathcal{R}}} A_{ijr}\lambda_{rt} \le 0, \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{21}$$

$$\sum_{r\in\hat{\mathcal{R}}} \sum_{j\in\mathcal{N}} A_{ijr}\lambda_{rt} \le 1, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{22}$$

$$\sum_{r\in\hat{\mathcal{R}}} \lambda_{rt} \le V, \qquad t \in \mathcal{T}, \tag{23}$$

$$\lambda_{rt} \in \{0,1\}, \qquad r \in \hat{\mathcal{R}}, t \in \mathcal{T}, \tag{24}$$

$$q_{it} \ge 0, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{25}$$

$$l_{ijt} \ge 0, \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{26}$$

The transportation and inventory holding costs are minimized in the objective function (19). Constraints (20) are flow balance constraints. Constraints (21) guarantee that the load on an arc does not exceed the capacity of a vehicle. In addition, constraints (22) ensure that a customer is not visited more than once in the same time period, while constraints (23) limit the number of routes used in a solution to be no more than the number of vehicles available. Finally, the variable domains are defined by constraints (24)–(26).

To strengthen the linear relaxation of the model, we propose new tightened versions of the three classes of valid inequalities presented in Coelho and Laporte (2014). The original inequalities are based on the fact that there has to be at least one visit to a customer in time periods $\{t_1, \dots, t_2\} \in \mathcal{T}$ if the sum of the demands in time period $t_1$ to $t_2$ is greater than the maximum inventory limit:

$$\sum_{r\in\hat{\mathcal{R}}} \sum_{t'=t_1}^{t_2} \sum_{j\in\mathcal{N}} A_{ijr}\lambda_{rt'} \ge \left\lceil \frac{\sum_{t'=t_1}^{t_2} D_{it'} - U_i}{U_i} \right\rceil, \qquad i \in \mathcal{N}^C, t_1, t_2 \in \mathcal{T}, t_2 \ge t_1. \tag{27}$$

The valid inequalities (27) can be further strengthened by replacing the upper inventory limit $U_i$ with the actual inventory $s_{i(t_1-1)}$ and instead of dividing by the upper inventory limit $U_i$, divide by $\min\{Q, U_i, \sum_{t'=t_1}^{t_2} D_{it'}\}$. This gives three different classes of valid inequalities. However, they can be further strengthened by realizing that deliveries are performed before consumption according to the problem description of the standard IRP (Archetti et al., 2007) and constraints (8) dictate that the maximum inventory held at a node at the beginning of a time period $t$ is $U_i - D_{i(t-1)}$. This gives us two tightened classes of valid inequalities (two of the classes presented in Coelho and Laporte (2014) can in fact be merged into one):

$$\sum_{r\in\hat{\mathcal{R}}} \sum_{t'=t_1}^{t_2} \sum_{j\in\mathcal{N}} A_{ijr}\lambda_{rt'} \ge \left\lceil \frac{\sum_{t'=t_1}^{t_2} D_{it'} - U_i - D_{i(t_1-1)}}{\min\{Q, U_i - D_{i(t_1-1)}\}} \right\rceil, \qquad i \in \mathcal{N}^C, t_1, t_2 \in \mathcal{T}, t_2 \ge t_1, \tag{28}$$

$$\sum_{r\in\hat{\mathcal{R}}} \sum_{t'=t_1}^{t_2} \sum_{j\in\mathcal{N}} A_{ijr}\lambda_{rt'} \ge \frac{\sum_{t'=t_1}^{t_2} D_{it'} - s_{i(t_1-1)}}{\min\{Q, U_i - D_{i(t_1-1)}, \sum_{t'=t_1}^{t_2} D_{it'}\}}, \qquad i \in \mathcal{N}^C, t_1, t_2 \in \mathcal{T}, t_2 \ge t_1. \tag{29}$$

In addition, if $t_1 = 1$, we can replace $U_i - D_{i(t_1-1)}$ in the numerator of valid inequalities (28) and $s_{i(t_1-1)}$ in the numerator of valid inequalities (29) with $I_i$ (the actual inventory held at the start of time period $t_1$).

#### 3.1.1. Route generation

Two methods are used, one of them new to the literature, to generate routes for the set $\hat{\mathcal{R}}$. The first method, called the giant tour

method, is taken from Vadseth et al. (2021). Here, we find the optimal solution of a traveling salesman problem (giant tour) defined on the graph $G^C = \{\mathcal{N}^C, \mathcal{A}^C\}$, where $\mathcal{A}^C = \{(i,j) \in \mathcal{N}^C \times \mathcal{N}^C | i \neq j\}$ is the set of arcs connecting every pair of customers. Each customer in the giant tour is assigned a delivered quantity equal to a predefined percentage $P$ of its inventory capacity. The giant tour with these corresponding delivered quantities is then converted to a capacitated vehicle routing problem (CVRP) solution by applying the splitting algorithm of Vidal (2016). The routes of the CVRP solutions are added to $\hat{\mathcal{R}}$. The allocation of delivered quantities is repeated iteratively with a decreasing $P$, so that we generate both short and long routes. We refer to Vadseth et al. (2021) for further details.

The second method, called the shifting assignment method, is outlined in Algorithm 1. Let $\hat{q}_{it}$ and $f_{vt}$ denote the quantity to deliver to customer $i$, and the load onboard vehicle $v$, respectively, in time period $t$. Further, let $v_{it}$ store the index of the vehicle serving customer $i$ in time period $t$, with an index of 0 indicating that the customer is not served in that time period. We also introduce $\mathcal{K} = \{1, \dots, V\}$ as the set of vehicles. The algorithm first assigns values to $\hat{q}_{it}$ on line 2, so that no product is assigned until the initial inventory runs out and from then on is assigned the customer's demand in each time period that cannot be covered by the initial inventory. Then, on lines 3–6, each $\hat{q}_{it}$ is assigned to the vehicle $v_{it}$ with the lowest load onboard $f_{vt}$ (line 4), and the vehicle load is updated (line 5).

Once the initial assignments are determined, the shifting assignment part of Algorithm 1 shifts delivery quantities from time period $t$ to time period $t-1$ on lines 8–21. It does this by iterating over each time period given that at least one vehicle departs in the previous time period, and each customer with a positive quantity $\hat{q}_{it}$ (lines 8 and 9). If that customer is not serviced in the previous time period, the quantity $\hat{q}_{it}$ is added to the vehicle with the largest load that can fit the quantity (lines 10–12). Otherwise, it is added to the vehicle already serving customer $i$ in the previous time period, given that this vehicle has sufficient space (line 13). Lines 14–18 update the vehicle loads and the delivered quantities. Finally, a CVRP instance is solved for each time period $t$ with delivery quantities $\hat{q}_{it}$, $i \in \mathcal{N}^C$ as input (line 23). Note that for the CVRP instance in time period $t$, only customers with $\hat{q}_{it} > 0$ are included. The resulting routes from the solutions of the CVRP instances are then added to $\hat{\mathcal{R}}$ (line 24).

As long as $\hat{q}_{it} \leq Q, i \in \mathcal{N}^C, t \in \mathcal{T}$ and $f_{vt} \leq Q,\ v \in \mathcal{K}, t \in \mathcal{T}$ after the initial assignment phase, the shifting assignment method is guaranteed to produce feasible CVRP instances. For readability, we have assumed that this is always the case in Algorithm 1. However, if this is not the case, Algorithm 1 will terminate, and no CVRP instances are solved. For this work, we use the hybrid genetic algorithm presented in Vidal et al. (2012) to solve the CVRP instances. More specifically, we use the open-source implementation which is described in Vidal (2022). If feasible CVRP instances are created, we know that the resulting routes and $\hat{q}_{it}, i \in \mathcal{N}^C, t \in \mathcal{T}$ make up a feasible solution to the original problem and can be used as an initial primal solution when solving the route-based model.

**Algorithm 1** The Shifting Assignment Method

1: *The initial assignment part*
2: $\hat{q}_{it} = \max\{D_{it} - I_{it}, 0\}, t \in \mathcal{T}, i \in \mathcal{N}^C$
3: **for** $i \in \mathcal{N}^C,\ t \in \mathcal{T}$ **do**
4: $\quad v_{it} = \text{argmin}_{v \in \mathcal{K}}\{f_{vt}\}$
5: $\quad f_{v_{it}t} = f_{v_{it}t} + \hat{q}_{it}$
6: **end for**
7: *The shifting assignments part*
8: **for** $t \in \mathcal{T} \setminus \{1\} : \sum_{v \in \mathcal{K}} f_{v(t-1)} \neq 0$ **do**
9: $\quad$ **for** $i \in \mathcal{N}^C$: $\hat{q}_{it} > 0$ **do**
10: $\quad\quad$ **if** $v_{i(t-1)} = 0$ **then**
11: $\quad\quad\quad v_{i(t-1)} = \text{argmax}_{v \in \mathcal{K}}\{f_{v(t-1)} | f_{v(t-1)} + \hat{q}_{it} \leq Q\}$
12: $\quad\quad$ **end if**
13: $\quad\quad$ **if** $f_{v_{i(t-1)}(t-1)} + \hat{q}_{it} \leq Q$ **then**
14: $\quad\quad\quad f_{v_{i(t-1)}(t-1)} = f_{v_{i(t-1)}(t-1)} + \hat{q}_{it}$
15: $\quad\quad\quad f_{v_{it}t} = f_{v_{it}t} - \hat{q}_{it}$
16: $\quad\quad\quad \hat{q}_{i(t-1)} = \hat{q}_{i(t-1)} + \hat{q}_{it}$
17: $\quad\quad\quad \hat{q}_{it} = 0$
18: $\quad\quad\quad v_{it} = 0$
19: $\quad\quad$ **end if**
20: $\quad$ **end for**
21: **end for**
22: *The CVRP part*
23: Solve $|\mathcal{T}|$ CVRP instances with $\hat{q}$ as input
24: Add all routes from the CVRP solutions to $\hat{\mathcal{R}}$

**Algorithm 2** Improvement heuristic

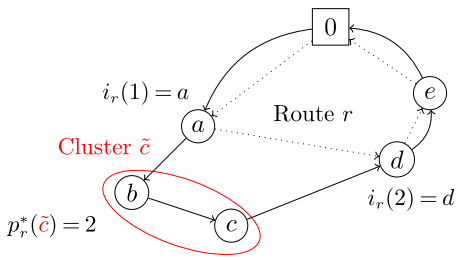
1: Input: $x$
2: $x^{best} = x$
3: $\hat{\mathcal{R}} = getRoutes(x^{best})$
4: **for** $h \in \{1, 2\}$ **do**
5: $\quad$ **for** $IT_h$ iterations **do**
6: $\quad\quad$ **if** $h = 2$ **then**
7: $\quad\quad\quad \hat{\mathcal{R}} = \hat{\mathcal{R}} \bigcup VRP(x^{best})$
8: $\quad\quad$ **end if**
9: $\quad\quad x^{current} = ImprovementMILP(\hat{\mathcal{R}})$
10: $\quad\quad$ **if** $f(x^{current}) < f(x^{best})$ **then**
11: $\quad\quad\quad x^{best} = x^{current}$
12: $\quad\quad\quad \hat{\mathcal{R}} = getRoutes(x^{best})$
13: $\quad\quad$ **else**
14: $\quad\quad\quad$ Break
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end for**
18: **return** $x^{best}$

### 3.2. Improvement heuristic

The improvement heuristic consists of solving a MILP that aims to improve a feasible solution within a well-defined neighborhood, for a finite number of iterations. In each iteration, the MILP explores a set of modifications of the routes of a feasible solution to the original problem. In the first iteration, the routes originate from the solution obtained by the construction heuristic, and for the remaining iterations, the routes are taken from the solution of the previous iteration. An outline of the improvement heuristic is given in Algorithm 2. The improvement MILP is presented in Section 3.2.1.

Algorithm 2 gets a feasible solution $x$ as input and sets this to the incumbent solution $x^{best}$ (lines 1 and 2). It then populates the route set $\hat{\mathcal{R}}$ with the routes from this solution using the $getRoutes()$ function (line 3). The main part of the heuristic is described on lines 4–17, where the algorithm iterates between solving the improvement MILP in the function $ImprovementMILP(\hat{\mathcal{R}})$ and updating the route set $\hat{\mathcal{R}}$. The route set is updated in two different ways, controlled by the parameter $h$ (line 4), with $IT_h$ setting an upper limit on the number of iterations (line 5). If $h = 1$ the improvement MILP is solved using only the routes of the solution from the previous iteration, while if $h = 2$ we add additional routes obtained by calling the function $VRP(x)$ described below. Once the route set is populated, the MILP is solved (line 9) to obtain a new current solution. If this solution has a better objective value than $x^{best}$ (line 10), it is set as the new best solution and its routes make up the new route set. Otherwise, we either increase $h$ or stop the algorithm (line 14) if $h = 2$. Finally, the best solution $x^{best}$ is returned (line 18).

The function $VRP(x)$ takes a feasible solution $x$ as input and solves three CVRP instances for each time period $t \in \mathcal{T}$, where the demand of each customer is set equal to its delivered quantity in time period $t$. However, the vehicle capacity $Q$ is set differently in the three instances. The motivation for this is that quantities can be shifted between time periods in the IRP, and better routing decisions can be obtained by




$$C_r^T = C_{0a} + C_{ad} + C_{de} + C_{e0}$$
$$C_{\tilde{c}r}^I = C^{SP}(i_r(1), \tilde{c}, i_r(2)) - C_{i_r(1),i_r(2)}$$
$$= C_{ab} + C_{bc} + C_{cd} - C_{ad}$$
$$0 \to a \to b \to c \to d \to e \to 0 : C_r^T + C_{\tilde{c}r}^I$$
$$= C_{0a} + C_{ab} + C_{bc} + C_{cd} + C_{de} + C_{e0}$$

**Fig. 2.** A route consisting of the depot and nodes $a$, $d$ and $e$ is changed to a route consisting of the depot and nodes $a$, $b$, $c$, $d$ and $e$ using cluster insertion. The cost of the route is then changed from $C_{0a} + C_{ad} + C_{de} + C_{e0}$ to $C_{0a} + C_{ab} + C_{bc} + C_{cd} + C_{de} + C_{e0}$.

making small adjustments. By varying the vehicle capacity $Q$, we mimic the effect of shifting quantities between time periods and are hopefully able to find better routes. The capacity is adjusted by multiplying $Q$ with 0.97, 1.03, and 1.06 in the three CVRP instances, respectively. The number of vehicles used in the solution of such a CVRP instance might decrease or increase compared with the corresponding time period of the current IRP solution, depending on whether the adjusted vehicle capacity $Q_{new}$ is greater or smaller than the actual vehicle capacity, respectively. This can be interpreted as a temporary expansion of the search space neighborhood, where we might find routes that the improvement MILP otherwise would not be able to encounter. Even though the aim of this procedure is to avoid getting stuck in a local optimum, we do not consider it disruptive enough to truly represent a diversification mechanism.

### 3.2.1. The improvement MILP

The improvement MILP is a modified version of the route-based model presented in Section 3.1, where the routes in $\hat{\mathcal{R}}$ can be modified by inserting or removing customers. All alterations of the original routes, even in cases where multiple changes are made simultaneously, are evaluated correctly in the objective function, i.e., an improved solution of the improvement MILP is also an improved solution of the original problem.

To present the improvement MILP, we introduce some additional notation. We let the set $\mathcal{N}_r$ denote the customers visited on route $r \in \hat{\mathcal{R}}$, while $\overline{\mathcal{N}}_r = \mathcal{N}^C \setminus \mathcal{N}_r$ is the complement set. To add customers to routes we introduce a set of clusters $C$, where a cluster $c \in C$ is a subset of the customers, $c \subseteq \mathcal{N}^C$. The subset $C_r$ of clusters associated with route $r$ is defined as $C_r = \{c \in C | c \cap \mathcal{N}_r = \emptyset\}$. We also introduce the variable $z_{crt}$ that is 1 if cluster $c \in C_r$ is inserted into route $r \in \hat{\mathcal{R}}$ in time period $t \in \mathcal{T}$, and 0 otherwise. A cluster is always inserted into the least cost position on the route, $p_r^*(c)$, calculated as:

$$p_r^*(c) = \underset{p \in \{1,\dots,|\mathcal{N}_r|+1\}}{\arg\min} \{C^{SP}(i_r(p-1), c, i_r(p)) - C_{i_r(p-1),i_r(p)}\}, \quad r \in \hat{\mathcal{R}}, c \in C_r, \tag{30}$$

where the function $i_r(p)$ gives the node placed in position $p$ in route $r$, and $C^{SP}(i_{start}, c, i_{end})$ returns the cost of the shortest path starting in node $i_{start} \in \mathcal{N}$, traveling through all customers in cluster $c \in C$ and ending in node $i_{end} \in \mathcal{N}$. Please note that the depot has both position 0 (the first position) and position $|\mathcal{N}_r| + 1$ (the last position) in all routes $r \in \hat{\mathcal{R}}$. The cost of adding cluster $c \in C_r$ to route $r \in \hat{\mathcal{R}}$, $C_{cr}^I$, is calculated as:

$$C_{cr}^I = C^{SP}(i_r(p_r^*(c) - 1), c, i_r(p_r^*(c))) - C_{i_r(p_r^*(c)-1),i_r(p_r^*(c))}, \quad r \in \hat{\mathcal{R}}, c \in C_r. \tag{31}$$

Fig. 2 shows an example where cluster $\tilde{c} = \{b, c\}$ (red ellipse) is inserted into the route $0 \to a \to d \to e \to 0$ (dotted arrows). The cheapest position to insert cluster $\tilde{c}$ is $p_r(\tilde{c}) = 2$ which is between node $a$ and $d$. In addition, the shortest route starting in node $a$, visiting all nodes in $\tilde{c}$, and ending in node $d$ is $a \to b \to c \to d$ with cost $C^{SP}(a, \tilde{c}, d) = C_{ab} + C_{bc} + C_{cd}$. The new route is marked with solid arrows.

In this work, a $k$-means algorithm (Ahmed et al., 2020), with a $k$-value $= \lfloor |\mathcal{N}^C|/2 \rfloor$ dictating the number of clusters, has been used to produce clusters with cardinality greater than 1. All clusters $c$ produced by the algorithm are placed in the set $C^k$, given that the size of $c$ is in the range $[2, \dots, clusterSize]$. We set the parameter $clusterSize = 3$, which is the largest cluster size we want to include in the model. We can then set $C = \left\{\bigcup_{i \in \mathcal{N}^C} \{i\} \bigcup C^k\right\}$.

Removing customers from a route is modeled by introducing variables $w_{prt}^{\phi}$ which is 1 if exactly $\phi \in \Phi$ consecutive customers are removed from route $r \in \hat{\mathcal{R}}$ starting from position $p \in \mathcal{P}_r^{\phi}$ in time period $t \in \mathcal{T}$, and 0 otherwise. We introduce the parameter $M$ as the maximum number of consecutive customers that can be removed and hence define the set $\Phi = \{1, \dots, M\}$. The set $\mathcal{P}_r^{\phi}$ contains all positions $p$ in route $r$ from where $\phi$ consecutive customers can be removed. For each route $r \in \hat{\mathcal{R}}$ in time period $t \in \mathcal{T}$ we let the variable $u_{irt}$ be 1 if node $i \in \mathcal{N}_r$ is removed from route $r$, and 0 otherwise. The cost reduction of removing $\phi$ customers from route $r$ starting at position $p$ is calculated as:

$$C_{\phi pr}^R = \sum_{p'=p}^{p+\phi} C_{i_r(p'-1),i_r(p')} - C_{i_r(p-1),i_r(p+\phi)}, \quad r \in \hat{\mathcal{R}}, \phi \in \Phi, p \in \mathcal{P}_r^{\phi}. \tag{32}$$

Fig. 3 shows the cost calculations of a route in the improvement MILP if a cluster consisting of a single customer ($e$) is added to a route and a sequence of three consecutive customers ($\phi = 3$) starting from position $p = 1$ are removed from the same route. The original route $0 \to a \to b \to c \to d \to 0$ is marked with dotted arrows. The new route is marked with solid arrows.

To formulate the improvement MILP, we must also define the variable $q_{irt}$ which denotes the quantity delivered by route $r$ to node $i$ in time period $t$. The improvement MILP can be formulated as follows:

$$\min \sum_{i \in \mathcal{N}^C} \sum_{t \in \mathcal{T}} C_i^H s_{it} + \sum_{r \in \hat{\mathcal{R}}} \sum_{t \in \mathcal{T}} C_r^T \lambda_{rt} - \sum_{r \in \hat{\mathcal{R}}} \sum_{\phi \in \Phi} \sum_{p \in \mathcal{P}_r^{\phi}} \sum_{t \in \mathcal{T}} C_{\phi pr}^R w_{prt}^{\phi} + \sum_{r \in \hat{\mathcal{R}}} \sum_{c \in C_r} \sum_{t \in \mathcal{T}} C_{cr}^I z_{crt} \tag{33}$$

Constraints (2)–(8),

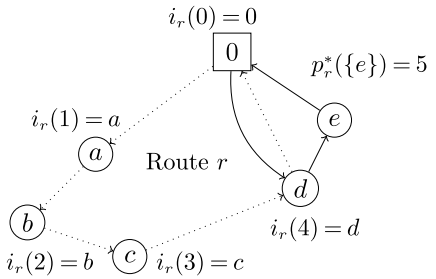
$$q_{it} = \sum_{r \in \hat{\mathcal{R}}} q_{irt}, \qquad i \in \mathcal{N}_r, t \in \mathcal{T}, \tag{34}$$

$$q_{irt} - \min(Q, U_i)(1 - u_{irt}) \le 0, \qquad r \in \hat{\mathcal{R}}, i \in \mathcal{N}_r, t \in \mathcal{T}, \tag{35}$$

$$q_{irt} - \min(Q, U_i) \sum_{c \in C_r : i \in c} z_{crt} \le 0, \qquad r \in \hat{\mathcal{R}}, i \in \overline{\mathcal{N}}_r, t \in \mathcal{T}, \tag{36}$$

$$\sum_{i \in \mathcal{N}^C} q_{irt} \le Q \lambda_{rt}, \qquad r \in \hat{\mathcal{R}}, t \in \mathcal{T}, \tag{37}$$

$$\sum_{r \in \hat{\mathcal{R}} : i \in \mathcal{N}_r} (\lambda_{rt} - u_{irt}) + \sum_{r \in \hat{\mathcal{R}} : i \in \overline{\mathcal{N}}_r} \sum_{c \in C_r} z_{crt} \le 1, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}, \tag{38}$$




$$C^T_r = C_{0a} + C_{ab} + C_{bc} + C_{cd} + C_{d0}$$
$$C^R_{31r} = C_{0a} + C_{ab} + C_{bc} + C_{cd} - C_{0d}$$
$$C^I_{\{e\}r} = C_{de} + C_{e0} - C_{d0}$$
$$0 \to d \to e \to 0 : C^T_r - C^R_{31r} + C^I_{\{e\}r} = C_{0d} + C_{de} + C_{e0}$$

**Fig. 3.** A route consisting of the depot and nodes $a$, $b$, $c$ and $d$ is changed to a route consisting of the depot and nodes $d$ and $e$ using a single insertion and multiple removals in a row. The cost of the route is then changed from $C_{0a} + C_{ab} + C_{bc} + C_{cd} + C_{d0}$ to $C_{0d} + C_{de} + C_{e0}$.

$$\sum_{c \in C_r : i \in c} z_{crt} \le \lambda_{rt}, \qquad r \in \hat{\mathcal{R}}, i \in \overline{\mathcal{N}}_r, t \in \mathcal{T}, \quad (39)$$

$$u_{irt} \le \lambda_{rt}, \qquad r \in \hat{\mathcal{R}}, i \in \mathcal{N}_r, t \in \mathcal{T}, \quad (40)$$

$$\sum_{p'=p}^{p+\phi-1} u_{i_r(p')rt} \ge \phi w^{\phi}_{prt}, \qquad \phi \in \Phi, r \in \hat{\mathcal{R}}, p \in \mathcal{P}^{\phi}_r, t \in \mathcal{T}, \quad (41)$$

$$\sum_{c \in C_r : p^*_r(c) = p} z_{crt} + \sum_{\phi \in \Phi} \sum_{p'=p-\phi+1}^{p} w^{\phi}_{p'rt} \le 1, \qquad r \in \hat{\mathcal{R}}, p \in 1, \dots, |\mathcal{N}_r| + 1, t \in \mathcal{T}, \quad (42)$$

$$\sum_{r \in \hat{\mathcal{R}}} \lambda_{rt} \le V, \qquad t \in \mathcal{T}, \quad (43)$$

$$\lambda_{rt} \in \{0,1\}, \qquad r \in \hat{\mathcal{R}}, t \in \mathcal{T}, \quad (44)$$

$$u_{irt} \in \{0,1\}, \qquad r \in \hat{\mathcal{R}}, i \in \mathcal{N}_r, t \in \mathcal{T}, \quad (45)$$

$$z_{crt} \in \{0,1\}, \qquad r \in \hat{\mathcal{R}}, c \in C, t \in \mathcal{T}, \quad (46)$$

$$w^{\phi}_{prt} \in \{0,1\}, \qquad \phi \in \Phi, r \in \hat{\mathcal{R}}, p \in \mathcal{P}^{\phi}_r, t \in \mathcal{T}, \quad (47)$$

$$q_{irt} \ge 0, \qquad i \in \mathcal{N}^C, r \in \hat{\mathcal{R}}, t \in \mathcal{T}. \quad (48)$$

The objective function (33) minimizes the sum of the transportation and inventory holding costs over the entire planning horizon. Constraints (34) link the old $q_{it}$ variable with the new $q_{irt}$ variable. Constraints (35) and (36) state that a customer can only be served if it is visited, while constraints (37) ensure that the vehicle capacity is respected. The fact that a customer can only be served at most once per time period is guaranteed by constraints (38). Constraints (39) and (40) enforce that a customer cannot be added to or removed from an unused route. Constraints (41) specify that all $u_{irt}$ variables in sequence $\phi$ starting at position $p$ are set to 1, if $w^{\phi}_{prt}$ is 1, and constraints (42) state that for each position $p$ in route $r$, at most one cluster can be inserted or at most one sequence including this position can be set to 1. Constraints (43) ensure that the fleet capacity is respected. Constraints (44)–(48) define the variable domains.

To ensure that the possible route modifications are evaluated correctly in the objective function, we must enforce some limitations on which modifications that can be allowed. This is achieved by adding the following constraints:

$$u_{i_r(p^*_r(c)-1)rt} + u_{i_r(p^*_r(c))rt} \le 2(1 - z_{crt}), \qquad r \in \hat{\mathcal{R}}, c \in C_r, t \in \mathcal{T}, \quad (49)$$

$$u_{i_r(p-1)rt} + u_{i_r(p+\phi)rt} \le 2(1 - w^{\phi}_{prt}), \qquad \phi \in \Phi, r \in \hat{\mathcal{R}}, p \in \mathcal{P}^{\phi}_r, t \in \mathcal{T}. \quad (50)$$

Constraints (49) make sure that if cluster $c$ is added to route $r$ then the customers in positions $p^*_r(c) - 1$ and $p^*_r(c)$ must be included in the route. Constraints (50) state that if a sequence of nodes is removed from the route, then the node preceding and succeeding the sequence cannot be removed. Please note that the improvement MILP assumes that the triangle inequality holds. Note that the 1's in constraints (35), (42), (49), and (50) can be replaced by $\lambda_{rt}$, thus tightening the constraints. However, preliminary testing showed that this had no significant effect when solving the MILP.

### 3.3. Branch-and-cut method

In this section, we present the B&C method used in our computational experiments. A B&C method is an extension of the well-known branch-and-bound (B&B) algorithm, where cutting planes or valid inequalities may be added to the linear relaxation of the model at every node of the B&B tree that does not satisfy the integer requirements. We use the customer schedule (CS) formulation presented by Skålnes et al. (2022), which was shown to generally perform better than the formulation defined by (1)–(18) in a B&C setting. Furthermore, Archetti and Ljubić (2022) showed that there are no advantages of using disaggregated formulations, i.e., formulations with a vehicle index, compared to aggregated formulations such as those presented in this paper.

The CS formulation is a Dantzig–Wolfe reformulation of the polyhedron defined by the linear relaxation of constraints (3), (5), (7), (8), (10), (15) and (16), which connects the customer visits to the delivered quantities. For each customer $i \in \mathcal{N}^C$, we define the set of customer schedules $\Omega_i$, i.e., the set of extreme points in the polyhedron mentioned. For each customer $i \in \mathcal{N}^C$ and corresponding customer schedule $\omega \in \Omega_i$, let the use of customer schedule $\omega$ for customer $i$ be denoted $y_{i\omega}$, let $B_{it\omega}$ be 1 if customer $i$ is visited in time period $t \in \mathcal{T}$ by using customer schedule $\omega$, and 0 otherwise, let $Q_{it\omega}$ be the delivered quantity to customer $i$ in time period $t \in \mathcal{T}$ by using customer schedule $\omega$, and let $S_{it\omega}$ be the inventory level at customer $i$ in time period $t \in \mathcal{T}$ if customer schedule $\omega$ is used. Using the notation defined above, we formulate the IRP as the following MILP:

$$\min \sum_{(i,j) \in \mathcal{A}} \sum_{t \in \mathcal{T}} C_{ij} x_{ijt} + \sum_{i \in \mathcal{N}^C} \sum_{t \in \mathcal{T}} \sum_{\omega \in \Omega_i} C^H_i S_{it\omega} y_{i\omega} + \sum_{t \in \mathcal{T}} C^H_0 s_{0t} \quad (51)$$

Constraints (2), (4), (6), (9), (11)–(18),

$$q_{it} = \sum_{\omega \in \Omega_i} Q_{it\omega} y_{i\omega}, \qquad t \in \mathcal{T}, i \in \mathcal{N}^C, \quad (52)$$

$$\delta_{it} = \sum_{\omega \in \Omega_i} B_{it\omega} y_{i\omega}, \qquad t \in \mathcal{T}, i \in \mathcal{N}^C, \quad (53)$$

$$\sum_{\omega \in \Omega_i} y_{i\omega} = 1, \qquad i \in \mathcal{N}^C, \quad (54)$$

$$y_{i\omega} \ge 0, \qquad i \in \mathcal{N}^C, \omega \in \Omega_i. \quad (55)$$

The objective function (51) minimizes the sum of the transportation cost, the inventory holding cost at the customers, and the inventory holding cost at the depot. Note that we can omit constraints (3), (5), (7) and (8), because they are implicitly handled within the customer schedules. Constraints (52) and (53) link the original decision variables for delivered quantities and customer visits to convex combinations of customer schedules. Constraints (54) make sure we only use a convex combination of customer schedules and constraints (55) define the non-negativity requirements for the customer schedule variables.

In addition, we propose an alternative formulation of the customer schedule. We observe that we can significantly reduce the number of customer schedules by substituting the equality sign of constraints (53)

with a $\geq$ sign:

$$\delta_{it} \geq \sum_{\omega \in \Omega_i} B_{it\omega} y_{i\omega}, \qquad i \in \mathcal{N}^C, t \in \mathcal{T}. \tag{56}$$

Then we no longer need to enumerate customer schedules that visit a customer $i \in \mathcal{N}^C$ without delivering anything, e.g., $B_{it\omega} = 1$ and $Q_{it\omega} = 0$. In constraints (53), we need these customer schedules to express all feasible solutions due to the equality sign. However, by using the updated version, we get the same dual bound of the linear relaxation, but with the cost of weaker branching decisions. Note that the up-branch on whether we visit a customer or not will have little to no effect on the convex combination of customer schedules. Preliminary testing indicated that the B&C method at termination, with the updated constraints (56), obtained higher average dual bounds for the large instances, but at the cost of slightly worse average dual bounds for the small instances. The large instances are closer to realistically sized instances, and therefore we value the performance on these more than that of the small instances. Also, note that the new version of the customer schedule formulation requires the triangle inequality to be satisfied to be a valid formulation for the IRP. This is handled by updating the cost matrix by finding the shortest path between all pairs of nodes so that the triangle inequality is guaranteed to be satisfied. For the instances where the triangle inequality was not originally satisfied, the solution is post-processed so that routes are updated according to the original cost matrix, i.e., the zero-delivery visits that violate the triangle inequality are inserted into the routes, so that the solution adheres to the original cost matrix. If this leads to an infeasible solution, i.e., more than one visit to the same customer in the same period, we remove the zero-delivery visits and re-calculate the cost with the original cost matrix.

The B&C method is based on the linear relaxation of (51), (2), (4), (6), (9), (11), (12), (15)–(18), (52), (54), (55) and (56), while dynamically adding the subtour elimination constraints (13) and (14), the capacity inequalities of Desaulniers et al. (2016) (adapted to a B&C context by Skålnes et al. (2022)), and the Disjoint Route (DR) inequalities of Avella et al. (2018). The customer schedules are enumerated a priori.

The integer feasible solution from the improvement heuristic is used to warm-start the B&C method. After solving the linear relaxation, we iteratively separate and add violated subtour elimination constraints and valid inequalities in the same manner as Skålnes et al. (2022) where the following separation order is used:

(i) SECs (13)
(ii) CSECs (14)
(iii) Capacity inequalities (Skålnes et al., 2022)
(iv) Simple DR inequalities (Avella et al., 2018)
(v) $h$-DR inequalities (Avella et al., 2018)

Due to the computational complexity of the latter three classes of valid inequalities, we only separate these in the root node. The SECs and the CSECs are separated at every node of the B&B tree. After stage (ii), we only move on to separate the next class of valid inequalities if the dual bound improvement from the previous iteration falls below a given threshold. All separation problems are solved in the same manner as in Skålnes et al. (2022), but with a minor modification of the separation algorithm for the $h$-DR inequalities to better handle the large instances. The default setting of this algorithm accounts for route lengths of $h = 8$, which means we may have to enumerate all routes of length 8 and less to verify that the separated valid inequality is in fact valid. From preliminary testing, this route enumeration seems to become too costly if the potential number of routes exceeds $10^{12}$. Therefore, we use $h = 7$ for $|\mathcal{N}^C| \geq 35$, $h = 6$ for $|\mathcal{N}^C| \geq 50$, and $h = 5$ for $|\mathcal{N}^C| \geq 200$.

### 3.4. Modifications of the B&C embedded matheuristic to solve the SDVRP

In this section, we demonstrate how the B&C embedded matheuristic can be modified to solve the SDVRP, which is a special case of the IRP where there is only a single time period, but where each customer can be visited multiple times.

Let $\mathcal{A}^C = \{(i,j) \in \mathcal{N}^C \times \mathcal{N}^C | i \neq j\}$ be the set of arcs connecting every pair of customers. A known property of the SDVRP is that there exists an optimal solution where no two routes can have more than one split customer in common, as long as the triangle inequality is fulfilled (Desaulniers, 2010). From this, it further follows that an arc $(i,j) \in \mathcal{A}^C$ between any pair of customers can appear at most once in an optimal solution. Thus, we can use the IRP model from Section 2 to formulate a relaxation of the SDVRP. We have $\mathcal{T} = \{1\}$ in the SDVRP and therefore omit the $t$-index in the previously presented notation. A relaxation of the SDVRP can now be defined by (1)–(14) and with the following variable domains:

$$q_i = D_i, \qquad i \in \mathcal{N}^C, \tag{57}$$
$$s_i = 0, \qquad i \in \mathcal{N}, \tag{58}$$
$$\delta_i \in \{0, 1, \dots, V\}, \qquad i \in \mathcal{N}, \tag{59}$$
$$x_{ij} \in \{0, 1\}, \qquad (i,j) \in \mathcal{A}^C, \tag{60}$$
$$x_{0j}, x_{j0} \in \{0, 1, \dots, V\}, \qquad j \in \mathcal{N}^C. \tag{61}$$

Since the customers can be visited several times, the capacitated subtour elimination constraints (14) may allow transshipments at some nodes, i.e., vehicles can swap loads at nodes they have in common, which is not feasible in the SDVRP. We, therefore, add cutting planes dynamically to the formulation every time the B&C method encounters such an infeasible solution. In this way, we ensure a correct formulation of the SDVRP.

Let $\mathcal{X}$ be the set of all feasible solutions defined by (2)–(14) and (57)–(61), and let $\mathcal{Y} \subset \mathcal{X}$ be the set of feasible solutions for the SDVRP. Furthermore, let $\mathcal{A}_k^T = \{(i,j) \in \mathcal{A} \mid x_{ij}^k \geq 1\}$ be the set of arcs in a feasible solution $x^k$ for a solution $k \in \mathcal{X} \setminus \mathcal{Y}$. We can then add feasibility cuts (62), as proposed by Archetti et al. (2014), to the relaxed formulation:

$$\sum_{(i,j) \in \mathcal{A} \setminus \mathcal{A}_k^T} x_{ij} \geq 1, \qquad k \in \mathcal{X} \setminus \mathcal{Y}. \tag{62}$$

Similar to the aforementioned paper, we use a mathematical program to determine whether a solution to the relaxed SDVRP is feasible for the full problem or not. However, we use a three-index vehicle flow formulation, equivalent to that of Coelho and Laporte (2014), where the arcs of a potentially feasible solution are allocated to vehicles, unlike the route-based procedure in Archetti et al. (2014). If this allocation problem is feasible, we have a feasible solution to the SDVRP, if not, we cut it off by adding the corresponding feasibility cut (62).

The B&C method is based on the model defined by (1)–(12) and (57)–(61), where the subtour elimination constraints (13) and (14) are added dynamically at every node of the B&B tree. In addition, we dynamically add the special case of the capacity inequalities (Skålnes et al., 2022), namely the rounded capacity inequalities for the CVRP (Laporte and Nobert, 1983) in the root node. The DR–inequalities are redundant for the SDVRP, so these are omitted. Whenever an infeasible solution is encountered, verified by a three-index vehicle flow model, we cut off the solution by adding the corresponding feasibility cut (62).

For the construction heuristic, the route-based MILP is significantly reduced as well. In addition, we must allow for multiple visits to the same customer. Please see Appendix A (Section C) for further details and a complete mathematical model. For the route generation, the first method (the giant tour method) must be slightly changed. Since the customers do not have an upper limit for inventory, we must instead use their demand as a starting point. The second method (the shifting

assignment method), is reduced to a simple CVRP for the SDVRP and is hence not applicable to this problem variant.

For the improvement heuristic, it follows that the improvement MILP is reduced since the SDVRP is a single-period problem. Please see Appendix A (Section D) for more details and a complete mathematical model. The function $VRP(x)$ must be updated to account for solutions where one or several customers are visited by multiple vehicles. This is achieved by splitting the relevant customer nodes into multiple nodes, to create a feasible CVRP instance. For instance, if a customer is visited by three different routes, it is split into three separate nodes. Their demand in the CVRP instance is equal to the demand that was delivered to them in the current SDVRP solution.

## 4. Computational study for the IRP

The proposed B&C embedded matheuristic has been tested on known benchmark instances from the literature, and the results have been compared with all methods we are aware of, where detailed computational results are publicly available. Moreover, we have analyzed each of the new features of the improvement heuristic and evaluated their effect on the overall solution. First, we describe the benchmark instances in Section 4.1, before we discuss parameter settings. The different components of the B&C embedded matheuristic are analyzed in Section 4.2, and the computational results for the benchmark instances are presented in Section 4.3. Further, we test the B&C embedded matheuristic on the DIMACS instances and compare the results with those of the DIMACS Implementation Challenge in Section 4.4. All computational experiments were run on a single thread on a 12 core Intel E5-2670v3 processor clocked at 2.3 GHz and 64 GB RAM. The algorithm is coded in C++ and the commercial solver Gurobi 9.5.1 has been used. Detailed solution files are publicly available at http://axiomresearchproject.com/

### 4.1. Benchmark instances, solution methods and parameter settings

Two sets of benchmark instances exist for the IRP with the ML inventory policy. The first set was proposed by Archetti et al. (2007) and consists of small single-vehicle instances with high or low inventory costs, multiples of five, $5k$, number of customers where $k = [1, 2, \dots, 10]$ and $k = [1, 2, \dots, 6]$ for three and six time periods, respectively. Each configuration has five versions, giving us a total of 160 instances. The second set was proposed by Archetti et al. (2012) and consists of large single-vehicle instances with six time periods, high or low inventory costs, and 50, 100, or 200 customers. Each configuration has ten versions, giving us a total of 60 instances. Both sets were made for the single-vehicle IRP, but were extended to up to five vehicles by Coelho et al. (2012). This gives us 878 multi-vehicle instances in total (two of the five-vehicle instances are infeasible).

For the DIMACS Implementation Challenge, specific instance files were made for the multi-vehicle instances, i.e., for two, three, four, and five vehicles. However, they have slightly different vehicle capacities from what has normally been used in the literature. Coelho et al. (2012) determined the capacity of each vehicle by taking the vehicle capacity of a single-vehicle instance, dividing it by the number of vehicles available, and then rounding it to the nearest integer. For the DIMACS Implementation Challenge, the vehicle capacities have been floored to the nearest integer instead. This gives, for some instances, a smaller solution space. In addition, the DIMACS instances include 160 new instances consisting of $5k$ customers and six time periods, where $k = [7, 8, 9, 10]$. Another difference is that the solutions to the DIMACS instances are reported without the inventory holding costs of the initial inventory in time period $t = 0$.

#### 4.1.1. State-of-the-art solution methods

As shown in the introduction, many solution methods exist for the IRP with an ML inventory policy. We have, to the best of our knowledge, compared our results with all solution methods with publicly available results. A detailed overview of the methods included in this study can be found in Table 1, where we list the reference to the paper (Reference), the abbreviation we use (Abbreviation), and the type of solution method (Sol), processor (CPU), number of threads (# Threads), and commercial solver (solver) used. Finally, we report their Passmark score (Passmark). Note that the code for the B&C-method of Coelho and Laporte (2014) was re-used by Desaulniers et al. (2016) and we have reported these results. Sakhri et al. (2022) did not provide detailed results.

#### 4.1.2. Parameter settings

The parameter settings in this work are a result of preliminary testing, and little effort has been put into parameter tuning. We used the same parameter settings as Vadseth et al. (2021) for the construction heuristic when relevant. The route-based MILP has a time limit of $1.5|\mathcal{N}^C|$ seconds. For the improvement heuristic, $IT_1 = 4$, $IT_2 = 5$, $k$-value $= |\mathcal{N}^C|/2$, $clusterSize = 3$ and $M = 3$. The improvement MILP has a time limit of $4|\mathcal{N}^C|$ seconds. The B&C method has a time limit of 600 s in the root node for the original instances and 200 s for the DIMACS instances. The entire B&C embedded matheuristic algorithm has a time limit of 7,200 s for the original instances. For the DIMACS instances, the time limit is set according to the rules of the DIMACS Implementation Challenge: $\frac{2000}{\text{Passmark score}}1800$ s. Please note that the time consumption of the route generation process of the construction heuristic is insignificant compared with the total time consumption and that the CVRP-solver terminates after 2000 iterations without improvements.

### 4.2. Analysis of the branch-and-cut embedded matheuristic

In this section, we analyze the different components of the B&C embedded matheuristic, mainly focusing on the matheuristic and the interaction between the matheuristic and the B&C method. To examine the impact of these features, we test various configurations on a subset consisting of 20% of the benchmark instances, i.e., version 1 of the small instances and versions 1 and 2 of the large instances. Here, we only present the conclusions, and we refer to Appendix A (Section A) for more extensive and detailed analyses.

First, we investigate whether our proposed improvements of the matheuristic, i.e., the construction and improvement heuristic, lead to better final solutions compared with the base version of the matheuristic with none of the proposed improvements. The base version only uses the giant tour route generation in the construction heuristic and single customer insertions and removals in the improvement heuristic. Here, our analyses show that adding the shifting assignment route generation method improves the solutions by 13.6% on average compared with the base version. This also highlights the importance of the starting solution, as the matheuristic is dependent on a good starting solution due to the limited number of iterations it can perform. Further, we found that including cluster insertions and removals of clusters with cardinality two and three improved the solutions on average by 11.1% compared with the base version. When combining both these components, i.e., both the shifting assignment method in the construction heuristic and the cluster insertions and removals in the improvement heuristic, the matheuristic found solutions that on average are 14.8% better than the base version. This indicates that our proposed improvements are crucial for the good results obtained by the B&C embedded matheuristic. We refer to Table 11 in Appendix A for more details.

Now that we have established that it is preferable to include insertions and removals of clusters of cardinality two and three, we examined how often these route modifications are performed. We find

**Table 1**
Benchmark solution methods. We present the solution approach, CPU, number of threads, MILP solver and Passmark score. Note: Sol: Solution approach, E: Exact, H: Heuristic/Metaheuristic, M: Matheuristic, Def: Default, Passmark: Passmark score.

| Reference | Abbreviation | Sol | CPU | #Threads | Solver | Passmark |
|---|---|---|---|---|---|---|
| Archetti et al. (2007) | A-BC | E | Pentium IV 2.8 GHz | Def | Cplex 9.0 | 236 |
| Archetti et al. (2012) | AR-M1 | M | Intel Dual Core 1.86 GHz | Def | Cplex 10.1 | – |
| Coelho and Laporte (2014) | CL-BC | E | Core i7-2600 3.4 GHz | 1 | Cplex 12.2 | 1742 |
| Adulyasak et al. (2014) | AD-BC | E | Intel Xeon 2.67 GHz | 8 | Cplex 12.3 | 5658 |
| Adulyasak et al. (2014) | AD-M | M | 2.10 GHz Duo CPU PC | Def | Cplex 12.3 | – |
| Desaulniers et al. (2016) | D-BPC | E | Core i7-2600 3.4 GHz | 1 | Cplex 12.2 | 1742 |
| Archetti et al. (2017) | AR-M2 | M | Xeon W3680, 3.33 GHz | 8 | Cplex 12.5 | 6913 |
| Avella et al. (2018) | AV-BC | E | Core i7-2620, 2.70 GHz | 1 | Xpress 7.6 | 1462 |
| Alvarez et al. (2018) | AL-SA | H | Core i7-2600 3.4 GHz | 1 | – | 1742 |
| Alvarez et al. (2018) | AL-ILS | H | Core i7-2600 3.4 GHz | 1 | – | 1742 |
| Chitsaz et al. (2019) | C-M | M | Xeon X5650 2.67 GHz | 1 | Cplex 12.6 | 1300 |
| Guimarães et al. (2023) | G-BC | E | Xeon E5-2630 v2 2.60 GHz | 6 | Gurobi 8.1 | 7490 |
| Manousakis et al. (2021) | M-BC | E | Intel Core i7-7700 CPU 3.60 GHz | 8 | Gurobi 8.1 | 8652 |
| Alvarez et al. (2020) | AL-M | M | Xeon X5650 2.67 GHz | 1 | Cplex 12.8 | 1300 |
| Diniz et al. (2020) | D-M | M | Intel Core i7-8700K 3.7 GHz | 1 | LEMON | 2759 |
| Vadseth et al. (2021) | V-M1 | M | Xeon Gold 6144 3.5 GHz | 1 | Gurobi 9.0 | 2523 |
| Archetti et al. (2021) | AR-M3 | M | Xeon E5-1620 v3 3.50 GHz | 1 | Cplex 12.10 | 2022 |
| Skålnes et al. (2022) | SK-BC | E | Intel E5-2670v3 2.3 GHz | 1 | Gurobi 9.0.2 | 1691 |
| Achamrah (2022) | AC-M | M | Quad-core Intel Core i7 3.3 GHz | Def | CPLEX 12.9 | – |
| Solyalı and Süral (2022) | S-M | M | Xeon X5650 2.67 GHz | 1 | CPLEX 12.7 | 1300 |
| Vadseth et al. (2023) | V-M2 | M | Xeon Gold 6144 3.5 GHz | 1 | Gurobi 9.1 | 2523 |
| This paper | | E | Intel E5-2670v3 2.3 GHz | 1 | Gurobi 9.5.1 | 1691 |

that for the small instances, clusters of cardinality one, two, and three are inserted on average 5.57, 0.55, and 0.19 times per instance, respectively. The corresponding number of removals are, on average, 4.16, 1.01, and 0.11 times per instance, respectively. For the large instances, we find that clusters of cardinality one, two and three are inserted on average 16.49, 1.04, and 0.24 times per instance, respectively, and they are removed on average 16.42, 3.48, and 0.55 times per instances, respectively. Not surprisingly, we can observe that the single-customer insertions and removals are the route modifications most frequently used by the improvement MILP. Please note that there could still be several modifications of a given route in these instances, but that each removal and insertion consisted, for the most part, of a single customer. An interesting observation is that the average number of cluster insertions and removals of cardinality two and three increases with the size of the instances, which might indicate that it is more crucial to include these features for large instances than for small instances. This is indeed natural since it is less likely to insert or remove several consecutive customers to improve the solution when there are few customers involved. We refer to Table 12 in Appendix A for more details.

Another important aspect of the B&C embedded matheuristic is how effective the loop between the B&C method and the improvement heuristic is. Every time the B&C embedded matheuristic is run, the improvement heuristic is called at least once, but on the benchmark instances, it was never called more than four times. In total, the B&C method was able to improve the warm-start solution provided by the improvement heuristic 157 and 53 times, for the small and large instances, respectively. Each such improvement induced a new call to the improvement heuristic which in turn improved these solutions in total 61 and 31 times for the small and the large instances, respectively. Finally, this resulted in that 54 and 28 instances were improved by a successive call to the improvement heuristic from within the B&C method, for the small and large benchmark instances respectively. Focusing on the number of iterations of the loop between the improvement heuristic and the B&C method, we found that the largest improvement from one iteration to the next occurs in the second iteration. This is indeed expected, as each improvement is one step closer to the optimal solution. We refer to Table 13 in Appendix A for more details.

Lastly, we found that the B&C method alone obtained the best average dual bounds. This indicates that the good primal bound from the matheuristic that potentially could be used to prune the B&B tree or perform variable fixing did not make up for the lost time running the matheuristic, which otherwise could be used to process more nodes of the B&B tree. We refer to Table 14 in Appendix A for more details.

### 4.3. Computational results on the benchmark instances

In this section, we present the computational results for the B&C embedded matheuristic on the 878 multi-vehicle benchmark instances ranging from two to five vehicles. In the overview displayed in Table 1 we see that the CPU Passmark score varies greatly across the various published methods. In addition, different software has been used in the various methods. Therefore, we find it hard to give a fair comparison of computational times between the various methods and have chosen to focus on solution quality for the computational study of this work. For the sake of readability, we only include the state-of-the-art (SOTA) solution methods in the presented tables, and we define this to be a method that has found a unique best-known solution (BKS) on at least one benchmark instance. Only 7 of the 22 methods included in this study qualify as SOTA methods according to this definition, and those are the matheuristics of Chitsaz et al. (2019) (C-M), Diniz et al. (2020) (D-M), Solyalı and Süral (2022) (S-M), Vadseth et al. (2021) (V-M1), and Vadseth et al. (2023) (V-M2), as well as the B&C methods of Guimarães et al. (2023) (G-BC) and Manousakis et al. (2021) (M-BC). A comparison with all published results can be found in the provided excel sheets. In addition, we would like to point out that Archetti et al. (2021) obtained high-quality solutions in a reasonable computational time, even though they have no unique BKSs. Also, we would like to highlight the simulated annealing method and the iterated local search algorithm of Alvarez et al. (2018), that found good solutions in an impressively short time, using less than 30 seconds for the small instances and less than 60 seconds for the large instances.

Table 2 reports the number of BKSs and unique BKSs (in parentheses) for each of the SOTA methods. We see from Table 2 that our B&C embedded matheuristic finds more BKSs and unique BKSs than the other methods, especially on the large instances. The only subset where we do not have the most BKSs are the small two-vehicle instances, where we have 13 less than the B&C method of Guimarães et al. (2023). We can also observe that the exact methods found many more BKSs on the small instances than the matheuristics, which is not surprising given that many of these instances are solved to optimality.

Apart from the number of BKSs, it is also interesting to investigate the overall quality of the solutions. We measure this by the average

**Table 2**
Overview of the number of BKSs (unique BKSs).

| Set | V | C-M | D-M | S-M | G-BC | M-BC | V-M1 | V-M2 | This paper | # inst. |
|---|---|---|---|---|---|---|---|---|---|---|
| Small | 2 | 19 (0) | 118 (0) | 38 (0) | **160 (5)** | 146 (0) | 72 (0) | – | 148 (0) | 160 |
| | 3 | 23 (5) | 83 (3) | 46 (3) | 124 (4) | **126 (6)** | 56 (3) | – | **131 (5)** | 160 |
| | 4 | 13 (5) | 47 (3) | 34 (4) | 88 (1) | **115 (16)** | 33 (1) | – | **125 (14)** | 160 |
| | 5 | 16 (4) | 40 (0) | 27 (0) | 73 (0) | 122 (15) | 31 (1) | – | **131 (23)** | 158 |
| Sum Small | | 71 (14) | 288 (6) | 145 (7) | 445 (10) | 509 (37) | 192 (5) | – | **534 (42)** | 638 |
| Large | 2 | 0 (0) | – | 3 (3) | 6 (5) | 9 (8) | 1 (1) | 1 (1) | **41 (41)** | 60 |
| | 3 | 0 (0) | – | 10 (10) | 0 (0) | 1 (1) | 1 (1) | 0 (0) | **48 (48)** | 60 |
| | 4 | 0 (0) | – | 0 (0) | 0 (0) | 1 (0) | 0 (0) | 2 (2) | **58 (57)** | 60 |
| | 5 | 0 (0) | – | 0 (0) | 0 (0) | 0 (0) | 0 (0) | 1 (1) | **59 (59)** | 60 |
| Sum Large | | 0 (0) | – | 13 (13) | 6 (5) | 11 (9) | 2 (2) | 4 (4) | **206 (205)** | 240 |
| Total Sum | | 71 (14) | 288 (6) | 158 (20) | 451 (15) | 520 (46) | 194 (7) | 4 (4) | **741 (247)** | 878 |

**Table 3**
Overview of average primal gaps (%).

| Set | $\lvert\mathcal{N}^C\rvert$ | C-M | D-M | S-M | G-BC | M-BC | V-M1 | V-M2 | This paper |
|---|---|---|---|---|---|---|---|---|---|
| Small | 5–50 | 3.04 | 0.47 | 0.51 | 0.38 | 0.06 | 1.16 | – | **0.05** |
| Large | 50 | 3.53 | – | 1.53 | 3.60 | 1.18 | 0.98 | 1.30 | **0.07** |
| | 100 | 2.91 | – | 1.44 | 17.25 | 1.75 | 1.09 | 1.29 | **0.03** |
| | 200 | 2.42 | – | 1.43 | 29.47 | – | 1.36 | 1.40 | **0.00** |
| Average Large | | 2.95 | – | 1.47 | 16.77 | 1.47 | 1.14 | 1.33 | **0.04** |

primal gap. Table 3 reports the average primal gaps, where the primal gap is defined as: Primal gap $= (UB_i - UB_{best})/UB_{best}$, where $UB_i$ is the upper bound obtained by method $i$ and $UB_{best}$ is the BKS across all methods listed in Table 1. The primal gaps are aggregated per number of customers separated between the sets of small and large instances. Here, our method obtains the best average primal gap for all instances with more than 30 customers. Compared with the other methods the average primal gaps are especially good for the large instances, which indicates that our solutions are also good for the instances where our method does not find the BKS.

So far we have focused on the upper bound obtained by our method, but since it is exact, it is also interesting to see how good lower bounds it obtains. Therefore, we calculate the deviation from the best-known lower bound (dev. BLB) for each method on each instance, defined as: dev. BLB $= (LB_{best} - LB_i)/LB_{best}$, where $LB_i$ is the lower bound obtained by method $i$ and $LB_{best}$ is the best-known lower bound across all methods listed in Table 1. Table 4 reports the average dev. BLB aggregated per number of customers and divided between the sets of small and large instances. In this table, we include all exact methods that have solved the multi-vehicle IRP and made their results publicly available, i.e., the BP&C method of Desaulniers et al. (2016) (D-BPC), the B&C methods of Coelho and Laporte (2014) (CL-BC), of Avella et al. (2018) (A-BC), of Guimarães et al. (2023) (G-BC) and of Manousakis et al. (2021) (M-BC). The detailed dual bounds were not available for the B&C method of Adulyasak et al. (2014). Here it is clear that Desaulniers et al. (2016) outperformed the other methods on the set of small instances. The B&C method which obtains the best average dev. BLB across the small instances is that of Manousakis et al. (2021), but our method is better on the small three-period instances of 35–50 customers. This is likely because the time spent in the improvement heuristic of the B&C embedded matheuristic has a small impact on the dual bound for the three-period instances. For the six-period instances, the time spent to obtain good primal bounds in the improvement heuristic leaves less time allocated to exploring the B&B tree, ultimately resulting in weaker dual bounds than the B&C method of Manousakis et al. (2021). The big difference in Passmark score between these methods might also play an important role, which makes the comparison harder.

Shifting the focus to the number of optimal solutions, as reported in Table 5, we see that all three B&C methods find a similar number of optimal solutions. In total, 480 instances are solved to proven optimality by at least one method, and the G-BC, M-BC, and the B&C of this paper found four, two, and one optimal solutions, respectively, that no other method is able to prove optimality for. However, here the inconsistencies of the results in the literature become apparent. Using an optimality gap tolerance of $10^{-4}$, as most commercial solvers use by default, we see that the methods of both Chitsaz et al. (2019) and Solyalı and Süral (2022) find more optimal solutions than BKSs. This is due to the fact that for 20 of the instances across all previously published results of our overview, the best lower bound entry is higher than the best upper bound entry. In some cases, it seems like there is an error in the reported lower bound, but in others in the reported upper bound. However, the only way to be sure would be to use a feasibility checker that guarantees that every solution is evaluated the exact same way. For the inconsistent instances where a solution is found feasible, we know that the lower bound must be wrong, and if not, that the upper bound must be wrong. This demonstrates the potential benefit of using a benchmark set where there is a feasibility checker available. This is also a motivation for why we recommend researchers to publish their solution files including routing decisions and corresponding delivered quantities.

**Table 4**
Overview of the average deviations from the best-known lower bounds (%) for each exact method.

| Set | $\lvert\mathcal{N}^C\rvert$ | CL-BC | D-BPC | A-BC | G-BC | M-BC | This paper |
|---|---|---|---|---|---|---|---|
| Small | 5–50 | 3.82 | **0.26** | 1.15 | 2.95 | 0.59 | 0.60 |
| Large | 50 | – | – | – | 13.19 | **0.28** | 0.40 |
| | 100 | – | – | – | 5.89 | **0.11** | 0.32 |
| | 200 | – | – | – | 9.14 | – | **0.00** |
| Average Large | | – | – | – | 9.41 | **0.19** | 0.24 |
| Average all | | 3.82 | **0.26** | 1.15 | 4.72 | 0.51 | 0.50 |

**Table 5**
Overview of the number of optimal solutions found.

| Set | V | C-M | D-M | S-M | G-BC | M-BC | V-M1 | This paper |
|---|---|---|---|---|---|---|---|---|
| Small | 2 | 28 | 114 | 51 | **149** | 142 | 71 | 145 |
| | 3 | 21 | 85 | 44 | **120** | 114 | 50 | 116 |
| | 4 | 12 | 44 | 36 | 86 | 92 | 32 | **99** |
| | 5 | 12 | 41 | 30 | 69 | **100** | 29 | 98 |
| Sum Small | | 73 | 284 | 161 | 424 | 448 | 182 | **458** |

**Table 6**
Average computational times in seconds on the 307 instances terminated by G-BC, M-BC, and the B&C embedded matheuristic before the two-hour time limit.

| Set | V | C-M | D-M | S-M | G-BC | M-BC | V-M1 | This paper |
|---|---|---|---|---|---|---|---|---|
| Small | 2 | 55 | 61 | **11** | 327 | 229 | 18 | 639 |
| | 3 | 29 | 37 | 15 | 425 | 502 | **14** | 814 |
| | 4 | 17 | 26 | 23 | 618 | 274 | **9** | 259 |
| | 5 | 16 | 33 | 80 | 702 | 287 | **10** | 245 |
| Average | | 35 | 44 | 25 | 466 | 316 | **14** | 552 |

Table 6 reports the average computational times for the 307 small instances where G-BC, M-BC and the B&C embedded matheuristic all terminated before the two-hour time limit. We believe this gives a fairer comparison of the computational time than including all instances, as that would skew the averages towards the time limit and perhaps cancel out some of the variations between the methods. Here we see that the matheuristics are considerably faster than the three B&C methods and that the B&C methods of Guimarães et al. (2023) and Manousakis et al. (2021) are faster than the B&C embedded matheuristic. However, adjusting the times with the Passmark score, we may multiply the average times of G-BC and M-BC by $7490/1691 \approx 4.42$ and $8652/1691 \approx 5.11$, respectively. This leads to average computational times for the small instances of 2064 and 1616 s for G-BC and M-BC, respectively. This is most likely an overestimate since the benefit of multi-thread computing is small in the root node, but increases when getting many nodes in the B&B tree. In addition, we have used Gurobi 9.5.1 while G-BC and M-BC have used Gurobi 8.1, which also impacts the results. On average, 2.76%, 13.73%, and 83.51% of the time is spent in the construction heuristic, the improvement heuristic, and the B&C method respectively. See Table 15 for more details.

An interesting observation from Table 6 is that our method has shorter computational times on the four- and five-vehicle instances than the two- and three-vehicle instances compared with the two other B&C methods. The main reason for this seems to be that there are more six-period instances solved to optimality for two and three vehicles than for four and five. Our preliminary testing indicated that the B&C method struggles to completely close the optimality gap when using the new customer schedule formulation on the small six-period instances. The dual bounds are good, but the method spends more time closing the last $0.\varepsilon\%$ of the optimality gap for the small six-period instances than if the B&C method was based on the customer schedule formulation of Skålnes et al. (2022). However, the benefit of the new customer schedule formulation is that the B&C method obtains strong dual bounds on the large instances, as seen in Table 4.

### 4.4. Computational results on the DIMACS instances

We have tested the B&C embedded matheuristic on the DIMACS instances and compared our results with all methods that were used in the DIMACS Implementation Challenge. This includes *MrOptimal*, which is an earlier version of the method presented in this paper. We have followed the rules as stated in the DIMACS Implementation Challenge, and all solutions have been checked by the provided feasibility checker. Table 7 reports the number of BKSs and unique BKSs. We can see from the table that our proposed method finds the second most BKSs for the set of small instances and the highest number for the large instances. Table 8 reports the primal gap, in the same manner as in Section 4.3, and it is clear that the proposed method has the best average primal bounds for both sets of instances. The results in Table 7 and Table 8 demonstrate that the method presented in this paper is a significant improvement of $MrOptimal$. It is also worth noting that $MrOptimal$ has lost several BKSs on the large instances to the version presented in this paper, which is why it now has fewer BKSs than the $2FHBC$ method.

**Table 7**
Overview of the BKSs (unique BKSs) on the DIMACS instances.

| Set | $\lvert\mathcal{N}^C\rvert$ | plaisir | TSMHA | IRPUC | 2FHBC | MrOptimal | This paper | # inst. |
|---|---|---|---|---|---|---|---|---|
| Small | 5–50 | 265 (0) | 429 (42) | 496 (26) | **587 (65)** | 508 (31) | **573 (75)** | 798 |
| Large | 50 | 0 (0) | 8 (7) | 8 (7) | 9 (8) | 15 (11) | **45 (42)** | 80 |
| | 100 | 0 (0) | 0 (0) | 2 (2) | 2 (2) | 19 (18) | **58 (57)** | 80 |
| | 200 | 0 (0) | 0 (0) | 0 (0) | 0 (0) | 27 (27) | **53 (53)** | 80 |
| Sum Large | | 0 (0) | 8 (7) | 10 (9) | 11 (10) | 61 (56) | **156 (152)** | 240 |
| Sum all | | 265 (0) | 437 (49) | 506 (35) | 598 (75) | 569 (87) | **729 (227)** | 1038 |

**Table 8**
Overview of average primal gaps (%) on the DIMACS instances.

| Set | $\lvert\mathcal{N}^C\rvert$ | plaisir | TSMHA | IRPUC | 2FHBC | MrOptimal | This paper |
|---|---|---|---|---|---|---|---|
| Small | 5–50 | 3.48 | 0.17 | 0.30 | 0.13 | 0.20 | **0.12** |
| Large | 50 | 8.98 | 0.55 | 1.16 | 0.69 | 0.35 | **0.14** |
| | 100 | 23.92 | 1.08 | 1.64 | 1.21 | 0.26 | **0.10** |
| | 200 | 35.04 | 2.83 | 2.47 | 2.10 | 0.13 | **0.06** |
| Average Large | | 22.65 | 1.49 | 1.76 | 1.33 | 0.25 | **0.10** |
| Average all | | 7.91 | 0.48 | 0.64 | 0.40 | 0.21 | **0.12** |

## 5. Computational study for other routing problems

In this section, we demonstrate that the B&C embedded matheuristic can be efficiently used on other routing problems to achieve high-quality solutions by solving the SDVRP. In addition, we demonstrate the potential of the improvement heuristic by running it on best-known solutions from the CVRP literature. The tests have been performed using the same computational settings as for the IRP.

### 5.1. Computational results on the split delivery vehicle routing problem

There exists several sets of benchmark instances for the SDVRP, that we have tested our method on. One set is the benchmark instances proposed by Belenguer et al. (2000), which consists of 25 instances with 22–101 nodes with either a non-rounded or rounded cost matrix, leading to a total of 50 distinct instances. Further, we have tested on the set of instances proposed by Archetti and Speranza (2008), which consists of 42 instances with 50–199 customers organized in six different groups. Here, the cost matrix is non-rounded. Lastly, we performed computational tests on the 21 instances with $8-288$ customers proposed by Chen et al. (2007). The distance matrix is non-rounded, and the customers are concentrically distributed around the depot. It is common practice in the literature to solve these instances with both an unlimited fleet and with the requirement of using exactly $K = \lceil(\sum_{i\in\mathcal{N}^C} D_i/Q)\rceil$ vehicles. This gives us a total of 226 instances.

We have compared our results with the state-of-the-art heuristics of Silva et al. (2015) (S–H) and He and Hao (2022) (H–H), and the exact methods of Archetti et al. (2011) (A–BP), Archetti et al. (2014) (A–BC), Ozbaygin et al. (2018) (O–BC), and Munari and Savelsbergh (2022) (MS–BC). All parameters are kept the same as for the IRP, except that the improvement MILP has a time limit of 300 s in each iteration.

Table 9 reports for each method, the number of BKSs, unique BKSs and the number of solved instances, respectively. The instances are divided into four subsets depending on whether the instances are solved with a rounded (R) or non-rounded (NR) cost matrix, and whether the vehicle fleet is limited (L) or unlimited (UL). Here we see that the memetic algorithm of He and Hao (2022) clearly obtains the most BKSs and unique BKSs. However, our method finds the second most BKSs and unique BKSs even though it is not specifically tailored to solve the SDVRP. In fact, on the instances with rounded cost matrix and limited vehicle fleet, our method obtains the most BKSs and unique BKSs, which demonstrates that the B&C embedded matheuristic works well also for the SDVRP, and can even compete with the state-of-the-art method on a subset of the instances.

**Table 9**
The number of best-known solutions (unique best-known solution) out of the number of solved instances in each instance set.

| Cost | Fleet | A–BP | A–BC | O–BC | MS–BC | S–H | H–H | This paper |
|---|---|---|---|---|---|---|---|---|
| NR | UL | 2(0)/39 | 0(0)/0 | 0(0)/0 | 27(0)/56 | 43(2)/88 | **83(35)/88** | 39(3)/88 |
| NR | L | 0(0)/0 | 24(0)/41 | 20(0)/28 | 23(0)/37 | 44(5)/88 | **81(36)/88** | 38(2)/88 |
| R | UL | 0(0)/0 | 0(0)/0 | 0(0)/0 | 12(1)/25 | 13(0)/25 | **21(4)/25** | 17(3)/25 |
| R | L | 0(0)/0 | 11(0)/16 | 8(0)/11 | 11(0)/14 | 14(1)/25 | 16(0)/25 | **22(6)/25** |
| Sum | | 2(0)/39 | 35(0)/57 | 28(0)/39 | 73(1)/132 | 114(8)/226 | **201(75)/226** | 116(14)/226 |

**Table 10**
Average primal gap (%) from best-known solution.

| Cost | Fleet | A–BP | A–BC | O–BC | MS–BC | S–H | H–H | This paper |
|---|---|---|---|---|---|---|---|---|
| NR | UL | 1.94 | – | – | 2.35 | 0.05 | **0.00** | 0.09 |
| NR | L | – | 1.24 | 0.10 | 0.40 | 0.06 | **0.00** | 0.10 |
| R | UL | – | – | – | 2.54 | 0.11 | **0.02** | 0.08 |
| R | L | – | 0.65 | 0.06 | 0.82 | 0.09 | 0.07 | **0.03** |
| Average | | 1.94 | 1.07 | 0.09 | 1.68 | 0.07 | **0.01** | 0.08 |

Moreover, we report the average primal gap in % for each of the methods in Table 10. This gives a good indication of the quality of the solutions of each method. Although a method does not find the BKS, it might not be far off. Here, we see that our method is, on average, 0.08% worse than the BKSs, making it one of the top three methods in terms of average primal gaps, and it actually obtains the lowest average primal gap for the subset of instances with rounded cost and limited fleet. We believe this demonstrates the potential the B&C embedded matheuristic has to find good quality solutions to a wider range of routing problems other than the IRP.

### 5.2. Computational results on the capacitated vehicle routing problem

To demonstrate the versatility of the new improvement MILP, we have tested it on the CVRP. Similar to the SDVRP, the new improvement MILP can be adapted to the CVRP by assuming that it is a single-period IRP with no initial inventories and inventory capacities equal to zero. The mathematical model is given in Appendix E. The new improvement MILP has been tested on the ten very large benchmark instances released by Arnold et al. (2019). The new improvement MILP was warm-started with the current best-known solution from CVRPLib (Uchoa et al., 2017) to see if it could improve the solution. A time limit of 10 000 s was used, and the new improvement MILP was able to improve three of the best-known solutions. These were *Brussels2* ($345,481 \rightarrow 345,468$), *Flanders1* ($7,240,124 \rightarrow 7,240,118$), and *Flanders2* ($4,373,320 \rightarrow 4,373,245$). These results indicate that the proposed improvement MILP complements existing neighborhood structures, as it is able to improve on instances where other state-of-the-art methods have reached a local optimum.

## 6. Concluding remarks

In this paper, we have presented a generalized and improved version of the solution method that won the IRP track of the 12th DIMACS Implementation Challenge. We have proposed a new method for generating initial routes, a new generalized improvement MILP, and updated the mathematical formulation used in the B&C method to account for larger instances. Our computational analyses show that these enhancements significantly improved the method. Further, the B&C embedded matheuristic is general and can be used to solve several different types of routing problems. For the IRP, it outperformed all published solution methods in terms of solution quality. The B&C embedded matheuristic found the best-known solution for 741 out of 878 multi-vehicle benchmark instances where 247 of them are strictly better than the previously best-known solutions found in the literature. Further, 458 of the 741 solutions are proven optimal. Thus, the method clearly establishes itself as state-of-the-art. In addition, the computational results show that the proposed method is a significant improvement of the version that won the DIMACS Implementation Challenge. For the SDVRP, the B&C embedded matheuristic produced competitive results compared with the state-of-the-art methods and found the best-known solution for 116 out of 226 well-established benchmark instances. Lastly, the new improvement heuristic was able to improve the best-known solution for three out of ten large benchmark instances for the CVRP released by Arnold et al. (2019).

To the best of our knowledge, we have gathered all published results from every published paper that solves the multi-vehicle benchmark instances of the IRP and made the most complete overview of IRP solutions ever created. We have made this overview easily available online, and we believe this contribution is of great value to anyone interested in doing further research on the IRP. However, our overview indicates that there are several inconsistencies across the results reported in the literature. This might be the result of numerical inaccuracies or varying interpretations of the instances. One possible explanation for some of the inconsistencies might be that some works have omitted constraints (8) and hence solved a relaxed version of the IRP as it is described in Archetti et al. (2007). Another reason might be that the triangle inequality does not hold for all instances, and adding valid inequalities which assume that it does (in reality making them invalid), might cut off the optimal solution. However, it is difficult for us to assess what is the true reason behind these inconsistencies. Prior to the DIMACS Implementation Challenge, an open-source feasibility checker (https://github.com/sbeyer/dimacs-irp-verifier) for the DIMACS instances was made publicly available for DIMACS instances, which is a great resource to reduce the type of errors found in several published articles. Hence, we encourage the research community to use the DIMACS instances moving forward since they have an open-source feasibility checker publicly available. We have provided our solutions for both the DIMACS instances and the original ones.

## CRediT authorship contribution statement

**Jørgen Skålnes:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Simen T. Vadseth:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Henrik Andersson:** Conceptualization, Supervision, Writing – review & editing. **Magnus Stålhane:** Conceptualization, Supervision, Writing – original draft, Writing – review & editing, Project administration.

## Data availability

The results are available at axiomresearchproject.com.

## Acknowledgments

We would like to thank the Norwegian Research Council for funding the research and our collaborators in the AXIOM project [grant number: 263031] for their valuable feedback. We also thank an anonymous reviewer for valuable feedback during the review process, which helped improve the quality of this paper.

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cor.2023.106353.

## References


Achamrah, F.E., 2022. A matheuristic for solving inventory sharing problems: A corrigendum to experimental results. URL https://orbi.uliege.be/bitstream/2268/292714/1/Corrigendum_to_CAOR_paper.pdf.

Achamrah, F.E., Riane, F., Di Martinelly, C., Limbourg, S., 2022. A matheuristic for solving inventory sharing problems. Comput. Oper. Res. 138, 105605.

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

Ahmed, M., Seraj, R., Islam, S.M.S., 2020. The k-means algorithm: A comprehensive survey and performance evaluation. Electronics 9 (8), 1295.

Aleman, R.E., Zhang, X., Hill, R.R., 2010. An adaptive memory algorithm for the split delivery vehicle routing problem. J. Heuristics 16 (3), 441–473.

Alvarez, A., Cordeau, J.F., Jans, R., Munari, P., Morabito, R., 2020. Formulations, branch-and-cut and a hybrid heuristic algorithm for an inventory routing problem with perishable products. European J. Oper. Res. 283 (2), 511–529.

Alvarez, A., Munari, P., Morabito, R., 2018. Iterated local search and simulated annealing algorithms for the inventory routing problem. Int. Trans. Oper. Res. 25 (6), 1785–1809.

Archetti, C., Bertazzi, L., Hertz, A., Speranza, M.G., 2012. A hybrid heuristic for an inventory routing problem. INFORMS J. Comput. 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. Transp. Sci. 41 (3), 382–391.

Archetti, C., Bianchessi, N., Speranza, M.G., 2011. A column generation approach for the split delivery vehicle routing problem. Networks 58 (4), 241–254.

Archetti, C., Bianchessi, N., Speranza, M.G., 2014. Branch-and-cut algorithms for the split delivery vehicle routing problem. European J. Oper. Res. 238 (3), 685–698.

Archetti, C., Boland, N., Speranza, M.G., 2017. A matheuristic for the multivehicle inventory routing problem. INFORMS J. Comput. 29 (3), 377–387.

Archetti, C., Guastaroba, G., Huerta-Muñoz, D.L., Speranza, M.G., 2021. A kernel search heuristic for the multivehicle inventory routing problem. Int. Trans. Oper. Res. 28 (6), 2984–3013.

Archetti, C., Ljubić, I., 2022. Comparison of formulations for the Inventory Routing Problem. European J. Oper. Res. 303 (3), 997–1008.

Archetti, C., Savelsbergh, M.W., Speranza, M.G., 2006a. Worst-case analysis for split delivery vehicle routing problems. Transp. Sci. 40 (2), 226–234.

Archetti, C., Speranza, M.G., 2008. The split delivery vehicle routing problem: A survey. In: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 103–122.

Archetti, C., Speranza, M.G., Hertz, A., 2006b. A tabu search algorithm for the split delivery vehicle routing problem. Transp. Sci. 40 (1), 64–73.

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

Arnold, F., Gendreau, M., Sörensen, K., 2019. Efficiently solving very large-scale routing problems. Comput. Oper. Res. 107, 32–42.

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. Transp. Sci. 52 (3), 497–508.

Baldacci, R., Hadjiconstantinou, E., Mingozzi, A., 2004. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Oper. Res. 52 (5), 723–738.

Belenguer, J.-M., Martinez, M., Mota, E., 2000. A lower bound for the split delivery vehicle routing problem. Oper. Res. 48 (5), 801–810.

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

Boudia, M., Prins, C., Reghioui, M., 2007. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In: International Workshop on Hybrid Metaheuristics. Springer, pp. 16–30.

Campos, V., Corberán, A., Mota, E., 2008. A scatter search algorithm for the split delivery vehicle routing problem. In: Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management. Springer, pp. 137–152.

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

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

Coelho, L.C., Cordeau, J.F., Laporte, G., 2012. Consistency in multi-vehicle inventory-routing. Transp. Res. C 24 (Supplement C), 270–287.

Coelho, L.C., Laporte, G., 2014. Improved solutions for inventory-routing problems through valid inequalities and input ordering. Int. J. Prod. Econ. 155, 391–397.

Derigs, U., Li, B., Vogel, U., 2010. Local search-based metaheuristics for the split delivery vehicle routing problem. J. Oper. Res. Soc. 61 (9), 1356–1364.

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

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

DIMACS, 2022. http://dimacs.rutgers.edu/programs/challenge/vrp/irp/. (Accessed 16 July 2022).

Diniz, P., Martinelli, R., Poggi, M., 2020. An efficient matheuristic for the inventory routing problem. In: Baïou, M., Gendron, B., Günlük, O., Mahjoub, A.R. (Eds.), Combinatorial Optimization. Springer International Publishing, Cham, ISBN: 978-3-030-53262-8, pp. 273–285.

Dror, M., Laporte, G., Trudeau, P., 1994. Vehicle routing with split deliveries. Discrete Appl. Math. 50 (3), 239–254.

Dror, M., Trudeau, P., 1989. Savings by split delivery routing. Transp. Sci. 23 (2), 141–145.

Dror, M., Trudeau, P., 1990. Split delivery routing. Nav. Res. Logist. 37 (3), 383–402.

Guimarães, T.A., Schenekemberg, C.M., Coelho, L.C., Scarpin, C.T., Jr., J.E.P., 2023. Mechanisms for feasibility and improvement for inventory-routing problems. Journal of the Operational Research Society 1–13. http://dx.doi.org/10.1080/01605682.2023.2174052.

He, P., Hao, J.-K., 2022. General edge assembly crossover-driven memetic search for split delivery vehicle routing. Transp. Sci. 57 (2), 482–511.

Jin, M., Liu, K., Eksioglu, B., 2008. A column generation approach for the split delivery vehicle routing problem. Oper. Res. Lett. 36 (2), 265–270.

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

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 J. Oper. Res. 290 (3), 870–885.

Munari, P., Savelsbergh, M., 2022. Compact formulations for split delivery routing problems. Transp. Sci. 56 (4), 1022–1043.

Ozbaygin, G., Karasan, O., Yaman, H., 2018. New exact solution approaches for the split delivery vehicle routing problem. EURO J. Comput. Optim. 6 (1), 85–115.

Sakhri, M.S.A., Tlili, M., Korbaa, O., 2022. A memetic algorithm for the inventory routing problem. J. Heuristics 28 (3), 351–375.

Shi, J., Zhang, J., Wang, K., Fang, X., 2018. Particle swarm optimization for split delivery vehicle routing problem. Asia-Pac. J. Oper. Res. 35 (02), 1840006.

Silva, M.M., Subramanian, A., Ochi, L.S., 2015. An iterated local search heuristic for the split delivery vehicle routing problem. Comput. Oper. Res. 53, 234–249.

Skålnes, J., Andersson, H., Desaulniers, G., Stålhane, M., 2022. An improved formulation for the inventory routing problem with time-varying demands. European J. Oper. Res. 302 (3), 1189–1201.

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

Solyalı, O., Süral, H., 2022. An effective matheuristic for the multivehicle inventory routing problem. Transp. Sci. 56 (4), 1044–1057.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A., 2017. New benchmark instances for the capacitated vehicle routing problem. European J. Oper. Res. 257 (3), 845–858.

Vadseth, S.T., Andersson, H., Stålhane, M., 2021. An iterative matheuristic for the inventory routing problem. Comput. Oper. Res. 131, 105262.

Vadseth, S.T., Andersson, H., Stålhane, M., Chitsaz, M., 2023. A Multi-Start Route Improving Matheuristic for the Production Routing Problem. Taylor & Francis, pp. 1–22. http://dx.doi.org/10.1080/00207543.2022.2154402.

Vidal, T., 2016. Split algorithm in $O(n)$ for the capacitated vehicle routing problem. Comput. Oper. Res. 69, 40–47.

Vidal, T., 2022. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. Comput. Oper. Res. 140, 105643.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. 60 (3), 611–624.

Wilck, IV, J.H., Cavalier, T.M., 2012. A genetic algorithm for the split delivery vehicle routing problem. Transp. Sci. 2 (2), 207–216.