Les Cahiers du GERAD ISSN: 0711–2440

A branch-price-and-cut algorithm for the multi-commodity two-echelon vehicle routing problem with time windows

T. Mhamedi, M. Cherkesly, G. Desaulniers

G–2024–79

December 2024

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2024
– Bibliothèque et Archives Canada, 2024

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2024
– Library and Archives Canada, 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A branch-price-and-cut algorithm for the multi-commodity two-echelon vehicle routing problem with time windows

Tayeb Mhamedi [a, c]

Marilène Cherkesly [b, c]

Guy Desaulniers [a, c]

[a] *Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, (Qc), Canada, H3T 1J4*

[b] *Département d'analytique, opérations et technologies de l'information, Université du Québec à Montréal, Montréal (Qc), Canada, H2X 1L7*

[c] *GERAD, Montréal (Qc), Canada, H3T 1J4*

tayeb.mhamedi@polymtl.ca
marilene.cherkesly@gerad.ca
guy.desaulniers@gerad.ca

December 2024
Les Cahiers du GERAD
G–2024–79

Abstract : In the multi-commodity two-echelon vehicle routing problem with time windows (MC-2E-VRPTW), first-echelon vehicles transport goods from depots to satellites while second-echelon vehicles ensure that goods are shipped from satellites to customers within their time windows. Given a set of customers, each with demand available at one depot, the MC-2E-VRPTW aims at determining least-cost and capacity-feasible first- and second-echelon routes such that each customer is serviced during its time window by a second-echelon route, and has a single first-echelon route supplying its whole demand. For this problem, we propose a route-based formulation that contains an exponential number of variables associated with second-echelon routes, and develop a tailored branch-price-and-cut algorithm. This algorithm considers one subproblem per satellite which is solved by a labeling algorithm to generate second-echelon routes and determine the first-echelon route supplying the load of each visited customer. We devise a recovery procedure to enforce integer solution feasibility in the presence of dual inequalities and propose a branching rule adapted to the multi-commodity context. Through extensive computational experiments on benchmark instances, we show that our algorithm outperforms a state-of-the-art algorithm.

Keywords : City logistics, two-echelon vehicle routing, multiple commodities, branch-price-and-cut, labeling algorithm, dual inequalities

Acknowledgements: We are grateful to the Natural Sciences and Engineering Research Council of Canada (NSERC) for providing financial support through the Discovery grants 2017–06106 and 2023–03791.

1 Introduction

In this paper, we propose a branch-price-and-cut algorithm for the multi-commodity two-echelon vehicle routing problem with time windows (MC-2E-VRPTW). This problem extends the two-echelon vehicle routing problem (2E-VRP, see Cuda et al. 2015, Guastaroba et al. 2016, Sluijk et al. 2023, for surveys on the 2E-VRP) to consider real-life characteristics that arise in city logistics, i.e., time windows and multiple commodities. Considering multiple commodities is of major importance when optimizing distribution processes on two-echelon network structures involving multiple origins with, e.g., customer-specific (non-substitutable) demands. The MC-2E-VRPTW involves depots, satellites and customers. Each customer is assigned to one depot that supplies its demand, which is referred to as a commodity. Therefore, there are as many commodities as there are customers. In addition, the service at each customer must start within a predefined time window. The first echelon corresponds to delivering goods from the depots to the satellites, and a set of identical first-echelon vehicles with limited capacity is available to conduct the first-echelon routes. Each first-echelon route starts and ends at the same depot, and visits a subset of satellites. We define two types of first-echelon routes: i) routes visiting only one satellite, referred to as back-and-forth routes, and ii) routes visiting more than one satellite. As often seen in practice, we assume that back-and-forth routes can be used more than once, whereas routes visiting more than one satellite cannot. The second echelon corresponds to delivering goods from the satellites to the customers, and a set of identical second-echelon vehicles with limited capacity is available to operate second-echelon routes. The merchandise is transferred from first- to second-echelon vehicles at the satellites, and transfer times at the satellites are assumed to be constant, i.e., they do not depend on the quantity transferred. The synchronization between first-and second-echelon vehicles is *exact* (see Drexl 2012, for a taxonomy), implying that each demand must be supplied by exactly one first-echelon route and one second-echelon route, but that a second-echelon route can be supplied by more than one first-echelon route. The MC-2E-VRPTW consists in determining a set of least-cost feasible first- and second-echelon routes such that each customer is visited by exactly one second-echelon route within its time window, and that its load is supplied by its assigned depot.

To the best of our knowledge, only the paper of Dellaert et al. (2021) studies the MC-2E-VRPTW, proposing several formulations and solution algorithms. Their best algorithm is a branch-and-price-based algorithm that solves their two-path formulation. It first generates subsets of first-echelon routes (called configurations) and sorts them in increasing order of a lower bound obtained by selecting the corresponding configuration as the first-echelon solution. These configurations with their lower bounds can then be used to define multiple root nodes in a branch-and-price algorithm that does not need to branch to achieve integrality at the first echelon. The algorithm starts with a single root node associated with the least-lower-bound configuration and proceeds as a standard branch-and-price algorithm, branching only to achieve integrality at the second echelon. When exploring the search tree, new root nodes (configurations) are added whenever the current lower bound in the tree exceeds the lower bounds associated with unexplored configurations. The algorithm stops when the current tree is fully explored and the best upper bound is less than or equal to the lower bounds of the unexplored configurations. Dellaert et al. (2021) report results on a small set of instances with up to 3 depots, 5 satellites, and 100 customers.

On the other hand, many heuristics (e.g., Breunig et al. 2016, Crainic et al. 2008, 2011, Dumez et al. 2023, Jia et al. 2023) and exact algorithms (e.g., Baldacci et al. 2013, Perboli et al. 2011) have been proposed for the classical 2E-VRP and its variants. For the exact algorithms, some of the best results are obtained by branch-and-price algorithms (e.g., Marques et al. 2020, Santos et al. 2015). In addition, many variants with time windows or multiple commodities have been studied. The two most relevant variants with time windows are the multi-depot 2E-VRP with time windows (2E-VRPTW, see Dellaert et al. 2019), and the 2E-VRPTW with freight consolidation at the satellites (see Marques et al. 2022). The 2E-VRPTW considers exact synchronization (Drexl 2012, Soares et al. 2024) between the first- and second-echelon vehicles, i.e., no freight consolidation occurs at the satellites and each second-echelon

vehicle is supplied from a single first-echelon vehicle. In the 2E-VRPTW with freight consolidation, the load aboard a second-echelon vehicle can be supplied by more than one first-echelon vehicle. The literature on the 2E-VRP with multiple commodities is limited, but has received more attention lately. In this literature, the commodities are defined either in a *many-to-many* or a *one-to-one* context, as highlighted in Gu et al. (2023). The two most relevant variants with multiple commodities are the multi-commodity 2E-VRP with satellite synchronization and the multi-commodity two-echelon distribution problem (MC-2EDP). In the first variant introduced by Jia et al. (2023), a commodity refers to an origin-destination pair in a one-to-one setting. Customers request two commodities, and the total demand of a customer must be delivered by a single second-echelon vehicle. In the MC-2EDP studied by Gu et al. (2022), commodities are sent from depots to satellites using back-and-forth routes before being transferred to second-echelon vehicles. A customer can request more than one commodity and each commodity needs to be supplied in a single visit. In the following, we provide a brief overview of the state-of-the-art exact algorithms for the most relevant variants of the 2E-VRP with time windows or multiple commodities, with an emphasis on branch-and-price algorithms.

To solve the 2E-VRPTW, Dellaert et al. (2019) propose the first branch-and-price-based algorithm, which is similar to the algorithm of Dellaert et al. (2021) described above and can solve to optimality instances with up to 3 depots, 5 satellites and 100 customers within a 3-hour time limit. Their results are outperformed by Mhamedi et al. (2022) who develop a branch-price-and-cut (BPC) algorithm involving dual inequalities and in which the labeling algorithm used to solve the pricing subproblems generates second-echelon routes and specifies their load-supplying first-echelon route. Marques et al. (2022) introduce a branch-and-price algorithm for the 2E-VRPTW with freight storage and consolidation at the satellites. They formulate the problem with a route-based formulation containing an exponential number of precedence constraints to enforce synchronization between first- and second-echelon vehicles. Their algorithm is adapted to consider two cases: the second-echelon routes contain a single trip each or they can involve multiple trips, possibly starting and ending at different satellites. The authors also present a post-processing procedure to verify whether feasible solutions can be transformed into same-cost ones requiring no storage or freight consolidation. Their algorithm performs well on instances from the literature, yielding new optimal solutions for 9 and 54 instances of the multi- and single-trip cases, respectively. To our knowledge, Petris et al. (2024) propose the only exact algorithm for the MC-2EDP, which is formulated as a set-covering model with an exponential number of variables, one for each second-echelon route. To solve it, the authors devise a BPC algorithm where second-echelon variables are generated dynamically by solving one subproblem per satellite. Each subproblem is modeled as an elementary shortest path problem with resource constraints (ESPPRC) and solved by a label setting algorithm. To tighthen the lower bounds, capacity inequalities and two families of cuts exploiting the multicommodity aspect of the problem are considered. Within a 1-hour time limit, their algorithm solves to optimality 439 of the 736 instances created by Gu et al. (2022).

In this paper, we address the MC-2E-VRPTW and develop an exact BPC algorithm that outperforms the algorithm of Dellaert et al. (2021). Our contributions are fourfold. First, we propose a BPC algorithm (inspired from that of Mhamedi et al. 2022) that does not rely on an a priori enumeration of first-echelon route subsets as in Dellaert et al. (2021). This algorithm generates dynamically the second-echelon routes using a labeling algorithm that determines which first-echelon route supplies each customer visited in the second-echelon routes. Second, we devise a recovery procedure that is applied to ensure integer solution feasibility in the presence of dual inequalities. Third, to tackle instances with a large number of first-echelon routes, we introduce a branching rule adapted to the multi-commodity context. Finally, we extend the benchmark dataset of Dellaert et al. (2021) by creating 140 new MC-2E-VRPTW instances based on the 2E-VRPTW instances of Dellaert et al. (2019) and test our algorithm on a total of 180 test instances with 30, 50 and 100 customers.

The rest of this paper is organized as follows. Section 2 presents notation and formulates the MC-2E-VRPTW as a set-partitioning problem with side constraints. Section 3 describes the proposed BPC algorithm. Computational results are reported in Section 4 to assess this algorithm and the importance of different algorithmic components. Conclusions are drawn in Section 5.

2 Mathematical formulation

The MC-2E-VRPTW is defined on a directed graph $G = (N, A)$, where N is its vertex set and A its arc set. Set $N = N^D \cup N^S \cup N^C$ comprises the sets of depots (N^D), satellites (N^S), and customers (N^C). For each depot $d \in N^D$, let $N^C_d \subseteq N^C$ be its set of assigned customers. Therefore, $N^C_{d_1} \cap N^C_{d_2} = \emptyset, \forall d_1, d_2 \in N^D, d_1 \neq d_2$. Each customer $i \in N^C$ has a demand q_i that needs to be supplied from its assigned depot $d_i \in N^D$. Service at customer i requires a time τ_i and can only start during time window $[\underline{w}_i, \overline{w}_i]$, where $\underline{w}_i$ and $\overline{w}_i$ are the earliest and latest start times. Each satellite $s \in N^S$ has a predefined transfer time τ_s representing the time to unload the merchandise from first-echelon vehicles and load the second-echelon vehicles. Arc set $A = A_1 \cup A_2$ comprises the subsets of arcs that can be traversed by first-echelon vehicles (A_1) and by second-echelon vehicles (A_2). Set A_1 contains three arc types: *i*) arcs from a depot to a satellite, *ii*) arcs from a satellite to a depot, and *iii*) arcs between satellites, that is, $A_1 = \{(d, s) : d \in N^D, s \in N^S\} \cup \{(s, d) : s \in N^S, d \in N^D\} \cup \{(s_1, s_2) : s_1, s_2 \in N^S, s_1 \neq s_2\}$. Similarly, set A_2 contains three arc types: i) arcs from a satellite to a customer, ii) arcs from a customer to a satellite, and iii) arcs between customers, that is, $A_2 = \{(s, i) : s \in N^S, i \in N^C\} \cup \{(i, s) : i \in N^C, s \in N^S\} \cup \{(i_1, i_2) : i_1, i_2 \in N^C, i_1 \neq i_2\}$. Each arc $(i, j) \in A$ is associated with a routing cost c_{ij} and a travel time t_{ij} which includes the service time at vertex i (i.e., 0 if $i \in N^D$ and τ_i if $i \in N^S \cup N^C$).

The set of first-echelon routes is denoted by $\mathcal{M}$, and each first-echelon route is performed by a first-echelon vehicle with a capacity Q_1 and a fixed cost f_1. Each first-echelon route $m = (i_0,\ i_1, \ldots,\ i_h,\ i_{h+1})$ starts and ends at the same depot $i_0 = i_{h+1} = d_m \in N^D$ and visits a subset of satellites $i_1, \ldots, i_h \in N^S$, such that $i_j \neq i_k, \forall j, k \in \{1, \ldots, h\},\ j \neq k$. Its total load cannot exceed vehicle capacity Q_1. For each pair of depot $d \in N^D$ and satellite $s \in N^S$, we create $\lceil \sum_{i \in N^C_d} q_i / Q_1 \rceil$ copies of the back-and-forth first-echelon route (d, s, d) and denote by $\mathcal{M}^1_{ds}$ the set of these copies that are included in $\mathcal{M}$. For each pair of first-echelon route $m \in \mathcal{M}$ and satellite $s \in N^S$, we define the binary parameter a_{sm} which is equal to 1 if route m visits satellite s and 0 otherwise. Each route m incurs a total cost c_m that includes a first-echelon vehicle fixed cost and its routing cost, i.e., $c_m = f_1 + \sum_{k=0}^{h} c_{i_k, i_{k+1}}$. We assume that every first-echelon route departs at time 0 from depot i_0 and we denote by t^s_m its arrival time at a visited satellite s. Finally, we denote by $\mathcal{M}_d$ the subset of first-echelon routes associated with depot d and by $\mathcal{M}_{ds} \subseteq \mathcal{M}_d$ the subset of those routes visiting satellite s.

The set of second-echelon routes is denoted by $\mathcal{L}$ and the second-echelon vehicles have a capacity $Q_2 < Q_1$ and a fixed cost f_2. A second-echelon route $l = (i_0,\ i_1, \ldots,\ i_h,\ i_{h+1})$ starts and ends at a satellite $i_0 = i_{n+1} = s_l \in N^S$ and visits a subset of customers $i_1, \ldots, i_h \in N^C$, such that $i_j \neq i_k$ for all $j, k \in \{1, \ldots, h\},\ j \neq k$. Its total load cannot exceed vehicle capacity Q_2. For every pair of second-echelon route $l \in \mathcal{L}$ and customer $i \in N^C$, we define the binary parameter a_{il} which is equal to 1 if route l visits customer i and 0 otherwise. In addition, the total cost c_l of a route l is composed of its fixed cost and its routing cost, i.e., $c_l = f_2 + \sum_{k=0}^{h} c_{i_k, i_{k+1}}$. Each route l is associated with a departure time t^{start}_l from satellite s_l that guarantees servicing customers $i_1, \ldots, i_h$ within their respective time windows. For each second-echelon route l, we denote by $\mathcal{M}_l \subseteq \{m \in \mathcal{M} : a_{s_l m} = 1 \text{ and } t^s_m \leq t^{start}_l - \tau_{s_l}\}$ the set of first-echelon routes that can supply it before its departure time and by $\mathcal{P}_l$ its set of supply patterns. A supply pattern $p \in \mathcal{P}_l$ specifies, for each visited customer, the first-echelon route in $\mathcal{M}_l$ that supplies its demand and is expressed as $p = \{(m_1, i_1), \ldots, (m_h, i_h)\}$, where $m_k \in \mathcal{M}_l \cap \mathcal{M}_{d_{i_k}}, \forall k \in \{1, \ldots, h\}$. Finally, let Q^l_{mp} be the total demand supplied by first-echelon route m when assigning supply pattern p to route l, i.e., $Q^l_{mp} = \sum_{\{i \in N^C:\ a_{il}=1\ \wedge\ (m,i) \in p\}} q_i$.

To better understand the concept of a supply pattern, Figure 1 presents a solution for a MC-2E-VRPTW instance with two depots ($|N^D| = 2$), three satellites ($|N^S| = 3$), and 12 customers ($|N^C| = 12$). Customers 1 to 7 are assigned to depot d_1, whereas customers 8 to 12 are assigned to depot d_2, i.e., $N^C_{d_1} = \{1, 2, \ldots, 7\}$ and $N^C_{d_2} = \{8, 9, \ldots, 12\}$. There are five feasible first-echelon routes $(m_1, m_2, \ldots, m_5)$, but only the first three are used in this solution, which also includes four second-

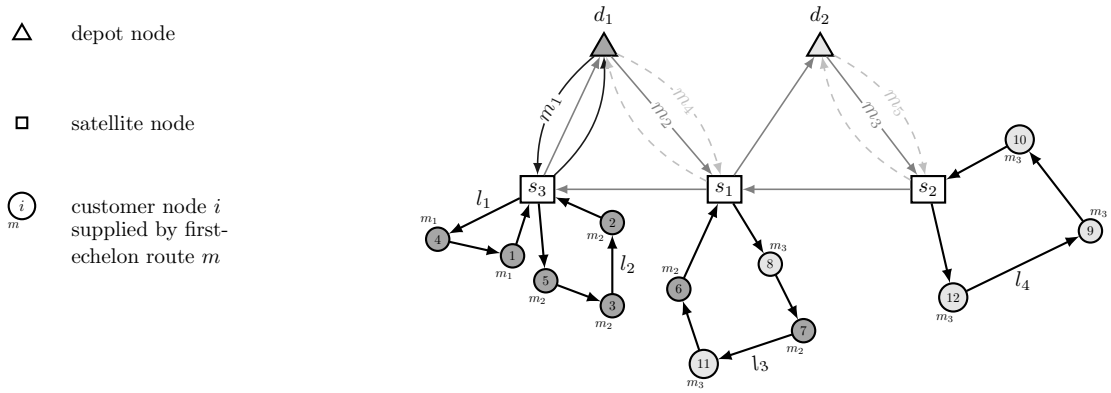

Figure 1: Example of a solution for an instance of the MC-2E-VRPTW

echelon routes l_1, l_2, l_3, and l_4. As an example, the supply pattern set for l_1 is $\mathcal{P}_{l_1} = \{(m_1, 1), (m_1, 4)\} \cup \{(m_1, 1), (m_2, 4)\} \cup \{(m_2, 1), (m_1, 4)\} \cup \{(m_2, 1), (m_2, 4)\}$, assuming that m_1 and m_2 arrive at satellite s_3 early enough to supply route l_1. Finally, in this solution, routes l_1 to l_4 are associated with supply patterns $\{(m_1, 1), (m_1, 4)\}$, $\{(m_2, 2), (m_2, 3), (m_2, 5)\}$, $\{(m_2, 6), (m_2, 7), (m_3, 8), (m_3, 11)\}$, and $\{(m_3, 9), (m_3, 10), (m_3, 12)\}$, respectively.

The MC-2E-VRPTW can then be formulated using two sets of binary variables. Let x_m be equal to 1 if first-echelon route $m \in \mathcal{M}$ is used in the solution, and 0 otherwise. For each pair of second-echelon route $l \in \mathcal{L}$ and supply pattern $p \in \mathcal{P}_l$, let y_{lp} be equal to 1 if route l is used in the solution and supplied using pattern p, and 0 otherwise. The problem can then be formulated as the following integer linear programming model:

$$\min \quad \sum_{m \in \mathcal{M}} c_m x_m + \sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} c_l y_{lp} \tag{1}$$

$$\text{s.t.} \quad \sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} a_{il} y_{lp} = 1, \qquad \forall i \in N^C, \tag{2}$$

$$\sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} Q^l_{mp} y_{lp} \leq Q_1 x_m, \qquad \forall m \in \mathcal{M}, \tag{3}$$

$$x_m \in \{0, 1\}, \qquad \forall m \in \mathcal{M}, \tag{4}$$

$$y_{lp} \in \{0, 1\}, \qquad \forall l \in \mathcal{L}, p \in \mathcal{P}_l. \tag{5}$$

Objective function (1) minimizes the total costs (sum of fixed vehicle and routing costs) incurred at both echelons. Set-partitioning constraints (2) ensure that each customer is serviced by a single second-echelon route. The synchronization constraints (3) between the first- and second-echelon routes ensure that the first-echelon routes involved in the supply pattern assigned to each selected second-echelon route are also selected in the solution. They also guarantee that the capacity of each first-echelon route is respected. Finally, (4) and (5) define the domain of the variables.

To model (1)–(5), we add symmetry-breaking constraints to distinguish the back-and-forth route copies in set $\mathcal{M}^1_{ds}$ for each pair of depot $d \in N^D$ and satellite $s \in N^S$. Without loss of generality, let us assume that each set $\mathcal{M}^1_{ds}$ is ordered, and let $\bar{m}^1_{ds}$ denote the last element of set $\mathcal{M}^1_{ds}$. The symmetry-breaking constraints express as follows:

$$x_m \leq x_{m^+}, \qquad \forall s \in N^S, d \in N^D, m \in \mathcal{M}^1_{ds} \setminus \{\bar{m}^1_{ds}\}, \tag{6}$$

where m^+ denotes the immediate successor of back-and-forth copy m in $\mathcal{M}^1_{ds}$.

3 Branch-price-and-cut algorithm

Model (1)–(6) usually contains a large number of variables. In the MC-2E-VRPTW, the cardinality of the first-echelon route set $\mathcal{M}$ is usually quite limited, but that of the second-echelon route set $\mathcal{L}$ can be very large. To avoid enumerating all second-echelon route variables, we resort to a branch-price-and-cut (BPC) algorithm, where the first-echelon route variables are all enumerated a priori. A BPC algorithm is a branch-and-cut algorithm where the linear relaxations are solved by column generation (CG, see, e.g., Costa et al. 2019). CG is an iterative algorithm that solves at each iteration a restricted master problem (RMP) and one or several (pricing) subproblems. In our case, the RMP corresponds to the linear relaxation of model (1)–(6) with a limited subset of second-echelon route variables, i.e., with index subsets $\mathcal{L}' \subseteq \mathcal{L}$ and $\mathcal{P}'_l \subseteq \mathcal{P}_l$. Given an optimal dual solution to the current RMP, the subproblems aim at finding negative reduced cost y_{lp} variables if some exist. In our case, there is one subproblem per satellite that can be cast as an ESPPRC (ESPPRC, Irnich and Desaulniers 2005). When negative reduced cost variables (columns) are found, they are added to the RMP before starting a new iteration. Otherwise, CG stops as the current RMP solution is also optimal for the complete linear relaxation. If this solution is not integer, branching and cutting is applied, re-optimizing each linear relaxation by CG.

In the following, we describe our BPC algorithm. First, we define the CG subproblems and develop a labeling algorithm to solve them. Second, we propose dual inequalities to stabilize the CG process and describe how the BPC algorithm handles them. Finally, we present the valid inequalities and branching rules used.

3.1 Subproblems

There is one subproblem per satellite $s \in N^S$, denoted SP_s and used to generate second-echelon variables y_{lp} associated with routes starting from satellite s. This subproblem is an ESPPRC that determines i) the customers visited in a second-echelon route and ii) the first-echelon routes supplying these customers while satisfying the customer-to-depot assignments, the second-echelon vehicle capacity, and the customer time windows. Given $(\sigma_i)_{i\in N^C}$ and $(\pi_m)_{m\in\mathcal{M}}$ the dual variables associated with constraints (2)–(3), respectively, the reduced cost of a variable y_{lp}, $l \in \mathcal{L}$ and $p \in \mathcal{P}_l$, is computed as

$$\bar{c}_{lp} = c_l - \sum_{i\in N^C} a_{il}\sigma_i - \sum_{m\in\mathcal{M}} Q_l^{mp}\pi_m. \tag{7}$$

Subproblem SP_s aims at finding a pair of second-echelon route l departing from s and supply pattern p with a minimum reduced cost $\bar{c}_{lp}$.

For a given second-echelon route l starting from satellite s, there might exist a large number of feasible supply patterns $p \in \mathcal{P}_l$. The feasibility of a pattern depends on the route starting time t_l^{start}. Given that this time is a priori unknown when solving SP_s, we propose to consider that $t_l^{start} = t + \tau_s$, where t belongs to the set T_s of the first-echelon route visiting times to satellite s (sorted in ascending order). For all starting times $t_l^{start} = t + \tau_s$, $t \in T_s$, we can then find the supply pattern $p \in \mathcal{P}_l$ that yields the least reduced cost $\bar{c}_{lp}$ among the supply patterns that are feasible if route l starts at time t_l^{start}. For $t \in T_s$, let $\mathcal{M}_s^t = \cup_{d\in N^D}\{m \in \mathcal{M}_d : t_s^m \leq t\}$ be the set of first-echelon routes visiting satellite s no later than t and $D_s^t = \{d \in N^D : \mathcal{M}_s^t \cap \mathcal{M}_d \neq \emptyset\}$ the set of depots associated with those routes. Furthermore, for $d \in D_s^t$, we denote by $m(d,t)$ a first-echelon route $m \in \mathcal{M}_s^t \cap \mathcal{M}_d$ with the largest dual value π_m, i.e., $m(d,t) \in \arg\max_{m\in\mathcal{M}_s^t\cap\mathcal{M}_d} \pi_m$. With this notation, the reduced cost defined in (7) can be rewritten as

$$\bar{c}_{lp} = c_l - \sum_{i\in N^C} a_{il}\sigma_i - \sum_{i\in N^C} a_{il}q_i\pi_{m(d_i,t)} \quad \text{if } t_l^{start} = t + \tau_s. \tag{8}$$

The subproblems are solved by the labeling algorithm (Irnich and Desaulniers 2005) described in Subsection 3.1.1. Two acceleration strategies, summarized in Subsection 3.1.2, are applied to speed up the solution process.

3.1.1 Labeling algorithm.

Each subproblem SP_s, $s \in N^S$, consists in finding a least reduced cost path on a directed graph $G_s = (N_s, A_s)$. Vertex set $N_s = N^C \cup \{n_s^{src}, n_s^{sk}\}$ comprises the set of customers as well as a source and a sink vertex, which represent copies of satellite s. We set $q_s^{src} = q_s^{sk} = 0$. Arc set $A_s = \{(n_s^{src}, i) : i \in N^C\} \cup \{(i, n_s^{sk}) : i \in N^C\} \cup \{(i_1, i_2 : i_1, i_2 \in N^C, i_2 \neq i_2)\}$. Each arc $(i, j) \in A_s$ is associated with the time t_{ij} of the corresponding arc in A_2 and with a modified cost $\bar{c}_{ij}$ defined as:

$$\bar{c}_{ij} = \begin{cases} c_{ij} + f_2 - \sigma_j, & \text{if } i = n_s^{src} \\ c_{ij} - \sigma_j, & \text{if } i, j \in N^C \\ c_{ij}, & \text{if } j = n_s^{sk}. \end{cases}$$

In a standard labeling algorithm, a label is a multi-dimensional vector that represents a partial path starting at the source node n_s^{src} and ending at a vertex $i \in N_s$. Starting from an initial label representing the path containing only vertex n_s^{src}, this algorithm recursively extends it towards the other vertices in N_s using resource extension functions (REFs) to generate longer partial paths until reaching the sink vertex n_s^{sk}. Infeasible paths are discarded as the search progresses. To avoid enumerating all feasible source-to-sink paths, a dominance rule comparing label pairs is applied. Below, we define the labels, describe how the algorithm is initialized with multiple labels (instead of a single one), and present the REFs and the dominance rule.

Label definition. A label represents a combination of a partial path l starting from n_s^{src} at a given time $t + \tau_s$, $t \in T_s$, and an associated supply pattern p which can be retrieved from the list of visited customers and time t. Such a label $E_{lt} = (T_{lt}^{rdc}, T_{lt}^{time}, T_{lt}^{load}, T_{lt}^{start}, U_{lt})$ associated with a path ending at vertex $i \in N_s$ holds the following information:

T_{lt}^{rdc}**:** the reduced cost of this path;
T_{lt}^{time}**:** the earliest time for starting service at vertex i;
T_{lt}^{load}**:** the total delivered demand;
T_{lt}^{start}**:** time t corresponding to the latest time a first-echelon route can visit satellite s to supply this partial route;
U_{lt}**:** the set of unreachable customers from label E_{lt}. A customer h is said to be unreachable from E_{lt} if $d_h \notin D_s^{T_{lt}^{start}}$, or h has already been visited in this path or cannot be due to time or load constraints (i.e., $T_{lt}^{time} + t_{ih} > \overline{w}_h$ or $T_{lt}^{load} + q_h > Q_2$).

Initialization. Recall from the discussion above that only $|T_s|$ starting times are relevant, i.e., the times $t + \tau_s$ for $t \in T_s$. Given one of these times, it becomes easy to identify the best supply pattern for a partial path. Therefore, our labeling algorithm starts with $|T_s|$ initial labels at source vertex n_s^{src}, namely, one for each $t \in T_s$. Denoting by 0 the partial path containing only the source vertex n_s^{src}, an initial label E_{0t}, $t \in T_s$, is set to $E_{0t} = (0, t, 0, t, \{h \in N^C : d_h \notin D_s^t \text{ or } t + t_{n_s^{src}h} > \overline{w}_h\})$.

Resource extension functions. To generate new labels, the following REFs are applied. Let $E_{lt} = (T_{lt}^{rdc}, T_{lt}^{time}, T_{lt}^{load}, T_{lt}^{start}, U_{lt})$ be a label associated with a partial path l ending at vertex $i \in N_s \backslash \{n_s^{sk}\}$ and a path starting time $t \in T_s$. Extending E_{lt} along an arc $(i, j) \in A_s$ such that $j \notin U_{lt}$ yields a label $E_{l't}$ whose components are computed as follows:

$$T_{l't}^{rdc} = T_{lt}^{rdc} + \bar{c}_{ij} - \begin{cases} q_j \cdot \pi_{m(d_j, T_{lt}^{start})} & \text{if } j \neq n_s^{sk}, \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

$$T_{l't}^{time} = \max\{\underline{w}_j, T_{lt}^{time} + t_{ij}\}, \tag{10}$$

$$T_{l't}^{load} = T_{lt}^{load} + q_j, \tag{11}$$

$$T_{l't}^{start} = T_{lt}^{start}, \tag{12}$$

$$U_{l't} = U_{lt} \cup \{j\} \cup \{h \in \bigcup_{d \in D_s^{T_{lt}^{start}}} N_d^C : T_{l't}^{time} + t_{jh} > \overline{w}_h \vee T_{l't}^{load} + q_h > Q_2\}. \tag{13}$$

Label $E_{l't}$ is always deemed feasible because $j \notin U_{lt}$. Observe that, if $j \in N^C$, the first-echelon route supplying j is route $m(d_j, T_{lt}^{start})$ as identified in the first case of (9).

Dominance rule. To discard non-promising labels, we apply the following dominance rule.

Definition 3.1. A label $E_{lt} = (T_{lt}^{rdc}, T_{lt}^{time}, T_{lt}^{load}, T_{lt}^{start}, U_{lt})$ dominates a label $E_{l't'} = (T_{l't'}^{rdc}, T_{l't'}^{time}, T_{l't'}^{load}, T_{l't'}^{start}, U_{l't'})$ if both paths l and l' end at the same vertex and the following conditions hold:

$$T_{lt}^{rdc} \leq T_{l't'}^{rdc} + (Q_2 - T_{l't'}^{load}) \cdot \Delta_s(T_{lt}^{start}, T_{l't'}^{start}) \tag{14}$$
$$T_{lt}^{time} \leq T_{l't'}^{time} \tag{15}$$
$$T_{lt}^{load} \leq T_{l't'}^{load} \tag{16}$$
$$U_{lt} \subseteq U_{l't'}, \tag{17}$$

where

$$\Delta_s(t_1, t_2) = \min\{0, \min_{d \in D_s^{t_1} \cap D_s^{t_2}} \pi_{m(d,t_1)} - \pi_{m(d,t_2)}\} \text{ for } t_1, t_2 \in T_s. \tag{18}$$

Function $\Delta_s(T_{lt}^{start}, T_{l't'}^{start})$ specifies the minimum (negative) difference between the dual values of the synchronization constraints (3) that could be activated in a feasible extension of the labels and, as such, is used in (14) to anticipate any decrease of the reduced cost in an extension of label $E_{l't'}$ that would not occur in the same extension of label E_{lt} due to a difference in their associated starting times. Note that, in definition (18), the computation may exclude depots that supply customers visited in a feasible extension of $E_{l't'}$, i.e., depots in $D_s^{T_{l't'}^{start}} \setminus D_s^{T_{lt}^{start}}$. If this is the case, then $U_{lt} \not\subseteq U_{l't'}$ and condition (17) would prevent label E_{lt} to dominate label $E_{l't'}$.

Proposition 3.1. Conditions (14)–(17) constitute a valid dominance rule for the considered ESPPRC variant.

See Appendix A for the proof.

3.1.2 Acceleration strategies.

To speed up the resolution of the subproblems, we make use of the *ng-path relaxation* and *decremental state-space relaxation* (DSSR), two well-known ESPPRC relaxations that we briefly describe in what follows.

- The *ng*-path relaxation, introduced by Baldacci et al. (2011), relies on defining for each vertex $i \in N$, a neighborhood $\mathcal{N}_i \subseteq N^C$ that contains its κ closest customers, where κ is a predefined parameter. A path is allowed to contain a cycle $(i_1, i_2, \ldots, i_h)$ with $i_1 = i_h$ if there exists $k \in \{2, \ldots, h-1\}$ such that $i_1 \notin \mathcal{N}_{i_k}$. For that matter, the REF (13) is adjusted as follows:

$$\begin{aligned} U_{l't} = (U_{lt} \cap \mathcal{N}_j) \cup \{j\} \cup \{h \in N^C : d_h \notin D_s^{T_{l't}^{start}}\} \\ \cup \{h \in \bigcup_{d \in D_s^{T_{lt}^{start}}} N_d^C : T_{l't}^{time} + t_{jh} > \overline{w}_h \vee T_{l't}^{load} + q_h > Q_2\}. \end{aligned} \tag{19}$$

- The DSSR (Boland et al. 2006, Righini and Salani 2008) attempts to solve an ESPPRC by first dropping the elementarity constraints on the customer vertices. If at least one elementary path with a negative reduced cost is found or no paths (elementary or not) with a negative reduced cost are found, the process is stopped. Otherwise, the labeling algorithm is restarted with elementarity requirements enforced on the customers visited more than once in the optimal non-elementary path found.

3.2 Stabilization by dual inequalities

To stabilize the column generation process, we propose to apply dual inequalities (see, e.g., Ben Amor et al. 2006, Gschwind and Irnich 2016). Let $\mu \in \mathbb{R}^n$ be the vector of dual variables associated with

a feasible bounded linear program and D^* the set of its dual optimal solutions. A dual inequality $\Lambda^\top \mu \leq \lambda$, with $\Lambda \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$, is a *dual-optimal inequality* (DOI) if $D^* \subseteq \{\mu \in \mathbb{R}^n : \Lambda^\top \mu \leq \lambda\}$. Furthermore, a set of dual inequalities $A\mu \leq a$, with $A \in \mathbb{R}^{m \times n}$ and $a \in \mathbb{R}^n$, forms a set of *deep-DOIs* (DDOIs) if $D^* \cap \{\mu \in \mathbb{R}^n : A\mu \leq a\} \neq \emptyset$. In the following, we first describe the DDOIs that we use before discussing integer solution feasibility and presenting a feasibility recovery procedure.

3.2.1 Transfer inequalities.

We consider the dual inequalities, called the *transfer inequalities* (TIs), that were introduced by Mhamedi et al. (2022) for the multi-depot 2E-VRPTW. For the MC-2E-VRPTW, the TIs are:

$$\pi_m \geq \pi_{m'}, \qquad \forall (m, m') \in \mathcal{K}, \tag{20}$$

where $\mathcal{K} = \bigcup_{d \in N^D} \left\{ (m_1, m_2) \in {\mathcal{M}_d}^2 \ : \ m_1 \neq m_2,\ S(m_1) \subseteq S(m_2) \ \wedge \ t_s^{m_2} \leq t_s^{m_1},\ \forall s \in S(m_1) \right\}$ is the set of first-echelon route pairs (m_1, m_2) such that any customer-satellite assignment feasible for m_1 is also feasible for m_2. These TIs are DDOIs for the linear relaxation of model (1)–(6).

To add these TIs, we introduce a non-negative primal variable $u_{m_1 m_2}$ with a zero cost for every route pair $(m_1, m_2) \in \mathcal{K}$ and rewrite constraints (3) as follows:

$$\sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} Q_l^{mp} y_{lp} - \sum_{m' \in \mathcal{M}_{d(m)} : (m, m') \in \mathcal{K}} u_{m,m'} + \sum_{m' \in \mathcal{M}_{d(m)} : (m', m) \in \mathcal{K}} u_{m',m} \leq Q_1 x_m, \quad \forall m \in \mathcal{M}. \tag{21}$$

These new variables are called *transfer variables* because they allow to transfer demand from one first-echelon route to another as discussed below. In the rest of this paper, we denote by F and $\tilde{F}$ the formulations (1)–(6) and (1)–(2),(21),(4)–(6), respectively.

3.2.2 Solution feasibility and recovery procedure.

Considering constraints (21) instead of constraints (3) yields a relaxation, i.e., $\tilde{F}$ is a relaxation of F. As a result, an integer solution $(\tilde{x}, \tilde{y}, \tilde{u})$ for $\tilde{F}$ is not feasible for F whenever the value $\tilde{u}_{mm'}$ of a transfer variable is positive and, moreover, it might not be possible to convert this solution into a feasible one with the same cost by transferring demands between the first-echelon routes according to the positive-valued transfer variables. See Appendix B for a detailed example.

To convert an integer solution $(\tilde{x}, \tilde{y}, \tilde{u})$ with a positive-valued transfer variable into a feasible integer solution for F with the same cost, we apply the following recovery procedure that consists in solving some small-sized bin packing problems. Let $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{L}}$ be the subsets of first- and second-echelon routes used in solution $(\tilde{x}, \tilde{y}, \tilde{u})$, respectively. For each route $l \in \tilde{\mathcal{L}}$, let $\tilde{p}_l \in \mathcal{P}_l$ denote the (unique) supply pattern assigned to l, i.e., $\tilde{y}_{l\tilde{p}_l} = 1$. Also, for each customer i visited along l, let $\tilde{m}_i \in \mathcal{M}_{d_i}$ be the first-echelon route assigned to i in supply pattern $\tilde{p}_l$, and let $\tilde{\mathcal{M}}_i = \{m \in \tilde{\mathcal{M}} \cap \mathcal{M}_{d_i} : m = \tilde{m}_i \text{ or } (\tilde{m}_i, m) \in \mathcal{K}\}$ be the set of active first-echelon routes that could have been used to supply the demand of customer i. Let $\tilde{\mathcal{K}} = \{(m, m') \in \mathcal{K} : \tilde{u}_{mm'} > 0\}$ denote the subset of route pairs in $\mathcal{K}$ associated with positive-valued transfer variables set. Finally, let $\tilde{\mathcal{D}} = \{d \in N^D : |\tilde{\mathcal{K}} \cap (\mathcal{M}_d \times \mathcal{M}_d)| \geq 1\}$ be the subset of depots associated with at least one pair in $\tilde{\mathcal{K}}$.

For each depot $d \in \tilde{\mathcal{D}}$, we consider a feasibility bin packing problem FP_d that consists in assigning the demand of each customer associated with d to the first-echelon routes in $\tilde{\mathcal{M}} \cap \mathcal{M}_d$, while preserving the selected second-echelon routes in $\tilde{\mathcal{L}}$. This problem involves a binary variable λ_m^i for each customer $i \in N_d^C$ and first-echelon route in $m \in \tilde{\mathcal{M}}_i$ that takes value 1 if customer i is assigned to first-echelon route m, 0 otherwise. It is formulated as:

$$\min \qquad 0 \tag{22}$$

$$\text{s.t.} \qquad \sum_{m \in \tilde{\mathcal{M}}_i} \lambda_m^i = 1, \qquad \forall i \in N_d^C \tag{23}$$

$$\sum_{i \in N_d^C : m \in \tilde{\mathcal{M}}_i} q_i \lambda_m^i \leq Q_1, \qquad \forall m \in \tilde{\mathcal{M}} \cap \mathcal{M}_d \tag{24}$$

$$\lambda_m^i \in \{0,1\}, \qquad \forall i \in N_d^C, m \in \tilde{\mathcal{M}}_i. \tag{25}$$

Constraints (23) ensure that each customer associated with depot d is assigned to a selected first-echelon route that can supply its second-echelon route on time. Constraints (24) guarantee that the capacity of each first-echelon route is not exceeded. Model (22)–(25) can be solved using a commercial mixed-integer programming solver.

If a feasible solution can be found for each problem FP_d, $d \in \tilde{\mathcal{D}}$, then solution $(\tilde{x}, \tilde{y}, \tilde{u})$ can be converted into a same-cost feasible solution for F, preserving the same first-echelon and second-echelon routes but rearranging the supply patterns according to the FP_d solutions. Otherwise, for all depots $d \in \tilde{D}$ for which FP_d turned out to be infeasible, we fix to zero the transfer variables $u_{mm'}$, $(m, m') \in \tilde{\mathcal{K}} \cap (\mathcal{M}_d \times \mathcal{M}_d)$, before restarting column generation.

3.3 Valid inequalities

To strengthen the lower bounds, we consider three families of valid inequalities: (lifted) *visited satellite inequalities* (VSIs), *rounded capacity inequalities* (RCIs), and *subset-row inequalities* (SRIs). When looking for violated inequalities, we apply the following priority order: VSIs first, RCIs second, and SRIs last. As soon as violated cuts are found for a family, we do not search for violated cuts of the subsequent families for the same fractional solution. Valid inequalities can be added in multiple rounds within the same branch-and-bound node. Moreover, because the subproblems can become much more difficult to solve in the presence of SRIs, we only look for violated SRIs in branching nodes of depth less than or equal to a predefined parameter value, set to 10 for our tests. In what follows, we briefly present each type of inequalities, explain how they are separated and how the column generation/labeling algorithm should be modified if need be.

Visited satellite inequalities. The VSIs, initially introduced by Marques et al. (2020) for the 2E-CVRP, ensure that, whenever a customer i is serviced from a satellite s, at least one first-echelon route visiting s and originating from d_i is used. In our BPC algorithm, we consider a lifted version of these inequalities:

$$\sum_{m \in \mathcal{M}_{d_i s} : t_s^m \leq \overline{w}_i - t_{si}} x_m \geq \sum_{l \in \mathcal{L}_s} \sum_{p \in \mathcal{P}_l} a_{il} y_{lp}, \qquad \forall s \in N^S, i \in N^C, \tag{26}$$

where $\mathcal{L}_s$ denotes the subset of second-echelon routes departing from satellite s. The left-hand side of (26) disregards first-echelon routes that would lead to an arrival at customer i after its time window in the best case. The VSIs are separated by enumeration. They are robust in the sense that they do not alter the structure of the subproblems. In fact, the dual value of a VSI associated with a customer i and a satellite s is added to the modified cost of customer's i outgoing arcs in set A_s.

Rounded capacity inequalities. The RCIs (Laporte and Nobert 1983) have been extensively used for different VRP variants, including those with two routing echelons (Marques et al. 2020, Santos et al. 2015, Mhamedi et al. 2022). For a subset of customers $\mathcal{C} \subseteq N^C$ such that $|\mathcal{C}| \geq 2$, let $\xi(\mathcal{C}) = \left\lceil \frac{\sum_{i \in \mathcal{C}} q_i}{Q_2} \right\rceil$ be a lower bound on the number of second-echelon vehicles required to service the demand of customers in $\mathcal{C}$. The RCIs are defined as:

$$\sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} \sum_{(i,j) \in \delta(\mathcal{C})} b_{ij}^l y_{lp} \geq \xi(\mathcal{C}), \qquad \forall \mathcal{C} \subseteq N^C \text{ such that } |\mathcal{C}| \geq 2, \tag{27}$$

where b_{ij}^l is a binary parameter indicating whether or not second-echelon route l traverses arc (i,j) and $\delta(\mathcal{C})$ is the set of arcs leaving $\mathcal{C}$, i.e., $\delta(\mathcal{C}) = \{(i,j) \in A_2 : i \in \mathcal{C}, j \notin \mathcal{C}\}$. The RCIs are separated

using the heuristic proposed by Lysgaard et al. (2004). They are also robust and the dual value of a RCI associated with a customer subset $\mathcal{C}$ is subtracted from the modified cost of all arcs in $\delta(\mathcal{C})$.

We also add a priori one RCI per depot d defined on its first-echelon routes:

$$\sum_{m \in \mathcal{M}_d} x_m \geq \left\lceil \frac{\sum_{i \in N_d^C} q_i}{Q_1} \right\rceil, \qquad \forall d \in N^D. \tag{28}$$

Subset-row inequalities. The SRIs, introduced by Jepsen et al. (2008) for the VRPTW, are valid inequalities for the set packing polytope and, thus, for the set partitioning polytope also. Similar to several previous works considering SRIs, we restrict ourselves to customer subsets $\mathcal{C} \subset N^C$ of cardinality three because their separation can be done by straighforward enumeration. The SRIs are expressed as:

$$\sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} \left\lfloor \frac{\sum_{i \in \mathcal{C}} a_{il}}{2} \right\rfloor y_{lp} \leq 1, \qquad \forall \mathcal{C} \subset N^C \text{ such that } |\mathcal{C}| = 3. \tag{29}$$

They are non-robust as each active SRI (associated with a negative dual price) requires an additional binary resource in the label definition. Furthermore, to take into account these additional label components, the dominance rule presented in Section 3.1.1 must be modified. For details, see Jepsen et al. (2008) or Costa et al. (2019).

3.4 Branching

The branch-and-bound tree is explored using a best-first search strategy and a hierarchical branching scheme. For each branching rule, the selected variable/entity has the fractional value closest to 0.5. Furthermore, for branching decisions involving modifications at the subproblem level, we remove from the MP, all columns that do not align with the imposed decisions. Let $(\hat{x}, \hat{y}, \hat{u})$ be a fractional solution at a given node. In what follows, we describe the considered branching rules in decreasing order of priority. The rules and their order have been selected based on preliminary computational experiments.

Depot-satellite supply. The first branching level decides on whether or not a depot d delivers freight to a satellite s. The selected depot-satellite pair $(\bar{d}, \bar{s})$ is defined by its $\max_{i \in N_d^C} \sum_{l \in \mathcal{L}_s, p \in \mathcal{P}_l} a_{il} \hat{y}_{lp}$ value. On one branch, we require depot $\bar{d}$ to use $\bar{s}$ as a supply satellite by adding to the MP a constraint forcing to select at least one route $m \in \mathcal{M}_{ds}$. On the other branch, we forbid depot d from using satellite s by setting $x_m = 0$, $\forall m \in \mathcal{M}_{ds}$.

Back-and-forth total flow. The second level of decisions targets the total flow on a back-and-forth route whenever it is fractional. To impose upper and lower bounds on the total back-and-forth flow, we set some of the route variables associated with copies of the selected back-and-forth route to zero (last copies) and one (first copies), respectively.

Vehicle usage. At a third level, we branch on different entities computing a number of vehicles used, by adding constraints to the MP. These entities are grouped in two priority sub-levels. The first sub-level considers branching on the total number of vehicles at the: *i*) first-echelon ($\sum_{m \in \mathcal{M}} \hat{x}_m$), *ii*) first-echelon visiting a satellite s ($\sum_{m \in \mathcal{M}_s} \hat{x}_m$), *iii*) first-echelon departing from depot d ($\sum_{m \in \mathcal{M}_d} \hat{x}_m$), *iv*) second-echelon ($\sum_{l \in \mathcal{L}, p \in \mathcal{P}_l} \hat{y}_{lp}$). At the second sub-level, we consider branching on the total number of vehicles at the: *i*) first-echelon departing from a depot d and visiting a satellite s ($\sum_{m \in \mathcal{M}_d \cap \mathcal{M}_s} \hat{x}_m$), *ii*) second-echelon departing from a satellite s ($\sum_{l \in \mathcal{L}_s, p \in \mathcal{P}_l} \hat{y}_{lp}$).

First-echelon route usage. The fourth level of branching imposes or forbids the use of a given first-echelon route by fixing bounds on the selected variable x_m.

Customer-satellite assignment. The fifth level decides whether to forbid or impose servicing a customer i from a satellite s. The selected customer-satellite pair (i, s) is defined by its $\sum_{l\in\mathcal{L}_s}\sum_{p\in\mathcal{P}_l} a_{il}\hat{y}_{lp}$ value. To forbid and impose servicing customer i from satellite s, we remove customer node i from node set N_s and from node sets $N_{s'}$ $(s' \neq s)$, respectively.

Arc flow. The sixth level imposes an integer flow on a second-echelon arc $(i, j) \in A_2$ which is selected according to its $\sum_{l\in\mathcal{L}}\sum_{p\in\mathcal{P}_l} b^l_{ij}\hat{y}_{lp}$ value. Fixing to zero the flow on this arc is done by removing (i, j) from all arc sets A_s, $\forall s \in N^S$. Fixing it to one is achieved by removing arcs (i, k), $k \neq j$, and (h, j), $h \neq i$, from all arc sets A_s, $\forall s \in N^S$.

Customer-satellite-route assignment. Integer flows on first- and second-echelon arcs do not guarantee integrality requirements (5) on second-echelon variables. In fact, there might exist fractional y_{lp} variables associated with the same second-echelon route but with different supply patterns, supplying at least one customer i with different first-echelon routes. This rule selects a triplet $(i, s, m) \in N^C \times N^S \times \mathcal{M}$ according to its $\sum_{l\in\mathcal{L}_s}\sum_{p\in\mathcal{P}_l:(m,i)\in p} a_{il}\hat{y}_{lp}$ value. On one branch, customer i must be serviced from satellite s by first-echelon route m. On the other branch, customer i must be serviced either through a satellite $s' \neq s$ or through satellite s as long as a first-echelon route $m' \neq m$ supplies the demand of customer i. To impose these branching decisions, the labeling algorithm needs to be modified as detailed in Appendix C.

4 Computational experiments

This section reports the results obtained by the proposed BPC algorithm described in Section 3. This algorithm was implemented in C/C++ using the GENCOL library (version 4.5). The RMPs are solved using CPLEX 22.1. All tests were performed on a Linux computer equipped with an Intel Core i7-8700 processor clocked at 3.2 GHz and 66 GB of RAM. For all tests, we impose a time limit of three hours.

This section is organized as follows. Section 4.1 describes the test instances. Section 4.2 summarizes the main computational results and compares the performance of our algorithm with that of Dellaert et al. (2021). Finally, Section 4.3 presents a sensitivity analysis on some of the main components of our BPC algorithm to highlight their contribution to the overall algorithm's performance.

4.1 MC-2E-VRPTW test instances

To evaluate our BPC algorithm, we consider the instances used by Dellaert et al. (2021), which are derived from the 2E-VRPTW benchmark instances of Dellaert et al. (2019). Let us start by describing the set of these 240 2E-VRPTW instances, which is composed of 12 groups of 20 instances. Each group is characterized by a pair of numbers of depots and satellites $(|N^D|, |N^S|) \in \{(2, 3), (3, 5), (4, 6)\}$ and a number of customers $|N^C| \in \{15, 30, 50, 100\}$. Each group is equally divided into four categories: `Ca`, `Cb`, `Cc`, and `Cd`, that differ by the time window length and the demand distribution. Across all instances, the first- and second-echelon vehicle capacities are $Q_1 = 200$ and $Q_2 = 50$, whereas their fixed costs are set to $f_1 = 50$ and $f_2 = 25$. The service time at every customer and satellite is set to 10.

To create MC-2E-VRPTW instances, Dellaert et al. (2021) introduce commodities by randomly assigning each customer to a depot. They only created 100 MC-2E-VRPTW instances: 60 instances with 15 customers (that we do not consider because they are too easy to solve, often in less than one second), 20 with 30 customers, 10 with 50 customers, and 10 with 100 customers. To complement these instances, we also generated commodities randomly for the other 140 2E-VRPTW instances. For each size and category, Table 1 presents the number of instances considered in our tests and their origin.

Table 1: Number of instances per size and category

Size		Category $\|N^C\| = 30$				Category $\|N^C\| = 50$				Category $\|N^C\| = 100$			
$\|N^D\|$	$\|N^S\|$	Ca	Cb	Cc	Cd	Ca	Cb	Cc	Cd	Ca	Cb	Cc	Cd
2	3	5†	5†	5†	5†	5†	5	5	5	5†	5	5	5
3	5	5	5	5	5	5†	5	5	5	5†	5	5	5
6	4	5	5	5	5	5	5	5	5	5	5	5	5

† From Dellaert et al. (2021)

4.2 Main computational results

This section presents computational results obtained with the proposed BPC algorithm. First, we report summarized computational results and compare the computational performance of our algorithm with that of the state-of-the-art algorithm of Dellaert et al. (2021). Second, we provide various algorithmic statistics to better understand the solution process. Finally, we list some of the features of the optimal solutions found.

4.2.1 Summarized results and comparison with Dellaert et al. (2021).

Table 2 presents the average results obtained by our BPC algorithm. For each group of 20 instances defined by its size ($|N^D|, |N^S|, |N^C|$), it indicates the average computational time in seconds for the instances solved to optimality (T) and the number of instances solved to optimality within the 3-hour time limit (# Opt). Detailed results can be found in Appendix D.

The results in Table 2 show that our BPC algorithm can solve 148 of the 180 instances within the time limit. Its overall performance is obviously impacted negatively by an increase in the number of customers: the average computational times increase rapidly and we can only solve 28 of the 60 instances with 100 customers. Moreover, instance groups with three depots and five satellites require the largest average computational times. The complexity of these instances largely stems from a larger number of first-echelon routes, which complicates the resolution of both the RMPs and the CG subproblems.

Table 2: Summary of the computational results

		$\|N^C\| = 30$		$\|N^C\| = 50$		$\|N^C\| = 100$	
$\|N^D\|$	$\|N^S\|$	T (s)	# Opt	T (s)	# Opt	T (s)	# Opt
2	3	1.1	20	38.2	20	3247.0	13
3	5	46.1	20	331.1	20	7914.8	3
6	4	18.1	20	126.3	20	4703.7	12

In Table 3, we compare our BPC algorithm with the best algorithm of Dellaert et al. (2021) (that based on their MC2E-2P model) on the 30-, 50-, and 100-customer instances they considered. For each instance group and each algorithm, we report the average computational time in seconds (T) computed over the instances solved to optimality, and the number of instances solved to optimality within the 3-hour time limit (# Opt). For the BPC algorithm, the number of new optimal solutions found (# New) is also provided. To make a fair comparison, we need to account for the bias induced by using a faster computer for our experiments than that used by Dellaert et al. (2021) for theirs. For that matter, the last column specifies the biased-minimal-speed-up ($BMSU$) factor, computed as T^D/T^{BPC}, where T^{BPC} and T^D are the average time of our BPC algorithm and the minimal average time of their algorithm (i.e., setting to 10800 seconds the time for each instance that is solved by our algorithm but not theirs), respectively.

Table 3 shows that we can solve all the 30 instances solved by Dellaert et al. (2021) within the time limit. Furthermore, we can also find 7 new solutions, leaving only 3 instances open out of the 40 used

for this comparison. Despite using a faster computer for our tests, the average BMSU factors, being more prominent than the speed-up factor provided by using our computer, clearly highlight the fact that our algorithm outperforms the current state-of-the-art algorithm for the MC-2E-VRPTW. The speed-up is, however, less important for the instances with 3 depots and 5 satellites, due to a larger number of first-echelon routes.

Table 3: Comparison with Dellaert et al. (2021)

			Dellaert et al. (2021) †		Our BPC algorithm ‡			BMSU
$\|N^D\|$	$\|N^S\|$	$\|N^C\|$	T (s)	# Opt	T (s)	# Opt	# New	factor
2	3	30	79.6	20/20	1.1	20/20	0	75.1
2	3	50	93.8	4/5	3.5	5/5	1	642.2
3	5	50	2794.2	5/5	424.5	5/5	0	6.6
2	3	100	–	0/5	1536.0	5/5	5	7.0
3	5	100	2919.0	1/5	9867.8	2/5	1	1.4
		Total		30/40		37/40		

† Tests ran on a computer equipped with an Intel Core i7-4770 processor
‡ Tests ran on a computer equipped with an Intel Core i7-8700 processor

4.2.2 Statistics on algorithm execution.

We now present statistics retrieved during the execution of our BPC algorithm. Table 4 reports collected statistics for each group of instances identified by their size in the first three columns. The average statistics are computed over instances solved to optimality. For each group, we report the average numbers of branch-and-bound nodes (# Nodes), enumerated first-echelon routes ($|\mathcal{M}|$), and TIs considered in the model (# TIs). We also specify, for each family of valid inequalities, the number of cuts generated during the solution process (# RCIs, # SRIs, # VSIs).

Table 4 asserts previous observations relating the difficulty of solving an instance with the numbers of customers and satellites it involves. For each customer count, the 3-depot, 5-satellite instances are consistently the most challenging to solve compared to their counterparts with depot-satellite pairs $(|N^D|, |N^S|) \in \{(2,3),(6,4)\}$ and the same number of customers $|N^C|$. The increased computational difficulty stems from the significant number of first-echelon routes, requiring many more branching nodes, SRIs, and RCIs. Table 4 also shows that the number of TIs is positively correlated with the number of first-echelon routes: an increase of $|\mathcal{M}|$ yields more route pairs in set $\mathcal{K}$. Finally, the VSIs are extensively used across the board. Their usage arises primarly from second-echelon routes requiring more than two first-echelon vehicles to supply their load.

Table 4: Summarized statistics on the algorithm execution

$\|N^D\|$	$\|N^S\|$	$\|N^C\|$	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs
2	3	30	15.0	29.9	26.7	32.7	6.9	84.2
3	5	30	65.4	486.7	461.9	65.1	36.9	141.2
6	4	30	51.7	180.0	147.4	57.1	28.7	112.9
2	3	50	139.4	35.2	35.5	85.9	38.3	142.4
3	5	50	366.3	685.7	688.6	228.2	111.4	235.9
6	4	50	125.2	227.3	198.8	164.7	71.6	186.2
2	3	100	2100.8	47.7	59.1	206.2	64.4	278.9
3	5	100	3254.0	965.7	1042.7	476.0	222.7	447.3
6	4	100	1824.8	318.9	321.3	644.1	447.8	369.9

4.2.3 Optimal solution statistics.

We now summarize some characteristics of the optimal solutions found. For each instance group, Table 5 presents the average numbers of satellites ($|N^S|^*$), first-echelon routes ($|\mathcal{M}|^*$), and second-echelon routes ($|\mathcal{L}|^*$) used, as well as the percentage of the first-echelon routes that are back-and-forth ones ($|\mathcal{M}^1|^*$). These results show that increasing the number of customers drives a higher satellite usage rate. In fact, this average rate increases from 70 to 96% as $|N^C|$ increases from 30 to 100. Obviously, the number of first- and second-echelon routes used increases with the customer count. Also, the usage of first-echelon vehicles tends to grow with the number of depots and satellites, especially because every depot must supply the demand of a subset of customers. Finally, the average percentages of first-echelon back-and-forth routes are quite high (between 50 and 83% per group) but much less than those reported for the 2E-VRPTW by Mhamedi et al. (2022) which vary between 72 and 100% per group. Hence, more first-echelon routes visiting multiple satellites are required for the multi-commodity case because the customers in the second-echelon routes must often be supplied by multiple first-echelon routes coming from different depots.

Table 5: Summarized optimal solution statistics

$\|N^D\|$	$\|N^S\|$	$\|N^C\|$	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
2	3	30	2.0	3.4	79.2	9.5
3	5	30	3.3	3.3	50.0	9.6
6	4	30	3.0	6.0	82.5	9.7
2	3	50	3.0	4.4	75.5	15.4
3	5	50	4.3	5.2	52.3	15.4
6	4	50	3.5	6.3	72.9	15.7
2	3	100	3.0	8.3	66.7	30.5
3	5	100	4.3	9.3	75.6	30.0
6	4	100	4.0	10.6	64.1	30.1

4.3 Sensitivity analysis

In this section, we conduct a sensitivity analysis to evaluate how some specific components of the BPC algorithm impact its overall computational performance. To this end, we selected a subset of 36 instances with $|N^C| \in \{50, 100\}$ that were challenging but solved to optimality in our previous tests. For the 50-customer instances, we considered all 20 instances with $(|N^D|, |N^S|) = (3, 5)$. For the 100-customer instances, we selected 13 instances with $(|N^D|, |N^S|) = (2, 3)$ and 3 instances with $(|N^D|, |N^S|) = (3, 5)$.

For this analysis, we focus on the following algorithmic components: the depot-satellite supply branching rule (see Section 3.4), the symmetry-breaking constraints (6), the VSIs (26), the TIs (see Section 3.2.1), and the recovery procedure (see Section 3.2.2). Other components such as the RCIs, the SRIs, and the other branching rules have been left out of this analysis because their impact has been well evaluated in previous works (RCIs and SRIs) or is relatively minor but essential to obtain the best results on all instances (the other branching rules). To assess the impact of each of the first four components, we have conducted separate computational tests, each time removing only the selected component from the algorithm. For the recovery procedure, we ran additional tests considering a modified procedure that fixes to zero all positive-valued transfer variables (if any) without solving the bin packing feasibility problems before restarting column generation.

Table 6 presents the computed average results. For each component and customer count, we report the number of instances solved to optimality within the time limit (# Opt), the average computational time in seconds (T), and the minimum computational time variation in percentage (ΔT) between the complete BPC algorithm and the BPC algorithm with the selected component removed or modified. We compute ΔT as $(T - \bar{T})/\bar{T}$, where $\bar{T}$ is the average computational time of the full BPC algorithm

and T that of its corresponding modified counterpart or three hours if it did not solve the instance within the time limit.

Table 6: Impact of omitting algorithmic components

	Instances with $\|N^C\| = 50$			Instances with $\|N^C\| = 100$		
Omitted component	# Opt	T (s)	ΔT (%)	# Opt	T (s)	ΔT (%)
Depot-satellite supply branching	17	4012.1	1111.8	14	3812.8	-7.5
Symmetry-breaking constraints	17	2573.3	677.2	0	–	162.0
Visited satellite inequalities	14	3696.4	1016.5	12	4381.7	6.3
Transfer inequalities	14	5833.6	1662.0	10	5859.9	42.2
Recovery procedure	20	444.4	34.2	14	4247.3	3.0

First, when removing the depot-satellite supply branching, 5 of the 36 instances cannot be solved anymore within the time limit, showing the effectiveness of using this branching rule. Furthermore, for the 50-customer instances, the average computational time increases by a minimum of 1111.8%, showing the high effectiveness of this rule when the numbers of depots and satellites (3 and 5 for all these instances) yield a large number of first-echelon routes. For the instances with 100 customers, the average computational time slightly decreases, possibly because 13 of these 16 instances do not involve a large number of first-echelon routes. On the other hand, for the other 3 instances, removing the depot-satellite supply branching increases the average time by at least 20.1%.

Second, when omitting the symmetry-breaking constraints, we can only solve 17 50-customer instances with a minimum average time increase of 677.2% and none of the 100-customer instances. Because the number of back-and-forth route copies increases with the number of customers, considering symmetry-breaking constraints is crucial when solving instances with $|N^C| = 100$.

Third, the VSIs impact positively the solution process, especially for instances with 3 depots and 5 satellites where these cuts greatly help improving the lower bounds. When discarded, we can only solve 14 of the 20 50-customer instances while increasing the computational time by a factor of at least 10. However, the impact of the VSIs is not as strong for instances with $(|N^D|, |N^S|) = (2, 3)$: the modified algorithm solves 11 of the 13 100-customer instances and yields a small minimum time increase of 5%.

Fourth, the absence of TIs significantly impacts the total times across the board. The total number of instances solved diminishes by 12 and the average times increase substantially, especially for the instances with a large number of first-echelon routes.

Finally, considering the full recovery procedure also has a positive impact on the overall solution process, but less than the above algorithmic components. With the simplified procedure, two instances with 100 customers cannot be solved within the time limit and the average computational time increases moderately (by 34.2%) for the instances with 50 customers. This time increase is due to the fact that there is a large number of first-echelon routes for the 50-customer instances (all with 3 depots and 5 satellites) and, thus, a large number of TIs. Consequently, the recovery procedure is invoked more often during the solution process as more integer solutions with active transfer variables are encountered.

Overall, this sensitivity analysis shows that all these algorithmic components play an important role to achieve the good results that were presented in Section 4.2.1. Some of them (depot-satellite supply branching, VSIs, recovery procedure) are, however, less useful when the number of first-echelon routes is small.

5 Conclusions

In this paper, we have developed a BPC algorithm for the MC-2E-VRPTW, which incorporates an ad hoc labeling algorithm for generating second-echelon routes and their associated supply patterns. We

use dual inequalities to speed up CG convergence and propose a recovery procedure to enforce integer solution feasibility. Our BPC algorithm also includes three families of valid inequalities to improve the lower bounds and a branching scheme featuring a problem-specific branching rule.

We extended the existing instances of Dellaert et al. (2021) by creating 140 new instances and showed through extensive computational experiments on 180 benchmark instances that our algorithm outperforms the best existing exact algorithm for the problem. It solves to optimality 148 instances within the 3-hour time limit, including 7 previously unsolved instances from Dellaert et al. (2021). We also showed the crucial role of some components of our BPC algorithm to achieve good computational performance. Three of these components proved essential when solving instances with a large number of first-echelon routes.

Nevertheless, solving instances with a large number of depots and satellites remains a challenge. A potential future research avenue could be the development of cuts to further strengthen the lower bounds. Another interesting one would be to extend the problem definition by considering load-dependent transfer times at the satellites.

Appendix A Proof of Proposition 3.1

In this proof, we use the symbol $\oplus$ to denote a concatenation of two paths. More precisely, let l be a path and ω a possible path extension of l. Then, $l \oplus \omega$ represents the path resulting from the concatenation of l and ω.

Proof. Let ω be a feasible extension of l' and $\mathcal{C}_\omega$ the subset of customers visited along this extension. To prove the proposition's statement, it suffices to show that i) any feasible extension ω of path l' such that $d_i \in D_s^{T_{lt}^{start}}$ for all $i \in \mathcal{C}_\omega$ (otherwise, condition (17) is violated and label E_{lt} does not dominate label $E_{l't'}$) is also feasible for path l and ii) the reduced cost of path $l \oplus \omega$ is less than or equal to that of path $l' \oplus \omega$. Given that time and load REFs (10)–(11) are non-decreasing, establishing the feasibility of path $l \oplus \omega$ with respect to the time window and capacity constraints is straightforward. Moreover, elementarity of path $l' \oplus \omega$ implies that no customers in $\mathcal{C}_\omega$ is in U_{lt} and thus $l \oplus \omega$ is elementary as well.

Let us now show that $T^{rdc}_{l\oplus\omega,t} \leq T^{rdc}_{l'\oplus\omega,t'}$. For convenience, let $c(\omega)$ denote the total routing cost of ω and $\sigma(\omega) = \sum_{j\in\mathcal{C}_\omega} \sigma_j$. With this notation, we get:

$$T^{rdc}_{l\oplus\omega,t} = T^{rdc}_{lt} + c(\omega) - \sigma(\omega) - \sum_{j\in\mathcal{C}_\omega} q_j \cdot \pi_{m(d_j, T^{start}_{lt})}$$
$$T^{rdc}_{l'\oplus\omega,t'} = T^{rdc}_{l't'} + c(\omega) - \sigma(\omega) - \sum_{j\in\mathcal{C}_\omega} q_j \cdot \pi_{m(d_j, T^{start}_{l't'})}$$

and the difference between these two reduced costs is:

$$T^{rdc}_{l\oplus\omega,t} - T^{rdc}_{l'\oplus\omega,t'} = T^{rdc}_{lt} - T^{rdc}_{l't'} - \sum_{j\in\mathcal{C}_\omega} q_j \cdot (\pi_{m(d_j, T^{start}_{lt})} - \pi_{m(d_j, T^{start}_{l't'})}). \tag{30}$$

Let us consider two cases. First, if $T^{start}_{lt} \geq T^{start}_{l't'}$, then $\mathcal{M}_s^{T^{start}_{lt}} \supseteq \mathcal{M}_s^{T^{start}_{l't'}}$ and

$$\pi_{m(d_j, T^{start}_{lt})} = \max_{m\in\mathcal{M}_{d_j}\cap\mathcal{M}_s^{T^{start}_{lt}}} \{\pi_m\} \geq \max_{m\in\mathcal{M}_{d_j}\cap\mathcal{M}_s^{T^{start}_{l't'}}} \{\pi_m\} = \pi_{m(d_j, T^{start}_{l't'})} \quad \forall j \in \mathcal{C}_\omega.$$

Establishing that $T^{rdc}_{l\oplus\omega,t} \leq T^{rdc}_{l'\oplus\omega,t'}$ in this case is deduced from (30) and the fact that condition (14) implies $T^{rdc}_{lt} \leq T^{rdc}_{l't'}$.

Second, if $T_{lt}^{start} < T_{l't'}^{start}$, the following ensues:

$$D_s^{T_{lt}^{start}} \subseteq D_s^{T_{l't'}^{start}} \implies \Delta(T_{lt}^{start}, T_{l't'}^{start}) = \min_{d \in D_s^{T_{lt}^{start}}} \pi_{m(d,T_{lt}^{start})} - \pi_{m(d,T_{l't'}^{start})} \leq 0.$$

and we can deduce that

$$\begin{aligned}\sum_{j \in \mathcal{C}_\omega} q_j \cdot (\pi_{m(d_j,T_{lt}^{start})} - \pi_{m(d_j,T_{l't'}^{start})}) &\geq \sum_{j \in \mathcal{C}_\omega} q_j \cdot \min_{d \in D_s^{T_{lt}^{start}}} (\pi_{m(d,T_{lt}^{start})} - \pi_{m(d,T_{l't'}^{start})}) \\ &\geq (Q_2 - T_{l't'}^{load}) \cdot \Delta(T_{lt}^{start}, T_{l't'}^{start})\end{aligned}$$

because $\sum_{j \in \mathcal{C}_\omega} q_j \leq Q_2 - T_{l't'}^{load}$ for a feasible extension ω. Consequently, relation (30) and condition (14) yield $T_{l\oplus\omega,t}^{rdc} - T_{l'\oplus\omega,t'}^{rdc} \leq T_{lt}^{rdc} - T_{l't'}^{rdc} - (Q_2 - T_{l't'}^{load}) \cdot \Delta(T_{lt}^{start}, T_{l't'}^{start}) \leq 0$, i.e., $T_{l\oplus\omega,t}^{rdc} \leq T_{l'\oplus\omega,t'}^{rdc}$. □

Appendix B Detailed example of solution feasibility with transfer inequalities

Figure 2 illustrates an MC-2E-VRPTW example with $|N^D| = 2$, $|N^S| = 3$, $|N^C| = 20$, $Q_1 = 100$, and $Q_2 = 40$, where it is not possible to convert a solution into a same-cost feasible one by transferring demands between first-echelon routes. The green and grey customers are assigned to depots d_1 and d_2, respectively, i.e., $N_{d_2}^C = \{2, 4, 6, 7, 9, 13, 19\}$ and $N_{d_1}^C = N^C \setminus N_{d_2}^C$. The solution uses three first-echelon routes m_2, m_3 and m_4 ($(\tilde{x}_2, \tilde{x}_3, \tilde{x}_4) = (1, 1, 1)$) and eight second-echelon routes l_i, $i \in \{1, \ldots, 8\}$. Furthermore, active transfer variables associated with route pairs (m_1, m_2) and (m_1, m_3) allow for an implicit use of first-echelon route m_1, i.e., $\tilde{u}_{m_1m_2} = \tilde{u}_{m_1m_3} = 20$ and $\tilde{x}_1 = 0$. The supply patterns $\tilde{p}_{l_i} \in \mathcal{P}_{l_i}$ retained for the second-echelon routes l_i, $i \in \{1, \cdots, 8\}$, along with the total loads associated with their supplying first-echelon routes are:

$$\begin{aligned}
\tilde{p}_{l_1} &= \{(m_1, 1), (m_4, 2)\}, & (Q_{l_1\tilde{p}_{l_1}}^{m_1}, Q_{l_1\tilde{p}_{l_1}}^{m_4}) &= (30, 10) \\
\tilde{p}_{l_2} &= \{(m_1, 3), (m_4, 4)\} & (Q_{l_2\tilde{p}_{l_2}}^{m_1}, Q_{l_2\tilde{p}_{l_2}}^{m_4}) &= (10, 20) \\
\tilde{p}_{l_3} &= \{(m_2, 5), (m_4, 6), (m_4, 7)\} & (Q_{l_3\tilde{p}_{l_3}}^{m_2}, Q_{l_3\tilde{p}_{l_3}}^{m_4}) &= (20, 20) \\
\tilde{p}_{l_4} &= \{(m_2, 8), (m_4, 9)\} & (Q_{l_4\tilde{p}_{l_4}}^{m_2}, Q_{l_4\tilde{p}_{l_4}}^{m_4}) &= (20, 20) \\
\tilde{p}_{l_5} &= \{(m_2, 10), (m_2, 11), (m_2, 12)\} & Q_{l_5\tilde{p}_{l_5}}^{m_2} &= 40 \\
\tilde{p}_{l_6} &= \{(m_4, 13), (m_3, 14), (m_3, 15)\} & (Q_{l_6\tilde{p}_{l_6}}^{m_3}, Q_{l_6\tilde{p}_{l_6}}^{m_4}) &= (20, 10) \\
\tilde{p}_{l_7} &= \{(m_3, 16), (m_3, 17)\} & Q_{l_7\tilde{p}_{l_7}}^{m_3} &= 30 \\
\tilde{p}_{l_8} &= \{(m_3, 18), (m_4, 19), (m_3, 20)\} & (Q_{l_8\tilde{p}_{l_8}}^{m_3}, Q_{l_8\tilde{p}_{l_8}}^{m_4}) &= (30, 10).
\end{aligned}$$

According to these supply patterns, first-echelon routes m_1, m_2, m_3, and m_4 have total loads of 40, 80, 80, and 90, respectively. The transfer variable values $\tilde{u}_{m_1m_2} = \tilde{u}_{m_1m_3}$ indicate that 20 units of load are transferred from m_1 to m_2 and from m_1 to m_3, reducing the load of m_1 to 0 and increasing the loads of m_2 and m_3 to $Q_1 = 100$. Given that the demand of customer 1, initially assigned to route m_1, is $q_1 = 30$, it cannot be transferred in its entirety to either route m_2 or m_3. Consequently, solution $(\tilde{x}, \tilde{y}, \tilde{u})$ cannot be transformed into a same-cost solution that is feasible for formulation F.

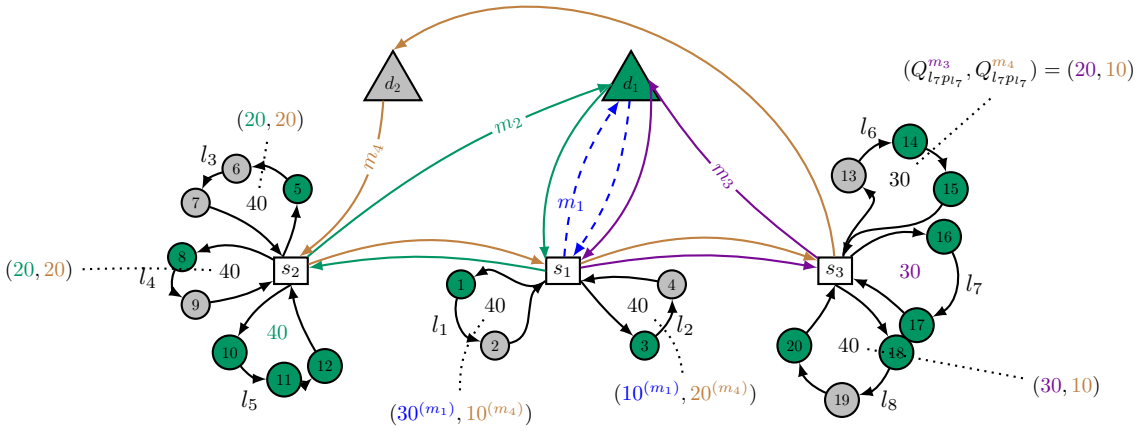

Figure 2: Example of an integer infeasible solution

Appendix C Adjustments to the labeling algorithm

To impose the decisions related to the *customer-satellite-route assignment* branching rule, the labeling algorithm described in Section 3.1.1 is modified as follows:

1. To impose that customer i is supplied from satellite s by first-echelon route m, customer i is removed from vertex sets $N_{s'}$ such that $s' \neq s$:
   (a) In the initial label E_{0t}, customer i is included in sets U_{0t} associated with time $t \in T_s$ such that $t < t_s^m$. Note that the unreachability status of i remains unchanged when considering ng-paths.
   (b) When extending a label E_{lt}, associated with a partial path l ending at vertex $h \in N_s \setminus \{n_s^{sk}\}$ and a time $t \in T_s$, along an arc $(h, j) \in A_s$ such that $T_{lt}^{start} \geq t_s^m$, the REF (9), computing this reduced cost of label $E_{l't}$ resulting from the extension, is adjusted as:

$$T_{l't}^{rdc} = T_{lt}^{rdc} + \bar{c}_{hj} - \begin{cases} q_j \cdot \pi_{m(d_j, T_{lt}^{start})}, & \text{if } j \notin \{n_s^{sk}, i\}, \\ q_j \pi_m, & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

2. To impose that customer i is supplied by a first-echelon route $m' \neq m$ whenever i is serviced through satellite s:
   (a) No changes need to be made when extending a label E_{lt}, associated with a partial path l ending at vertex $h \in N_s \setminus \{n_s^{sk}\}$ and a time $t \in T_s$ such that $T_{lt}^{start} < t_s^m$.
   (b) When extending a label E_{lt}, associated with a partial path l ending at vertex $h \in N_s \setminus \{n_s^{sk}\}$ and a time $t \in T_s$, along an arc $(h, j) \in A_s$ such that $T_{lt}^{start} = t \geq t_s^m$
   the REF (9) for the reduced cost is adjusted as:

$$T_{l't}^{rdc} = T_{lt}^{rdc} + \bar{c}_{hj} - \begin{cases} q_j \cdot \pi_{m(d_j, T_{lt}^{start})}, & \text{if } j \notin \{n_s^{sk}, i\} \text{ or } (j = i \text{ and } m(d_j, T_{lt}^{start}) \neq m), \\ q_j \cdot \pi_{m_i}, & \text{if } j = i \text{ and } m(d_j, T_{lt}^{start}) = m, \\ 0 & \text{otherwise} \end{cases}, \tag{32}$$

   where $m_i \in \arg\max \left\{\pi_m : m \in \hat{\mathcal{M}}_s^{T_{lt}^{start}} \cap (\mathcal{M}_{d_i} \setminus \{m\})\right\}$, and $\hat{\mathcal{M}}_s^t$ refers to routes in $\mathcal{M}_s^t$ that are not fixed to zero in the current branching node.

Appendix D Detailed computational results

This section presents the detailed computational results obtained by our BPC algorithm. Results for instances with $|N^C| = 30, 50$ and 100 are reported in Tables 7–9, respectively. For each row in each table, the first column indicates the instance name in the format `MC-Cxs-`$|N^D|, |N^S|, |N^C|$, where `x` $\in \{a, b, c, d\}$ is the instance category and `s` $\in \{1, 2, 3, 4, 5\}$ the instance number in this category. The next two columns specify the optimal value z^* found by our algorithm, and the computational time (T) in seconds as reported by Dellaert et al. (2021), respectively. Then for our BPC algorithm, we report the computational time (T) in seconds, the integrality gap (Gap) in percentage computed as $100(z^* - \underline{z})/\underline{z}$ where $\underline{z}$ is the root node lower bound before adding cuts, the number of branch-and-bound nodes explored (# Nodes), and the number of enumerated first-echelon routes ($|\mathcal{M}|$). Columns # TIs, # RCIs, # SRIs and # VSIs give, respectively, the number of TIs (20), RCIs (27), SRIs (29) and lifted VSIs (26) considered during the solution process. The last four columns describes some characteristics of the computed optimal solution, namely, the number of satellites used ($|N^S|^*$), the number of first-echelon routes used ($|\mathcal{M}|^*$), the percentage of these routes that are back-and-forth ones ($|\mathcal{M}^1|^*$), and the number of second-echelon routes used ($|\mathcal{L}|^*$).

Table 7: Detailed computational results for instances with $|N^C| = 30$

		Dellaert et al. (2021) †	Our BPC algorithm ‡											
Instance	z^*	T(s)	T (s)	Gap (%)	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
MC-Ca1-2,3,30	1257.82	76.5	0.6	0.77	16	36	36	4	60	80	2	4	100.0	10
MC-Ca2-2,3,30	1189.67	4.6	0.7	1.21	15	29	26	10	92	77	3	4	100.0	10
MC-Ca3-2,3,30	1207.82	0.8	0.2	0.44	1	29	26	0	21	77	1	4	50.0	10
MC-Ca4-2,3,30	1101.32	2.3	0.3	0.65	3	32	29	4	43	78	1	3	66.7	9
MC-Ca5-2,3,30	1210.97	2.0	0.4	1.01	6	31	29	8	31	82	2	4	75.0	9
MC-Cb1-2,3,30	1259.51	59.9	4.1	3.79	123	27	22	31	90	79	2	3	100.0	10
MC-Cb2-2,3,30	1078.84	1.2	0.5	1.36	8	33	30	19	7	79	3	3	100.0	9
MC-Cb3-2,3,30	1136.04	2.4	2.2	0.82	43	26	22	6	46	77	1	3	66.7	10
MC-Cb4-2,3,30	1172.14	1.8	0.3	1.18	10	36	36	8	8	81	2	4	75.0	10
MC-Cb5-2,3,30	1030.38	2.9	0.7	1.94	13	30	26	23	24	80	2	3	66.7	9
MC-Cc1-2,3,30	989.81	15.9	1.9	1.39	5	32	29	0	22	90	2	3	66.7	8
MC-Cc2-2,3,30	1168.48	1108.1	2.4	1.24	24	36	36	0	23	90	3	4	75.0	10
MC-Cc3-2,3,30	1038.19	211.7	2.8	0.38	7	24	17	6	92	90	2	3	100.0	10
MC-Cc4-2,3,30	1103.19	68.3	1.5	0.48	5	29	26	0	28	90	2	4	75.0	10
MC-Cc5-2,3,30	1052.6	1.7	1.1	0.22	2	28	25	0	12	90	2	3	100.0	9
MC-Cd1-2,3,30	1036.89	1.0	0.3	0.04	3	32	29	10	0	90	2	3	100.0	8
MC-Cd2-2,3,30	1090.12	11.3	0.1	0.03	2	30	27	0	13	87	2	3	66.7	10
MC-Cd3-2,3,30	1094.12	2.1	0.4	0.98	6	22	16	5	10	88	2	3	100.0	10
MC-Cd4-2,3,30	1076.57	15.1	0.3	1.47	3	28	23	2	11	90	2	3	33.3	10
MC-Cd5-2,3,30	1093.93	2.0	0.4	1.06	4	27	23	1	20	88	2	3	66.7	9
MC-Ca1-3,5,30	1247.26	-	29.3	4.26	83	644	625	42	36	137	3	3	33.3	9
MC-Ca2-3,5,30	1220.89	-	27.9	5.72	101	470	449	100	87	125	4	3	66.7	10
MC-Ca3-3,5,30	1231.52	-	12.8	1.66	17	629	599	14	95	139	3	3	0.0	9
MC-Ca4-3,5,30	1151.18	-	5.7	4.77	17	490	466	13	14	134	3	3	100.0	9
MC-Ca5-3,5,30	1146.37	-	5.7	0.73	6	444	426	10	9	130	3	4	100.0	9
MC-Cb1-3,5,30	1212.19	-	24.1	4.73	69	465	430	36	35	141	3	3	66.7	10
MC-Cb2-3,5,30	1216.03	-	16.2	3.03	33	598	572	19	37	135	4	3	33.3	10
MC-Cb3-3,5,30	1264.63	-	8.2	1.02	39	495	469	5	17	129	3	4	50.0	10
MC-Cb4-3,5,30	1173.25	-	5.6	2.30	19	500	464	9	22	120	4	3	66.7	10
MC-Cb5-3,5,30	1166.95	-	19.6	4.07	49	480	447	37	43	143	4	3	33.3	11
MC-Cc1-3,5,30	1304.17	-	73.2	6.07	112	395	382	59	198	150	4	4	50.0	11
MC-Cc2-3,5,30	1130	-	175.9	6.75	138	497	465	102	140	150	3	3	33.3	9
MC-Cc3-3,5,30	1198.81	-	240.3	6.13	258	393	366	39	76	150	4	3	0.0	10
MC-Cc4-3,5,30	1080.66	-	65.4	3.35	31	617	592	35	23	150	3	3	66.7	9
MC-Cc5-3,5,30	1076.84	-	32.6	1.43	15	423	412	10	36	150	3	4	75.0	8
MC-Cd1-3,5,30	1250.54	-	25.4	6.52	75	421	389	39	102	148	4	3	0.0	11
MC-Cd2-3,5,30	1275.63	-	27.9	5.00	88	400	380	44	72	148	3	4	50.0	9
MC-Cd3-3,5,30	1151.79	-	26.4	4.75	63	436	416	32	76	148	2	3	33.3	10
MC-Cd4-3,5,30	1188.72	-	5.4	1.45	9	414	394	9	30	148	3	4	75.0	9
MC-Cd5-3,5,30	1115.27	-	93.4	6.02	86	522	495	83	154	148	3	3	66.7	9

Continued on next page

Table 7 – *Continued from previous page*

		Dellaert et al. (2021) †	Our BPC algorithm ‡										
Instance	z^*	T (s)	T (s)	Gap (%)	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
MC-Ca1-6,4,30	1518.43	-	10.8	3.67	69	195	160	45	94	108	4	6	83.3	11
MC-Ca2-6,4,30	1521.19	-	1.1	1.36	7	193	162	10	11	100	4	6	66.7	10
MC-Ca3-6,4,30	1462.26	-	4.3	1.30	33	173	138	17	4	106	3	6	83.3	10
MC-Ca4-6,4,30	1477.12	-	8.8	2.03	55	184	157	26	36	111	2	6	100.0	10
MC-Ca5-6,4,30	1526.62	-	1.7	1.37	11	175	139	10	4	107	4	6	66.7	10
MC-Cb1-6,4,30	1601.61	-	12.2	1.96	81	239	213	20	71	112	2	6	83.3	10
MC-Cb2-6,4,30	1469.43	-	3.8	2.19	28	189	162	21	29	103	4	6	66.7	9
MC-Cb3-6,4,30	1604.21	-	9.7	3.77	97	201	170	27	72	110	3	6	83.3	10
MC-Cb4-6,4,30	1440.76	-	8.8	2.54	75	151	113	20	21	109	2	6	100.0	10
MC-Cb5-6,4,30	1533.93	-	2.1	2.50	21	178	142	6	11	103	3	6	83.3	11
MC-Cc1-6,4,30	1405.72	-	45.1	4.24	50	196	162	63	140	120	2	6	83.3	9
MC-Cc2-6,4,30	1471.62	-	35.4	4.61	58	188	154	48	123	120	4	6	66.7	10
MC-Cc3-6,4,30	1435.69	-	18.2	3.44	28	182	148	24	15	120	4	6	83.3	9
MC-Cc4-6,4,30	1416.02	-	56.2	4.10	87	182	145	34	92	120	3	6	50.0	9
MC-Cc5-6,4,30	1430.97	-	102.0	3.09	128	181	152	101	130	120	3	6	83.3	9
MC-Cd1-6,4,30	1425.04	-	9.7	2.74	45	167	134	31	32	120	2	6	100.0	9
MC-Cd2-6,4,30	1438.07	-	7.8	2.24	29	141	108	10	76	117	2	6	100.0	9
MC-Cd3-6,4,30	1475.18	-	8.0	1.82	33	166	133	19	13	118	3	6	100.0	9
MC-Cd4-6,4,30	1472.13	-	9.1	1.99	35	158	123	25	80	116	3	6	83.3	9
MC-Cd5-6,4,30	1507.26	-	6.5	1.79	63	161	132	16	87	117	3	6	83.3	11

† Tests ran on a computer equipped with an Intel Core i7-4770 pocessor.

‡ Tests ran on a computer equipped with an Intel Core i7-8700 processor.

Table 8: Detailed computational results for instances with $|N^C| = 50$

		Dellaert et al. (2021) †	Our BPC algorithm ‡											
Instance	z^*	T (s)	T (s)	Gap (%)	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
MC-Ca1-2,3,50	1808.2	43.0	1.0	1.16	7	39	42	8	46	132	3	5	60.0	16
MC-Ca2-2,3,50	1743.18	-	5.7	1.95	51	34	33	23	216	133	2	4	75.0	16
MC-Ca3-2,3,50	1766.5	186.0	0.9	0.33	11	37	38	2	27	136	3	5	100.0	16
MC-Ca4-2,3,50	1607.81	30.0	1.1	0.58	8	34	33	7	56	138	3	4	100.0	16
MC-Ca5-2,3,50	1617.26	116.0	8.7	0.63	85	36	36	24	64	132	3	4	75.0	15
MC-Cb1-2,3,50	1722.39	-	128.1	1.61	567	39	42	75	132	143	3	5	80.0	15
MC-Cb2-2,3,50	1638.38	-	9.9	1.46	51	36	36	32	22	134	3	4	75.0	14
MC-Cb3-2,3,50	1694.12	-	208.0	2.68	1165	34	33	210	311	142	3	4	100.0	16
MC-Cb4-2,3,50	1648.86	-	24.1	1.45	143	36	36	29	44	129	3	4	100.0	15
MC-Cb5-2,3,50	1694.42	-	30.4	1.15	171	39	42	77	52	143	3	5	40.0	15
MC-Cc1-2,3,50	1547.9	-	12.7	0.67	11	34	33	37	34	150	3	4	50.0	15
MC-Cc2-2,3,50	1573.41	-	5.6	0.24	4	32	30	0	29	150	3	4	75.0	15
MC-Cc3-2,3,50	1529.43	-	21.5	0.81	27	36	36	21	81	150	3	4	75.0	15
MC-Cc4-2,3,50	1566.67	-	245.1	1.47	241	29	26	58	102	150	3	4	75.0	15
MC-Cc5-2,3,50	1579.08	-	19.8	0.99	19	32	30	21	52	150	3	4	75.0	15
MC-Cd1-2,3,50	1602.73	-	7.7	0.91	47	36	36	32	98	146	3	4	75.0	15
MC-Cd2-2,3,50	1787.01	-	10.6	0.92	65	37	40	34	109	148	3	5	60.0	16
MC-Cd3-2,3,50	1724.4	-	4.3	0.72	17	36	37	17	74	147	3	5	40.0	16
MC-Cd4-2,3,50	1708.93	-	15.6	1.52	73	35	37	37	107	150	3	5	80.0	15
MC-Cd5-2,3,50	1780.23	-	3.1	1.31	25	33	33	21	62	145	3	5	100.0	16
MC-Ca1-3,5,50	1814.7	4386.0	773.8	2.32	1479	689	706	90	314	229	4	6	66.7	16
MC-Ca2-3,5,50	1828.37	4758.0	80.4	3.08	163	622	627	74	209	210	5	6	50.0	16
MC-Ca3-3,5,50	1664.81	2158.0	213.4	3.28	190	834	841	106	276	231	4	5	40.0	14
MC-Ca4-3,5,50	1711.82	200.0	143.4	2.74	178	572	570	160	225	218	4	4	25.0	15
MC-Ca5-3,5,50	1685.09	2469.0	911.6	6.63	1334	624	613	194	506	229	3	4	75.0	14
MC-Cb1-3,5,50	1822.96	-	77.6	2.30	195	760	761	23	71	223	5	6	83.3	16
MC-Cb2-3,5,50	1849.23	-	655.0	3.79	1451	710	710	472	505	223	4	5	60.0	16
MC-Cb3-3,5,50	1839.83	-	121.6	2.38	149	798	821	23	110	225	4	6	66.7	17
MC-Cb4-3,5,50	1726.29	-	201.7	2.47	339	727	737	113	202	220	4	5	40.0	16
MC-Cb5-3,5,50	1683.58	-	408.2	3.58	733	740	731	102	92	224	4	4	25.0	14
MC-Cc1-3,5,50	1549.84	-	899.5	3.50	163	584	570	126	269	250	4	4	25.0	15
MC-Cc2-3,5,50	1683.01	-	551.3	3.16	138	760	767	112	302	250	4	5	40.0	15
MC-Cc3-3,5,50	1613.64	-	193.7	2.56	29	762	769	29	74	250	5	5	40.0	15
MC-Cc4-3,5,50	1827.61	-	528.7	3.62	177	657	664	118	334	250	5	6	66.7	17
MC-Cc5-3,5,50	1730.19	-	238.1	3.89	102	477	463	40	146	250	4	6	50.0	15
MC-Cd1-3,5,50	1738.81	-	124.3	2.92	73	757	769	87	160	248	4	6	83.3	15
MC-Cd2-3,5,50	1764.77	-	124.4	4.07	108	666	682	122	182	250	5	6	50.0	15
MC-Cd3-3,5,50	1652.72	-	85.0	2.76	62	660	655	50	142	250	4	5	80.0	16
MC-Cd4-3,5,50	1717.1	-	82.5	3.84	75	694	691	63	121	244	5	5	40.0	16
MC-Cd5-3,5,50	1663.41	-	207.3	3.37	188	620	624	124	324	243	5	5	40.0	15

Continued on next page

Table 8 – *Continued from previous page*

		Dellaert et al. (2021) †	Our BPC algorithm ‡										
Instance	z^*	T (s)	T (s)	Gap (%)	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
MC-Ca1-6,4,50	2110.31	-	72.6	2.71	203	243	216	96	268	172	4	6	83.3	16
MC-Ca2-6,4,50	2017.13	-	26.4	1.32	47	245	216	14	82	180	3	6	83.3	16
MC-Ca3-6,4,50	2214.22	-	20.2	1.63	48	238	220	37	72	176	4	8	87.5	16
MC-Ca4-6,4,50	1973.93	-	7.8	2.70	21	228	199	21	43	166	4	6	66.7	15
MC-Ca5-6,4,50	2075.04	-	42.4	2.56	96	240	217	94	73	176	3	7	85.7	15
MC-Cb1-6,4,50	2173.65	-	45.1	2.74	165	263	234	72	168	167	4	6	83.3	17
MC-Cb2-6,4,50	2177.47	-	6.5	1.86	15	216	187	16	23	176	3	7	100.0	17
MC-Cb3-6,4,50	2167.62	-	23.2	2.19	71	256	233	49	28	177	4	7	100.0	16
MC-Cb4-6,4,50	1985.13	-	54.8	4.01	137	279	254	102	113	175	3	6	83.3	16
MC-Cb5-6,4,50	2075.33	-	128.9	3.80	357	188	157	107	331	176	4	6	50.0	16
MC-Cc1-6,4,50	1960.1	-	913.9	5.05	245	225	195	232	284	200	4	6	50.0	15
MC-Cc2-6,4,50	1904.53	-	87.6	2.18	39	224	197	32	28	200	3	6	83.3	15
MC-Cc3-6,4,50	1948.02	-	76.0	1.90	27	180	139	32	44	200	4	6	50.0	16
MC-Cc4-6,4,50	1870.07	-	199.6	3.73	81	215	184	49	130	200	4	6	50.0	15
MC-Cc5-6,4,50	2166.48	-	197.8	4.22	116	217	190	60	198	200	4	7	85.7	17
MC-Cd1-6,4,50	1980.33	-	63.8	3.26	81	196	165	81	207	197	2	6	66.7	15
MC-Cd2-6,4,50	1960.23	-	319.3	3.23	459	252	223	137	536	197	3	6	50.0	15
MC-Cd3-6,4,50	2038.08	-	144.9	3.00	168	220	192	77	411	195	3	6	66.7	16
MC-Cd4-6,4,50	1867.35	-	31.3	2.85	46	225	192	43	47	197	4	6	66.7	15
MC-Cd5-6,4,50	1980.33	-	63.2	3.26	81	196	165	81	207	197	2	6	66.7	15

† Tests ran on a computer equipped with an Intel Core i7-4770 pocessor.

‡ Tests ran on a computer equipped with an Intel Core i7-8700 processor.

Table 9: Detailed computational results for instances with $|N^C| = 100$

		Dellaert et al. (2021) †	Our BPC algorithm ‡											
Instance	z^*	T (s)	T (s)	Gap (%)	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
MC-Ca1-2,3,100	3126.86	-	1172.8	0.39	1357	51	66	48	71	268	3	9	66.7	32
MC-Ca2-2,3,100	3270.5	-	1244.9	0.72	845	51	66	26	134	265	3	9	66.7	32
MC-Ca3-2,3,100	3004.22	-	69.3	1.00	57	46	55	58	217	249	3	8	37.5	30
MC-Ca4-2,3,100	3305.93	-	4827.3	0.88	3688	47	59	55	98	280	3	8	75.0	31
MC-Ca5-2,3,100	3095.43	-	365.6	0.86	182	48	60	35	166	277	3	8	62.5	30
MC-Cb1-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb2-2,3,100	3017.68	-	10624.2	0.95	8406	46	55	147	88	268	3	8	75.0	30
MC-Cb3-2,3,100	3000.1	-	9720.1	1.31	5805	48	60	137	182	270	3	8	75.0	30
MC-Cb4-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb5-2,3,100	3035.56	-	5796.1	0.78	4993	48	60	78	56	267	3	8	75.0	29
MC-Cc1-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc2-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc3-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc4-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc5-2,3,100	2944.71	-	1391.5	0.67	173	48	60	15	165	300	3	8	75.0	30
MC-Cd1-2,3,100	2932.14	-	1508.5	0.72	335	49	61	50	218	298	3	9	66.7	30
MC-Cd2-2,3,100	3157.21	-	1617.5	1.83	484	49	61	71	665	293	3	9	66.7	32
MC-Cd3-2,3,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd4-2,3,100	3111.62	-	3607.8	0.75	932	48	60	51	182	296	3	8	75.0	30
MC-Cd5-2,3,100	2976.11	-	265.8	1.08	53	41	45	66	438	295	3	8	50.0	30
MC-Ca1-3,5,100	3206.67	2919.0	9686.1	1.48	5453	966	1051	207	235	424	5	10	60.0	31
MC-Ca2-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Ca3-3,5,100	3029.88	-	10049.4	1.22	3946	978	1051	212	329	428	5	9	66.7	29
MC-Ca4-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Ca5-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb1-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb2-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb3-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb4-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb5-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc1-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc2-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc3-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc4-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc5-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd1-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd2-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd3-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd4-3,5,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd5-3,5,100	3048.75	-	4008.8	1.15	363	953	1026	249	864	490	3	9	100.0	30

Continued on next page

Table 9 – *Continued from previous page*

		Dellaert et al. (2021) †	Our BPC algorithm ‡											
Instance	z^*	T (s)	T (s)	Gap (%)	# Nodes	$\|\mathcal{M}\|$	# TIs	# RCIs	# SRIs	# VSIs	$\|N^S\|^*$	$\|\mathcal{M}\|^*$	$\|\mathcal{M}^1\|^*$ (%)	$\|\mathcal{L}\|^*$
MC-Ca1-6,4,100	3364.65	-	3810.1	1.99	1657	350	352	362	827	358	4	10	40.0	30
MC-Ca2-6,4,100	3465.14	-	4823.9	2.27	3348	324	325	1485	1527	352	4	10	50.0	28
MC-Ca3-6,4,100	3960.03	-	110.8	1.49	83	310	312	82	168	341	4	12	83.3	31
MC-Ca4-6,4,100	3374.9	-	7514.2	2.18	2999	333	330	698	1086	355	4	9	66.7	29
MC-Ca5-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb1-6,4,100	3402.62	-	5883.6	2.01	2887	341	343	724	210	362	4	10	60.0	28
MC-Cb2-6,4,100	3678.83	-	762.1	1.67	768	362	383	231	302	352	4	12	75.0	31
MC-Cb3-6,4,100	3725.96	-	4876.3	1.50	3615	311	320	196	151	358	4	11	81.8	32
MC-Cb4-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cb5-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc1-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc2-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc3-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc4-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cc5-6,4,100	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MC-Cd1-6,4,100	3474.89	-	3036.7	1.74	429	329	333	256	574	393	4	11	36.4	31
MC-Cd2-6,4,100	3374.88	-	999.5	1.79	541	337	338	359	269	390	4	10	80.0	30
MC-Cd3-6,4,100	3280.52	-	7594.8	2.78	1997	302	300	411	913	396	4	10	60.0	28
MC-Cd4-6,4,100	3545.37	-	6757.5	2.48	2359	263	252	316	979	389	4	11	81.8	32
MC-Cd5-6,4,100	3469.71	-	10275.3	1.46	1215	265	267	253	723	393	4	11	54.5	31

† Tests ran on a computer equipped with an Intel Core i7-4770 pocessor.

‡ Tests ran on a computer equipped with an Intel Core i7-8700 processor.

References

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

Baldacci R, Mingozzi A, Roberti R, Calvo RW (2013) An exact algorithm for the two-echelon capacitated vehicle routing problem. Operations Research 61(2):298–314.

Ben Amor H, Desrosiers J, Carvalho J (2006) Dual-optimal inequalities for stabilized column generation. Operations Research 54:454–463.

Boland N, Dethridge J, Dumitrescu I (2006) Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 34(1):58–68.

Breunig U, Schmid V, Hartl R, Vidal T (2016) A large neighbourhood based heuristic for two-echelon routing problems. Computers & Operations Research 76:208–225.

Costa L, Contardo C, Desaulniers G (2019) Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science 53(4):946–985.

Crainic TG, Mancini S, Perboli G, Tadei R (2008) Clustering-based heuristics for the two-echelon vehicle routing problem. Technical Report CIRRELT–2008–46, CIRRELT, Canada.

Crainic TG, Mancini S, Perboli G, Tadei R (2011) Multi-start heuristics for the two-echelon vehicle routing problem. Merz P, Hao JK, eds., Evolutionary Computation in Combinatorial Optimization, 179–190 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Cuda R, Guastaroba G, Speranza M (2015) A survey on two-echelon routing problems. Computers & Operations Research 55:185–199.

Dellaert N, Dashty Saridarq F, Van Woensel T, Crainic TG (2019) Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. Transportation Science 53(2):463–479.

Dellaert N, Van Woensel T, Crainic TG, Saridarq FD (2021) A multi-commodity two-echelon capacitated vehicle routing problem with time windows: Model formulations and solution approach. Computers & Operations Research 127:105154.

Drexl M (2012) Synchronization in vehicle routing – A survey of VRPs with multiple synchronization constraints. Transportation Science 46(3):297–316.

Dumez D, Tilk C, Irnich S, Lehuédé F, Olkis K, Péton O (2023) A matheuristic for a 2-echelon vehicle routing problem with capacitated satellites and reverse flows. European Journal of Operational Research 305(1):64–84.

Gschwind T, Irnich S (2016) Dual inequalities for stabilized column generation revisited. INFORMS Journal on Computing 28(1):175–194.

Gu W, Archetti C, Cattaruzza D, Ogier M, Semet F, Speranza MG (2022) A sequential approach for a multi-commodity two-echelon distribution problem. Computers & Industrial Engineering 163:107793.

Gu W, Archetti C, Cattaruzza D, Ogier M, Semet F, Speranza MG (2023) Vehicle routing problems with multiple commodities: A survey. European Journal of Operational Research .

Guastaroba G, Speranza MG, Vigo D (2016) Intermediate facilities in freight transportation planning: A survey. Transportation Science 50(3):763–789.

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. Desaulniers G, Desrosiers J, Solomon MM, eds., Column Generation, chapter 2, 33–65 (Boston, MA: Springer US), 1st edition.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research 56(2):497–511.

Jia S, Deng L, Zhao Q, Chen Y (2023) An adaptive large neighborhood search heuristic for multi-commodity two-echelon vehicle routing problem with satellite synchronization. Journal of Industrial and Management Optimization 19(2):1187–1210.

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

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

Marques G, Sadykov R, Deschamps JC, Dupas R (2020) An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. Computers & Operations Research 114:104833.

Marques G, Sadykov R, Dupas R, Deschamps JC (2022) A branch-cut-and-price approach for the single-trip and multi-trip two-echelon vehicle routing problem with time windows. Transportation Science 56(6):1598–1617.

Mhamedi T, Andersson H, Cherkesly M, Desaulniers G (2022) A branch-price-and-cut algorithm for the two-echelon vehicle routing problem with time windows. Transportation Science 56(1):245–264.

Perboli G, Tadei R, Vigo D (2011) The two-echelon capacitated vehicle routing problem: models and math-based heuristics. Transportation Science 45(3):364–380.

Petris M, Archetti C, Cattaruzza D, Ogier M, Semet F (2024) A branch-price-and-cut algorithm for the multi-commodity two-echelon distribution problem. EURO Journal on Transportation and Logistics 13:100139.

Righini G, Salani M (2008) New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks 51(3):155–170.

Santos FA, Mateus GR, da Cunha AS (2015) A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. Transportation Science 49(2):355–368.

Sluijk N, Florio AM, Kinable J, Dellaert N, Van Woensel T (2023) Two-echelon vehicle routing problems: A literature review. European Journal of Operational Research 304(3):865–886.

Soares R, Marques A, Amorim P, Parragh SN (2024) Synchronisation in vehicle routing: classification schema, modelling framework and literature review. European Journal of Operational Research 313(3):817–840.