Article:
He, Y., Liu, P., Xu, S. et al. (2026) Exact algorithm for a half-open multi-depot multi-commodity unpaired pickup and delivery problem with redistribution in omnichannel retailing. European Journal of Operational Research. ISSN: 0377-2217 (In Press)

https://doi.org/10.1016/j.ejor.2026.03.016

Exact Algorithm for a Multi-Depot Unpaired Pickup and Delivery Problem with Redistribution in Omnichannel Retailing

Ying He [a, b], Peng Liu [c], Shuxian Xu [a, b, *], Ronghui Liu [d]

[a] College of Management and Economics, Tianjin University, Tianjin 300072, China

[b] Laboratory of Computation and Analytics of Complex Management Systems (CACMS), Tianjin University, Tianjin 300072, China

[c] MOE Key Laboratory of Complex System Analysis and Management Decision, School of Economics and Management, Beihang University, Beijing 100191, China

[d] Institute for Transport Studies, University of Leeds, Leeds LS2 9JT, UK

Abstract: This paper investigates a multi-depot unpaired pickup and delivery problem with inventory redistribution and recovery in an integrated omnichannel retail transportation system. In this setting, a retailer operates multiple warehouses and physical stores, served by a homogeneous fleet that fulfills online customer orders while replenishing and recovering inventory for stores. We consider two product categories: saleable items, which can be redistributed among stores and customers, and unsaleable items, which must be collected and returned to warehouses. We formulate the problem as a mixed-integer programming (MIP) model that jointly optimizing routing, pickup–delivery pairing, and warehouse loading to minimize transportation and vehicle costs. The proposed model captures several key operational features, including multiple depots and commodities, unpaired pickup and delivery, time windows, and a hybrid routing structure that allows vehicles to follow closed or half-open routes, returning to any depot after task completion. To solve the model efficiently, we develop a branch-and-price-and-cut algorithm incorporating a bidirectional labeling scheme for the pricing subproblem, subset-row inequalities to strengthen the LP relaxation, and acceleration strategies to enhance computational performance. Extensive numerical experiments demonstrate the effectiveness of the proposed algorithm and offer managerial insights into the impact of redistribution strategies, demand heterogeneity, routing flexibility, and online–offline integration.

Keywords: Vehicle routing problem; unpaired pickup and delivery; redistribution and recovery; Branch-and-price-and-cut; omnichannel retailing

1. Introduction

1.1. Background and motivation

With the rapid expansion of e-commerce, traditional retailers are increasingly transforming their operations to adopt omnichannel retailing models that integrate physical stores with online channels. Omnichannel retailing aims to provide customers with a seamless shopping experience but introduces significant challenges in logistics and fulfillment. To meet the demands of both online and offline channels,

* Corresponding author.

E-mail addresses: 2023209113@tju.edu.cn (Y. He), p.liu@buaa.edu.cn (P. Liu), shuxianxu@tju.edu.cn (S. Xu) R.Liu@its.leeds.ac.uk (R. Liu).

retailers must replenish store inventories and deliver products to online customers. Simultaneously, consumer returns are particularly prevalent. According to David Sobie, CEO of Happy Returns, the return rate for online orders typically ranges from 15% to 40%, compared to 5% to 10% for in-store purchases. Additionally, many stores face the challenge of managing unsold inventory. As a result, retailers must handle two critical tasks: collecting unsold products from stores and returned items from online customers, and delivering replenishment stock to physical stores as well as fulfilling orders for online customers. These logistics operations are essential to meet customer expectations and improve the operational efficiency of the omnichannel model. However, existing studies on omnichannel retailing often focuses on inventory optimization or customer order fulfillment (Acimovic and Farias, 2019; Bayram and Cesaret, 2021; Goedhart et al., 2023), while neglecting the simultaneous integration of inventory replenishment, order fulfillment, and reverse logistics. Moreover, the distribution demands in this process lead to high transportation cost. For example, in 2018, Amazon's shipping cost accounted for 11.89% of its net sales (Wei et al., 2018). Therefore, efficient fleet organization and route planning has become a critical issue, and designing a distribution system tailored to the omnichannel model has also emerged as an urgent challenge for retailers, which have received limited attention in existing research.

In reality, some omnichannel retailers, such as JD.com and Tmall, manage distribution tasks for physical stores and individual customers separately. Their logistics infrastructure typically include a central warehouse, satellite hubs, and physical retail stores. Products for stores are replenished directly from the central warehouse using large vehicles, while items for online customers are initially stocked at the central warehouse, transferred to satellite hubs by trucks, and then delivered to customers by smaller vehicle (Li and Wang, 2025). Many studies focus on either delivering products from warehouses to stores (Paul et al., 2019a, 2019b; Qiu et al., 2025) or modeling these systems as a multi-echelon vehicle routing problem (Janjevic et al., 2021; Tahirov et al., 2025). However, this siloed approach often results in inefficiencies, such as the need for larger fleet sizes and higher travel costs (Abdulkader et al., 2018). Therefore, coordinating and organizing the distribution tasks for online and offline channels is critical for omnichannel retailing. Based on the above analysis, this paper proposes a novel pickup-and-delivery framework that integrates store replenishment, recovery of unsold products, and pickup/delivery for online customers, aiming to address the vehicle routing problem within this integrated context.

1.2. Problem statement and contributions

This paper contributes an unpaired pickup and delivery problem with inventory redistribution and recovery (UPDP-RR). **Figure 1** illustrates the product flows in this novel integrated omnichannel distribution system. The physical retailer operates multiple warehouses (D), several retail stores (R), and serves numerous online customers (C). Most product returns stem from reasons such as size, color, or personal preferences, with only a small proportion due to defects (Ratcliff, 2014). Unsold products in stores are often linked to demand mismatch rather than quality issues. The system considers two product categories: saleable products, which are delivered to stores or customers and returned in good condition, and unsaleable products, which are defective. Stores require replenishment of saleable products and collection of returns, while online customers place delivery orders for saleable items and return both

saleable and unsaleable products. Saleable products recovered from stores or customers can be redistributed to other stores or customers, while unsaleable products need to be returned directly to warehouses and cannot be temporarily stored elsewhere.

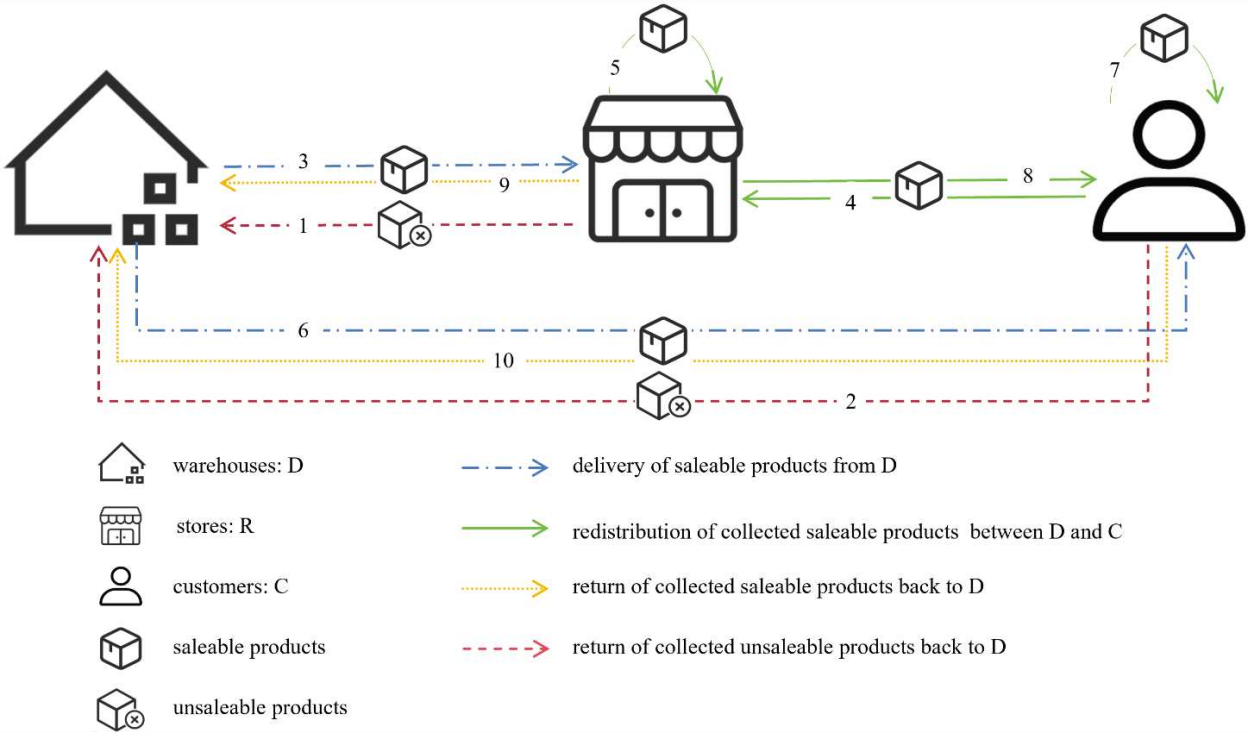

Fig. 1. The product flows among warehouses, stores and customers[1]

In practice, many retailers have adopted strategies to support such distribution flexibility. For instance, ship-from-store-to-store (SFSTS) strategies enable retailers like Best Buy to allow inter-store replenishment. Stores can also receive returns from both online and offline customers, as implemented by IKEA, J. Crew, Sephora, Walmart, and J.C. Penney. Moreover, online order fulfillment can be jointly handled by warehouses and stores, as practiced by Urban Outfitters, Walmart, and Amazon. A single vehicle fleet is dispatched from multiple warehouses to serve both stores and customers along optimized routes. We allow both closed routes and half-open routes (i.e., routes that start and end at the same warehouse, or start at one and end at another, respectively), as illustrated in **Fig. 2**.

We formulate UPDVRP-RR as a closed and half-open mixed multi-depot multi-commodity unpaired pickup and delivery problem with time windows (HMM-UPDPTW). Due to the allowance of redistribution among stores and customers, this model relaxes the constraint that online customer orders must be fulfilled by stores (Abdulkader et al., 2018; Martins et al., 2020) and the constraint that stores can only be replenished by the central warehouse (Li and Wang, 2025), thereby expanding the route selection space. Additionally, both types of products can be stored in the warehouses, so it is also necessary to decide the quantity of saleable products to be dispatched from the warehouse, which is interdependent with both the routing and pairing decisions, resulting in a highly challenging optimization task. Despite the extensive

[1] Unsaleable products need be returned directly to warehouses (line "1" and "2"). Saleable products delivered to stores/customers may originate from warehouses (line "3" / "6"), other online customers (line "4" / "7"), or other stores (line "5" / "8"). Collected saleable products that are not redistributed are returned to warehouses (line "9" and "10").

research on unpaired pickup and delivery vehicle routing problem (UPDVRP) (Erdoğan et al., 2014; Zhang et al., 2019; Osorio et al., 2021), to the best of our knowledge, little attention has been given to simultaneous collaborative routing for recovery and inventory replenishment for retail stores and pickup and delivery for online customers in the context of omnichannel retailing. Moreover, most studies addressing vehicle routing problems in omnichannel retailing rely on heuristic algorithms (Abdulkader et al., 2018; Schubert et al., 2021), while this paper proposes an exact algorithm to bridge this research gap.

Stores and customers	Delivery	Pickup	
R1	0	20	4
R2	30	0	0
R3	0	2	20
C1	0	2	0
C2	0	1	0
C3	0	1	0
C4	4	0	0

saleable products
unsaleable products

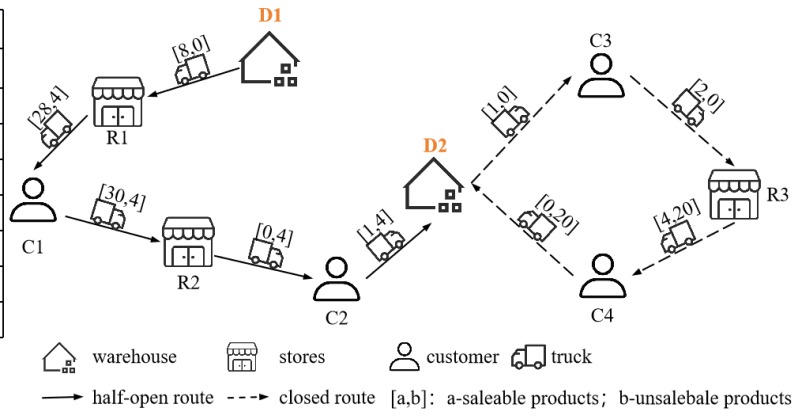

Fig. 2. An example of HMM-UPDPTW with two feasible routes[2]

The contributions of our study are threefold: First, we introduce and formulate a novel variant of the vehicle routing problem (VRP), termed HMM-UPDPTW. This problem captures the operational complexities of omnichannel retailing by integrating multiple depots, multiple commodities (saleable and unsaleable), unpaired pickup and delivery, redistribution and recovery, closed and half-open routing structures, and time window constraints. A comprehensive mixed-integer programming (MIP) formulation is developed to represent these interacting components. Second, we propose an exact solution method based on the branch-and-price-and-cut (BPC) framework, which incorporates an efficient bidirectional labeling algorithm with tailored dominance rules to solve the pricing subproblem. In addition, we implement several acceleration strategies, including two-cycle elimination, subset-row inequalities, time window tightening, and feasible solution recovery techniques, to enhance computational performance. Third, we conduct extensive computational experiments using customized instances adapted from Solomon's benchmark datasets. The algorithm's effectiveness is evaluated through performance metrics and sensitivity analyses. We examine the influence of critical operational factors, such as redistribution mechanisms, customer demand heterogeneity, the use of half-open routes, and integrated fulfillment between stores and customers—on overall system cost and efficiency. These results offer managerial insights into the design of resilient and efficient omnichannel distribution systems.

The remainder of this paper is organized as follows. **Section 2** presents a comprehensive literature review of relevant research. **Section 3** describes the HMM-UPDPTW in detail and formulate the problem as a MIP model. In **section 4**, a BPC algorithm is proposed. **Section 5** reports the computational

[2] In the **half-open** route, saleable products for store 2 are supplied by warehouse 1, store 1 and customer 1. Meanwhile, collected unsaleable products from store 1 and saleable products from customer 2 are returned to warehouse 2. In the **closed route**, customer 4 is supplied by warehouse 2, customer 3, and store 3, while collected unsaleable products from store 3 are returned to warehouse 2.

experiment results. Finally, the conclusions and promising research directions are discussed in **Section 6**.

2. Literature review

2.1. Omnichannel vehicle routing problem

The transportation of products in omnichannel retailing has garnered significant attention (Guo et al., 2021; Schubert et al., 2021; Yang and Li, 2023). Many studies have considered the fulfillment of online orders through retail stores, focusing on the vehicle routing problem associated with the transportation of products from warehouses to stores, consisting of products replenished to stores and to be delivered for online orders. However, the subsequent delivery of online orders from stores to end customers is often neglected. For example, Li and Wang (2025) examined the distribution of both store inventory and online orders from central warehouses to physical stores or satellite facilities in omnichannel retail systems, and developed an adaptive large neighborhood search algorithm for solution optimization. Paul et al. (2019a; 2019b) and Qiu et al. (2025) investigated scenarios where online customers could pick up their orders in stores. Their studies addressed the issue of store replenishment and online order delivery. Additionally, the products replenished to stores and those delivered to online orders come from different dedicated warehouses. They also allowed some online orders to be transferred from online delivery vehicles to replenishment delivery vehicles through a capacity-sharing strategy. Furthermore, some scholars have investigated multi-echelon vehicle routing problems that considered delivery to customers' doorsteps. For example, Janjevic et al. (2021) developed a three-echelon capacitated location-routing model and analyze the strategic configuration of three-tiered multi-modal distribution networks, where the hubs were restricted from directly serving online customers. Tahirov et al. (2025) proposed an omnichannel model for a distribution scenario that integrates a location-routing problem within a two-tier supply chain network, where online orders are fulfilled via the manufacturer, dark stores, or buy-online-pick-up-in-store. They further developed a heuristic method to solve the problem.

With the integration of online and offline channels in omnichannel retailing, some studies have focused on joint optimization problems. These studies employ a fleet of vehicles to simultaneously replenish stores and fulfill online orders. Abdulkader et al. (2018) were the first to address this problem, proposing a two-phase heuristic method and a multi-ant colony algorithm. The most solutions in this work were improved by Bayliss et al. (2020) via a two-phase local search with a discrete-event heuristic. Schubert et al. (2021) designed an algorithm based on general variable neighborhood search to address the integrated order picking-vehicle routing problem. However, these studies overlook key aspects such as product returns and the redistribution of inventory between stores and customers. Furthermore, they mostly assume that online orders can only be fulfilled by either stores or a central warehouse. In contrast, our study addresses these limitations by offering greater flexibility and a broader decision space.

2.2. Unpaired pickup and delivery vehicle routing problem（UPDVRP）

UPDVRP can be categorized into single-commodity UPDVRP (1-UPDVRP) and multi-commodity (M-UPDVRP). For 1-UPDVRP, Raviv et al. (2013) first introduced the UPDVRP with single-depot, multiple vehicles and solved it using a two-stage exact algorithm. Lei and Ouyang (2018) developed a

Lagrangian relaxation based algorithm to solve their proposed hybrid model. Dubey and Tanksale (2023) developed an elitist genetic algorithm and a hybrid genetic algorithm to address 1-UPDVRP. For M-UPDVRP, Chen et al. (2014) were the first to study the UPDVRP with multiple vehicles and commodities within a multi-plant tobacco manufacturing system, assuming that each plant's material inventory levels were either zero or unbounded. This means that the loads picked up from each support vertex are uncertain and it only need to ensure the full satisfaction of the delivery demand. And they propose an easy-to-implement heuristic and later improving its performance with a VNS (variable neighborhood search)-based local search algorithm. Zhang et al. (2019) designed an adaptive memory programming-based algorithm to solve a multi-commodity many-to-many vehicle routing problem with simultaneous pickup and delivery. Zhao et al. (2023) addressed a green split m-UPDVRP and introduced a two-phase search quantum particle swarm optimization method. All these studies assumed that all types of commodities can be balanced, which is a great distinction from our work. A notable application of UPDP is the bike relocation problem (BRP), which had been extensively studied. For example, Ho and Szeto (2017) introduced a hybrid large neighborhood search to address a BRP. Szeto and Shui (2018) proposed an improved artificial bee colony algorithm to solve a static BRP, which allowed demand dissatisfaction. Dell'Amico et al. (2018) developed branch-and-cut algorithm to address a BRP with stochastic demand. Zhang et al. (2020) studied a BRP combined bicycle repositioning and recycling challenge and developed an adaptive tabu search algorithm integrated with six neighborhood structures. However, most BRP studies do not consider time window constraints at each station. In the omnichannel retail system, it is essential to account for the time preferences of retail stores and online customers, ensuring that vehicles provide services within specified time windows to enhance customer satisfaction. Additionally, in BRP, multiple visits are typically allowed so that vehicles can temporarily store bikes at stations and retrieve them later (Chemla et al., 2013). This characteristic differs significantly from our problem, where such flexibility is not permitted.

2.3. Multi-depot pickup and delivery problem（MDPDP）

The HMM-UPDPTW considers multiple warehouses, which is a variant of MDPDP, where each depot can dispatch vehicles to provide pickup and delivery services for customers. MDPDP can be classified into four subcategories: closed multi-depot pickup and delivery problem (CMDPDP), open multi-depot pickup and delivery problem (OMDPDP), close-open mixed multi-depot pickup and delivery problem (COMDPDP), and close and half-open mixed multi-depot pickup and delivery problem (CHMDPDP).

In CMDPDP, all vehicles must return to their originating depots after completing their pickup and delivery tasks. Dayarian et al. (2015) proposed a branch-and-price methodology to solve a milk collection problem. Dridi et al. (2019) introduced the MDPDP with time windows and multiple vehicles (m-MDPDPTW) and proposed an innovative particle swarm optimization-based algorithm. Nafstad et al. (2021) formulated a helicopter flight scheduling problem as compact and extended mathematical model and apply delayed constraint generation to accelerate the solution process. In OMDPDP, vehicles are allowed to park at arbitrary locations after completing their tasks, meaning the travel cost of returning to

the depot from the last visited vertex is generally not considered. Tarantilis and Kiranoudis (2002) were the first to study the open vehicle routing problem with multiple depots and proposed a list-based threshold-accepting metaheuristic approach. Lahyani et al. (2019) designed a hybrid adaptive large neighborhood search algorithm, incorporating three insertion heuristics, five removal operators, and four local search strategies, while Brandão (2020) developed a memory-based iterated local search algorithm. Recently, an innovative adaptive variable neighborhood search approach was specifically designed by Hwang et al. (2024). In COMDPDP, both closed and open route modes are allowed. Zhen et al. (2022) proposed column generation-based solution methods, while Hamid et al. (2023) designed an efficient self-adaptive hyper-heuristic to solve COMDPDP and address a scheduling problem for homemade meal delivery involving drones and a crowd-sourced fleet. In CHMDPDP, vehicles can return to either their originating depot or a different depot. Ensafian et al. (2023) designed an adaptive backtracking-simulated annealing metaheuristic algorithm to address this variant.

Table 1 summarizes the most closely relevant works, aiming to highlight the contributions of this paper. As can be seen, most studies on pickup and delivery problem with open routes and multiple depots mainly employ heuristic algorithms. In contrast, we propose an exact algorithm to address the UPDP with multi-depot and half-open routes, offering a more flexible solution with greater cost reduction potential.

Table 1
Overview of literature of omnichannel retailing relevant to the problem in this paper.

Reference	Time windows	Depot	Pick-up and delivery	Warehouses can dispatch or receive products	Commodity	Algorithm
Li and Wang (2025)	√	SD, CR	delivery-only	√, ×	MC	Heuristic
Paul et al. (2019a)	×	MD, CR	delivery-only	√, ×	MC	BC, Heuristic
Paul et al. (2019b)	×	MD, CR	delivery-only	√, ×	MC	Heuristic
Qiu et al. (2025)	×	MD, CR	delivery-only	√, ×	MC	A three-layer procedure
Janjevic et al. (2021)	×	MD, CR	delivery-only	√, ×	MC	Solver
Tahirov et al. (2025)	×	SD, CR	delivery-only	√, ×	MC	Heuristic
Abdulkader et al. (2018)	×	SD, CR	unpaired	√, ×	MC	Heuristic
Bayliss et al. (2020)	×	SD, CR	unpaired	√, ×	MC	Heuristic
Schubert et al. (2021)	√	SD, CR	delivery-only	√, ×	MC	Heuristic
This study	**√**	**MD, CR, HOR**	**unpaired**	**√, √**	**MC**	**BPC**

MD: multi-depot; SD: single-depot; CR: Closed routes; OR: Open routes; HOR: half-open routes; Paired: paired pick-up and delivery; Unpaired: unpaired pick-up and delivery; MC: multi-commodity; SC: single-commodity.

3. Problem description and model formulation

In this section, we first describe the HMM-UPDPTW, and then present an arc-based mixed integer formulation of the problem. In order better to track the narrative, the main notations used in the paper are listed in **Table A.1** of **Appendix A**.

3.1. Problem description

We consider an omnichannel retailing company that operates a network of multiple warehouses and retail stores. Each day, the company must replenish inventory and manage recovery at retail stores while simultaneously fulfilling online delivery and pick-up demand from customers. These two operations require the design of efficient vehicle routes to visit both stores and customers. To improve vehicle

utilization, the company integrates these two operations, allowing stores and customers to be visited in a single route. This integration leads to a variant of the pickup and delivery problem. Formally, the problem is defined on a directed graph $G = \{V, A\}$, where the vertex set $V = D \cup R \cup C \cup D'$ consists of a warehouse vertex set D, a dummy warehouse vertex set D', a retail store vertex set R and a customer vertex set C. The arc set is defined as $A = \{(i, j) \mid i, j \in V,\ i \neq j\}$.

As mentioned earlier, products are categorized as saleable and unsaleable ones. The former refer to products that are originated from warehouses or returned in good condition and can be delivered to stores and customers directly, while the latter denote defective items that cannot be delivered. Each retailer store $i \in R$ has a replenishment demand of d_i units saleable products, and a pickup demand of p_i units saleable products and u_i units unsaleable products, respectively. It is reasonable to assume that $d_i p_i \equiv 0$, meaning that each single retail store $i \in R$ cannot simultaneously have both replenishment and saleable pickup demand. Similarly, each customer $i \in C$ has either a delivery demand of d_i units saleable products or a pick-up demand of either p_i units saleable products or u_i units unsaleable products.

Each required visited vertex $i \in R \cup C$ is associated with a strict time window $[e_i, l_i]$, meaning that if the vehicle arrives before e_i, it must wait until e_i to begin service. Additionally, the vehicle must not arrive later than l_i. We assume each warehouse $o \in D$ has a sufficiently large fleet of homogeneous vehicles. Each vehicle incurs a fixed cost γ, has a maximum range of L kilometers, and a capacity limit of Q units products. Vehicles are not required to return to their originating warehouse, allowing for both closed and half-open routes, as long as they return to any warehouse $d \in D'$ before the operational time l_d ends.

For a homogeneous fleet, the travel time on each arc $(i, j) \in A$ is given by $t_{ij} = s_i + d_{ij} / v$, where s_i denotes the service time at vertex i, d_{ij} represents the travel distance along arc (i, j), and v is the vehicle speed. The transportation cost c_{ij} along arc $(i, j) \in A$ is calculated as $c_{ij} = \alpha d_{ij}$, where α is a constant that represents the per-kilometer transportation cost.

The objective of this problem is to identify the vehicle routes and determine the outbound quantity of saleable products from a warehouse for each vehicle, ensuring that the demands of retail stores and customers for saleable products are satisfied, including those collected along the route. Additionally, the goal is to minimize the total cost, which consists of transportation cost and fixed vehicle cost.

3.2. Arc-based formulation

Several decision variables are introduced in this problem. Let x_{ijk} be a binary variable that equals 1 if vehicle k travels along arc $(i, j) \in A$, and 0 otherwise. Continuous variables T_{ik} and b_{ik} represent the time at which vehicle $k \in K$ starts offering services at vertex $i \in V$, and the accumulated travel distance when vehicle $k \in K$ arrives at vertex $i \in V$, respectively. The quantities of saleable and unsaleable products carried by vehicle $k \in K$ when arriving at vertex $i \in V$ are denoted by the non-negative integer variables P_{ik} and U_{ik}, respectively. Similarly, the quantities of saleable and unsaleable products carried by vehicle $k \in K$ when departing from vertex $i \in V$ are represented by the non-negative integer variables P'_{ik} and U'_{ik}, respectively. Furthermore, the non-negative integer variable N_{ik} is defined as the quantity of saleable products carried by vehicle $k \in K$ when departing from warehouse $i \in D$. This

variable contributes to the complexity of the problem due to its interdependence with vehicle routes selection. Using the notations and variables above, the HMM-UPDPTW is formulated as an arc-based formulations in the following:

$$\min Z=\sum_{k\in K}\sum_{(i,j)\in A} c_{ij}x_{ijk}+\sum_{k\in K}\sum_{i\in D}\sum_{j\in R\cup C} x_{ijk}\gamma \tag{1a}$$

$$\text{s.t.} \sum_{k\in K}\sum_{i\in V\setminus D',\, i\neq j} x_{ijk}=1,\ \forall j\in R\cup C \tag{1b}$$

$$\sum_{j\in V\setminus D,\, i\neq j} x_{ijk}=\sum_{j\in V\setminus D',\, i\neq j} x_{jik},\ \forall i\in R\cup C,\ k\in K \tag{1c}$$

$$\sum_{i\in D}\sum_{j\in R\cup C} x_{ijk}=\sum_{j\in R\cup C}\sum_{i\in D'} x_{jik},\ \forall k\in K \tag{1d}$$

$$T_{ik}+t_{ij}-M(1-x_{ijk})\le T_{jk},\ \ \forall (i,j)\in A,\ k\in K \tag{1e}$$

$$T_{ik}\ge e_i,\ \forall i\in V,\ k\in K \tag{1f}$$

$$T_{ik}\le l_i,\ \forall i\in V,\ k\in K \tag{1g}$$

$$U'_{ik}=U_{ik}=0,\ \forall i\in D,\ k\in K \tag{1h}$$

$$P'_{ik}=N_{ik}\le Q,\ \forall i\in D,\ k\in K \tag{1i}$$

$$P'_{ik}=P_{ik}-(d_i-p_i),\ \forall i\in R\cup C,\ k\in K \tag{1j}$$

$$U'_{ik}=U_{ik}+u_i,\ \forall i\in R\cup C,\ k\in K \tag{1k}$$

$$P_{jk}+M(1-x_{ijk})\ge P'_{ik},\ \forall (i,j)\in A,\ k\in K \tag{1l}$$

$$P_{jk}\le P'_{ik}+M(1-x_{ijk}),\ \forall (i,j)\in A,\ k\in K \tag{1m}$$

$$U_{jk}\le U'_{ik}+M(1-x_{ijk}),\ \forall (i,j)\in A,\ k\in K \tag{1n}$$

$$U_{jk}+M(1-x_{ijk})\ge U'_{ik},\ \forall (i,j)\in A,\ k\in K \tag{1o}$$

$$P'_{ik}+U'_{ik}\le Q,\ \forall i\in V,\ k\in K \tag{1p}$$

$$P_{ik}+U_{ik}\le Q,\ \forall i\in V,\ k\in K \tag{1q}$$

$$b_{ik}+d_{ij}-M(1-x_{ijk})\le b_{jk},\ \forall (i,j)\in A,\ k\in K \tag{1r}$$

$$b_{ik}\le L,\ \forall i\in V,\ k\in K \tag{1s}$$

$$x_{ijk}\in\{0,\ 1\},\ \forall (i,j)\in A,\ k\in K \tag{1t}$$

$$T_{ik}\ge 0,\ b_{ik}\ge 0,\ P_{ik}\ge 0,\ P'_{ik}\ge 0,\ U_{ik}\ge 0,\ U'_{ik}\ge 0,\ N_{ik}\ge 0,\ \forall i\in V,\ k\in K \tag{1u}$$

The objective of this problem is to minimize the total cost, which consists of the travel cost and fixed cost of the used vehicles, as shown in Eq. (1a). Constraints (1b) ensure that each retail store or online customer is visited exactly once by one vehicle. Constraints (1c) define the flow conservation at each retail store or online customer. Constraints (1d) guarantee that a vehicle departs from one warehouse, and serves several vertices, and finally returns to any warehouse, allowing that the starting and ending warehouses are different. Constraints (1e) represent the relationship between the service start time T_{ik} and T_{jk} when vehicle $k\in K$ travels arc (i,j). Constraints (1f) and (1g) restrict the hard time window constraints for each vertex $i\in V$. Constraints (1h) claim that no vehicle carries unsaleable products when departing from a warehouse. Constraints (1i) require that each vehicle may carry some saleable products when leaving a

warehouse, which necessitates decision-making. Constraints (1j) and (1k) describe the change in the quantities of saleable and unsaleable products, respectively, loaded on a vehicle before and after serving a retail store or an online customer. Constraints (1l) - (1o) impose that there is no change in the quantities of saleable and unsaleable products carried by vehicle k when traveling along arc $(i, j) \in A$. Constraints (1p) and (1q) ensure that the quantity of products loaded on vehicle k throughout the entire trip does not exceed its capacity limitation Q. Constraints (1r) imply the accumulation of mileage. Constraints (1s) restrict the accumulated mileage so that it never exceeds the range limitation L. Constraints (1t) and (1u) state the domains of the decision variables.

4. The branch-and-price-and-cut algorithm

The solution process for the arc-based formulation proposed above is significantly constrained by the instance size. Specifically, its application to large-scale examples is extremely time-consuming and may even fail to identify a feasible solution. To overcome this limitation, we reformulate the HMM-UPDPTW as a set partitioning model, where feasible routes are used as decision variables based on Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960; Lübbecke and Desrosiers, 2005; Desaulniers et al., 2005). The reformulated model is referred to as the master problem (MP). From its dual solution, we derive the pricing subproblem (SP), which corresponds to an elementary shortest-path problem with resource constraints (ESPPRC).

We develop an enhanced branch-and-price-and-cut (BPC) algorithm to solve the HMM-UPDPTW. Before delving into the algorithm, **Figure 3** illustrates an overview with a flowchart. The BPC algorithm integrates a branch-and-bound algorithm with column generation. A greedy insertion algorithm is applied to generate an initial feasible solution, thereby creating the root node of the branch-and-bound tree. Column generation involves an iterative procedure to solve the linear relaxation of the master problem (LMP). At each iteration, the restricted LMP (RLMP), which contains only a subset of all the route variables, is solved using the simplex method. The dual solution of the RLMP is then used as the input of SP. A bidirectional labeling algorithm is employed to identify columns with negative reduced cost. To accelerate the process, several acceleration strategies are applied. If some such columns are found, they are added to the RLMP and trigger the next iteration. Otherwise, if Condition 1 is satisfied, a violating subset-row inequality is searched for and added to the RLMP to lift the lower bound of the LMP, continuing the next iteration. The column generation terminates when no new columns are found and Condition 1[3] is not satisfied. If Condition 2[4] is satisfied, an integer feasible solution can be recovered from all the current columns, and the upper bound is updated. In the branch-and-bound framework, if the solution of the LMP is not fractional, the upper bound is updated, and the termination conditions are checked. Otherwise, three branching strategies are introduced to generate new branches and continue the column generation process. The details of the BPC algorithm are described in the following.

[3] Condition 1 represents that either (the current node depth is no more than 3 with no more than 2 cuts found), or (the current node depth exceeds 3 with no cuts found).

[4] Condition 2 will be introduced in section 4.5.3.

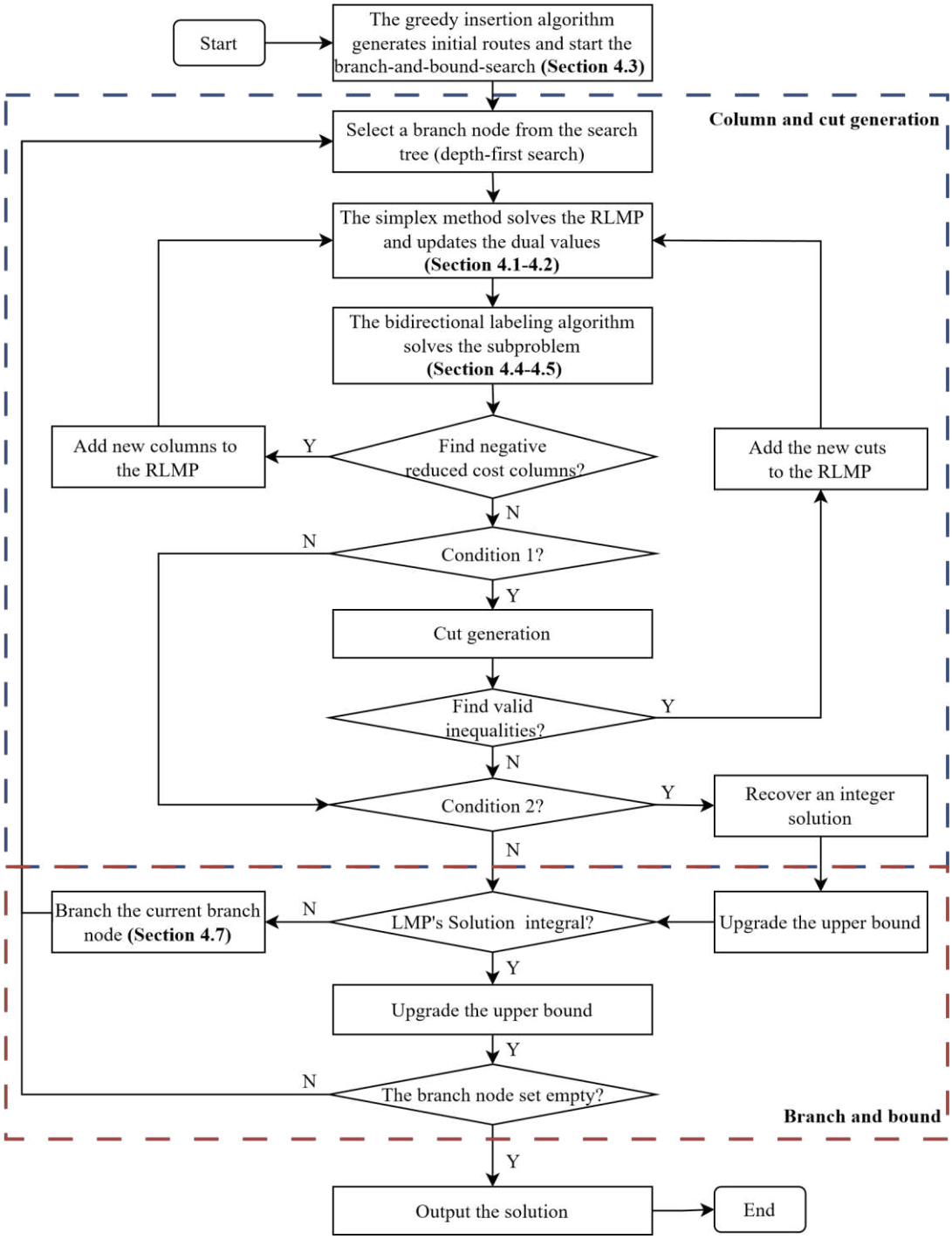

Fig. 3. The flowchart of the proposed BPC algorithm

4.1. The master problem（MP)

We reformulate the problem into a set partitioning model. Let $\Re$ denote the set of all the feasible routes. Associate a binary decision variable θ_r with a route $r \in \Re$, which equals 1 if and only if the route is selected in the optimal solution, and 0 otherwise. Let λ_{ri} be a binary variable indicating whether route r visits vertex $i \in R \cup C$. Moreover, the cost of route $r \in \Re$ consisting of the travel cost and the fixed cost, is denoted by c_r. Based on these notations, the formulation is expressed as follows:

$$\min \sum_{r\in\Re} c_r\theta_r \tag{2a}$$

$$s.t. \quad \sum_{r\in\Re} \lambda_{ri}\theta_r = 1, \ \forall i \in R\cup C \tag{2b}$$

$$\theta_r = \{0,\ 1\}, \ \forall r \in \Re \tag{2c}$$

where $c_r = \sum_{(i,j)\in A} z_{ijr}c_{ij} + \gamma$, and z_{ijr} is a binary variable specifying whether or not route r travels arc $(i,j)\in A$. Objective (2a) minimizes the total cost of selected routes. Constraints (2b) ensure that each retail store or customer is visited exactly once. Constraints (2c) define the feasible domains for the binary decision variables, allowing for an exponential number of them.

To tighten the lower bound on the LMP, we consider the subset-row (SR) inequalities introduced by Jepsenet al. (2008). An SR inequality for our problem is defined over a vertex set $S\subset R\cup C$ that contains exactly three vertices. Let Λ denote the set of identified subsets (i.e., $S\in\Lambda$). $\lfloor \sum_{i\in S}\lambda_{ir}/2 \rfloor$ equals 1 if route r visits at least two vertices in $S\subset R\cup C$ and 0 otherwise. Eqs. (3) ensure that no more than one route visiting at least two vertices in S can be selected in the optimal solution. Otherwise, there exists at least one vertex that is visited more than once, thereby violating Constraints (2b).

$$\sum_{r\in\Re}\left\lfloor \sum_{i\in S}\lambda_{ir}/2 \right\rfloor \theta_r \le 1, \ \forall S\in\Lambda, \ |S| = 3 \tag{3}$$

4.2. The pricing subproblem (SP)

At each iteration, based on the optimal dual solution of the RLMP, the pricing problem is constructed to find feasible routes $r\in\Re$ with negative reduced cost, denoted as $\bar{c}_r$. Let π_i, $i\in R\cup C$ and ϑ_S, $S\in\Lambda$ denote the dual variables associated with Constraints (2b) and Eqs. (3), respectively. The reduced cost of route $r\in\Re$ is then calculated by

$$\begin{aligned}\bar{c}_r &= c_r - \sum_{i\in R\cup C}\lambda_{ir}\pi_i - \sum_{S\in\Lambda}\left\lfloor \sum_{i\in S}\lambda_{ir}/2 \right\rfloor \vartheta_S \\ &= \sum_{(i,j)\in A} x_{ijr}\bar{c}_{ij} + \gamma - \sum_{S\in\Lambda}\left\lfloor \sum_{i\in S}\lambda_{ir}/2 \right\rfloor \vartheta_S\end{aligned} \tag{4}$$

where $\bar{c}_{ij}$ represents the redefined cost of arc $(i,j)\in A$ and $\bar{c}_{ij} = \begin{cases} c_{ij}, & \forall i\in D \\ c_{ij}-\pi_i, & \forall i\in R\cup C \end{cases}$.

4.3. Initial solution

A high-quality initial solution can significantly reduce subsequent iteration time. To obtain such a solution within a reasonable timeframe, we propose a greedy insertion algorithm, which is based on Li' et al. (2020) work. The pseudocode of the algorithm is shown in **Algorithm 1**. First, the set V_u containing uninserted customer and store vertices and the set R_{Init} of closed and half-open empty routes whit all possible warehouse combinations are defined and initialized (lines 1-4, **Algorithm 1**). Then, the customer closest to all warehouses in distance is selected to generate the shortest route r_{new}, which includes the customer and the corresponding two warehouses, and r_{new} is added to R_{Init} (lines 5, **Algorithm 1**). Note that during subsequent insertions, if a vertex is inserted into an empty route, another new route visiting this vertex and corresponding two warehouses should be generated and added to R_{Init}, ensuring that R_{Init}

always maintains the empty routes of all warehouse combinations as the initial set. Then, at each iteration, a vertex that brings the lowest cost increment is selected and inserted into the appropriate position of the corresponding route. This process is repeated until all customers and stores are inserted into the routes.

Algorithm 1. The greedy insertion

Let V_u be the set of uninserted customer and store vertices;
Let R_{Init} be the set of initial routes, and $r_{oo'}$ represent the empty route from warehouse $o \in D$ to warehouse $o' \in D'$.
Initialize $V_u \leftarrow R \cup C$;
Initialize $R_{Init} \leftarrow \{r_{oo'} \mid \forall o \in D, o' \in D'\}$;
Select the corresponding customer or store $i \in R \cup C$ with $\min_{\forall o \in D, o' \in D', i \in R \cup C} \{d_{oi} + d_{io'}\}$ to generate a new route r_{new}. Then add r_{new} to R_{Init} and remove vertex i from V_u;
While $V_u \neq \varnothing$ **do**
  $c_{\Delta_{\min}} \leftarrow +\infty$;
  for all $i \in V_u$ **do**
    for all $r \in R_{Init}$ **do**
      for all insertion position pos of r **do**
        if it is feasible to insert i into pos **then**
          if r is an empty route **then**
            Calculate the cost increment $c_\Delta = c_{\Delta_d} + \gamma$ after the insertion, where c_{Δ_d} represents the increased travel cost.
          else
            $c_\Delta = c_{\Delta_d}$;
          end if
        end if
        if $c_\Delta < c_{\Delta_{\min}}$ **then**
          $c_{\Delta_{\min}} = c_\Delta$, $r^* = r$, $v^* = i$, $pos^* = pos$;
        end if
      end for
    end for
  end for
  if r^* is an empty route **then**
    Generate a new open route r_{new} that is exactly the same as r^* in r^*, and add it to R_{Init};
  end if
  Insert v^* into the position pos^* of route r^*;
  Remove v^* from V_u;
end while
Return the set of non-empty routes in R_{Init}.

4.4. The bidirectional labeling algorithm

In this subsection, we propose a tailored bidirectional labeling algorithm to solve the SP. The bidirectional labeling technique has been widely applied to various variants of vehicle routing problem (Righini and Salani, 2006; Li et al., 2020; Su et al., 2023). Its key feature is the simultaneous generation and extension of forward and backward labels, which are eventually merged to form complete routes. In this approach, the critical resource is defined as the time duration of warehouses. Forward labels and backward labels, originating form a warehouse $o \in D$ and a dummy warehouse $d \in D'$, respectively, propagate until the time consumption exceeds the defined limit $T/2$. In the following, we provide a detailed explanation of the forward labeling, backward labeling and merging procedures.

Algorithm 2. The bidirectional labeling algorithm

1: Initialize $LC \leftarrow \varnothing$, $n_c \leftarrow 0$;
2: **for** all $o \in D$ **do**
3: Generate a new forward label $l_o^f = \{\backslash, 0, e_o, 0, Q, 0, 0, \{0,...,0\}_{S\in\Lambda}, \varnothing, \varnothing\}$;
4: Add l_o^f to UL_o^f, $TL_o^f \leftarrow \varnothing$;
5: **end for**
6: **for** all $o' \in D'$ **do**
7: Generate a new backward label $l_{o'}^b = \{\backslash, 0, l_{o'}, 0, 0, 0, \{0,...,0\}_{S\in\Lambda}, \varnothing, \varnothing\}$;
8: Add $l_{o'}^b$ to $UL_{o'}^b$, $TL_{o'}^b \leftarrow \varnothing$;
9: **end for**
10: **for** all $i \in R \cup C$ **do**
11: $UL_i^f \leftarrow \varnothing$, $TL_i^f \leftarrow \varnothing$, $UL_i^b \leftarrow \varnothing$, $TL_i^b \leftarrow \varnothing$;
12: **end for**
13: **while** $\bigcup_{\forall i\in V}(UL_i^f \cup UL_i^b) \neq \varnothing$ **do**
14: $L_u \leftarrow \bigcup_{\forall i\in V}(UL_i^f \cup UL_i^b)$, $\bigcup_{\forall i\in V}(UL_i^f \cup UL_i^b) \leftarrow \varnothing$;
15: **for** all $l_i \in L_u$ (l_i can be a forward or a backward label) **do**
16: Move l_i^{f*} from UL_i^f to TL_i^f (assume l_i is a forward label l_i^{f*});
17: **if** $i \in R \cup C$ **then**
18: *Dominance Test* (i, UL_i^f, TL_i^f, UL_i^b, TL_i^b, L_u)[5];
19: **end if**
20: **if** l_i^{f*} is dominated **then**
21: Go to line 16;
22: **end if**
23: **if** $i \in R \cup C$ **then**
24: *Merge* (i, UL_i^f, TL_i^f, UL_i^b, TL_i^b, LC, n_c)[6];
25: **end if**
26: **if** $n_c \geq n_{\max}$ **then**
27: Go to line 38;
28: **end if**
29: **if** $t_i^{f*} > \frac{T}{2}$ **then**
30: Go back to line 16;
31: **end if**
32: **for** $j \in R \cup C \setminus U_i^{f*}$ **do**
33: Extend l_i^{f*} to j and generate a new label l_j^{f*}, if l_j^{f*} is feasible according to conditions (5a) - (5d);
34: Add l_j^{f*} to UL_j^f;
35: **end for**
36: **end for**
37: **end while**

4.4.1. Forward labeling

Denote by $l_i^f = \{\tau_i^f, c_i^f, a_i^f, t_i^f, \rho_i^f, \varphi_i^f, \alpha_i^f, \beta_i^f, \{\theta_{Si}^f\}_{S\in\Lambda}, \mathrm{X}_i^f, U_i^f\}$ the label of a forward partial route r_i^f with the lasted visited vertex $i \in V \setminus D'$, where

- τ_i^f : the predecessor label of l_i^f;
- c_i^f : the reduced cost of the partial route r_i^f;
- t_i^f : the earliest feasible service start time at vertex i;

[5] The retails are demonstrated in Algorithm C.1.
[6] The retails are demonstrated in Algorithm C.2.

- ρ_i^f : the accumulated mileage of the partial route r_i^f ;
- φ_i^f : the maximum possible quantity of saleable products on the vehicle, sourced from the warehouse, after serving vertex i ;
- α_i^f : the current quantity of saleable products collected from retail stores and online customers by the vehicle along the partial route r_i^f after serving vertex i . Note that it refers to the quantity of collected saleable products currently on the vehicle, not the accumulated quantity of collected saleable products. For example, when the vehicle redistributes some of those to a vertex with delivery demand along the partial route, the current quantity on the vehicle will be less than the accumulated quantity.
- β_i^f : the accumulated quantity of unsaleable products collected from retail stores and online customers by the vehicle along the partial route r_i^f after serving vertex i .
- θ_{Si}^f : the accumulated number of visits to vertices in $S \in \Lambda$ along the partial route r_i^f .
- X_i^f : the set of retail store and online customer vertices visited along the partial route r_i^f .
- U_i^f : the set of retail stores and online customers that are visited along the partial route r_i^f or are unreachable when further propagating. A vertex $n \in R \cup C$ is called unreachable if any of the following conditions holds: $t_i^f + t_{in} > l_n$, $\rho_i^f + d_{in} > L$ or $\beta_i^f + \max\{u_n, d_n\} > Q$. In such cases, vertex n cannot be part of any feasible forward propagation of the partial route r_i^f .

A forward label l_i^f can propagate to vertex $j \in R \cup C \setminus U_i^f$ to form a new forward label l_j^f . The label l_j^f is considered infeasible if any one of the following conditions is satisfied, and it is discarded.

$$t_i^f + t_{ij} > l_j \tag{5a}$$

$$\rho_i^f + d_{ij} > L \tag{5b}$$

$$\alpha_i^f - d_j + p_j + \beta_i^f + u_j > Q, \text{ if } d_j \le \alpha_i^f \tag{5c}$$

$$\varphi_i^f - d_j + \alpha_i^f < 0 \ \vee \ \beta_i^f + u_j > Q, \text{ if } d_j > \alpha_i^f \tag{5d}$$

Forward label extension. The initial forward label originated from one warehouse $o \in D$ is defined as $l_o^f = \{\setminus, 0, e_o, 0, Q, 0, 0, \{0,...,0\}_{S\in\Lambda}, \varnothing, \varnothing\}$. If none of the conditions (5a) - (5d) satisfies, a new label $l_j^f = \{\tau_j^f, c_j^f, t_j^f, \rho_j^f, \varphi_j^f, \alpha_j^f, \beta_j^f, \mathrm{X}_i^f, U_j^f\}$ can be generated through the propagation of label l_i^f along arc $(i, j) \in A, j \in R \cup C \setminus U_i^f$, where

$$\tau_j^f = l_i^f \tag{6a}$$

$$c_j^f = c_i^f + \overline{c}_{ij} - \sum_{j\in S,\ S\in\Lambda,\ \theta_{Si}^f = 1} \vartheta_S \tag{6b}$$

$$t_j^f = \max\{t_i^f + t_{ij},\ e_j\} \tag{6c}$$

$$\rho_j^f = \rho_i^f + d_{ij} \tag{6d}$$

$$\theta_{Sj}^f = \begin{cases} \theta_{Si}^f + 1, & \text{if } j \in S \\ \theta_{Si}^f, & \text{otherwise} \end{cases}, \forall S \in \Lambda \tag{6e}$$

$$\mathrm{X}_j^f = \mathrm{X}_i^f \cup \{j\} \tag{6f}$$

$$U_j^f = U_i^f \cup \aleph_j^f \cup \{j\} \tag{6g}$$

$$\beta_j^f = \beta_i^f + u_j \tag{6h}$$

$$\alpha_j^f = \max\{0,\ \alpha_i^f - d_j + p_j\} \tag{6i}$$

$$\varphi_j^f = \begin{cases} \min\{\varphi_i^f,\ Q - \alpha_j^f - \beta_j^f\}, & \text{if } d_j \le \alpha_i^f \\ \min\{\varphi_i^f - d_j + \alpha_i^f,\ Q - \beta_j^f\}, & \text{otherwise} \end{cases} \tag{6j}$$

where $\aleph_j^f$ is the subset of the vertices in $(R \cup C) \setminus U_i^f$ that are unreachable from the partial route associated with label l_j^f. Extension conditions (6a) - (6g) are straightforward. Condition (6h) represents the accumulated unsaleable products picked up from previous visited vertices along the partial route r_j^f of label l_j^f. The explanation for conditions (6i) and (6j) is as follows: When $\mathrm{d}_j \le \alpha_i^f$, the saleable products collected from previous visited vertices along the partial route r_j^f are sufficient to meet the delivery demand for saleable products of vertex j. As a result, the remaining saleable products collected form retailer stores and customers by the vehicle after servicing vertex j will be $\alpha_i^f - d_j + p_j$. This implies that there is no need to use the saleable products originated from the warehouse. Given the load capacity limitation, $\varphi_j^f = \min\{\varphi_i^f,\ Q - \alpha_j^f - \beta_j^f\}$ is satisfied. On the other hand, when $\mathrm{d}_j > \alpha_i^f$, the saleable products collected from previous visited vertices are insufficient to meet the demand for saleable products of vertex j. In this case, an additional $d_j - \alpha_i^f$ units must be sourced from the warehouse, making $\alpha_j^f = 0$ and $\varphi_j^f = \min\{\varphi_i^f - d_j + \alpha_i^f,\ Q - \beta_j^f\}$.

Forward dominance rules. Label algorithms can generate an exponentially large number of labels. To accelerate this process and efficiently identify columns that significantly reduce the objective of RLMP, we propose novel effective dominance rules. These rules ensure the optimality of the LMP while discarding as many unpromising columns as possible.

For any two forward labels $l_i^f = \{\tau_i^f,\ c_i^f,\ t_i^f,\ \rho_i^f,\ \varphi_i^f,\ \alpha_i^f,\ \beta_i^f,\ \mathrm{X}_i^f,\ U_i^f\}$ and $l_i^{f'} = \{\tau_i^{f'},\ c_i^{f'},\ t_i^{f'},\ \rho_i^{f'},\ \varphi_i^{f'},\ \alpha_i^{f'},\ \beta_i^{f'},\ \mathrm{X}_i^{f'},\ U_i^{f'}\}$, the former dominates the latter if

$$(R \cup C) \setminus U_i^{f'} \subset (R \cup C) \setminus U_i^f \tag{7a}$$

$$\rho_i^f \le \rho_i^{f'} \tag{7b}$$

$$t_i^f \le t_i^{f'} \tag{7c}$$

$$\varphi_i^f + \alpha_i^f \ge \varphi_i^{f'} + \alpha_i^{f'} \tag{7d}$$

$$\alpha_i^f + \beta_i^f \le \alpha_i^{f'} + \beta_i^{f'} \tag{7e}$$

$$\beta_i^f \le \beta_i^{f'} \tag{7f}$$

$$c_i^f \le c_i^{f'} \tag{7g}$$

$$\theta_{Si}^f \ne 1 \ \vee\ (\theta_{Si}^f = 1 \ \wedge\ \theta_{Si}^{f'} \bmod 2 = 1),\ \forall S \in \Lambda \tag{7h}$$

Proposition 1. Forward dominance rules (7a) - (7h) are valid.

Proof. See **Appendix B.1**.

4.4.2. Backward labeling

Denote by $l_i^b = \{\tau_i^b,\ c_i^b,\ t_i^b,\ \rho_i^b,\ \omega_i^b,\ \xi_i^b,\ \{\theta_{Si}^b\}_{S \in \Lambda},\ \mathrm{X}_i^b,\ U_i^b\}$ the label of a backward partial route r_i^b with the lasted visited vertex $i \in V \setminus D$. The meanings of components of the label l_i^b are analogous to those of the forward label, except that:

- t_i^b: the latest feasible service start time at the vertex i;

- ω_i^b: the minimum quantity of saleable products loaded on vehicle when arriving at vertex i that ensures the delivery demand along the partial route r_i^b can be fully satisfied. Note that ω_i^b is not the accumulated delivery demand of the partial route r_i^b, and the former never exceeds the latter. We assume that the saleable products collected from retail stores and online customers can be redistributed to satisfy the delivery demands of other vertices. Therefore, if the partial or the total delivery demand of a vertex visited later than vertex i along the partial route r_i^b is satisfied by saleable products collected from other retail stores or customers, ω_i^b is smaller than the accumulated delivery demand of the partial route r_i^b.
- ξ_i^b: the quantity of products loaded on the vehicle when the loads along this partial route is at its maximum, updated after serving vertex i.
- U_i^f: the set of retail stores and online customers that are visited along the partial route r_i^b or that are unreachable when further propagating. A vertex $n \in R \cup C$ is called unreachable if $t_i^b - t_{ni} < e_n$, $\rho_i^b + d_{ni} > L$ or $\omega_i^b + u_n > Q$, in which case it cannot be part of any feasible forward propagation of the partial route r_i^b.

A backward label l_i^b can propagate to vertex j along arc $(j, i) \in A, j \in R \cup C \setminus U_i^b$ to form a new backward label l_j^b. The label l_j^b is considered infeasible if any of the following conditions are satisfied, and it is discarded.

$$t_i^b - t_{ji} < e_j \tag{8a}$$

$$\rho_i^b + d_{ji} > L \tag{8b}$$

$$\max\{d_j + \omega_i^b - p_j,\ \omega_i^b + u_j\} > Q,\ \text{if } p_j < \omega_i^b \tag{8c}$$

$$\xi_i^b + u_j + \max\{0,\ p_j - \omega_i^b\} > Q \tag{8d}$$

Backward label extension. The initial backward label originated from $o' \in D'$ is defined as $l_{o'}^b = \{\backslash,\ 0,\ l_{o'},\ 0,\ 0,\ 0,\ \{0,...,0\}_{S\in\Lambda},\ \varnothing,\ \varnothing\}$. If none of the conditions (8a) - (8d) satisfies, a new label $l_j^b = \{\tau_j^b,\ c_j^b,\ t_j^b,\ \rho_j^b,\ \omega_j^b,\ \xi_j^b,\ \{\theta_{Sj}^b\}_{S\in\Lambda},\ \mathrm{X}_j^b,\ U_j^b\}$ can be generated through the propagation of label l_i^b along arc $(j,\ i) \in A, j \in R \cup C \setminus U_i^b$, where

$$\tau_j^b = l_i^b \tag{9a}$$

$$c_j^b = c_i^b + \bar{c}_{ji} - \sum_{j\in S,\ S\in\Lambda,\ \theta_{Si}^b=1} \vartheta_S \tag{9b}$$

$$t_j^b = \min\{t_i^b - t_{ji},\ l_j\} \tag{9c}$$

$$\rho_j^b = \rho_i^b + d_{ji} \tag{9d}$$

$$\theta_{Sj}^b = \begin{cases} \theta_{Si}^b + 1, & \text{if } j \in S \\ \theta_{Si}^b, & \text{otherwise} \end{cases},\ \forall S \in \Lambda \tag{9e}$$

$$\mathrm{X}_j^b = \mathrm{X}_i^b \cup \{j\} \tag{9f}$$

$$U_j^b = U_i^b \cup \aleph_j^b \cup \{j\} \tag{9g}$$

$$\omega_j^b = d_j + \max\{0,\ \omega_i^b - p_j\} \tag{9h}$$

$$\xi_j^b = \max\{\max\{p_j,\ \omega_i^b\} + u_j,\ \xi_i^b + u_j + \max\{0,\ p_j - \omega_i^b\}\} \tag{9i}$$

where $\aleph_j^b$ is the subset of the vertices in $(R \cup C) \setminus U_i^b$ that are unreachable from the partial route

associated with label l_j^b. Extension conditions (9a) - (9g) are straightforward. The explanation for conditions (9h) and (9i) is as follows: When $p_j \geq \omega_i^b$, it indicates that the saleable products picked up from vertex j are sufficient to meet the delivery demand for saleable products of the vertices along the partial route r_i^b of the label l_i^b. Therefore, the saleable products loaded on the vehicle when arriving at vertex j only need to meet the delivery demand for saleable products of vertex j, which gives $\omega_j^b = d_j$. Meanwhile, $p_j - \omega_i^b$ represents the additional units of saleable products loaded on the vehicle along r_i^b. Thus, the maximum load of the vehicle along r_i^b is $\xi_i^b + p_j - \omega_i^b + u_j$, and the total load on the vehicle when it leaves vertex j is $p_j + u_j$. Hence, the condition $\xi_j^b = \max\{p_j + u_j, \xi_i^b + u_j + p_j - \omega_i^b\}$ holds. When $p_j < \omega_i^b$, it indicates that the saleable products picked up from vertex j are insufficient to meet the delivery demand for saleable products of the vertices along the partial route r_i^b. In this case, the saleable products arriving at vertex j satisfies $\omega_j^b = d_j + \omega_i^b - p_j$, where d_j units are used to meet the delivery demand of vertex j and $\omega_i^b - p_j$ units of saleable products transported by the vehicle to vertex j are delivered to vertices along r_i^b. This shows that no additional saleable products, except for ω_i^b units need to be transported to the predecessor vertex of l_i^b. The maximum load on the r_i^b is $\xi_i^b + u_j$, and the total load on the vehicle when vehicle leaves vertex j is $\omega_i^b + u_j$. Hence, the condition $\xi_j^b = \max\{\omega_i^b + u_j, \xi_i^b + u_j\}$ holds.

Backward dominance rules. Similarly, we propose novel effective backward dominance rules to ensure the optimality of LMP, thereby discarding as many unpromising columns as possible.

For any two backward labels $l_i^b = \{\tau_i^b, c_i^b, t_i^b, \rho_i^b, \omega_i^b, \xi_i^b, \{\theta_{Si}^b\}_{S\in\Lambda}, \mathrm{X}_i^b, U_i^b\}$ and $l_i^{b'} = \{\tau_i^{b'}, c_i^{b'}, t_i^{b'}, \rho_i^{b'}, \omega_i^{b'}, \xi_i^{b'}, \{\theta_{Si}^{b'}\}_{S\in\Lambda}, \mathrm{X}_i^{b'}, U_i^{b'}\}$, the former dominates the latter if (7a) - (7b) and (7g) - (7h) are satisfied, together with conditions

$$t_i^b \geq t_i^{b'} \tag{10a}$$

$$\omega_i^b \leq \omega_i^{b'} \tag{10b}$$

$$\xi_i^b \leq \xi_i^{b'} \tag{10c}$$

$$\xi_i^b - \omega_i^b \leq \xi_i^{b'} - \omega_i^{b'} \tag{10d}$$

Proposition 2. Backward dominance rules (10a) - (10d) are valid.

Proof. See **Appendix B.2**.

4.4.3. Merging forward and backward labels

During the generation of forward and backward labels, all forward and backward labels propagating to the same vertex must be merged to form complete routes. Complete routes with negative reduced cost are then added to the RLMP, while others are discarded. Given that a forward label l_i^f and a backward label l_i^b that have latest visited vertex i, denote by $l_{pre}^b = \{\tau_{pre}^b, c_{pre}^b, t_{pre}^b, \rho_{pre}^b, \omega_{pre}^b, \xi_{pre}^b, \{\theta_{S\,pre}^b\}_{S\in\Lambda}, \mathrm{X}_{pre}^b, U_{pre}^b\}$ the predecessor label of l_i^b. And then l_i^f and l_i^b can be merged if all the following conditions are satisfied.

$$t_i^f \leq t_i^b \tag{11a}$$

$$\rho_i^f + \rho_i^b \leq L \tag{11b}$$

$$\mathrm{X}_i^f \cap \mathrm{X}_i^b = \{i\} \tag{11c}$$

$$\max\{\alpha_i^f + \beta_i^f,\ \alpha_i^f + \beta_i^f + \xi_{pre}^b - \omega_{pre}^b\} \le Q,\ \text{if } \omega_i^b = d_i \tag{11d}$$

$$\alpha_i^f + \varphi_i^f \ge \omega_{pre}^b \ \wedge\ \xi_i^b - u_i + \max\{0,\ \alpha_i^f - \omega_{pre}^b\} + \beta_i^f \le Q,\ \text{if } \omega_i^b > d_i \tag{11e}$$

Condition (11a) ensures the time feasibility of the complete route. Condition (11b) indicates that the accumulated travel mileage cannot exceed L. Condition (11c) respects that l_i^f and l_i^b cannot serve the same vertex, except for vertex i. Condition (11d) and (11e) guarantee that the delivery demand for saleable products of each vertex along the l_i^b can be totally satisfied and the loads carried by the vehicle along the complete route can never exceed the capacity limitation.

Relying on the merge at vertex i, the reduced cost $\bar{c}_r$ of the complete route r in the concatenation is calculated by

$$\bar{c}_r = c_i^f + c_i^b + \gamma - \sum_{S\in\Lambda} \zeta_S \vartheta_S \tag{12}$$

where $\zeta_S = \begin{cases} 1, & \text{if } \theta_{Si}^f = 1 \ \wedge\ \theta_{Si}^b = 1 \ \wedge\ i \notin S \\ -1, & \text{if } \theta_{Si}^f = 2 \ \wedge\ \theta_{Si}^b = 2 \ \wedge\ i \in S,\ \forall S \in \Lambda \\ 0, & \text{otherwise} \end{cases}$ is aimed at refining the number of visiting vertices in $S \in \Lambda$ along the complete route.

The overview of the bidirectional labeling algorithm described in **Algorithm 2, Algorithm C.1** and **Algorithm C.2** (cf. **Appendix C**). UL_i^f and TL_i^f are the set containing all unprocessed and processed forward labels that propagating to vertex i, respectively. Similarly, UL_i^b and TL_i^b are the corresponding sets of backward labels. L_u is a set aiming at storing labels. LC is the set of complete routes with negative reduced costs. And let n_c and $n_{\max}$ be the size of LC and a constant representing the controllable quantity of complete routes with negative reduced cost that need to be found at each iteration of column generation, respectively.

4.5. Acceleration Strategies

4.5.1. Time windows adjustments

We adopt the method introduced by Desrochers et al. (1992) to tighten the time windows of retail stores and customers based on four conditions: minimal arrival time from predecessors, minimal arrival time to successors, maximal departure time from predecessors, and maximal departure time to successors. At each vertex, these conditions are sequentially applied to eliminate arcs that violate time window constraints, rendering them infeasible. This process is repeated iteratively until no further tightening of time windows is possible at any vertex. This step is performed prior to executing the BPC algorithm.

4.5.2. Two-cycle Pricing subproblem

The pricing subproblem aims to find an elementary route that visits each vertex no more than once, which is essentially an ESPPRC. The ESPPRC is proven to be NP hard. Due to the high time complexity of solving it, we relax the elementary condition and adopt the shortest path problem with resource constraints (SPPRC) with two-cycle elimination, first introduced by Houck et al. (1980), as the pricing subproblem. This relaxation accelerates the process of solving the SP but changes the structure of the SP above. For simplicity, we only explain the changes in forward labeling and label concatenation, as the

analysis for backward labeling is analogous.

Given that vertices in $R \cup C$ is allowed to be visited multiple times, the component X_i^f is discarded, and U_i^f represents the set of unreachable vertices of label l_i^f. Therefore, each vertex visited along the partial route of l_i^f cannot be added to U_j^f, unless it is unreachable. Furthermore, in the two-cycle SP, a two-cycle "$v-u-v$" is not allowed in any partial route, thereby preventing propagation to its predecessor vertex. The parameter θ_{Si}^f denotes the number of visiting vertices in $S \in \Lambda$, which can be any non-negative integer, not necessarily no more than three. Thus, the new reduced cost c_j^f can be computed as Eq. (13a) and the dominance rule (9f) is refined as condition (13b).

$$c_j^f = c_i^f + \bar{c}_{ij} - \sum_{j \in S,\, S \in \Lambda,\, \theta_{Si}^f \bmod 2 = 1} \vartheta_S \tag{13a}$$

$$\theta_{Si}^f \bmod 2 \le \theta_{Si}^{f'} \bmod 2,\ \forall S \in \Lambda \tag{13b}$$

The dominance rule for ESPPRC described above is very strict on the relationship between the potentially accessible vertices sets of two labels, as defined by condition (7a). Kohl (1995) and Jesper (1999) proposed an enhanced dominance rule for the two-cycle-SP, where introduces three types of dominant labels. This enhancement eliminates the need to consider condition (7a). In our problem, we adopt the enhanced dominance rule, which provides improved computational efficiency. For further details, refer to Irnich and Villeneuve (2006).

When merging two labels at the same vertex i, condition (11c) is no longer obligatory. Considering the two-cycle elimination, the predecessor vertices of the forward label and backward label must differ. Furthermore, the reduced cost of complete route $\bar{c}_r$ is refined by updating the formula for ζ_S as follows

$$\zeta_S = \begin{cases} 1, & \text{if } \theta_{Si}^f \bmod 2 = 1 \wedge \theta_{Si}^b \bmod 2 = 1 \wedge i \notin S \\ -1, & \text{if } \theta_{Si}^f \bmod 2 = 0 \wedge \theta_{Si}^b \bmod 2 = 0 \wedge i \in S,\ \forall S \in \Lambda \\ 0, & \text{otherwise} \end{cases}$$

4.5.3. Recovering a feasible solution

After solving a branch node in the branch-and-bound tree using the CG algorithm, the pricing problem generates a set of routes. If the LMP's optimal solution of a branch is fractional and the gap between the current upper bound and lower bound exceeds 0.02 (i.e., Condition 2), we attempt to solve the MP of the branch node with the integrality constraints (2c) using CPLEX before branching. This yields a feasible integer solution, which serves as a new upper bound (UB) if the feasible integer solution is less than the current UB. The improved UB enables us to cut more branches, thereby accelerating the convergence of the algorithm.

4.6. Branching strategies

We explore the branch-and-bound tree through depth-first search strategy introduced by Desaulniers et al. (2014). When the optimal objective of LMP is fractional, branching is performed. At each fractional branch node of the search tree, the following branching strategies are evaluated sequentially until two branches are successfully generated.

(i) Branching on the number of vehicles. If the total number of utilized vehicles in the current

solution of LMP is fractional, we implement two branches $\sum_{r\in\Re}\theta_r \le \lfloor \sum_{r\in\Re}\theta_r \rfloor$ and $\sum_{r\in\Re}\theta_r \le \lceil \sum_{r\in\Re}\theta_r \rceil$, and these branching constraints are incorporated into the RLMP. Let μ be dual variable associated with the branch constraint, which is eventually passed to the SP. The reduced cost (Eq. 12) of complete route needs to be reformed as $\bar{c}_r = c_i^f + c_i^b + \gamma - \sum_{S\in\Lambda}\zeta_S\vartheta_S - \mu$.

(ii) Branching on the outflow of a set Γ with two vertices in $R\cup C$. We choose a set $\Gamma \subset R\cup C$ whose outflow $F_\Gamma = \sum_{r\in\Re}\sum_{i\in\Gamma}\sum_{j\in(R\cup C\cup D')\setminus\Gamma} x_{ijr}\theta_r$ is closest to 1.5, and create two branches $F_\Gamma \le 1$ and $F_\Gamma \ge 2$. Assume that the set Φ is composed of the sets associated with all the identified inequalities, let ε_Γ be the dual value of the corresponding inequality of $\Gamma\in\Phi$. Therefore, the reduced cost of the forward label needs to be reformed as $c_j^f = c_i^f + \bar{c}_{ij} - \sum_{j\in S,\, S\in\Lambda,\, \theta_S^f=1}\vartheta_S - \sum_{i\in\Gamma,\, j\notin\Gamma,\, \Gamma\in\Phi}\varepsilon_\Gamma$.

(iii) Branching on an individual arc variable f_{ij}, which presents the flow of arc (i,j). We select an arc (i,j) with f_{ij} closest to 0.5, and create two branches $f_{ij}=0$ and $f_{ij}=1$. $f_{ij}=0$ indicates that the arc (i,j) is forbidden in any route. In this case, we remove (i,j) from the physical network and discard all the routes containing (i,j) from the columns set. $f_{ij}=1$ implies that vertex i can only propagate to vertex j. If $i,j\in R\cup C$, we need to eliminate all the ingoing arcs ending at vertex j and all the outgoing arcs starting from vertex i except for (i,j) in the physical network. Additionally, we discard any columns that do not traverse arc (i,j) but visit vertex i or j. If $i\in D$ and $j\in R\cup C$, we need to eliminate all the ingoing arcs ending at vertex j except for (i,j) in the physical network. And we discard any columns that do not traverse arc (i,j) but visit vertex j. If $i\in R\cup C$ and $j\in D$, we need to eliminate all the outgoing arcs starting from vertex i except for (i,j) in the physical network. And we discard any columns that do not traverse arc (i,j) but visit vertex i.

5. Computational study

5.1. Experimental setup

The algorithm is coded in Java, and the RLMP is implemented using CPLEX 22.1.1. All experiments are carried out on an HP personal computer equipped with an Intel Core i5-1340 1.9GHz CPU, 32 GB of RAM, and running the Windows 11 operating system. Computational times are recorded in CPU seconds, with a time limit of 7200 seconds set for each run on each instance.

5.2. Instance generation

To the best of our knowledge, no existing benchmark instances perfectly align with our problem. Therefore, we generate new instances based on Solomon's benchmark datasets (Solomon, 1987). The Solomon instances consist of 100 customer vertices and 1 warehouse vertex. We selected 100 customer vertices from the C101–C109 and R101–R112 instances to represent retail store and customer vertices in our large-scale instances. Additionally, to generate small- and medium-scale instances, we select the first 25 and 50 customer vertices from each instance, respectively. The coordinates, time windows, and service time information remain consistent with those provided in Solomon's benchmark datasets.

Based on product demand, retail stores are categorized into five types: 1) $d>0$, $p=0$, $u=0$; 2) $d=0$, $p>0$, $u=0$;3) $d=0$, $p=0$, $u>0$; 4) $d=0$, $p>0$, $u>0$ and 5) $d>0$, $p=0$, $u>0$. Customers are categorized into three types: 1) $d>0$, $p=0$, $u=0$; 2) $d=0$, $p>0$, $u=0$;3) $d=0$,

$p = 0$, $u > 0$, which are defined as C_A, C_B and C_C respectively. The delivery and recovery demands for saleable products, as well as the pick-up demand for unsaleable products at retail stores, are generated as random numbers in the range of [30, 50]. For online customers, the demands are random numbers in the range of [7, 13].

For the selection of retail store and online customer vertices, we apply the k-medoids method introduced by Kaufman and Rousseeuw (1987). For example, in the large-scale instance, we cluster the 100 vertices into 10 groups and select the center of each cluster as retail stores, while the remaining vertices are designated as customer vertices. To determine warehouse locations, we use the k-means method (MacQueen, 1967). For instance, by clustering the 100 vertices into 8 clusters, the centroids of these 8 clusters are chosen as warehouse locations. Finally, the selected 8 warehouses replace the single warehouse in Solomon's benchmark datasets.

Based on the number of warehouses, retail stores and customers, we generate baseline instances of different sizes and name them 'X_dm_cn', where 'X' is the original Solomon's instance name (e.g., C101, R101), 'm' denotes the number of warehouses, 'n' represents the number of online customers and retail stores. For the large-scale instances, generated from Solomon's instances with 100 vertices, there are 10 retail stores, with each type having 2 stores. The remaining vertices are customers, consisting of 35 C_A, 35 C_B, and 20 C_C. We consider 6, 7, and 8 warehouses, resulting in a total of 63 instances. For the medium-scale instances, generated from Solomon's instances with 50 vertices, there are 5 retail stores, with each type having 1 store. The remaining vertices are customers, including 18 C_A, 17 Type B C_B, and 10 C_C. We consider 5 warehouses, resulting in a total of 21 instances. For the small-scale instances, generated from Solomon instances with 25 vertices, there are 5 retail stores, with each type having 1 store. The remaining vertices are customers, including 8 C_A, 8 C_B, and 4 C_C. We consider 2 warehouses, resulting in a total of 21 instances. In total, we generate 105 instances. Finally, other parameters used in the HMM-UPDPTW formulations (i.e., in Eqs. (1a) - (1u)) are also specified. The capacity, range limitation and speed of each vehicle are set to 150 units, 80 km and 60km/h, respectively. And the unit travel cost is set to 0.5 dollars per kilometer, and each used vehicle incurs a fixed cost of 25 dollars.

5.3. Performance of the Branch-and-Price-and-Cut algorithm

5.3.1. Effectiveness of the SR inequalities

We evaluate the effectiveness of the SR inequalities (SRCs) by comparing the proposed BPC algorithm and the algorithm without SRCs, denoted as BP algorithm. We conduct experiments on 21 large-scale instances with 8 warehouses. The results are summarized in **Table 3**. The "Instance" column represents the name of each instance. The "Gap (%)" column displays the integrity gap between the upper bound obtained by our BPC algorithm (UB) and the corresponding optimal value of the LMP (LB) in percentage, which is calculated as $100 \times (UB - LB) / UB$. The "rObj" column indicates the optimal value of the LMP at the root branch node. The "rCuts", "Cuts" and "Nodes" columns record the number of SRCs added to the root branch node, the number of SRCs added during the whole computational process and the number of branch nodes explored on the branch-and-bound tree, respectively. The "Time (s)" column

reports the total computational time in seconds. If an instance is not solved within the time limit, the "Time (s)" column includes a "\" to indicate this.

Table 3
The effectiveness of SRCs.

Instance	Without SRCs				With SRCs					
	Gap (%)	rObj	Nodes	Time (s)	Gap (%)	rObj	rCuts	Cuts	Nodes	Time (s)
C101_d8_c100	0.00	478.50	1	3.82	0.00	478.50	0	0	1	3.72
C102_d8_c100	0.00	474.80	5	**42.06**	**0.00**	475.20	3	3	1	**36.94**
C103_d8_c100	0.00	462.33	67	**1195.00**	**0.00**	462.94	3	16	7	**514.82**
C104_d8_c100	1.64	453.94	62	\	1.52	454.47	3	15	9	\
C105_d8_c100	0.00	477.40	1	4.41	0.00	477.40	0	0	1	4.56
C106_d8_c100	0.00	477.10	1	7.19	0.00	477.10	0	0	1	7.45
C107_d8_c100	0.00	476.90	1	7.49	0.00	476.90	0	0	1	7.36
C108_d8_c100	0.00	475.40	1	14.58	0.00	475.40	0	0	1	14.55
C109_d8_c100	2.48	463.63	290	\	**0.00**	465.41	3	29	17	**749.57**
R101_d8_c100	0.00	1004.05	11	**3.21**	**0.00**	1005.10	3	3	1	**2.29**
R102_d8_c100	0.00	904.15	213	**187.74**	**0.00**	905.23	3	17	11	**37.45**
R103_d8_c100	0.00	770.40	21	**48.50**	**0.00**	770.83	3	3	3	**27.72**
R104_d8_c100	5.21	603.16	90	\	**3.91**	604.38	3	17	9	\
R105_d8_c100	0.00	822.40	381	**379.59**	**0.00**	824.53	3	52	55	**130.33**
R106_d8_c100	2.33	755.47	816	\	**1.65**	756.49	3	173	261	\
R107_d8_c100	3.01	665.62	163	\	**1.73**	666.69	3	24	15	\
R108_d8_c100	3.87	585.55	31	\	**3.18**	588.20	3	9	3	\
R109_d8_c100	3.19	714.37	784	\	**2.73**	715.35	3	56	67	\
R110_d8_c100	3.73	665.67	306	\	**3.36**	668.30	3	32	24	\
R111_d8_c100	2.92	667.16	201	\	**2.48**	668.69	3	24	16	\
R112_d8_c100	4.37	600.18	6	\	**4.10**	601.85	3	3	1	\

The results clearly highlight the effectiveness of the SR inequalities. Specifically, our BPC algorithm finds the optimal solution for one additional instance within the time limit compared to the BP algorithm and is able to exactly solve 12 instances. For example, the BP algorithm fails to optimally solve C109_8_100 within the time limit of 7200 seconds, whereas the BPC algorithm achieves the optimal solution in only 749.57 seconds. To better illustrate the impact of the SRCs, we only analyze the instances where at least one SR inequality is added by the BPC algorithm, referred as Set*[7]. The introduction of SRCs reduces the integrity gap for the instances[8] n Set* that cannot be optimally solved by the BP algorithm. For example, the average integrity gap is 3.27% for the BP algorithm, but decreases to 2.47% for the BPC algorithm, as shown in column "Gap (%)". Furthermore, from columns "rObj" and "rCuts", we observe that the optimal value of the LMP on the root branch node for Set* increases by 0.2% on average when an average of 3 SRCs are added to the root branch node. It is well known that lifted lower bounds allow for pruning more branch nodes. This is supported by column "Nodes", where the number of branch nodes explored by the BPC algorithm is 87.77% fewer on average than that of the BP algorithm for Set*. This strongly indicates that the SRCs significantly accelerate the solution process. Finally, columns "Time (s)" and "Cuts" reveal that, for the instances[9] in Set* that are exactly solved by both algorithms, the computational time is reduced by 47.72% on average, due to the introduction of an average of 16 SRCs in total.

[7] Set*: 102_d8_c100 - C104_d8_c100, C109_d8_c100, R101_d8_c100 - R112_d8_c100.
[8] These instances conclude C104_d8_c100, C109_d8_c100, R104_d8_c100, R106_d8_c100 - R112_d8_c100.
[9] These instances conclude C102_d8_c100, C103_d8_c100, R101_d8_c100 - R103_d8_c100, R105_d8_c100.

5.3.2. Effectiveness of acceleration strategies

The incorporation of bidirectional labeling and two-cycle elimination strategies are designed to enhance the efficiency of the column generation process. To evaluate the effectiveness of these acceleration techniques, we compare four versions of the algorithm based on 21 large-scale instances with 8 warehouses: (i) BS – the baseline version without either acceleration strategy, (ii) AT – with two-cycle elimination but without bidirectional labeling, (iii) AB – with bidirectional labeling but without two-cycle elimination, and (iv) ALL – incorporating both acceleration strategies. The experimental results, presented in **Table 4**, clearly demonstrate that both bidirectional labeling and two-cycle elimination significantly improve the algorithm's performance and complement each other. These strategies enable the algorithm to optimally solve more instances, narrow the integrity gap for unsolved instances, and reduce the average computational time for solved instances. Comparative analysis shows that while two-cycle elimination demonstrates superior individual efficacy, their combined implementation produces the most significant improvement in computational efficiency.

Table 4
The effectiveness of acceleration strategies.

Instance	Baseline		AT		AB		ALL	
	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	gap (%)
C101_d8_c100	76.88	0.00	5.09	0.00	16.88	0.00	**3.72**	**0.00**
C102_d8_c100	\	100.00	860.00	0.00	565.36	0.00	**36.94**	**0.00**
C103_d8_c100	\	100.00	\	0.52	6528.16	0	**514.82**	**0.00**
C104_d8_c100	\	100.00	\	100	\	100	\	**1.52**
C105_d8_c100	421.07	0.00	7.83	0.00	42.30	0.00	**4.56**	**0.00**
C106_d8_c100	975.03	0.00	11.10	0.00	51.45	0.00	**7.45**	**0.00**
C107_d8_c100	1669.71	0.00	10.60	0.00	57.35	0.00	**7.36**	**0.00**
C108_d8_c100	\	100.00	27.60	0.00	170.81	0.00	**14.55**	**0.00**
C109_d8_c100	\	100.00	\	2.35	1427.73	0.00	**749.57**	**0.00**
R101_d8_c100	52.61	0.00	4.40	0.00	11.92	0.00	**2.29**	**0.00**
R102_d8_c100	\	100.00	347.95	0.00	231.50	0.00	**37.45**	**0.00**
R103_d8_c100	\	100.00	179.04	0.00	173.91	0.00	**27.72**	**0.00**
R104_d8_c100	\	100.00	\	100.00	\	100.00	\	**3.91**
R105_d8_c100	\	100.00	2689.73	0.00	3763.43	0.00	**130.33**	**0.00**
R106_d8_c100	\	100.00	\	2.59	\	1.67	\	**1.65**
R107_d8_c100	\	100.00	\	2.66	\	100.00	\	**1.73**
R108_d8_c100	\	100.00	\	100	\	100.00	\	**3.18**
R109_d8_c100	\	100.00	\	3.42	\	100.00	\	**2.73**
R110_d8_c100	\	100.00	\	5.17	\	100.00	\	**3.36**
R111_d8_c100	\	100.00	\	3.97	\	100.00	\	**2.48**
R112_d8_c100	\	100.00	\	100.00	\	100.00	\	**4.10**

Specifically, the results show that the BPC algorithm, with both acceleration strategies, can exactly solve 12 out of 21 instances. In contrast, applying only two-cycle elimination, only bidirectional labeling and no acceleration strategy optimally solve 10, 12 and 5 instances, respectively, with 4, 8, and 16 instances even failing to find a feasible solution. Furthermore, for the instances[10] that are not exactly solved in baseline scenario (i.e., BS), the addition of only two-cycle elimination, only bidirectional labeling and both strategies can contribute to an average reduction of the integrity gap by 73.71%, 49.90% and

[10] These instances conclude C102_d8_c100 - C104_d8_c100, C108_d8_c100, C109_d8_c100, R102_d8_c100 - R112_d8_c100.

98.46%, respectively. A more detailed examination of the instances[11] optimally solved across all four scenarios reveals that the average computational times decrease by 98.78%, 93.05% and 99.21%, respectively, compared to the baseline results when only two-cycle elimination, only bidirectional labeling, and both strategies are applied. This dramatic reduction is attributed to the fact that the majority of the computation time in the CG algorithm is spent on solving the pricing problem using the labeling algorithm.

5.3.3. Computational results of small-, medium-, and large-scale instances

The effectiveness of the developed BPC algorithm is evaluated through a direct comparison with the arc-based formulation solved using CPLEX, applied to both small-scale and medium-scale instances, as shown in **Table 5** and **6**. It presents several key performance metrics, including "Times (s)", "Gap (%)", the final upper bound achieved either when the instance is optimally solved or when the procedure is forcefully terminated due to time limit (Obj) and the integrity gap on the root branch node (rGap (%)), respectively.

Table 5
Performance of the BPC compared with CPLEX on the small-scale instances.

Instance	CPLEX			BPC			
Small-scale	Times (s)	Obj	Gap (%)	Times (s)	Obj	Gap (%)	rGap (%)
C101_d2_c25	0.08	136.9	0.00	**0.16**	136.9	**0**	0
C102_d2_c25	85.20	135.1	0.00	**3.35**	135.1	**0**	0
C103_d2_c25	\	135.1	28.42	**181.83**	135.1	**0**	2.92
C104_d2_c25	\	133.6	43.43	\	133.6	**3.64**	3.64
C105_d2_c25	0.15	136.9	0.00	**0.18**	136.9	**0**	0
C106_d2_c25	0.08	136.9	0.00	**0.15**	136.9	**0**	0
C107_d2_c25	0.16	136.9	0.00	**0.19**	136.9	**0**	0
C108_d2_c25	2.33	136.9	0.00	**0.85**	136.9	**0**	0
C109_d2_c25	1154.85	136.1	0.00	**288.32**	136.1	**0**	17.47
R101_d2_c25	0.11	459.6	0.00	**0.06**	459.6	**0**	0
R102_d2_c25	1869.12	430.7	0.00	**0.69**	430.7	**0**	1.73
R103_d2_c25	\	371.2	25.42	**0.69**	371.2	**0**	0
R104_d2_c25	\	347.1	28.88	**0.34**	347.1	**0**	0
R105_d2_c25	1.38	412	0.00	**0.53**	412	**0**	0.74
R106_d2_c25	\	390.8	13.01	**23.34**	390.8	**0**	4.21
R107_d2_c25	\	358.3	29.16	**0.91**	358.3	**0**	0.18
R108_d2_c25	\	347	37.92	**0.77**	347	**0**	1.87
R109_d2_c25	104.75	361.7	0.00	**1.02**	361.7	**0**	0.26
R110_d2_c25	\	353.1	26.52	**12.58**	349.7	**0**	3.52
R111_d2_c25	\	368.1	30.11	**1.60**	359.3	**0**	0.05
R112_d2_c25	\	343.9	35.47	**183.07**	343.9	**0**	7.36

Notably, CPLEX optimally solves only 19 out of 42 instances, while the BPC algorithm successfully solves 40 out of 42 instances. The instances solved optimally by CPLEX are generally less complex, as evidenced by the fact that the BPC algorithm can solve them at the root node, which can be inferred from column "Gap_{root} (%)". And these instances primarily belong to the "C" series, characterized by clustered geographical distributions and narrow time windows. Conversely, CPLEX performs poorly on "R" series instances, which feature random geographical distributions, solving only 6 out of 24 instances to optimality, all with narrow time windows. Specifically, only the instances C104_d2_c25 and C104_d5_c50 could not be solved optimally by the BPC algorithm within the time limit, with minor integrity gaps of 3.64% and

[11] These instances conclude C101_d8_c100, C105_d8_c100 - C107_d8_c100, R101_d8_c100.

2.21%, respectively. In contrast, the average integrity gap for instances unsolved by CPLEX within the time limit is significantly larger, reaching 27.12% for small-scale instances and 39.68% for medium-scale instances.

Table 6
Performance of the BPC compared with CPLEX on the medium-scale instances.

Instance	CPLEX			BPC			
Medium-scale	Times (s)	Obj	Gap (%)	Times (s)	Obj	Gap (%)	rGap (%)
C101_d5_c50	0.22	220.7	0.00	**0.91**	220.7	**0**	0
C102_d5_c50	673.97	219.9	0.00	**8.45**	219.9	**0**	0
C103_d5_c50	\	216.1	30.55	**36.75**	216.1	**0**	0.12
C104_d5_c50	\	214.4	47.27	\	214.2	2.21	2.21
C105_d5_c50	0.31	219.3	0.00	**0.96**	219.3	**0**	0
C106_d5_c50	0.25	219.6	0.00	**0.85**	219.6	**0**	0
C107_d5_c50	0.31	219.3	0.00	**1.32**	219.3	**0**	0
C108_d5_c50	28.51	218.7	0.00	**3.12**	218.7	**0**	0
C109_d5_c50	\	218.3	28.93	**55.71**	218.3	**0**	1.76
R101_d5_c50	0.85	669.8	0.00	**4.81**	669.8	**0**	0.45
R102_d5_c50	\	582.5	28.74	**1.92**	580.1	**0**	0.14
R103_d5_c50	\	513.5	40.12	**79.61**	502.1	**0**	0.96
R104_d5_c50	\	451.4	40.48	**1668.82**	426.8	**0**	1.45
R105_d5_c50	123.38	577.4	0.00	**6.07**	577.4	**0**	0.93
R106_d5_c50	\	525.1	37.68	**36.14**	516.9	**0**	1.95
R107_d5_c50	\	500.1	45.65	**482.08**	475.3	**0**	2.87
R108_d5_c50	\	460.4	47.10	**122.01**	426.8	**0**	2.47
R109_d5_c50	\	537.5	28.77	**7.37**	516.6	**0**	1.29
R110_d5_c50	\	503.8	46.08	**262.24**	471.6	**0**	3.24
R111_d5_c50	\	496.1	45.03	**15.87**	472.1	**0**	1.91
R112_d5_c50	\	463.9	49.40	**222.39**	421.5	**0**	4.28

Table 7
Performance of the BPC on the large-scale instances with 6, 7 and 8 warehouses.

Instance	6 warehouses		7 warehouses		8 warehouses	
	Time (s)	Gap (%)	Time (s)	gap (%)	Time (s)	Gap (%)
C101	3.02	**0.00**	3.66	**0.00**	3.72	**0.00**
C102	114.23	**0.00**	147.97	**0.00**	36.94	**0.00**
C103	1162.83	**0.00**	744.22	**0.00**	514.82	**0.00**
C104	\	**1.73**	\	**1.38**	\	**1.52**
C105	3.06	**0.00**	4.33	**0.00**	4.56	**0.00**
C106	4.75	**0.00**	6.56	**0.00**	7.45	**0.00**
C107	4.51	**0.00**	8.18	**0.00**	7.36	**0.00**
C108	15.43	**0.00**	15.53	**0.00**	14.55	**0.00**
C109	346.04	**0.00**	435.76	**0.00**	749.57	**0.00**
R101	9.87	**0.00**	2.37	**0.00**	2.29	**0.00**
R102	8.80	**0.00**	11.91	**0.00**	37.45	**0.00**
R103	309.29	**0.00**	36.47	**0.00**	27.72	**0.00**
R104	\	**4.61**	\	**4.40**	\	**3.91**
R105	\	**1.90**	\	**1.89**	130.33	**0.00**
R106	\	**2.24**	\	**1.86**	\	**1.65**
R107	\	**3.19**	\	**3.67**	\	**1.73**
R108	\	**3.24**	\	**4.24**	\	**3.18**
R109	\	**2.71**	\	**3.53**	\	**2.73**
R110	\	**4.25**	\	**4.23**	\	**3.36**
R111	\	**3.05**	\	**3.03**	\	**2.48**
R112	\	**4.40**	\	**4.67**	\	**4.10**

Furthermore, the BPC algorithm demonstrates superior computational efficiency compared to CPLEX. It solves the vast majority of small- and medium-scale instances in significantly less time. Specifically, for the 21 small-scale instances, the BPC algorithm solves 12 instances within 1 second and 17 instances

within 25 seconds. For the 21 medium-scale instances, it solves 15 instances within 80 seconds and 19 instances within 500 seconds. Notably, for certain instances, the BPC algorithm reduces computational time by more than three orders of magnitude compared to CPLEX.

We also present a comparative analysis of the computational results of the BPC algorithm applied to large-scale instances with varying numbers of warehouses. **Table 7** summarizes the results for instances with 6, 7, and 8 warehouses, where "k" represents the fleet size. The BPC algorithm successfully identifies the optimal solution for 11 out of 21 instances with 6 warehouses, 11 out of 21 instances with 7 warehouses, and 12 out of 21 instances with 8 warehouses. For instances where the optimal solutions are obtained within the time limit, the average computational times are 180.17 seconds, 128.81 seconds, and 128.06 seconds, respectively. For instances where the optimal solution is not achieved, all gaps remain below 5%. Moreover, the results indicate that "R" series instances are more challenging to solve than "C" series instances. Additionally, instances with wider time windows are more difficult to solve than those with narrower time windows, primarily due to the larger search space.

5.4. Sensitivity analysis and Managerial insights

5.4.1. Impacts of redistribution varying from different vehicle capacity

We first analyze the benefits of redistribution for saleable products by comparing our transportation model, which allows redistribution, with a model that does not, under varying the vehicle capacity limitation. The vehicle capacity Q is varied within the range of $\{90, 100, 110, 120, 130, 140, 150, 160\}$. The instances selected for analysis include C101_d5_c50 - C103_d5_c50 and C105_d5_c50 - C109_d5_c50 of medium-scale instances. To enable a more effective comparison, we select instances that can be optimally solved across the entire capacity range in both transportation modes (i.e., C101_d5_c50, C105_d5_c50, and C106_d5_c50) and conduct a detailed analysis, as presented in **Fig. 4.**

As shown in **Fig. 4(a)**, as vehicle capacity increases, the average fixed cost (av.FixC)，the average transportation cost (av.TranC) and the average total cost (av.TolC) for both models initially decrease and then stabilize, with "av.TranC" experiencing a particularly notable decline. This can be attributed to the fact that increasing vehicle capacity allows each vehicle to serve more customers and stores, thereby expanding the solution space. In **Fig. 4(b)**, according to the index "av.fdLoad", which refers to average loads from warehouses, when redistribution is not allowed, all 270 units of saleable products must be sourced from the warehouses. However, when redistribution is allowed, only an average of 148 units are sourced from warehouses, with the remainder coming from customers and stores. Furthermore, **Figure 4(a)** demonstrates that our model, which allows redistribution, consistently achieves a lower "av.TolC" when vehicle capacity is less than 150 units, with an average reduction of 9.1%. For capacity greater than 150, the "av.TolC" values for both models converge. Notably, thresholds exist for both models: once the vehicle capacity reaches or exceeds a certain threshold, the optimal vehicle routes and "av.TolC" remain unchanged. This indicates that vehicle routing is no longer limited by capacity but instead influenced by other factors, such as time windows. Additionally, the threshold $Q=130$ for our model with redistribution occurs at a lower capacity than that of $Q=150$ for the model without redistribution. The observation highlights that redistribution enhances vehicle capacity utilization, reduces the quantity of products sourced

from the warehouses, and allows a smaller fleet with lower capacity to complete the service, thereby reducing costs and improving efficiency. Under the current parameter settings, a vehicle capacity of 130 units is sufficient to achieve the optimal reduction in total cost.

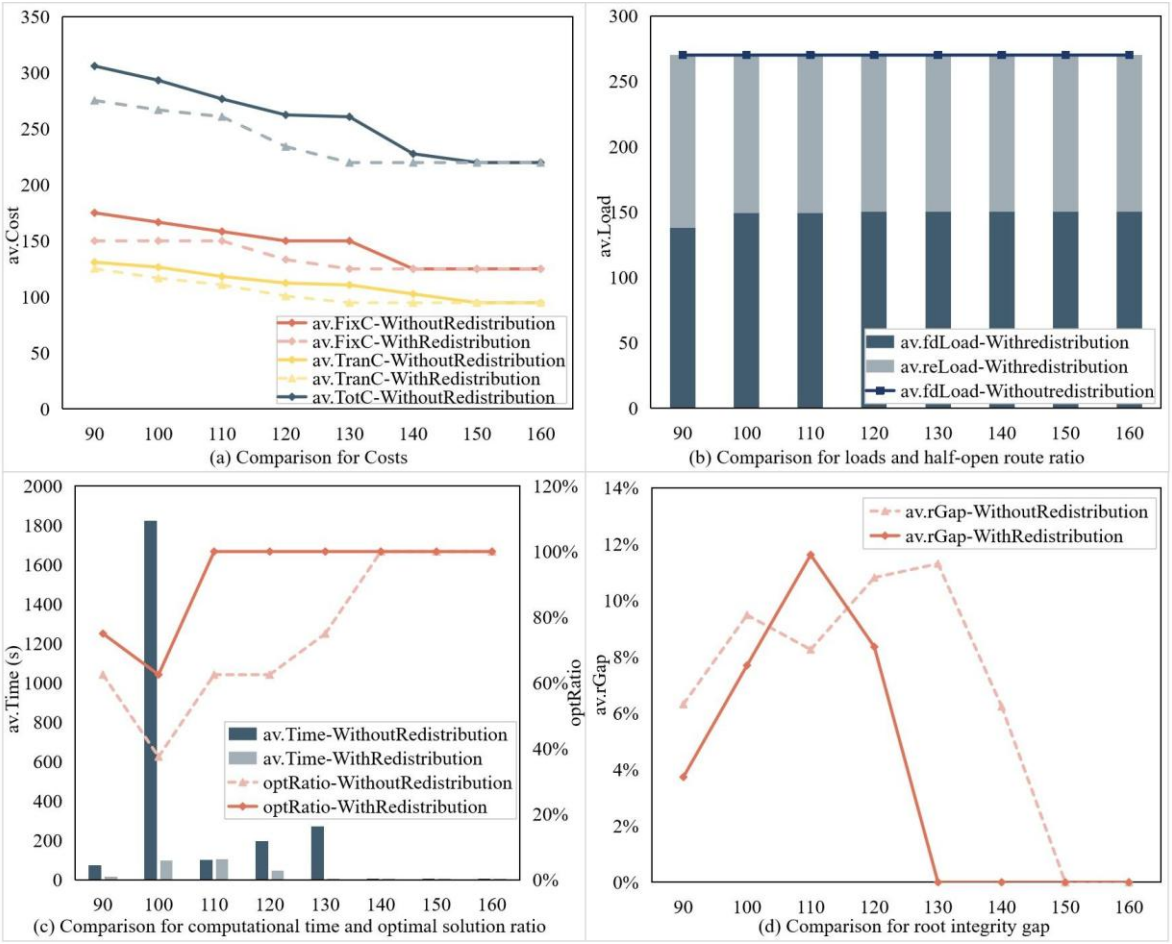

Fig. 4. The impact of redistribution for saleable products with different vehicle capacity limit

We further observe from **Fig. 4(b)** that as vehicle capacity increases, average loads for saleable products collected from customers or retail stores and redistributed (av.reLoad) initially decrease and then stabilize. This is because, when vehicle capacity is small, the capacity constraints (1p) and (1q) become more significant, leading to the selection of more circuitous routes with bigger "av.reLoad" values. This trade-off helps control fleet size. As capacity increases, these constraints become less restrictive, and more direct routes with smaller "av.reLoad" values are preferred. This variation reflects the balance between transportation cost and fixed vehicle cost, as well as the joint decision-making involved in route selection and delivery-pickup demand pairing. **Figure 4(c)** shows that when the vehicle capacity increases to a certain value, the average computational time (av.Time) becomes very small. For the eight selected instances, the optimal solution ratio (i.e., the proportion of instances solved optimally within the time limit (optRatio)) first decreases, then increases, and finally stabilizes. This phenomenon contrasts with the intuitive expectation that larger vehicle capacities, which expand the solution spaces, would make the problem more difficult to solve. Additionally, **Figure 4(d)** reveals that in both scenarios, the average root

integrity gaps (av.rGap) first increase and then decrease, indicating an initial increase in the number of branch nodes explored, followed by a decline. This suggests that computational time is influenced not only by column generation but also by branch-and-bound procedure.

5.4.2. Analysis on the customer demand heterogeneity

We consider three types of customers with heterogeneous demands. Due to the mechanism of redistribution for saleable products and recovery of unsaleable products, different customer combinations have a significant impact on vehicle routing and overall cost. In this section, we generate new instances based on the medium-sized instance C101_d5_c50 by adjusting the demand data of store and customer nodes. In the experiment, the vehicle capacity limitation is set to 100 units, and other parameters remain consistent with the baseline experiment.

Analysis on the delivery-to-pickup customer ratio for saleable products. We adjust the delivery-to-pickup customer ratio for saleable products (i.e., the ratio of customers with delivery demand to customers with pick-up demand for saleable products, ($ratio_{d-p}$)) to investigate the impact of delivery and recovery demands for saleable products on the logistics system. We create seven instance sets with different "$ratio_{d-p}$", where the variation in the number of customers of C_A, C_B, and C_C falls within the following ranges: {(0_40_5), (5_35_5), (10_30_5), (20_20_5), (30_10_5), (35_5_5), (40_0_5)}. Each instance set includes five sub-instances with the same "$ratio_{d-p}$" but different demand data. Therefore, a total of 35 distinct instances are generated. The average experimental results for each set are visualized in **Fig. 5(a)**, which shows the variation in average total costs ($av.\mathrm{Cos}t$) and average fleet size (av.k) under different "$ratio_{d-p}$". As "$ratio_{d-p}$" increases, "$av.\mathrm{Cos}t$" generally exhibits a trend of first dramatically decreasing and then increasing at a slower rate. It reaches its minimum when the $ratio_{d-p}$ approaches 1. This phenomenon can be attributed to the fact that when the number of customers with saleable product delivery demand is similar to that with saleable product pickup demand, vehicles can perform redistribution more frequently. Such a high frequency of redistribution improves vehicle capacity utilization and further reduces the number of vehicles, as demonstrated by the chart of "av.k". Specifically, "av.k" decreases when $ratio_{d-p} \le 1$. However, when $ratio_{d-p}$ exceeds 1, "k" remains unchanged and the increase in "$av.\mathrm{Cos}t$" becomes more gradual, with some results even deviating from the overall trend. This is mainly because with a larger $ratio_{d-p}$, more saleable products are delivered to customers, freeing up more space to collect both types of products, thereby enlarging the solution space to some extent. Therefore, when the ratio approaches 1, the balance between the delivery demand and the pick-up demand for saleable products leads to more efficient transportation, optimizing vehicle utilization and achieving the lowest total cost.

Analysis on the ratio of customers with pickup demand for unsaleable products. In addition to the demand for saleable products, the quantity of unsaleable product returns from customers also plays a great role.. Similar to the previous section, we generate six new instance sets, with different combinations of C_A, C_B and C_C customer quantities: { (20_20_5), (18_18_9), (15_15_15), (10_10_25), (5_5_35), (0_0_45) }. Each set includes five sub-instances with different demand data, resulting in a total of 30 distinct instances.

As shown in **Fig. 5(b),** it is noteworthy that as the number of customers with pickup demand for unsaleable products increases, the average total cost and the fleet size exhibit an upward trend. Since unsaleable products cannot be redelivered, more vehicles are required to meet the increased demand for product returns. This increase in fleet size contributes to the observed rise in "av.Cost".

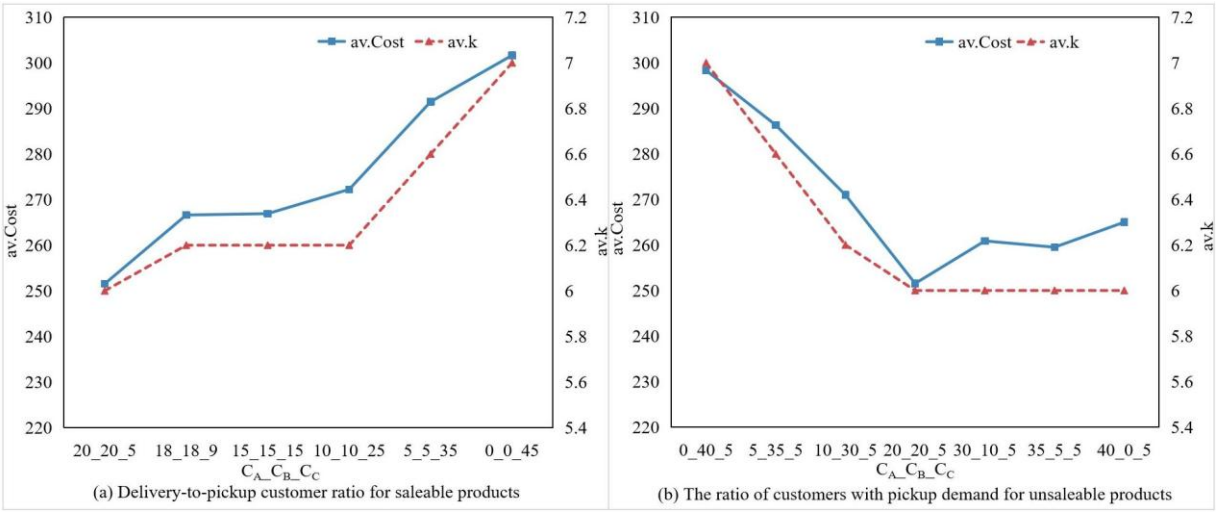

Fig. 5. The impact of the ratio of the quantities of different types of customers

5.4.3. Impacts of half-open routes

Experiments are conducted using the "R" series medium-scale instances to analyze the impacts of allowing half-open routes on routes selection. We compare our model (which allows half-open and closed routes, i.e., a hybrid model) with the closed-route model (i.e., a closed model). The experimental results are visualized in **Fig. 6**, which demonstrates the fixed cost (FixC), transportation cost (TranC), redistribution loads (reLoad), and half-open route ratio (oRatio) for each instance in both models.

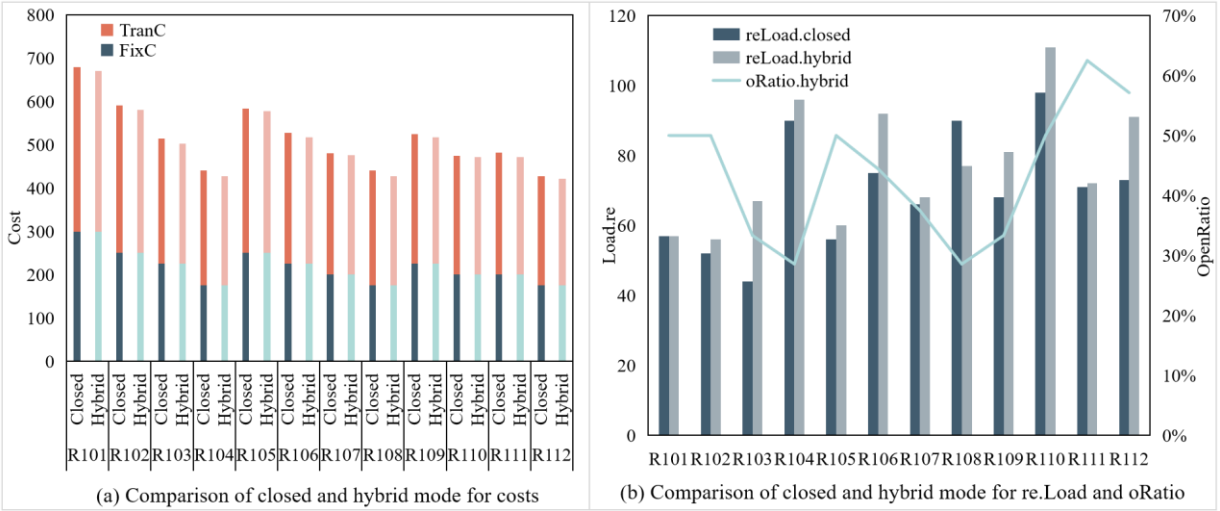

Fig. 6. The impacts of half-open routes.

From **Fig. 6(a),** it can be observed that, "TranC" for all instances in the hybrid model is lower than that of the closed model, with an average reduction of 3.11%. However, "FixC" remains unchanged, indicating that allowing half-open routes plays a significant role in reducing transportation cost. **Figure 6(b)** shows that in the hybrid model, "oRatio" is 43.78%, whereas no half-open routes are selected under the closed model. Furthermore, in most cases, more saleable products collected from stores and customers are redistributed in the hybrid model, with "reLoad" increasing by an average of 11.96%. This is mainly due to the expanded route selection space enabled by half-open routes, which allows the selection of more advantageous routes with larger "reload", thereby contributing to a reduction in total cost.

5.4.4. Effectiveness of integrated redistribution system

To evaluate the effectiveness of our proposed integrated distribution system, we conduct an experiment comparing it with a three-echelon independent distribution system. In the three-echelon independent distribution system, three separate fleets are employed to handle the distribution tasks. The first echelon delivers saleable products from warehouses to retail stores to replenish store inventory while reserving saleable products for subsequent customer deliveries. The fleet then collects both saleable and unsaleable products and returns them to the warehouses. The second echelon picks up saleable products from retail stores, delivers them to customers, and returns both types of products to the stores. In the third echelon, the fleet departs from the warehouses, collects two types of products that are returned to stores by customers in the second echelon, and ultimately returns to the warehouses.

Table 8
Impact of Integrated distribution system.

Instance	Three-echelon		Integrated			
	Obj_{Three}	k_{Three}	Obj_{Inte}	k_{Inte}	Δ_{Obj} (%)	Δ_{k} (%)
C101_5_50	384.7	9	220.7	5	**42.63**	**44.44**
C102_5_50	383.9	9	219.9	5	**42.72**	**44.44**
C103_5_50	380.3	9	216.1	5	**43.18**	**44.44**
C104_5_50	383.8	9	214.2	5	**44.19**	**44.44**
C105_5_50	384.4	9	219.3	5	**42.95**	**44.44**
C106_5_50	384.7	9	219.6	5	**42.92**	**44.44**
C107_5_50	384.4	9	219.3	5	**42.95**	**44.44**
C108_5_50	383.4	9	218.7	5	**42.96**	**44.44**
C109_5_50	383.3	9	218.3	5	**43.05**	**44.44**
R101_5_50	843.5	15	669.8	12	**20.59**	**20.00**
R102_5_50	770.4	14	580.1	10	**24.70**	**28.57**
R103_5_50	683.2	12	502.1	9	**26.51**	**25.00**
R104_5_50	611.6	11	426.8	7	**30.22**	**36.36**
R105_5_50	765.9	14	577.4	10	**24.61**	**28.57**
R106_5_50	708.2	13	516.9	9	**27.01**	**30.77**
R107_5_50	658.2	12	475.3	8	**27.79**	**33.33**
R108_5_50	611.6	11	426.8	7	**30.22**	**36.36**
R109_5_50	687.3	12	516.6	9	**24.84**	**25.00**
R110_5_50	636.1	11	471.6	8	**25.86**	**27.27**
R111_5_50	658.4	12	472.1	8	**28.30**	**33.33**
R112_5_50	603.3	11	421.5	7	**30.13**	**36.36**
Average					**33.73**	**36.24**

Experiments are conducted on the medium-scale instances, with the results presented in **Table 8.** Column "Δ_{Obj} (%)" gives the total cost reduction rate, which is calculated as $\Delta_{Obj} = 100 \times (Obj_{Three} -$

$Obj_{Inte}) / Obj_{Three}$, where Obj_{Three} and Obj_{Inte} are the total costs obtained in the three-echelon and the integrated distribution system, respectively. Column "Δ_k (%)" gives the fleet size reduction rate, which is calculated as $\Delta_k = 100 \times (k_{Three} - k_{Inte}) / k_{Three}$, where k_{Three} and k_{Inte} are the fleet size used in the three-echelon and the integrated distribution system, respectively. The results reveal that, compared to the three-echelon independent distribution system, the integrated distribution system achieves an average total cost reduction of 33.73% and an average fleet size reduction of 36.24%. These findings demonstrate that the integrated distribution system significantly outperforms the three-echelon system by sharing vehicle resources, maximizing vehicle space utilization, and dynamically selecting and combining multiple types of tasks.

6. Conclusions

In the context of omnichannel retailing, the traditional independent transportation system for the replenishment and recovery of stores and the pick-up/delivery order fulfillment of customers must adapts to the increasing demands for integrated logistics. This paper proposes an integrated logistics and distribution system, modeled as a closed and half-open mixed multi-depot multi-commodity unpaired pickup and delivery problem with time windows. This problem involves making simultaneous decisions on vehicle routing, matching pickup and delivery demands between stores and customers, and determining the quantity of products dispatched from warehouses, making it computationally challenging. To address these challenges, we propose an enhanced branch-and-price-and-cut algorithm, which introduces an innovative bidirectional labeling algorithm, along with several acceleration strategies to enhance the efficiency of subproblem solving. Additionally, we identify that the subset-row inequalities can lift the lower bound, further improving the algorithm's performance. Extensive computational experiments demonstrate the algorithm's superior performance, especially when compared with the CPLEX solver. Key findings indicate that allowing redistribution, using half-open routes, and adopting the integrated distribution model can significantly improve operational efficiency and reduce logistics cost. Furthermore, sensitivity analyses on vehicle capacity and customer demand heterogeneity provide practical insights for logistics operations.

There are certain limitations in this paper and the following possible directions can be done for future research. This paper assumes fixed time windows and pre-determined delivery and pickup demands. However, in practical applications, businesses may face more complex demand fluctuations and variations in time windows. Future research could explore how to incorporate uncertainties in time windows and real-time demand changes into logistics planning to enhance the system's flexibility and responsiveness. Additionally, since redistribution can reduce the quantity of saleable products dispatched from warehouses, warehouse processing costs could further be incorporated into the optimization objective. Finally, the proposed BPC algorithm could be combined with some heuristic algorithms to accelerate the solution of subproblem, thereby improving the performance of the proposed BPC algorithm.

CRediT authorship contribution statement

Ying He: Conceptualization, Methodology, Software, Writing - original draft, Writing – review & editing. **Peng Liu:** Conceptualization, Writing - original draft, Writing – review & editing. **Shuxian Xu:** Conceptualization, Supervision, Writing - original draft, Writing – review & editing. **Ronghui Liu**: Formal analysis, Writing - review & editing.

Data availability

Data will be made available on request.

Acknowledgments

The authors sincerely thank anonymous reviewers and editors for their invaluable suggestions and comments that significantly improved the paper. This work is supported by grants from the National Natural Science Foundation of China (72271175, 72431006, 72471017, 72394374, 72288101).

Appendix A. Notations

Table A.1
Notations and definitions.

Notation	Definition
Sets	
R	Set of retail store vertices
C	Set of online customer vertices
D	Set of warehouse vertices
$D^{'}$	Set of dummy warehouse vertices
V	Set of vertices in a physical network（$V = D \cup R \cup C \cup D^{'}$）
A	Set of arcs in a physical network
K	Set of a sufficiently large fleet of homogeneous vehicles
Parameters	
c_{ij}	Travel cost along arc $(i, j) \in A$
t_{ij}	Travel time along arc $(i, j) \in A$
d_{ij}	Travel distance along arc $(i, j) \in A$
d_i	Delivery demand for saleable products of vertex $i \in R \cup C$
p_i	Pickup demand for saleable products of vertex $i \in R \cup C$
u_i	Pickup demand for unsaleable products of vertex $i \in R \cup C$
e_i	Earliest service start time for vertex $i \in V$
l_i	Latest service start time for vertex $i \in V$
Q	Capacity limitation of a vehicle
L	Travel range limitation of a vehicle
v	Travel speed of a vehicle
γ	Fixed cost of a vehicle
s_i	Service time at vertex $i \in V$
M	A Large positive number
Variables	
x_{ijk}	$x_{ijk} = 1$ if vehicle $k \in K$ travels arc $(i, j) \in A$; 0, otherwise
T_{ik}	Service start time for vehicle $k \in K$ at vertex $i \in V$
b_{ik}	Accumulated travel distance for vehicle $k \in K$ when arriving at vertex $i \in V$
P_{ik}	Quantity of saleable products carried by vehicle $k \in K$ when arriving at vertex $i \in V$
U_{ik}	Quantity of unsaleable products carried by vehicle $k \in K$ when arriving at vertex $i \in V$
$P_{ik}^{'}$	Quantity of saleable products carried by vehicle $k \in K$ when departing from vertex $i \in V$
$U_{ik}^{'}$	Quantity of unsaleable products carried by vehicle $k \in K$ when departing from vertex $i \in V$
N_{ik}	Quantity of saleable products carried by vehicle $k \in K$ when departing from warehouse $i \in D$

Appendix B. Proofs of forward and backward dominance rules.

Appendix B.1. Proof of Proposition 1

$l_i^{f'}$ is dominated by l_i^f if and only if all feasible extensions of $l_i^{f'}$ are also feasible extensions of

l_i^f, and the reduced cost of the complete route generated by extending $l_i^{f'}$ is no less than that generated by extending l_i^f. Let $\Im = \{v_1, v_2, v_3, \cdots, v_n, o\}$, $v_m \in R \cup C$, $\forall m \in \{1, 2, 3, \cdots, n\}$, $o \in D'$ be any one feasible extension of $l_i^{f'}$, and the feasible complete route r_2 is generated by merging the partial route of $l_i^{f'}$ and $\Im$. Suppose that l_i^f can be extended along $\Im$ to from a complete route r_1. Clearly, based on conditions (7a) - (7c), it can be inferred that the elementary, maximum travel range limits, and time window constraints are satisfied. Furthermore, conditions (7d) - (7f) ensure that r_1 does not violate the capacity constraints. The detailed proof is as follows: Suppose $l_{v_1}^f = \{l_i^f, c_{v_1}^f, t_{v_1}^f, \rho_{v_1}^f, \varphi_{v_1}^f, \alpha_{v_1}^f, \beta_{v_1}^f, \mathrm{X}_{v_1}^f, U_{v_1}^f\}$ and $l_{v_1}^{f'} = \{l_i^{f'}, c_{v_1}^{f'}, t_{v_1}^{f'}, \rho_{v_1}^{f'}, \varphi_{v_1}^{f'}, \alpha_{v_1}^{f'}, \beta_{v_1}^{f'}, \mathrm{X}_{v_1}^{f'}, U_{v_1}^{f'}\}$ are the label extended from l_i^f and $l_i^{f'}$. Apparently, $l_i^{f'}$ is feasible. The forward extension condition (6h) and forward dominance rule (7f) ensure $\beta_{v_1}^f = \beta_i^f + u_{v_1} \le \beta_{v_1}^{f'} = \beta_i^{f'} + u_{v_1}$. Based on the relationship among $\alpha_i^{f'}, \alpha_i^f$ and d_{v_1}, it can be divided into four cases: 1) if $\alpha_i^{f'} \ge d_{v_1}$ and $\alpha_i^f \ge d_{v_1}$, it can be deduced from the extension conditions (6i) and (6j) that $\alpha_{v_1}^{f'} + \beta_{v_1}^{f'} = \alpha_i^{f'} + \beta_i^{f'} + p_{v_1} + u_{v_1} - d_{v_1} \le Q$, $\alpha_{v_1}^f + \beta_{v_1}^f = \alpha_i^f + \beta_i^f + p_{v_1} + u_{v_1} - d_{v_1}$, $\alpha_{v_1}^{f'} + \varphi_{v_1}^{f'} = \min\{\alpha_i^{f'} + p_{v_1} - d_{v_1} + \varphi_i^{f'}, Q - \beta_{v_1}^{f'}\}$ and $\alpha_{v_1}^f + \varphi_{v_1}^f = \min\{\alpha_i^f + p_{v_1} - d_{v_1} + \varphi_i^f, Q - \beta_{v_1}^f\}$ holds. It follows from the dominance rule (7d) and (7e) that $\alpha_{v_1}^f + \beta_{v_1}^f \le \alpha_{v_1}^{f'} + \beta_{v_1}^{f'} \le Q$ and $\alpha_{v_1}^f + \varphi_{v_1}^f \ge \alpha_{v_1}^{f'} + \varphi_{v_1}^{f'}$ is satisfied. 2) if $\alpha_i^{f'} \ge d_{v_1}$ and $\alpha_i^f < d_{v_1}$, then $p_{v_1} = 0$ holds. It can be deduced from the extension conditions (6i) - (6j) and dominance rule (7d) - (7f) that $\varphi_{v_1}^f + \alpha_{v_1}^f = \varphi_{v_1}^f = \min\{\varphi_i^f + \alpha_i^f - d_{v_1}, Q - \beta_{v_1}^f\} \ge \min\{\varphi_i^{f'} + \alpha_i^{f'} - d_{v_1}, Q - \beta_{v_1}^{f'}\} = \varphi_{v_1}^{f'} + \alpha_{v_1}^{f'} \ge 0$ and $\alpha_{v_1}^f + \beta_{v_1}^f = \beta_{v_1}^f \le \alpha_{v_1}^{f'} + \beta_{v_1}^{f'} \le Q$. 3) if $\alpha_i^{f'} < d_{v_1}$ and $\alpha_i^f \ge d_{v_1}$, then $p_{v_1} = 0$ holds, it can be deduced from the extension conditions (6i) - (6j) and dominance rule (7d) - (7f) that $\varphi_{v_1}^f + \alpha_{v_1}^f = \min\{\varphi_i^f + \alpha_i^f - d_{v_1}, Q - \beta_{v_1}^f\} \ge \min\{\varphi_i^{f'} + \alpha_i^{f'} - d_{v_1}, Q - \beta_{v_1}^{f'}\} = \varphi_{v_1}^{f'} + \alpha_{v_1}^{f'}$ and $\alpha_{v_1}^f + \beta_{v_1}^f = \beta_i^f + u_{v_1} - d_{v_1} \le \alpha_i^{f'} + \beta_i^{f'} + u_{v_1} - d_{v_1} \le \beta_i^{f'} + u_{v_1} = \alpha_{v_1}^{f'} + \beta_{v_1}^{f'} \le Q$. 4) if $\alpha_i^{f'} < d$ and $\alpha_i^f < d$, then $p_{v_1} = 0$ holds. it can be deduced from the extension conditions (6i) - (6j) and dominance rule (7d) - (7f) that $\varphi_{v_1}^f + \alpha_{v_1}^f = \varphi_{v_1}^f = \min\{\varphi_i^f + \alpha_i^f - d_{v_1}, Q - \beta_{v_1}^f\} = \varphi_{v_1}^{f'} + \alpha_{v_1}^{f'} \ge 0$ and $\alpha_{v_1}^f + \beta_{v_1}^f = \beta_{v_1}^f \le \beta_{v_1}^{f'} = \alpha_{v_1}^{f'} + \beta_{v_1}^{f'} \le Q$.

In conclusion，$l_{v_1}^f$ does not violate the capacity feasibility constraints (namely, both condition (5c) and (5d) are not satisfied) and is feasible. Moreover, $l_{v_1}^f$ and $l_{v_1}^{f'}$ still satisfy the dominance rules (7a) - (7f), it follows that when $l_{v_1}^{f'}$ can sequentially expanded along $\{v_2, v_3, \cdots, v_n, o\}$, $v_m \in R \cup C$, $\forall m \in \{2, 3, \cdots, n\}$, $o \in D'$ to obtain a feasible complete route, $l_{v_1}^f$ will also lead to a feasible complete route r_1 by extending along $\{v_2, v_3, \cdots, v_n, o\}$, $v_m \in R \cup C$, $\forall m \in \{2, 3, \cdots, n\}$, $o \in D'$. Furthermore, Since the dual variables of the SR inequalities are negative, dominance conditions (7g) and (7h) ensure that the reduced cost of r_1 is no greater than that of r_2. Therefore, forward dominance rules (7a) - (7h) are valid.

Appendix B.2. Proof of Proposition 2

We only prove rules (10b) - (10d) related to load resource, because the proofs for the other rules and the proofs of proposition 1 are similar. Let $\wp = \{v_1, v_2, v_3, \cdots, v_n, o\}$, $v_m \in R \cup C$, $\forall m \in \{1, 2, 3, \cdots, n\}$, $o \in D$ be any one feasible extension of $l_i^{b'}$, and the feasible complete route ι_2 is generated by merging the partial route of $l_i^{b'}$ and $\wp$. Suppose that l_i^b can be extended along $\wp$ to from a complete route ι_1. Conditions (10b) - (10d) ensure that ι_1 does not violate the capacity constraints. The detailed proof is as follows: Suppose $l_{v_1}^b = \{l_i^b, c_{v_1}^b, t_{v_1}^b, \rho_{v_1}^b, \omega_{v_1}^b, \xi_{v_1}^b, \{\theta_{Sv_1}^b\}_{S\in\Lambda}, \mathrm{X}_{v_1}^b, U_{v_1}^b\}$ and $l_{v_1}^{b'} = \{l_{v_1}^{b'}, c_{v_1}^{b'}, t_{v_1}^{b'}, \rho_{v_1}^{b'}, \omega_{v_1}^{b'}, \xi_{v_1}^{b'}, \{\theta_{Sv_1}^{b'}\}_{S\in\Lambda}, \mathrm{X}_{v_1}^{b'}, U_{v_1}^{b'}\}$ are the label extended from l_i^b and $l_i^{b'}$ respectively. Apparently, $l_i^{b'}$ is feasible. Based on the relationship among $\omega_i^{b'}$, ω_i^b and p_{v_1}, it can be divided into three cases according to the

backward extension conditions (9h) - (9i) and backward dominance rules (10b) - (10d), i.e., 1) If $p_{v_1} \geq \omega_i^{b'} \geq \omega_i^b$, we can derive $\omega_{v_1}^b = d_{v_1} = \omega_{v_1}^{b'} \leq Q$, $\xi_{v_1}^b = \max\{p_{v_1} + u_{v_1},\ \xi_i^b + p_{v_1} - \omega_i^b + u_{v_1}\} \leq \max\{p_{v_1} + u_{v_1},\ \xi_i^{b'} + p_{v_1} - \omega_i^{b'} + u_{v_1}\} \leq \xi_{v_1}^{b'} \leq Q$, and further we obtain $\xi_{v_1}^b - \omega_{v_1}^b = \xi_{v_1}^b - d_{v_1} \leq \xi_{v_1}^{b'} - d_{v_1} = \xi_{v_1}^{b'} - \omega_{v_1}^{b'}$. 2) If $\omega_i^b \leq p_{v_1} < \omega_i^{b'}$, it is easy to derive $\omega_{v_1}^b = d_{v_1} \leq d_{v_1} + \omega_i^{b'} - p_{v_1} = \omega_{v_1}^{b'} \leq Q$, $\xi_{v_1}^b = \max\{p_{v_1} + u_{v_1},\ \xi_i^b - \omega_i^b + p_{v_1} + u_{v_1}\} \leq \max\{\omega_i^{b'} + u_{v_1}, \xi_i^{b'} - \omega_i^{b'} + p_{v_1} + u_{v_1}\} \leq \max\{\omega_i^{b'} + u_{v_1},\ \xi_i^{b'} + u_{v_1}\} = \xi_{v_1}^{b'} \leq Q$ and $\xi_{v_1}^b - \omega_{v_1}^b = \max\{p_{v_1} + u_{v_1} - d_{v_1},\ \xi_i^b - \omega_i^b + p_{v_1} + u_{v_1} - d_{v_1}\} \leq \max\{p_{v_1} + u_{v_1} - d_{v_1},\ \xi_i^{b'} - \omega_i^{b'} + p_{v_1} + u_{v_1} - d_{v_1}\} = \xi_{v_1}^{b'} - \omega_{v_1}^{b'}$. 3) If $p_{v_1} < \omega_i^b \leq \omega_i^{b'}$, we can deduce $\omega_{v_1}^b = d_{v_1} + \omega_i^b - p_{v_1} \leq d_{v_1} + \omega_i^{b'} - p_{v_1} = \omega_{v_1}^{b'} \leq Q$, $\xi_{v_1}^b = \max\{\omega_i^b + u_{v_1},\ \xi_i^b + u_{v_1}\} \leq \max\{\omega_i^{b'} + u_{v_1},\ \xi_i^{b'} + u_{v_1}\} = \xi_{v_1}^{b'} \leq Q$ and $\xi_{v_1}^b - \omega_{v_1}^b = \max\{p_{v_1} + u_{v_1} - d_{v_1},\ \xi_i^b - \omega_i^b + p_{v_1} + u_{v_1} - d_{v_1}\} \leq \max\{p_{v_1} + u_{v_1} - d_{v_1},\ \xi_i^{b'} - \omega_i^{b'} + p_{v_1} + u_{v_1} - d_{v_1}\} = \xi_{v_1}^{b'} - \omega_{v_1}^{b'}$ hold.

In conclusion, $l_{v_1}^b$ does not violate the capacity feasibility constraints (namely, both condition (8c) and (8d) are not satisfied) and is feasible. Similarly, the complete route ι_1 is feasible and its reduced cost is no more than that of ι_2. Therefore, backward dominance rules (10a) - (10d) are valid.

Appendix C. Dominance Test algorithm and Merge algorithm.

In **Algorithm C.1**, we check the dominance relationship of forward labels and backward labels according to the rules (7a) – (7h) and (10a) – (10d), respectively. And **Algorithm C.2** uses the criterions (11a) – (11e) to determine whether or not a forward label and a backward label can be merged.

Algorithm C.1. Dominance Test (i, UL_i^f, TL_i^f, UL_i^b, TL_i^b, L_u)

for all $l_i^{f_1}, l_i^{f_2} \in UL_i^f \cup TL_i^f$, $l_i^{f_1} \neq l_i^{f_2}$ **do**
  if the dominance relationship between $l_i^{f_1}$ and $l_i^{f_2}$ did not been test **then**
    Determine the dominant relationship between $l_i^{f_1}$ and $l_i^{f_2}$ based on conditions (7a) - (7h);
    Discard the dominated label from $UL_i^f \cup TL_i^f$ and L_u;
  end if
end for
for all $l_i^{b_1}, l_i^{b_2} \in UL_i^b \cup TL_i^b$, $l_i^{b_1} \neq l_i^{b_2}$ **do**
  if the dominance relationship between $l_i^{b_1}$ and $l_i^{b_2}$ did not been test **then**
    Determine the dominant relationship between $l_i^{b_1}$ and $l_i^{b_2}$ based on conditions (10a) - (10d);
    Discard the dominated label from $UL_i^b \cup TL_i^b$ and L_u;
  end if
end for

Algorithm C.2. Merge (i, UL_i^f, TL_i^f, UL_i^b, TL_i^b, LC, n_c).

for $\forall l_i^f \in UL_i^f \cup TL_i^f$, $\forall l_i^b \in UL_i^b \cup TL_i^b$ **do**
  if l_i^f and l_i^b have never been merged before **do**
    Merge l_i^f and l_i^b to create a complete route l_c, if l_c is feasible based on conditions (11a) - (11e);
    if the reduced cost of l_c is negative **then**
      Add l_c to LC, $n_c \leftarrow n_c + 1$;
    end if
  end if
end for

References

Abdulkader, M.M.S., Gajpal, Y., ElMekkawy, T.Y., 2018. Vehicle routing problem in omni-channel retailing distribution systems.

International Journal of Production Economics 196, 43-55. https://doi.org/10.1016/j.ijpe.2017.11.011

Acimovic, J., Farias, V.F., 2019. The Fulfillment-Optimization Problem. INFORMS TutORials in Operations Research 218-237. https://doi.org/10.1287/educ.2019.0199

Bayliss, C., Martins, L.d.C., Juan, A.A., 2020. A two-phase local search with a discrete-event heuristic for the omnichannel vehicle routing problem. Computers & Industrial Engineering 148, 106695. https://doi.org/10.1016/j.cie.2020.106695.

Bayram, A., Cesaret, B., 2021. Order fulfillment policies for ship-from-store implementation in omni-channel retailing. European Journal of Operational Research 294 (3), 987-1002. https://doi.org/10.1016/j.ejor.2020.01.011

Brandão, J., 2020. A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. European Journal of Operational Research 284 (2), 559-571. https://doi.org/10.1016/j.ejor.2020.01.008

Chemla, D., Meunier, F., Calvo, R.W., 2013. Bike sharing systems: Solving the static rebalancing problem. Discrete Optimization 10 (2), 120-146. https://doi.org/10.1016/j.disopt.2012.11.005

Chen, Q., Li, K., Liu, Z., 2014. Model and algorithm for an unpaired pickup and delivery vehicle routing problem with split loads. Transportation Research Part E: Logistics and Transportation Review 69, 218-235. https://doi.org/10.1016/j.tre.2014.06.010

Dantzig, G.B., Wolfe, P., 1960. Decomposition Principle for Linear Programs. Operations Research 8 (1), 101-111. https://doi.org/10.1287/opre.8.1.101

Dayarian, I., Crainic, T.G., Gendreau, M., Rei, W., 2015. A column generation approach for a multi-attribute vehicle routing problem. European Journal of Operational Research 241 (3), 888-906. https://doi.org/10.1016/j.ejor.2014.09.015

Dell'Amico, M., Iori, M., Novellani, S., Subramanian, A., 2018. The Bike sharing Rebalancing Problem with Stochastic Demands. Transportation Research Part B: Methodological 118, 362-380. https://doi.org/10.1016/j.trb.2018.10.015

Desaulniers, G., Desrosiers, J., Solomon, M.M. (Eds.), 2005. Column Generation. Springer.

Desaulniers, G., Madsen, O., Ropke, S., 2014. The vehicle routing problem with time windows. In: Vehicle Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization (SIAM), pp. 119-159. https://doi.org/10.1137/1.9781611973594.ch5

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research 40 (2), 342-354. https://doi.org/10.1287/opre.40.2.342

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research 40 (2), 342-354. https://doi.org/10.1287/opre.40.2.342

Dridi, I. H., Alaïa, E.B., Borne, P., Bouchriha, H., 2019. Optimisation of the multi-depots pick-up and delivery problems with time windows and multi-vehicles using PSO algorithm. International Journal of Production Research 58 (8), 1-4. https://doi.org/10.1080/00207543.2019.1650975

Dubey, N., Tanksale, A., 2023. A multi-depot vehicle routing problem with time windows, split pickup and split delivery for surplus food recovery and redistribution. Expert Systems with Applications 232, 120807. https://doi.org/10.1016/j.eswa.2023.120807

Ensafian, H., Andaryan, A.Z., Bell, M.G.H., Geers, D.G., Kilby, P., Li, J., Cost-optimal deployment of autonomous mobile lockers co-operating with couriers for simultaneous pickup and delivery operations. Transportation Research Part C: Emerging Technologies 146, 103958. https://doi.org/10.1016/j.trc.2022.103958

Erdoğan, G., Laporte, G., Calvo, R.W., 2014. The static bicycle relocation problem with demand intervals. European Journal of Operational Research 238 (2), 451-457. https://doi.org/10.1016/j.ejor.2014.04.013

George B. Dantzig, Philip Wolfe, (1960) Decomposition Principle for Linear Programs. Operations Research 8(1):101-111. https://doi.org/10.1287/opre.8.1.101

Goedhart, J., Haijema, R., Akkerman, R., Leeuw, S.d., 2023. Replenishment and fulfilment decisions for stores in an omni-channel retail network. European Journal of Operational Research 311 (3),1009-1022. https://doi.org/10.1016/j.ejor.2023.06.018.

Guo, C., Thompson, R.G., Foliente, G., Kong, X.T.R., 2021. An auction-enabled collaborative routing mechanism for omnichannel on-demand logistics through transshipment. Transportation Research Part E: Logistics and Transportation Review 146,

102206. https://doi.org/10.1016/j.tre.2020.102206

Hamid, M., Nasiri, M.M., Rabbani, M., 2023. A mixed closed-open multi-depot routing and scheduling problem for homemade meal delivery incorporating drone and crowd-sourced fleet: A self-adaptive hyper-heuristic approach. Engineering Applications of Artificial Intelligence 120, 105876. https://doi.org/10.1016/j.engappai.2023.105876

Ho, S.C., Szeto, W.Y., 2017. A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. Transportation Research Part B: Methodological 95, 340-363. https://doi.org/10.1016/j.trb.2016.11.003

Houck, D.J.J., Picard, J.C., Queyranne, M., Vemuganti, R.R., 1980. The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. Opsearch 17 (2&3), 93-109. https://publications.polymtl.ca/5977/

Hwang, F. J., Hu, B., & Kovalyov, M. Y. (2024). Supermarket-chain grocery delivery optimization through crowdshipping. International Journal of Production Research 63(5), 1725–1752. https://doi.org/10.1080/00207543.2024.2389550

Ikotun, A.M., Ezugwu, A.E., Abualigah, L., Abuhaija, B., Heming, J., K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. Information Sciences 622, 178-210. https://doi.org/10.1016/j.ins.2022.11.139

Irnich, S., Villeneuve, D., 2006. The Shortest-Path Problem with Resource Constraints and k-Cycle Elimination for k ≥ 3. INFORMS Journal on Computing 18 (3), 283-406. https://doi.org/10.1287/ijoc.1040.0117

Janjevic, M., Merchán, D., Winkenbach, M., 2021. Designing multi-tier, multi-service-level, and multi-modal last-mile distribution networks for omni-channel operations. European Journal of Operational Research 294 (3), 1059-1077. https://doi.org/10.1016/j.ejor.2020.08.043

Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., 2007. Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows. Operations Research 56 (2), 497-511. https://doi.org/10.1287/opre.1070.0449

Jesper, L., 1999. Parallelization of the Vehicle Routing Problem with Time Windows. Unpublished doctoral thesis, IMM-DTU Technical University of Denmark, Lyngby, Denmark. https://orbit.dtu.dk/en/publications/Parallelization of the Vehicle Routing Problem with Time Windows - Welcome to DTU Research Database

Jspsen, 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. https://doi.org/10.1287/opre.1070.0449

Kaufman, L., Rousseeuw, P.J., 1987. Clustering by means of medoids. In: Statistical Data Analysis Based on the L1-Norm, pp. 405-416. https://www.researchgate.net/publication/Clustering by Means of Medoids

Lahyani, R., Gouguenheim, A.L., Coelho, L.C., 2019. A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. International Journal of Production Research 57 (22), 1-14. https://doi.org/10.1080/00207543.2019.1572929

Lei, C., Ouyang, Y., 2018. Continuous approximation for demand balancing in solving large-scale one-commodity pickup and delivery problems. Transportation Research Part B: Methodological 109, 90-109. https://doi.org/10.1016/j.trb.2018.01.009Kohl, N., 1995. Exact methods for time constrained routing and related scheduling problems. Unpublished doctoral thesis, Technical University of Denmark, Lyngby, Denmark. https://orbit.dtu.dk/en/publications/Exact methods for time constrained routing and related scheduling problems - Welcome to DTU Research Database

Li, J., Qin, H., Baldacci, R., Zhu, W., 2020. Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. Transportation Research Part E: Logistics and Transportation Review 140, 101955. https://doi.org/10.1016/j.tre.2020.101955

Li, N., Wang, Z., 2025. Vehicle routing problem for omnichannel retailing including multiple types of time windows and products. Computers & Operations Research 173, 106828. https://doi.org/10.1016/j.cor.2024.106828

Lübbecke, M.E., Desrosiers, J., 2005. Selected Topics in Column Generation. Operations Research 53 (6), 1007-1023. https://doi.org/10.1287/opre.1050.0234

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. 1, 281–297. https://digitalassets.lib.berkeley.edu/math/ucb/text/math_s5_v1_article-17.pdf

MacQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, pp. 281-297. https://digitalassets.lib.berkeley.edu/math/ucb/text/math_s5_v1_article-17.pdf

Marco E. Lübbecke, Jacques Desrosiers, (2005) Selected Topics in Column Generation. Operations Research 53(6):1007-1023. https://doi.org/10.1287/opre.1050.0234

Martins, L.d.C., Bayliss, C., Juan, A.A., Panadero, J., Marmol, M., 2020. A savings-based heuristic for solving the omnichannel vehicle routing problem with pick-up and delivery. Transportation Research Procedia 47, 83-90. https://doi.org/10.1016/j.trpro.2020.03.082

Nafstad, G.M., Haugseth, A., Høyland, V., Stålhane, M., 2021. An exact solution method for a rich helicopter flight scheduling problem arising in offshore oil and gas logistics. Computers & Operations Research 128, 105158. https://doi.org/10.1016/j.cor.2020.105158

Osorio, J., Lei, C., Ouyang, Y., 2021. Optimal rebalancing and on-board charging of shared electric scooters. Transportation Research Part B: Methodological 147, 197-219. https://doi.org/10.1016/j.trb.2021.03.009

Paul, J., Agatz, N., Savelsbergh, M., 2019b. Optimizing Omni-Channel Fulfillment with Store Transfers. Transportation Research Part B: Methodological 129, 381-396. https://doi.org/10.1016/j.trb.2019.10.002

Paul, J., Agatz, N., Spliet, R., Koster, R.D., 2019a. Shared Capacity Routing Problem − An omni-channel retail study. European Journal of Operational Research 273 (2), 731-739. https://doi.org/10.1016/j.ejor.2018.08.027

Qiu, R., Yuan, M., Sun, M., Fan, Z.P., Xu, H., 2025. Optimizing omnichannel retailer inventory replenishment using vehicle capacity-sharing with demand uncertainties and service level requirements. European Journal of Operational Research 320 (2), 417-432. https://doi.org/10.1016/j.ejor.2024.08.005

Ratcliff, C., 2014. How Fashion Ecommerce Retailers Can Reduce Online Returns.https://econsultancy.com/blog/65026-how-fashion-ecommerce-retailers-can-reduce-online-returns/. [Online; accessed 24-April-2025].

Raviv, T., Tzur, M., Forma, I.A., 2013. Static repositioning in a bike-sharing system: models and solution approaches. EURO Journal on Transportation and Logistics 2 (3), 187-229. https://doi.org/10.1007/s13676-012-0017-6

Righini, G., Salani, M., 2006. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization 3 (3), 255-273. https://doi.org/10.1016/j.disopt.2006.05.007

Schubert, D., Kuhn, H., Holzapfel, A., 2021. Same-day deliveries in omnichannel retail: Integrated order picking and vehicle routing with vehicle-site dependencies. Naval Research Logistics 68 (6), 721-744. https://doi.org/10.1002/nav.21954.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35 (2), 254-265. https://dl.acm.org/doi/10.5555/2778348.2778358

Su, E., Qin, H., Li, J., Pan, K., 2023. An exact algorithm for the pickup and delivery problem with crowdsourced bids and transshipment. Transportation Research Part B: Methodological 177, 102831. https://doi.org/10.1016/j.trb.2023.102831

Szeto, W.Y., Shui, C.S., 2018. Exact loading and unloading strategies for the static multi-vehicle bike repositioning problem. Transportation Research Part B: Methodological 109, 176-211. https://doi.org/10.1016/j.trb.2018.01.007

Tahirov, N., Akhundov, N., Emde, S., Glock, G.H., 2025. Configuration of last-mile distribution networks for an encroaching manufacturer. Annals of Operations Research 344, 679–720. https://doi.org/10.1007/s10479-024-06031-3.

Tarantilis, C.D., Kiranoudis, C.T., 2002. Distribution of fresh meat. Journal of Food Engineering 51 (1), 85-91. https://doi.org/10.1016/S0260-8774(01)00040-1

Wei, L., Kapuscinski, R., Jasin, S., 2020. Shipping consolidation across two warehouses with delivery deadline and expedited options for e-commerce and omni-channel retailers. Manufacturing & Service Operations Management 23 (6), 1634-1650. https://doi.org/10.1287/msom.2020.0903

Yang, J., Li, Y., 2023. A multicommodity pickup and delivery problem with time windows and handling time in the omni-channel last-mile delivery. International Transactions in Operational Research 32 (3), 1524-1565. https://doi.org/10.1111/itor.13362

Zhang, D., Xu, W., Ji, B., Li, S., Liu, Y., 2020. An adaptive tabu search algorithm embedded with iterated local search and route

elimination for the bike repositioning and recycling problem. Computers & Operations Research 123, 105035 https://doi.org/10.1016/j.cor.2020.105035

Zhang, Z., Cheang, B., Li, C., Lim, A., 2019. Multi-commodity demand fulfillment via simultaneous pickup and delivery for a fast fashion retailer. Computers & Operations Research 103, 81-96. https://doi.org/10.1016/j.cor.2018.10.020

Zhao, J., Dong, H., Wang, N., 2023. Green split multiple-commodity pickup and delivery vehicle routing problem. Computers & Operations Research 159, 106318. https://doi.org/10.1016/j.cor.2023.106318

Zhen, L., Baldacci, R., Tan, Z., Wang, S., Lyu, J., 2022. Scheduling heterogeneous delivery tasks on a mixed logistics platform. European Journal of Operational Research 298 (2), 680-698. https://doi.org/10.1016/j.ejor.2021.06.057