informs
https://pubsonline.informs.org/journal/msom

MANUFACTURING & SERVICE OPERATIONS MANAGEMENT

Vol. 00, No. 0, Xxxxx 0000, pp. 000–000

Submitted to *Manufacturing & Service Operations Management*

ISSN 1523-4614, EISSN 1526-5498

A Decomposition Framework for Practical Customized Supply-Chain Decisions and a Computational Study

Kuo Liang[a], Chuwen Zhang[ad], Jinsong Liu[a], Dongdong Ge[bd*], Zizhuo Wang[cd]

[a]Research Institute for Interdisciplinary Sciences, School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai, China, kuoliang.cora@gmail.com, chuwzhang@gmail.com, liujinsong@163.sufe.edu.cn.
[b]Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China, ddge@sjtu.edu.cn.
[c]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China, wangzizhuo@cuhk.edu.cn.
[d]Cardinal Operations (Beijing) Company Limited, Beijing, China.
[*]Corresponding authors

Abstract. ***Problem definition:*** This paper explores a practice-driven integrated supply chain optimization problem involving dynamic network configuration, inventory placement, and traffic planning over a finite horizon. We formulate it as a multi-period mixed-integer program with heterogeneous customers and end-to-end service constraints, aiming to minimize the total operational costs. ***Academic/practical relevance:*** Quick and effective dynamic adjustment is crucial for online retailers and third-party logistics companies that face varying demand fulfillment pressure and low profit margins. However, existing off-the-shelf solvers struggle to obtain high-quality solutions within a reasonable time limit due to complex constraints and dynamic supply chain networks. ***Methodology:*** We propose a decomposition framework based on column generation that tackles the integrated problem by breaking it into customer-oriented subproblems with complex service constraints. Strategies and valid inequalities are introduced to accelerate the framework and improve solution quality. ***Results:*** Our numerical experiments demonstrate that the proposed framework achieves near-optimal solutions with an absolute gap of just 0.70% while reducing computation time by 66% compared to Gurobi, significantly outperforming more naive reformulations. ***Managerial implications:*** Our approach offers a computationally efficient method for solving integrated supply chain network problems that can be generalizable to similar settings. We also analyze the impact of different network scales, constraint configurations, and strategies, highlighting the flexibility and adaptability of our method.

Funding: This research is partially supported by the National Natural Science Foundation of China (NSFC) [Grant NSFC-72225009, 72394360, 72394365].

Key words: Third-party Logistics, Integrated Decisions, Decomposition Methods, Mixed-Integer Optimization, Parallel Computing

1. Introduction

Optimizing supply chain decisions in online retail involves three key levels: strategic, tactical, and operational levels. Strategic choices involve substantial capital investments and long-term contractual agreements in the supply chain network configuration, including determining new warehouse locations and selecting transportation routes. Tactical decisions address inventory management problems, including inventory placement and replenishment strategies in warehouses, which are usually adjusted every few months. The operational level determines more detailed planning, such as last-mile delivery and peer-transshipment, aiming to allocate transportation capabilities to satisfy customer demands. In practice, these decisions often follow a hierarchical approach, where lower-level problems are addressed after setting higher-level solutions.

However, the rapid growth of online retail has highlighted limitations in this hierarchical approach. During peak shopping periods, fluctuating demand can lead to situations where customer orders cannot be fulfilled promptly due to limited warehouses and delivery capacities (Rao 2015, Carey 2015, Huang et al. 2016). In response to these challenges, third-party warehousing and logistics services such as UPS, FedEx, and Sinotrans have emerged as attractive alternatives to self-built supply chains for online retailers (Song and Zhao 2022). This shift is driven primarily by two critical factors: *elasticity* and *customization*. Elasticity allows retailers to flexibly adjust their usage of facilities on a per-period basis, in contrast to the rigid, predefined setups of traditional self-built networks (Huang et al. 2016). Specifically, in a self-built network, it can be prohibitively expensive to expand the logistics resource. But with the third-party services, the retailers can improve their service levels by paying extra usage costs during peak seasons (Liu 2016). Customization offers retailers the ability to define specific end-to-end service requirements to achieve better service levels, such as maintaining minimum traffic levels and limiting the number of upstream service nodes (Balakrishnan et al. 2017). However, such shifts also present significant challenges to third-party warehousing and logistics service providers. On one hand, they face with the complex *integrated* decision-making problem, which consists of network configuration, inventory placement, and transportation planning over a given time horizon. The need for dynamic network adjustments further amplifies these computational challenges. On the other hand, the specific requirements of retailers can lead to costly unfulfilled demands (Stevens et al. 2014) and place significant pressure on service providers to maintain timely responsive operations. In our experience, even state-of-the-art optimization solvers like Gurobi (Gurobi Optimization, LLC 2024) and COPT (Ge et al. 2022)

struggle to solve a weekly planning problem involving 519 customers, 30 plants, 25 warehouses, and 140 commodities within a two-hour limit.

To resolve the above challenges, this paper studies the typical *integrated* problem faced by a third-party company and proposes a customer-oriented decomposition framework. This framework utilizes a reformulation based on the column generation method, combined with valid inequalities and a rounding heuristic. Our numerical experiments demonstrate that this approach successfully balances computational efficiency with high-quality solutions. Particularly, our work makes the following contributions.

(1) Inspired by our collaboration with our industrial partner, we address an integrated model that encompasses strategic, tactical, and operational decisions over a finite time horizon. We also incorporate a range of service constraints into the framework, such as transportation lower bounds, cardinality constraints, distance limitations, and customer backlog constraints. These constraints reflect the complex requirements frequently encountered by companies but are rarely examined in existing literature.

(2) We propose a decomposition framework for large-scale integrated supply chain problems under service constraints. The framework consists of two main steps: First, we introduce two reformulations that are suitable for a column generation approach to handle heterogeneous subproblems arising from customers with end-to-end service constraints. This approach enables effective parallelization of the subproblems. Then, column selection strategies and valid inequalities are also discussed. Finally, we introduce a rounding heuristic to recover integer-feasible solutions and enhance computational efficiency.

(3) We present a comprehensive empirical study under practical size and applicability in various scenarios. To evaluate the robustness and efficiency of our approach across diverse settings, we develop a companion procedure that generates synthetic problems using available geographic datasets, as detailed in the Electronic Companion. We conduct computational experiments on three real-world cases from our partner and challenging synthetic datasets. The results show that our method achieves an average 66% reduction in computation time compared to Gurobi while maintaining high solution quality with an average absolute gap of just 0.70% from Gurobi's optimal solution.

The remainder of this paper is organized as follows: Section 2 reviews the relevant literature related to our study. Section 3 introduces our integrated supply chain model within a finite horizon. Section 4 outlines our decomposition framework, which leverages the column generation method

and incorporates acceleration strategies to enhance performance. Section 5 presents numerical experiments that demonstrate the effectiveness and superiority of the proposed framework. Finally, Section 6 concludes the paper and discusses potential directions for future research.

2. Literature Review

This review covers several streams of literature that are most relevant to our study. Classical problems in supply-chain management consist of facility location problems (Mirchandani and Francis 1990), multi-item commodity flow problems (Ahuja et al. 1993), lot-sizing problems (Wagner and Whitin 1958, Federgruen and Tzur 1991, Pochet and Wolsey 2006), and various vehicle routing problems (Christofides et al. 1981). Most of the basic models can now be found in standard textbooks, such as Conforti et al. (2014) and Wolsey (2020).

Generally, our problem is closely related to recent studies on extensions of facility location problems that focus on enhancing customer satisfaction and enabling quicker responses to changing customer demands. For example, Balakrishnan et al. (2017) introduce several classes of strong cuts and develop an optimization-based heuristic to solve the Network Design with Service Requirements (NDSR) problem. The objective of this problem is to optimize the design of network arcs and decide how to route commodities while adhering to end-to-end service constraints. A computational study on the practical solvability of the problem is conducted in Gudapati et al. (2022b). To address a dynamic problem, Kang et al. (2022) consider an annual multi-period facility location model for JD.com, where, similar to our work, facility coverage and interconnections are no longer one-shot decisions. Instead, the goal is to enable timely adjustments within a business-to-consumer e-commerce environment. In our model, apart from deciding the network structure with service requirements, inventory decisions and transshipment are also considered. These elements are introduced to deal with the practical problems from our industry collaborators.

Another topic emerging from e-commerce is tackling computational difficulties in inventory placement and replenishment planning, arising from the complexity introduced by bills of materials and a large number of commodities. To address the first challenge, Wang and Hong (2023) propose a neural network-inspired approach in large-scale multi-period inventory optimization. The method has shown great practical value by utilizing efficient back-propagation schemes in TensorFlow (Abadi 2016). For the second, Chen and Graves (2021) explore inventory placement, focusing on the coverage fractions of facilities over customers. By efficiently aggregating items, they propose a column generation method to solve problems involving over 1 million commodities, although dynamic inventory decisions are not considered in their work. In a similar vein, our method is also

based on a column generation framework. However, different from their work, the computational challenge in this work arises from managing a large number of heterogeneous customers.

The column generation method is a common technique to solve large-scale optimization problems. It is particularly useful for problems with a large number of variables. The method works like a primal simplex method: the master problem iteratively adds more columns generated from the pricing problem, which finds the most promising columns based on the dual variables. Classical applications include flight scheduling problem vehicle routing problems (Barnhart et al. 1998a,b) and capacitated vehicle routing problem (Desrosiers et al. 1984, Fukasawa et al. 2006). More recently, apart from aforementioned supply-chain problems (Gudapati et al. 2022b, Chen and Graves 2021), the column generation method has been applied to the scheduling of electric vehicle fleets (Parmentier et al. 2023), unit commitment problem (Sugishita et al. 2024), and distributionally robust multi-item newsvendor problems (Wang and Delage 2024).

Compared to these successful applications, the column generation framework presented in this paper is designed to handle cases with a large number of subproblems. As we will clarify, both reformulations used in this framework differ from the single-block case (Wolsey 2020, Chapter 11.5) and from cases involving multiple identical blocks (Fukasawa et al. 2006). Instead, our framework addresses heterogeneous blocks, such as customers with varying requirements. To tackle this practical challenge, we focus on developing a framework that is well-suited for distributed computing environments and capable of handling a large number of customers in practice.

3. Problem Description and Formulation

We consider a joint decision-making problem faced by third-party logistics companies that incorporate dynamic network configuration, inventory placement, and transportation planning over a finite horizon. The network structure of these companies consists of three levels: suppliers, warehouses, and customers. The goal of the problem is to design a weekly plan to assign warehouses and allocate traffic from suppliers, through warehouses, to customers. The primary objective is to minimize total operational costs while meeting customer demands for various commodities. Set covering constraints are incorporated, reflecting the flexibility to activate or deactivate warehouses at any time with fixed costs. Additional constraints include capacity limitations, complex operational requirements, and customized service constraints such as minimum transshipment levels and restrictions on the number of upstream warehouses. Satisfying these constraints necessitates careful planning and coordination across the network to ensure overall performance and service delivery. In the sequel, we refer to this problem as the Dynamic Network Planning problem (DNP after that).

3.1. Notation

In this paper, we let $\mathbb{R}, \mathbb{B} = \{0, 1\}$ denote the set of real numbers and binary variables, respectively. Let $\mathcal{K}$ denote the set of various commodities considered, and $t \in [T] := 1, \ldots, T$ represent the discrete planning horizons. We introduce a directed graph $\mathcal{G} = (V, E)$ to model the three-tier supply chain network. The node set V is partitioned into three subsets: $\mathcal{C}$, $\mathcal{S}$, and $\mathcal{W}$, which represent customers, suppliers, and warehouses, respectively. The edge set E represents the available transshipment routes e within the network, allowing for peer-to-peer transshipment between warehouses. For any node $i \in V$, δ_i^+ and δ_i^- denote the sets of inflow and outflow edges of node i, respectively.

3.2. Formulation

We model the integrated three-level decision problem as a multi-period mixed-integer optimization problem. For each warehouse $i \in \mathcal{W}$, the binary decision variable $y_{i,t} = 1$ if warehouse i is activated in period t, incurring a fixed activation cost f_i per period, and $y_{i,t} = 0$ if it remains inactive. Similarly, for each edge $e \in E$, the variable $y_{e,t} \in \mathbb{B}$ indicates whether the edge is selected in period t, with an associated setup cost f_e per period. The transportation decision variable $x_{e,t,k}$ represents the quantity of commodity k transported along edge e in period t, incurring a per-unit transportation cost of g_e^d. For each customer $i \in \mathcal{C}$, the variable $s_{i,t,k}$ represents the backlogged demand for commodity k in period t, incurring a unit backlog cost $g_{i,k}^b$. Inventory levels for each warehouse $i \in \mathcal{W}$ are captured by $I_{i,t,k}$, representing the amount of commodity k held at warehouse i in period t, with a unit holding cost $g_{i,k}^h$. Lastly, for each supplier $i \in \mathcal{S}$, the variable $q_{i,t,k}$ denotes the supply quantity of commodity k produced by supplier i in period t.

In the rest of this paper, we repetitively use *plain* x, y, s, and I without the subscripts to denote the vectors for the conciseness of our representation. Besides, we use $\mathbf{x} = (x, y, s, I)$ to denote all variables. The objective of the decision-maker is to reduce the total operational costs, including setup costs, delivery expenses, backlogged costs, and holding costs. Specifically, the objective function can be expressed as follows:

$$v(\mathbf{x}) = \sum_{\forall t \in [T]} \left(\sum_{i \in \mathcal{W}} f_i y_{i,t} + \sum_{e \in E} f_e y_{e,t} + \sum_{k \in \mathcal{K}} \left(\sum_{e \in E} g_e^d x_{e,t,k} + \sum_{i \in \mathcal{C}} g_{i,k}^b s_{i,t,k} + \sum_{i \in \mathcal{W}} g_{i,k}^h I_{i,t,k} \right) \right). \quad (1)$$

The model is subject to three types of constraints. *Flows and layout constraints* encompass flow conservation and set-covering conditions. *Operational constraints* represent real-world restrictions on traffic and inventory decisions, including warehouse capacity, variable lower bounds on the traffic, and upper bounds on inflows. *Customized services constraints* represent the personalized requirements, such as the distance requirements for particular commodities (e.g., perishable products). We now describe these constraints below.

Flows and layout constraints Constraints (2)-(4) ensure consistent flow conservation among warehouses, suppliers, and customers:

$$q_{i,t,k} = \sum_{e \in \delta_i^-} x_{e,t,k}, \quad \forall i \in \mathcal{S},\ \forall k \in \mathcal{K},\ \forall t \in [T], \tag{2}$$

$$I_{i,t,k} = I_{i,t-1,k} - \sum_{e \in \delta_i^-} x_{e,t,k} + \sum_{e \in \delta_i^+} x_{e,t,k}, \quad \forall i \in \mathcal{W},\ \forall k \in \mathcal{K},\ \forall t \in [T], \tag{3}$$

$$\sum_{e \in \delta_i^+} x_{e,t,k} + s_{i,t,k} = s_{i,t-1,k} + D_{j,t,k}, \quad \forall i \in \mathcal{C},\ \forall k \in \mathcal{K},\ \forall t \in [T]. \tag{4}$$

Here, $D_{i,t,k}$ denotes the demand of customer i for the commodity k in period t, while $s_{i,t,k}$ and $s_{i,t-1,k}$ represent the backlogged demand of customer i for the commodity k in periods t and $t-1$, respectively. For sufficiently large M, the set covering constraints (5)-(7) are defined as follows:

$$y_{e,t} \leq y_{i,t}, \quad \forall i \in V,\ \forall e \in \delta_i^+ \cup \delta_i^-,\ \forall t \in [T], \tag{5}$$

$$x_{e,t,k} \leq M y_{e,t}, \quad \forall e \in E,\ \forall k \in \mathcal{K},\ \forall t \in [T], \tag{6}$$

$$y_{j,t} = 1, \quad \forall i \in \mathcal{C},\ \forall t \in [T]. \tag{7}$$

Constraint (5) ensures that the selection of edge e is contingent on the activation of its connected nodes, both upstream and downstream. Constraint (6) restricts the transportation quantity along edge e based on its activation status. Finally, constraint (7) specifies that customer nodes remain consistently open throughout the planning horizon.

Operational constraints Constraints (8)-(10) further guarantee the adherence to capacity limits, encompassing delivery upper bound C_e for transportation route e, storage limits C_i for warehouse i and production limits S_i for supplier i.

$$\sum_{k \in \mathcal{K}} x_{e,t,k} \leq C_e y_{e,t}, \quad \forall e \in E,\ \forall t \in [T], \tag{8}$$

$$\sum_{k \in \mathcal{K}} I_{i,t,k} \leq C_i y_{i,t}, \quad \forall i \in \mathcal{W},\ \forall t \in [T], \tag{9}$$

$$\sum_{k \in \mathcal{K}} q_{i,t,k} \leq S_i y_{i,t}, \quad \forall i \in \mathcal{S},\ \forall t \in [T]. \tag{10}$$

Constraint (11) sets an upper limit $U_{i,t}$ on the inflow for warehouse i during period t. For example, this can model warehouse throughput limitations due to human resource,

$$\sum_{e \in \delta_i^+} \sum_{k \in \mathcal{K}} x_{e,t,k} \leq U_{i,t}, \quad \forall i \in \mathcal{W},\ \forall t \in [T]. \tag{11}$$

To prevent meaningless transshipment and encourage the consolidation of small, long-tailed quantities, constraints (12) incorporates lower bounds L_e for both transfer edges between warehouses and termination edges from warehouses to customers,

$$\sum_{k\in\mathcal{K}} x_{e,t,k} \geq L_e y_{e,t}, \quad \forall e \in E,\ \forall t \in [T]. \tag{12}$$

Customized service constraints End-to-end customized service constraints, as introduced by Balakrishnan et al. (2017), are incorporated to address customer concerns and ensure operational efficiency. Specifically, to avoid split-order losses, constraint (13) is used to limit the number of warehouses employed to fulfill demands for each customer i:

$$\sum_{e\in\delta_i^+} y_{e,t} \leq N_{i,t}, \quad \forall i \in \mathcal{C},\ \forall t \in [T], \tag{13}$$

where $N_{i,t}$ specifies the maximum number of upstream service nodes (warehouses) that can serve customer i in period t. Additionally, constraint (14) sets a limit P_i on the total transportation distances for each customer i, addressing quality concerns, particularly for perishable products:

$$\sum_{e\in\delta_i^+} d_e y_{e,t} \leq P_i, \quad \forall i \in \mathcal{C},\ \forall t \in [T]. \tag{14}$$

Overall, the problem reads as a compact form

$$\begin{aligned} v^* := \min\ & v(\mathbf{x}) \\ \text{s.t.}\ & \mathcal{X} = \{\mathbf{x} \mid (2)-(14)\}, \end{aligned} \tag{15}$$

where the bounded feasible set is denoted as $\mathcal{X}$.

Remark Note that (4) models the increments of backlogged demands for $t \in [T]$. To our practical knowledge, incorporating backlogged demands typically presents greater challenges compared to the lost-sales setting. In the lost-sales setting, the flow conservation constraint for customers can be expressed as:

$$\sum_{e\in\delta_i^+} x_{e,t,k} + s_{i,t,k} = D_{i,t,k}, \quad \forall i \in \mathcal{C},\ \forall k \in \mathcal{K},\ \forall t \in [T], \tag{16}$$

where $s_{i,t,k}$ represents the lost sales of customer i for commodity k in period t.

If one sets $T = 1$ and focuses solely on the layout and customized service constraints, this problem reduces to a static network planning problem with service requirements (Balakrishnan et al. 2017, Gudapati et al. 2022a). In a dynamic problem with $T > 1$, the inflow upper constraints (11) and the variable lower bounds (12) considerably raise the computational burden. In the following section, we will propose a method to efficiently solve this problem.

4. A Decomposition Reformulation Framework for DNP

The problem defined in (15) cannot be efficiently solved using standard commercial solvers, particularly in practical scenarios like weekly planning problems involving approximately 500 nodes and over 100 commodities. A significant portion of the computational challenge arises from the large number of customers with diverse service requirements and complex operational constraints. If *a priori* delivery plans for customers were available, the problem could be reformulated as a multi-commodity, multi-source, multi-sink minimum-cost flow problem, which is considerably more tractable. This observation motivates the development of a decomposition framework for DNP that addresses each customer's requirements individually, simplifying the overall computational process.

4.1. Reformulations of DNP

Our motivation for exploring reformulations arises naturally from the fact that the customer size is usually larger than the size of the supply network, that is, $|\mathcal{C}| \gg |\mathcal{S} \cup \mathcal{W}|$. Hence, we decompose the original problem into subproblems, each corresponding to an individual customer. This enables a *customized* and *distributed* approach to address the hard requirements (11)-(14) by setting appropriate subproblems and corresponding feasible sets.

4.1.1. Primal Decomposition by Customers To facilitate the discussion on reformulations, we present a modular view in the style of the Dantzig-Wolfe decomposition. Without further clarification, we repetitively use j to denote warehouses or suppliers, while i is for customers. For each customer $i \in \mathcal{C}$, we define a subproblem with corresponding variables $\mathbf{x}_i$ that are specifically tailored to meet the decomposed requirements. Let n_i and b_i denote the number of continuous and binary variables in this subproblem, respectively, and let the feasible set be $\mathbf{x}_i \in \mathcal{X}_i \subseteq \mathbb{R}_+^{n_i} \times \mathbb{B}^{b_i}$. The continuous relaxation of the feasible set $\mathcal{X}_i$ is denoted by $\bar{\mathcal{X}}_i$.

In addition to the variables within $\mathcal{X}_i, \forall i \in \mathcal{C}$, we denote the remaining variables by $\mathbf{x}_0$, which belong to another mixed-integer set with n_0 continuous and b_0 binary variables:

$$\mathcal{X}_0 = \{\mathbf{x}_0 \in \mathbb{R}_+^{n_0} \times \mathbb{B}^{b_0} \mid B_0\mathbf{x}_0 \leq d_0\},$$

where the linear inequalities represented by the matrix B_0 and the vector $d_0 \in \mathbb{R}^{n_0+b_0}$ are constraints that apply exclusively to $\mathbf{x}_0$.

Then the original set $\mathcal{X}$ and the problem (15) can be expressed in the following form:

$$\begin{aligned} \min\ & v_0(\mathbf{x}_0) + \sum_{i\in\mathcal{C}} v_i(\mathbf{x}_i) \\ \text{s.t.}\ & \mathbf{x}_0 \in \mathcal{X}_0,\ \mathbf{x}_i \in \mathcal{X}_i,\ \forall i \in \mathcal{C}, \\ & A_0\mathbf{x}_0 + \sum_{i\in\mathcal{C}} A_i\mathbf{x}_i \le d. \end{aligned} \tag{17}$$

Here, v_0 and v_i denote the cost functions of $\mathbf{x}_0$ and $\mathbf{x}_i$, respectively, which can be derived from the original objective function (1). The matrices A_0 and A_i represent the coupling constraint matrices, and $d \in \mathbb{R}^m$ is the right-hand side vector with m constraints.

We assume that the set $\mathcal{X}_i$ is always bounded, allowing us to rewrite the system in terms of its extreme points $\mathbf{x}_i^1, \ldots, \mathbf{x}_i^{m_i}$:

$$\mathcal{X}_i = \left\{ \sum_{\nu=1}^{m_i} \mathbf{x}_i^\nu \lambda_i^\nu \mid \lambda_i \in \mathbb{B}^{m_i}, \lambda_i^T \mathbf{1} = 1 \right\} \quad \text{and} \quad \tilde{A}_i = [A_i\mathbf{x}_i^1, \ldots, A_i\mathbf{x}_i^{m_i}], \tag{18}$$

so that the cost and the coupling constraints can be rewritten as:

$$\tilde{v}_i(\lambda_i) = \sum_{\nu=1}^{m_i} \lambda_i^\nu \cdot v_i(\mathbf{x}_i^\nu) \quad \text{and} \quad A_0\mathbf{x}_0 + \sum_{i\in\mathcal{C}} \tilde{A}_i\lambda_i \le d. \tag{19}$$

Note that $\tilde{v}_i(\cdot)$ is simply a reformulation of the cost function due to the linearity of $v_i(\cdot)$. We omit recession directions since $\mathcal{X}_i$ is bounded. Given that m_i can be exponentially large for a polyhedral set, we typically work with a limited set of columns and iteratively add new columns from $\mathcal{X}_i$. Namely, at each iteration $n = 1, \ldots, N$, we use a truncated set of columns for each $i \in \mathcal{C}$:

$$\mathcal{X}_i^n = \{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^\nu, \ldots, \mathbf{x}_i^n\},$$

and then introduce the restricted master problem (RMP):

$$\begin{aligned} \text{(RMP)} \quad v_{\text{CG}}^n := \min\ & v_0(\mathbf{x}_0) + \sum_{i\in\mathcal{C}} \tilde{v}_i^n(\lambda_i) \\ \text{s.t.}\ & \mathbf{x}_0 \in \mathcal{X}_0,\ \lambda_i \in \mathbb{B}^n,\ \forall i \in \mathcal{C} \\ & \lambda_i^T \mathbf{1} = 1,\ \forall i \in \mathcal{C} \\ & A_0\mathbf{x}_0 + \sum_{i\in\mathcal{C}} \tilde{A}_i^n \lambda_i \le d. \end{aligned} \tag{20}$$

$\tilde{v}_i^n, \tilde{A}_i^n$ refers to the cost and constraint matrix of customer i at iteration n:

$$\tilde{v}_i^n(\lambda_i) = \sum_{\nu=1}^{n} \lambda_i^\nu \cdot v_i(\mathbf{x}_i^\nu) \qquad \text{and} \quad \tilde{A}_i^n = [A_i\mathbf{x}_i^1, ..., A_i\mathbf{x}_i^n].$$

We also denote the LP relaxation of (20) (rRMP) by relaxing λ_i and the binary variables in $\mathcal{X}_0$

$$\text{(rRMP)} \quad \bar{v}^n_{\text{CG}} := \min \; v_0(\mathbf{x}_0) + \sum_{i\in\mathcal{C}} \tilde{v}^n_i(\lambda_i) \tag{21a}$$

$$\text{s.t.} \;\; \mathbf{x}_0 \in \bar{\mathcal{X}}_0, \; \mathbf{0} \le \lambda_i \le \mathbf{1}, \; \forall i \in \mathcal{C} \tag{21b}$$

$$\lambda_i^T \mathbf{1} = 1, \; \forall i \in \mathcal{C} \tag{21c}$$

$$A_0\mathbf{x}_0 + \sum_{i\in\mathcal{C}} \tilde{A}^n_i \lambda_i \le d. \tag{21d}$$

By solving (21), we obtain $(\theta^n, \pi^n) \in \mathbb{R}^{|\mathcal{C}|} \times \mathbb{R}^m_+$ as the dual variable to (21c) and (21d), respectively.

A new column will be appended to the column set by solving the pricing problem:

$$\text{(P1)} \qquad \mathbf{x}^n_i := \arg\min_{\mathbf{x}_i\in\mathcal{X}_i} \; r^n_i(\mathbf{x}_i) := v_i(\mathbf{x}_i) + \mathbf{x}_i^T A_i^T \pi^n - \theta^n_i \quad \text{and} \quad r^n_i := r^n_i(\mathbf{x}^n_i); \tag{22}$$

Once the reduced cost r^n_i in (22) is non-negative for all customers, we terminate and gain a relaxed solution of (17). Otherwise, we proceed to the next iteration $n+1$.

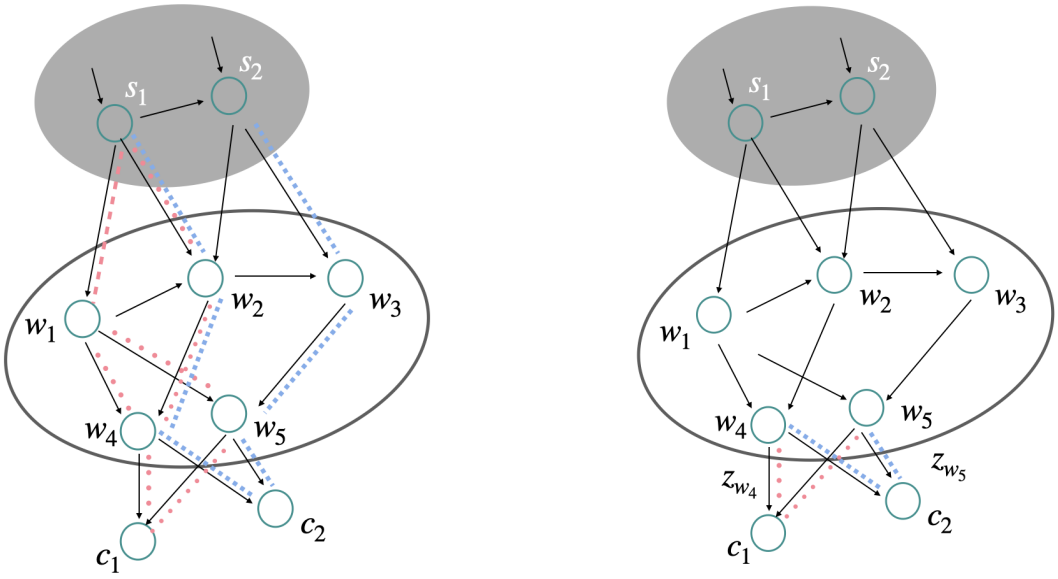
In the rest of this subsection, we explore two different reformulations, and their special representation of $(\mathcal{X}_0, \mathcal{X}_i)$. We discuss a few preliminary polyhedral properties and valid inequalities that improve continuous relaxation. To simplify the presentation, for the ν-th column of customer i at the iteration n, we have $\mathbf{x}^\nu_i \in \mathcal{X}^n_i$ with $\nu \le n$. We denote its corresponding transportation quantities as $x^{[i,\nu]}$ using Matlab-style indexing notation. Similarly, other decision variables such as $\{y, s, I\}$ can be defined in this manner (see Section 3 for the specific definitions).

In the following subsections, we introduce a *fully-customized* (FC) formulation, where for each customer $i \in \mathcal{C}$, the integrated decision from suppliers through warehouses to the customer is modeled as a completely separate subproblem, as depicted in Figure 1a. Although this formulation is intuitive and operates on a smaller subgraph than the entire supply chain network, each subproblem still tackles an integrated problem with all the complex constraints inherent to the customer's subnetwork, leading to high computational costs. Additionally, the subproblems are heterogeneous due to differences in commodities, demand, and required service nodes, leading to significant variability in computation times across subproblems, which can negatively affect parallel implementations.

Next, we propose a *preparation-allocation* (PA) reformulation, as illustrated in Figure 1b, which introduces an additional main block along with a set of easily separable subproblems for each customer. The *preparation* block handles warehouse configuration, inventory placement, and transportation decisions from suppliers to warehouses, focusing only on the combined throughput to

Figure 1 Decomposition and Reformulations of DNP.

(a) Fully customized reformulation. (b) Preparation-allocation reformulation.



Note. The red dashed line represents a column for c_1, while the blue dashed line denotes a column for c_2.

customers $\mathcal{C}$, represented by z_j for all $j \in \mathcal{W}$. This setup allows the *allocation blocks* (serving as the pricing problems) to concentrate solely on delivery plans to $\mathcal{C}$, without considering operational constraints in the upstream supply chain. These allocation blocks are relatively inexpensive to solve for both large and small customers, making them ideal for parallel implementations. The trade-off, however, is the increased complexity of the master problem.

4.1.2. A Fully-Customized Reformulation Note that all constraints (2)-(14) can be efficiently decomposed into separate customer-oriented subproblems, each based on the three-level sub-network associated with the respective customer. To achieve this, we first subdivide the entire supply chain network $\mathcal{G}$ into subnetworks $\{\mathcal{G}_i\}$, each responsible for a specific customer. This is done by collecting connected edges and nodes, effectively solving a connected subgraph using a breadth-first search algorithm using the customer i as the starting node. The feasible set $\mathcal{X}_i$ in this context includes the relevant flow conservation constraints and the customized service constraints for customer i:

$$\mathcal{X}_i = \{\mathbf{x}_i \in \mathbb{R}_+^{n_i} \times \mathbb{B}^{b_i} \mid (2)-(4); (5)-(7); (12)-(14) \quad \text{on} \quad \mathcal{G}_i\}. \tag{23}$$

It is understood that $\mathcal{X}_i$ is defined only on the subgraph $\mathcal{G}_i$. In this reformulation, $\mathcal{X}_0$ in the master problem is simply the domain of definition: $\mathcal{X}_0 = \mathbb{R}_+^{n_0} \times \mathbb{B}^{b_0}$. The binding block contains the

operational constraints (8)-(12) for nodes $\mathcal{S} \cup \mathcal{W}$ and edges connecting $\mathcal{S}, \mathcal{W}$. A subset of (12) is readily included in $\mathcal{X}_i$. To illustrate, the edge capacity constraints (8) can be written in n^{th} iteration:

$$\sum_{k\in\mathcal{K}} \left(\sum_{i\in\mathcal{C}} \sum_{\nu\le n} \lambda_i^{\nu} \cdot x_{e,t,k}^{[i,\nu]} \right) \le C_e y_{e,t}, \quad \forall e \in E,\ \forall t \in [T]. \tag{24}$$

Other rows in the binding blocks (cf. (20)) can be reformulated similarly. Notably, the flow conservation constraints (2)-(4) are no longer needed. This leads to the following proposition.

PROPOSITION 1. *Suppose each generated column is feasible with respect to* (23)*. At each iteration* n*, if* $\lambda_i \ge \mathbf{0}$ *and* $\lambda_i^T \mathbf{1} = 1$ *for all* $i \in \mathcal{C}$*, the combined flow satisfies the conservation constraints* (2)-(4) *for the original network* $\mathcal{G}$*.*

The next observation ensures the convergence of this reformulation under the conditions of no lower-bound requirements and sufficiently large capacities.

PROPOSITION 2. *Suppose the lower bound* $L_e = 0$ *for all* $e \in E$ *without considering layout decisions, and that the node capacity* C_j *(including* S_j *and* U_j*) and edge capacity* C_e *are sufficiently large, i.e.,*

$$C_e \ge \left\lceil \sum_{t\in[T]} \sum_{i\in\mathcal{C}} \sum_{k\in\mathcal{K}} D_{i,t,k} \right\rceil,\ \forall e \in E \quad \textit{and} \quad C_j \ge \left\lceil \sum_{t\in[T]} \sum_{i\in\mathcal{C}} \sum_{k\in\mathcal{K}} D_{i,t,k} \right\rceil,\ \forall j \in \mathcal{S} \cup \mathcal{W},$$

then the basic column generation algorithm will terminate at $n \le 1$*.*

As the RMP (20) employs fully decomposed subproblems for each customer, a significant drawback of this approach is the difficulty in solving a series of highly imbalanced subproblems. Additionally, there are inherent limitations associated with (20), particularly in managing the coupling lower bound constraints for transfer edges connecting nodes in $\mathcal{W}$ and $\mathcal{S}$, as expressed in (12). If the lower bound L_e is non-zero, the bound provided by the column generation procedure can be poor.

4.1.3. A Preparation-Allocation Reformulation To address the aforementioned shortcomings, we consider an alternative reformulation that introduces cheaper subproblems but involves a larger complicating block in the RMP. We define the subgraph needed for each customer $i \in \mathcal{C}$:

$$E_i := \delta_i^+,\ V_i := \{j \in V \mid (j,i) \in E_i\} \cup \mathcal{C},\ \mathcal{G}_i = (V_i, E_i);\ E_{\mathcal{C}} = \bigcup_{i\in\mathcal{C}} E_i, \tag{25}$$

and the graph for the preparation block can be defined as $\mathcal{G}_0 = (V \setminus \mathcal{C}, E \setminus E_{\mathcal{C}})$.

We first describe the master problem, where the flow conservation constraint is reformulated using an additional variable $z_{j,t,k}$, which represents the total flow from warehouse $j \in \mathcal{W}$ to customers $\mathcal{C}$ for each commodity k in period t. It can be seen as the quantity that flows "out" from the subgraph $\mathcal{G}_0$.

$$q_{j,t,k} = \sum_{e \in \delta_j^-} x_{e,t,k}, \quad \forall j \in \mathcal{S},\ \forall k \in \mathcal{K},\ \forall t \in [T], \tag{26}$$

$$I_{j,t-1,k} + \sum_{e \in \delta_j^+} x_{e,t,k} - \sum_{e \in \delta_j^- \setminus E_{\mathcal{C}}} x_{e,t,k} - z_{j,t,k} = I_{j,t,k}, \quad \forall j \in \mathcal{W},\ \forall k \in \mathcal{K},\ \forall t \in [T], \tag{27}$$

$$z_{j,t,k} - \sum_{e \in \delta_j^- \bigcap E_{\mathcal{C}}} x_{e,t,k} = 0, \quad \forall j \in \mathcal{W},\ \forall k \in \mathcal{K},\ \forall t \in [T]. \tag{28}$$

Summarizing, at iteration n, we have:

$$\mathcal{X}_0 = \left\{ \mathbf{x}_0 \in \mathbb{R}_+^{n_0} \times \mathbb{B}^{b_0} \mid (5) - (12);\ (26) - (28) \quad \text{on} \quad \mathcal{G}_0 \right\}.$$

The feasible set of the restricted master problem (21) is defined as:

$$\mathbf{x}_0 \in \bar{\mathcal{X}}_0,\ \mathbf{0} \le \lambda_i \le \mathbf{1}, \quad \forall i \in \mathcal{C}, \tag{29a}$$

$$\lambda_i^T \mathbf{1} = 1, \quad \forall i \in \mathcal{C}, \tag{29b}$$

$$z_{j,t,k} - \sum_{(j,i) \in \delta_j^- \cap E_{\mathcal{C}}} \sum_{\nu \le n} \lambda_i^\nu \cdot x_{(j,i),t,k}^{[i,\nu]} = 0, \quad \forall j \in \mathcal{W},\ \forall k \in \mathcal{K},\ \forall t \in [T]. \tag{29c}$$

The master problem essentially excludes (12)-(14) on $\mathcal{G}_i$; the binding constraint (29c) means the outflow z meets with total "allocated" quantities to customers. The pricing problem for each customer i generates a delivery plan of such "allocated" quantities that accommodates any required customizations. The feasible set $\mathcal{X}_i$ in this case is denoted as:

$$\mathcal{X}_i = \{ \mathbf{x}_i \in \mathbb{R}_+^{n_i} \times \mathbb{B}^{b_i} \mid (28); (4); (8); (12) - (14) \quad \text{on} \quad \mathcal{G}_i \}$$

These constraints are easy to solve and $\mathcal{G}_i$ is now simple (from i to its adjacent nodes, a subset of $\mathcal{W}$). Similarly, let $\pi_{j,t,k}^n$ be the dual variable associated with (29c) and θ_i^n be the dual variable of (29b). The pricing problem for each customer $i \in \mathcal{C}$ at iteration n is as follows:

$$\text{(P2)} \qquad \mathbf{x}_i^n := \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} r_i^n(\mathbf{x}_i) := v_i(\mathbf{x}_i) + \sum_{t \in [T]} \sum_{j \in \mathcal{W}} \sum_{(j,i) \in \delta_j^- \bigcap E_{\mathcal{C}}} \sum_{k \in \mathcal{K}} \pi_{j,t,k}^n x_{(j,i),t,k}^{[i,\nu]} - \theta_i^n; \tag{30}$$

The lengthy expressions in cost functions are omitted for brevity since they can be straightforwardly derived from the original problem.

4.1.4. Valid Inequalities for the Restricted Master Problem When solving the LP relaxation of RMP, the variable lower bound constraints (12) may be violated. Since these constraints are naturally separated by edges connecting customers, RMP only has to handle the coupling constraints (12) in the subgraph without customers. Specifically, recall the definition of $E_{\mathcal{C}}$, the RMP focuses on:

$$\sum_{k\in\mathcal{K}} x_{e,t,k} \geq L_e y_{e,t},\ \forall e \notin E_{\mathcal{C}}. \tag{31}$$

Cutting planes can be introduced into the RMP to ensure the effectiveness of the coupling constraints. We utilize flow cover inequalities for the so-called single-node fixed cost set, a special case of mixed 0-1 rows (Van Roy and Wolsey 1987):

$$F = \left\{ (x,y) : \sum_{j\in N_-} x_j - \sum_{j\in N_+} x_j \leq d,\ l_j y_j \leq x_j \leq u_j y_j,\ y_j \in \mathbb{B},\ j \in N_+ \cup N_- \right\}. \tag{32}$$

Clearly, if we take $N_+ \cup N_-$ as the set of inflow and outflow edges, then F describes the set of feasible flow allocations x_j paired with their corresponding indicator variables y_j, for all $j \in N_+ \cup N_-$.

An intuitive idea is to construct (32) from the inventory inequalities (3). The crux to do so is to introduce (a) artificial edges and (b) a surrogate operator $\ell(\cdot)$ so that the variable lower bound inequalities can be modeled in a similar fashion of (32). We first assume the ending inventory of warehouse j (where $j \in \mathcal{W}$) flows into some j_-, extending δ_j^- to include (j, j_-). Similarly, if the inventory at the beginning of period t flows from some j_+, we have $\delta_j^+ = \delta_j^+ \cup (j_+, j), , \forall j \in \mathcal{W}$. We also assume that the production quantity of the supplier j (where $j \in \mathcal{S}$) flows from some j_+, we have $\delta_j^+ = \delta_j^+ \cup (j_+, j), \forall j \in \mathcal{S}$. For any $e \in E$ and $t \in [T]$, the surrogate operator $\ell(\cdot)$ can be applied in two ways: $\ell(y)$ for an auxiliary binary variable y, and $\ell(x)$ for a continuous variable x. Specifically, we consider five possible usages:

- When $e \in \delta_j^- \backslash E_{\mathcal{C}}$ and $e = (j, j_-)$ is a virtual edge, take $\ell_{e,t,k}(x) = I_{j,t,k}$ and $\ell_{e,t,k}(y) = 1$.
- When $e \in \delta_j^- \backslash E_{\mathcal{C}}$ and e is a normal edge, take $\ell_{e,t,k}(x) = x_{e,t,k}$ and $\ell_{e,t,k}(y) = y_{e,t,k}$.
- When $e \in \delta_j^+ \backslash E_{\mathcal{C}}$ and $(j_+, j), \forall j \in \mathcal{W}$ is a virtual edge, take $\ell_{e,t,k}(x) = I_{j,t-1,k}$ and $\ell_{e,t,k}(y) = 1$.
- When $e \in \delta_j^+ \backslash E_{\mathcal{C}}$ and $(j_+, j), \forall j \in \mathcal{S}$ is a virtual edge, take $\ell_{e,t,k}(x) = q_{j,t,k}$ and $\ell_{e,t,k}(y) = 1$.
- When $e \in \delta_j^+ \backslash E_{\mathcal{C}}$ and e is a normal edge, take $\ell_{e,t,k}(x) = x_{e,t,k}$ and $\ell_{e,t,k}(y) = y_{e,t,k}$.

PROPOSITION 3. *By the above abstraction, we can extract a system from the RMP in the following form.*

$$\begin{aligned}
\sum_{e\in\delta_j^+\setminus E_{\mathcal{C}}} \ell_{e,t,k}(x) - \sum_{e\in\delta_j^-\setminus E_{\mathcal{C}}} \ell_{e,t,k}(x) \le Q_{j,t,k}, \quad \forall j\in\mathcal{W}\cup\mathcal{S},\ \forall k\in\mathcal{K},\ \forall t\in[T],\\
L_e\ell_{e,t,k}(y) \le \ell_{e,t,k}(x) \le C_e\ell_{e,t,k}(y), \quad \forall e\in(\delta_j^+\cup\delta_j^-)\setminus E_{\mathcal{C}},\ \forall k\in\mathcal{K},\ \forall t\in[T],\\
\ell_{e,t,k}(y)\in\mathbb{B}, \ell_{e,t,k}(x)\ge 0, \quad \forall e\in(\delta_j^+\cup\delta_j^-)\setminus E_{\mathcal{C}},\ \forall k\in\mathcal{K},\ \forall t\in[T],
\end{aligned} \tag{33}$$

where $Q_{j,t,k}=0, \forall j\in\mathcal{S}$. *When considering the backlogged demands, we have* $Q_{j,t,k}=\sum_{t'\in[t]}\sum_{i\in\mathcal{C}} d_{i,t',k}, \forall j\in\mathcal{W}$, *while* $Q_{j,t,k}=\sum_{i\in\mathcal{C}} d_{i,t',k}, \forall j\in\mathcal{W}$ *in the lost-sales setting.*

The following generalized flow cover inequality is readily available. We introduce the following proposition.

PROPOSITION 4. *For* $\forall t\in[T], \forall j\in\mathcal{W}, \forall k\in\mathcal{K}$, *for any* $C_1\subseteq\delta_j^+, C_2\subseteq\delta_j^-, R\subseteq(\delta_j^+\cup\delta_j^-), Q_2\subseteq\delta_j^-\setminus C_2$, *the cutting plane for each time period* t *and commodity* k *:*

$$\begin{aligned}
&\sum_{e\in C_1\setminus R}\left[\ell_{e,t,k}(x)+\max\{C_e-\Lambda,0\}\left(1-\ell_{e,t,k}(y)\right)\right]+\sum_{e\in C_1\cap R}\left[\max\{L_e-\Lambda,0\}+\min\left(L_e,\Lambda\right)\ell_{e,t,k}(y)\right]\\
&-\sum_{e\in C_2\setminus R}\left[\ell_{e,t,k}(x)+L_e\left(1-\ell_{e,t,k}(y)\right)\right]-\sum_{e\in C_2\cap R} C_e\\
&-\sum_{e\in Q_2\setminus R}\left[\ell_{e,t,k}(x)-\max\{L_e-\Lambda,0\}\ell_{e,t,k}(y)\right]-\sum_{e\in Q_2\cap R}\min\left(C_e,\Lambda\right)\ell_{e,t,k}(y)\\
&-\sum_{e\in\delta_j^-\setminus(C_2\cup Q_2)}\ell_{e,t,k}(x)\ -\ Q_{j,t,k}\le 0
\end{aligned} \tag{34}$$

is valid for (33) *if the following condition holds,*

$$\Lambda=\sum_{e\in C_1\cap R} L_e+\sum_{e\in C_1\setminus R} C_e-\sum_{e\in C_2\cap R} C_e-\sum_{e\in C_2\setminus R} L_e-Q_{j,t,k}>0.$$

For succinctness, a companion separation problem for constructing appropriate cutting planes is relegated to (EC.15).

Although a cutting plane in the form of (34) is expressed by the original variables x, y, it can be easily re-expressed if an alternative formulation is involved. For example, for the fully customized reformulation, one can facilitate a similar inequality by replacing $\ell_{e,t,k}(x)=\sum_{\nu,i}\lambda_i^\nu x_{e,t,k}^{[i,\nu]}$ and $\ell_{e,t,k}(y)=\sum_{\nu,i}\lambda_i^\nu y_{e,t,k}^{[i,\nu]}$ for normal edges. For virtual edges, we can replace $\ell_{e,t,k}(x)=\sum_{\nu,i}\lambda_i^\nu I_{i,t,k}^{[i,\nu]}$ or $\sum_{\nu,i}\lambda_i^\nu I_{i,t-1,k}^{[i,\nu]}$ or $\sum_{\nu,i}\lambda_i^\nu q_{i,t,k}^{[i,\nu]}$ and $\ell_{e,t,k}(y)=1$.

4.2. Implementation Details

For practical performance, we implement the customer-oriented column generation algorithm in Ray (Moritz et al. 2018), which is an open-source distributed computing framework. We exploit techniques including parallelization, efficient initialization, and rounding schemes to improve the numerical efficacy in Algorithm 1.

Algorithm 1: A Distributed Customer-Oriented Column Generation Framework

Input: Maximum iteration number $N_{\max}$, fixed-point tolerance ε_f, pricing tolerance ε_p;
The graph $\mathcal{G}$ and the set of customers $\mathcal{C}$, available distribution workers number P.

Initialize a distributed pool of P workers;
Obtain the thread-customer mapping $\mathcal{T}_p$,

$$\mathcal{T}_p \subset \mathcal{C}, p = 1, ..., P; \quad \bigcup_{p=1}^{P} \mathcal{T}_p = \mathcal{C};$$

for $p \in P$ **do in parallel**
  forall $i \in \mathcal{T}_p$ ***sequentially*** **do**
    Create the MIP oracle $\mathcal{X}_i$ for the pricing problem.
  end
end

Initialize $\mathcal{X}_i^0 = \emptyset,\ \forall i \in \mathcal{C}$; `// Optional, by Algorithm 2.`
forall $n = 0, \ldots, N_{\max}$ **do**
  Solve the RMP (20) based on $\{\mathcal{X}_i^n, i \in \mathcal{C}\}$;
  Record the optimal value v_{CG}^n and the dual solution (π^n, θ^n);

  for $p \in P$ **do in parallel**
    forall $i \in \mathcal{T}_p$ ***sequentially*** **do**
      Solve the pricing problem (22) with the optimal value r_i^n and the column $\mathbf{x}_i^n$;
      if $r_i^n < -\varepsilon_p$ **then**
        Append the new column: $\mathcal{X}_i^{n+1} := \mathcal{X}_i^n \cup \{\mathbf{x}_i^n\}$;
      end
    end
  end

  if $\min_{i \in \mathcal{C}}\{r_i^n\} \geq -\varepsilon_p$ *and* $\frac{|v_{\mathrm{CG}}^n - v_{\mathrm{CG}}^{n-1}|}{|v_{\mathrm{CG}}^n|+1} \leq \varepsilon_f$ **then**
    Record all columns, and current weights $\{\lambda_i,\ \forall i \in \mathcal{C}\}$;
    Break; `// Column generation phase terminated.`
  end
end
Calculate the integer solutions; `// Optional, by Algorithm 3.`
Return: feasible set $\{\mathbf{x}_i \in \mathcal{X}_i, i \in \mathcal{C}\}$ and $\mathbf{x}_0 \in \mathcal{X}_0$.

We make a few comments regarding the approach. Both initialization, storage, and solving of the subproblems can be distributed across P workers, as shown by the shaded lines in Algorithm 1. Ideally, the performance of a distributed method should have a similar overhead to solving a problem

with $|\mathcal{C}|/P$ customers when addressing the pricing problems. To initialize the columns (Line 3), one can start with "empty" columns or use a set of iteratively constructed feasible columns generated by Algorithm 2. To accelerate the solving of the RMP (Line 5), various ad-hoc strategies can be employed. It is important to note that the column generation method without an integral RMP only provides a relaxation of the problem. When all reduced costs are nonnegative, we switch to an integral version (Line 13) to obtain feasible solutions. For medium-sized problems (fewer than 100 customers), a direct MILP solver can be used. For larger problems, a simple rounding algorithm that converges within a few additional iterations (Algorithm 3) is recommended.

4.2.1. Initialization and Selection of Columns Motivated by the coupling constraints (8)-(10), we design a simple algorithm that sweeps the customers and iteratively occupies the capacity. For each customer $i \in \mathcal{C}$, we solve the subproblem by satisfying other remaining constraints. Then, we lower the RHS in (8)-(10) by its used resources. We proceed to the next customer and terminate if the capacity is fully occupied. Obviously, the procedure produces a primal feasible solution. This provides a set of initial columns.

Algorithm 2: A sweeping initialization algorithm

Input: Initialize remaining capacities: $b_j, j \in V$, and $b_e, e \in E$
1 **forall** $i \in \mathcal{C}$ **do**
2 Solve the problem (15) for *only* i with constraints in $\mathcal{X}_i$ excluding (8)-(10);
3 Update remaining capacities:

$$C_e \leftarrow C_e - \sum_{k\in\mathcal{K}} \sum_{t\in[T]} x^{[i,0]}_{e,t,k}, \quad \forall e \in E_i \backslash E_{\mathcal{C}};$$

$$C_j \leftarrow C_j - \sum_{k\in\mathcal{K}} \sum_{t\in[T]} x^{[i,0]}_{j,t,k}, \quad \forall j \in V_i;$$

4 **end**
5 **Return:** initial columns $\{\mathbf{x}^0_i, i \in \mathcal{C}\}$

As the column generation algorithm iterates, the increasing number of columns generated by the pricing problem can make the RMP challenging to solve. Observing the solution patterns, we notice that the column weights λ are highly sparse, suggesting that only a small subset of the generated columns contribute valuable information.

Thus, it is natural to dynamically remove non-essential columns and retain only those that tighten the bound. The key challenge is defining which columns are useful. For example, we can manage the *usage* of the columns and delete column x based on its historical column weights, e.g., if its historical column weight in the last iteration falls below a predefined threshold.

4.2.2. A Rounding Heuristic for Integral Solutions We introduce a rounding heuristic method after the column generation phase is terminated. For simplicity, we let $\mathbf{y}_0 \in \mathbb{B}^{m_0}$ be all binary variables in $\mathbf{x}_0$. The purpose of the rounding scheme is to provide binary $\mathbf{y}_0$ and $\{\lambda_i,\ i \in \mathcal{C}\}$. Note if $\lambda_i \in \mathbb{B}^{m_i}$ holds for all $i \in \mathcal{C}$, the solution $\mathbf{x} = (\mathbf{x}_0, \{\mathbf{x}_i\}_{i\in\mathcal{C}})$ is an integral feasible solution.

At a high level, the heuristic collects the weights $\forall i \in \mathcal{C}, \lambda_i \in \mathbb{R}^n_+$ in (29) terminated at some iterate n. With a slight abuse of notations, we iteratively update the set $B^n, n = 1, ..., N_{\max}$, the current set of customers, whose weights readily belong to $\mathbb{B}$,

$$B^n = \{i \in \mathcal{C} \mid \exists \lambda_i^\nu \in \mathbb{B}\} \quad \text{and} \quad d_i = \tilde{A}_i \lambda_i, \forall i \in \mathcal{C}. \tag{35}$$

And the rest belongs to Q^n, the *working set* for the rounding method. We fix the solution in B^n, then impose the following penalized problem that operates on Q^n:

$$\varphi(Q^n) := \min\ v_0(\mathbf{x}_0) + \rho_\lambda \cdot \sum_{i\in Q^n} \sum_{\nu=1}^{n} v_i^\nu \cdot (1 - \lambda_i^\nu) \tag{36a}$$

$$\text{s.t.}\ \ \mathbf{x}_0 \in \mathcal{X}_0,\ \mathbf{0} \le \lambda_i \le \mathbf{1},\ \forall i \in Q^n \tag{36b}$$

$$\lambda_i^T \mathbf{1} \le 1,\ \forall i \in Q^n \tag{36c}$$

$$A_0 \mathbf{x}_0 + \sum_{i\in Q^n} \tilde{A}_i \lambda_i \le d - \sum_{i\in B^n} d_i. \tag{36d}$$

In (36), ρ_λ is the penalty coefficient. The problem is reduced since we enlarge B at each at each iteration. We present the method in Algorithm 3.

Algorithm 3: A rounding method for integer feasible solutions

Input: Maximum iteration count $N_{\max}$
Initialize binary variable set B^1 and Q^1 (35);
forall $n = 1, ..., N_{\max}$ **do**
  Solve (36) and collect $\lambda_i, \forall i \in \mathcal{C}$;
  Update the variable sets B^{n+1}, Q^{n+1} by (35);
  if $|Q^{n+1}| \ge |Q^n|$ **then**
    Set $\lambda_i = \mathbf{0}, \forall i \in Q^{n+1}$; `// No room for improvement`
    Break;
  end
  if $Q^{n+1} = \emptyset$ **then**
    Break; `// Finite convergence`
  end
end
Return: integer solution $\{\lambda_i, i \in \mathcal{C}\}$ and $\mathbf{y}_0$.

Note that at each iteration, (36) is a mixed-integer program. Due to the independence of $i \in \mathcal{C}$, it is evident that Algorithm 3 converges in a finite number of steps. At each step n, $\lambda_i = \mathbf{0}$ for all

$i \in Q^n$ is a trivial feasible solution. If $|Q^{n+1}| \geq |Q^n|$, there is no further room for improvement, and the algorithm terminates. Otherwise, $|Q^n|$ decreases strictly, ensuring finite convergence. In practice, the use of mixed-integer programs does not slow down the approach (see Table 7); for large problems, the total runtime of Algorithm 3 is typically a few tens of seconds.

5. Case Study

Our computational experiments explore three main aspects. Firstly, we compare the computational performance of our proposed algorithm with that of a state-of-the-art solver using practical datasets from VX Logistics and synthetic datasets. Our ablation study also assesses the impact of problem scales and constraint configurations on the overall difficulty of the problem. In several scenarios, we highlight the advantages of the preparation-allocation reformulation approach compared to the fully customized reformulation regarding the quality of the solution and computational times. We also evaluate the improvement effects of integrating various algorithmic strategies into the preparation-allocation method. Finally, we conduct a sensitivity analysis on the impact of critical parameters in the rounding algorithm.

5.1. Experimental Setup

All experiments are conducted on an Ubuntu 20.04 Desktop with Intel Xeon Gold 6226R 2.90GHz CPU processors with 32 cores (64 total) and 256GB DDR4 3200 MHz RAM. Both pricing problems and RMP in Algorithm 1 are solved by Gurobi $10.0.1$. We use 24 distributed workers implemented in Ray (Moritz et al. 2018). We also use Gurobi to directly solve different DNP instances as benchmarks, with $10,000$ seconds time limit and 0.01% relative integrality gap. For Algorithm 1, we use PA (Subsection 4.1.3) as the default reformulation, with a termination criteria of $N_{\max} = 25, \varepsilon_p = 10^{-4}, \varepsilon_f = 10^{-5}$.

In real-life operations, weekly planning with $|T| = 7$ plays a critical role, which, thus, is the setting throughout our experiments. To fully demonstrate the superiority and robustness of our methods, we have conducted extensive experiments on a wide range of instances, which differ in the following attributes:

(1) **Network topology** (`G`). We evaluate our methods on both synthetic and real-world datasets. The synthetic data is generated from the supply chain networks in the U.S. and China, resulting in two heterogeneous graphs denoted as $\mathtt{S}_1$ and $\mathtt{S}_2$, respectively. The real-world data sources are from VX Logistics, a Vanke Group subsidiary. We rank the network structures from easy to complex, denoted as $\mathtt{R}_1$, $\mathtt{R}_2$, and $\mathtt{R}_3$.

(2) **Demand type** (D). For synthetic datasets, demand types range from 1 to 5 and represent particular demand patterns like seasonal demands, whereas, in real-world datasets, it is marked with $\mathtt{D} = 0$.

(3) **The number of customers** (C). It is a critical factor influencing the scale and difficulty of the problem. Therefore, instances with different numbers of customers are evaluated, with $|\mathcal{S}| = 30$, $|\mathcal{W}| = 25$, $|\mathcal{K}| = 140$ unless otherwise stated.

(4) **Constraint configuration** (CC). The parameter represents different combinations of constraints and takes values in $\{\mathtt{fo}, \mathtt{foc}, \mathtt{focb}\}$. The "fo" configuration represents "flows and layout constraints" related to "operational constraints", specifically tied to the lost-sales constraint (16); the "foc" additionally adds "customized service constraints"; the "focb" additionally adds "backlogged constraints".

Complete descriptions on generating the synthetic data are relegated to Section EC.3.

To summarize, we can use the following notation

$$\mathtt{G/D/C/CC}$$

to identify the different instances. Since the real-world datasets have 519 customer nodes, we present the problem size with $\mathtt{C} = 519$ after presolving by Gurobi, including the number of rows N_{row}, columns N_{col}, integer variables N_{int} and nonzeros N_{non} (See Table 1). Synthetic datasets generally turn out to be more difficult than datasets based on real-world topology.

Table 1 Size of Problem Instances After Presolving

No	N_{row}	N_{col}	N_{int}	N_{non}
$\mathtt{S}_1\mathtt{/1/519/foc}$	7,665,450	15,651,406	101,822	74,464,974
$\mathtt{S}_1\mathtt{/2/519/foc}$	8,281,776	15,649,525	101,822	75,646,839
$\mathtt{S}_1\mathtt{/3/519/foc}$	9,227,549	15,640,020	101,822	77,281,750
$\mathtt{S}_1\mathtt{/4/519/foc}$	5,855,912	15,649,505	101,822	70,794,571
$\mathtt{S}_1\mathtt{/5/519/foc}$	8,194,745	15,648,823	101,822	75,453,823
$\mathtt{S}_2\mathtt{/1/519/foc}$	1,893,155	7,250,451	16,371	26,039,065
$\mathtt{S}_2\mathtt{/2/519/foc}$	1,949,693	7,236,351	16,271	26,093,634
$\mathtt{S}_2\mathtt{/3/519/foc}$	1,999,037	7,247,329	16,349	26,233,513
$\mathtt{S}_2\mathtt{/4/519/foc}$	1,672,866	7,256,937	16,417	25,628,022
$\mathtt{S}_2\mathtt{/5/519/foc}$	1,954,993	7,245,798	16,338	26,146,357
$\mathtt{R}_1\mathtt{/0/519/foc}$	332,741	1,134,052	13,433	3,937,060
$\mathtt{R}_2\mathtt{/0/519/foc}$	344,336	1,169,199	13,584	4,058,769
$\mathtt{R}_3\mathtt{/0/519/foc}$	2,036,832	7,215,488	16,128	26,175,456

Metrics Before defining the specific metrics, we introduce some notations used in the formulas. The subscript "Alg" denotes different solution algorithms: "Gur" (for Gurobi), "PA" (for Algorithm 1, based on the Preparation-Allocation Reformulation), and "FA" (for Algorithm 1, based on the Fully-Customized Reformulation). The term $z^{\text{UB}}_{\text{Alg}}$ represents the objective value of the best-known feasible integer solution obtained by the respective method. Meanwhile, $z^{\text{LB}}_{\text{Alg}}$ denotes the lower bound on the best achievable objective value, either from Gurobi's optimal bound or the Lagrangian relaxation bound from the column generation process. The main evaluation metrics include:

(1) The Total Running Time T_{Alg}: Measured in seconds, this metric includes both the time spent on solving the column generation process ($T^{\text{C}}_{\text{Alg}}$) and the rounding phase ($T^{\text{R}}_{\text{Alg}}$).

(2) Acceleration Ratio in Running Time:

$$R_{\text{Acc}} = \left(1 - \frac{T_{\text{Alg1}}}{T_{\text{Alg2}}}\right) \times 100\%,$$

where "Alg1" and "Alg2" can take values from $\{\text{Gur}, \text{PA}, \text{FA}\}$.

(3) Absolute Integer Optimality Gap: This metric evaluates how closely the algorithm's solution approximates the optimal value achieved by Gurobi.

$$G_{\text{Alg}} = \frac{z^{\text{UB}}_{\text{Alg}} - z^{\text{LB}}_{\text{Gur}}}{z^{\text{LB}}_{\text{Gur}}} \times 100\%.$$

(4) Relative Integer Optimality Gap: This metric measures the difference between the obtained solution and its own best-known bound.

$$G^{\star}_{\text{Alg}} = \frac{z^{\text{UB}}_{\text{Alg}} - z^{\text{LB}}_{\text{Alg}}}{z^{\text{LB}}_{\text{Alg}}} \times 100\%.$$

5.2. Main Results

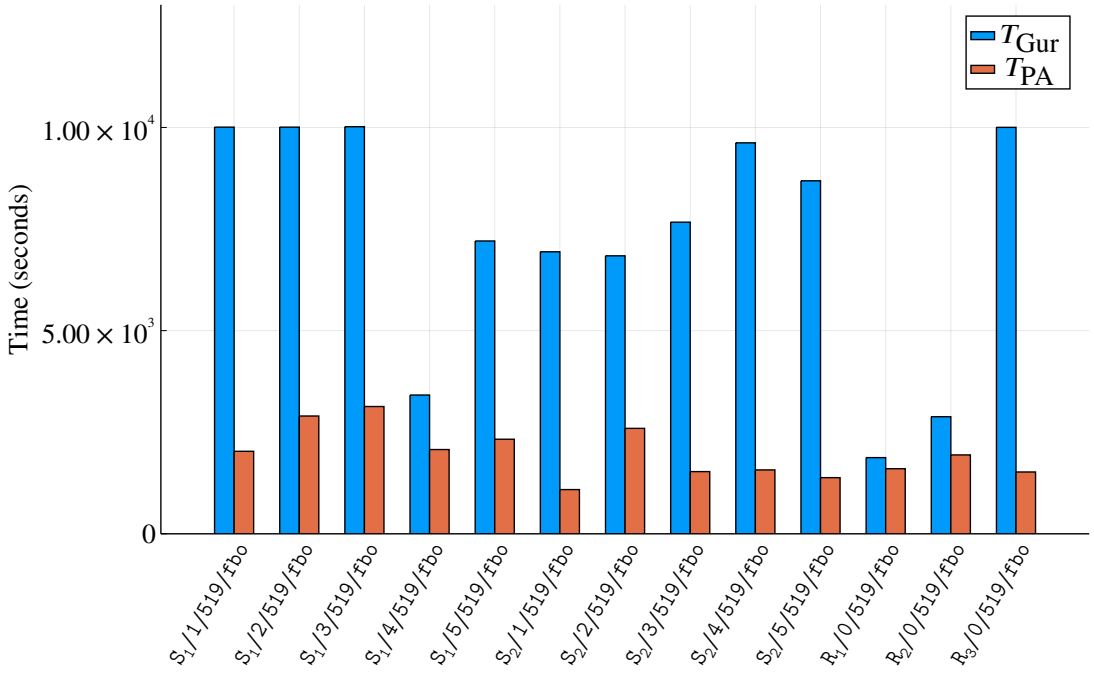
We compare our proposed methods with Gurobi across all instances, as detailed in Table 2. For instances where the Gurobi fails to find a solution within a 1% integer optimality gap, the "G_{PA}" is marked with a "-", as the best bound provided by Gurobi is not meaningful. The column "$T_{\leq 1\%}$" denotes the required time for the Gurobi to gain an integer feasible solution within 1% optimality gap.

The results in Table 2 demonstrate that Algorithm 1 successfully solves all 13 instances with an average absolute integer optimality gap of approximately 0.70%. In contrast, Gurobi only solves 8 out of 13 instances within the $10,000$ second time limit. Additionally, Algorithm 1 achieves an average reduction of about 65.59% in computation time compared to Gurobi. Our algorithm exhibits

Table 2 Performance Comparison between Gurobi and NCS on Different Instances

No	Gurobi					Algorithm 1					R_{Acc}
	$z^{\text{UB}}_{\text{Gur}}$	$z^{\text{LB}}_{\text{Gur}}$	G_{Gur}	$T_{\leq 1\%}$	T_{Gur}	$z^{\text{UB}}_{\text{PA}}$	$z^{\text{LB}}_{\text{PA}}$	G_{PA}	$G^{\star}_{\text{PA}}$	T_{PA}	
S1/1/519/foc	7.46E+11	3.97E+08	99.95%	-	10,007.50	7.20E+11	7.16E+11	-	0.63%	2,030.31	79.71%
S1/2/519/foc	7.22E+11	3.14E+09	99.57%	-	10,007.50	6.96E+11	6.92E+11	-	0.64%	2,898.67	71.04%
S1/3/519/foc	7.90E+11	4.37E+10	94.47%	-	10,018.03	7.62E+11	7.60E+11	-	0.25%	3,132.17	68.73%
S1/4/519/foc	1.20E+12	1.20E+12	0.00%	3,410	3,414.97	1.21E+12	1.20E+12	0.68%	0.68%	2,073.50	39.28%
S1/5/519/foc	7.25E+11	7.32E+09	98.99%	-	7,206.00	6.97E+11	6.95E+11	-	0.27%	2,327.36	67.70%
S2/1/519/foc	6.75E+11	6.75E+11	0.00%	6,937	6,940.03	6.78E+11	6.75E+11	0.43%	0.43%	1,088.31	84.32%
S2/2/519/foc	6.52E+11	6.52E+11	0.00%	6,838	6,840.76	6.54E+11	6.52E+11	0.29%	0.29%	2,594.77	62.07%
S2/3/519/foc	6.01E+11	6.01E+11	0.00%	7,666	7,669.15	6.03E+11	6.01E+11	0.38%	0.38%	1,530.78	80.04%
S2/4/519/foc	1.17E+12	1.17E+12	0.00%	9,617	9,621.20	1.17E+12	1.17E+12	0.51%	0.51%	1,572.30	83.66%
S2/5/519/foc	6.35E+11	6.35E+11	0.00%	8685	8,687.05	6.37E+11	6.35E+11	0.42%	0.42%	1,382.01	84.09%
R1/0/519/foc	8.90E+10	8.90E+10	0.00%	1,873	1,874.76	8.90E+10	8.90E+10	0.01%	0.00%	1,601.65	14.57%
R2/0/519/foc	8.95E+10	8.95E+10	0.00%	2,879	2,881.73	9.20E+10	8.95E+10	2.89%	2.88%	1,941.19	32.64%
R3/0/519/foc	7.73E+11	-3.68E+10	104.76%	-	10,002.19	7.45E+11	7.40E+11	-	0.73%	1,521.21	84.79%

Figure 2 Running time of Gurobi (T_{Gur}) and Algorithm 1 (T_{PA}) over different instances.



substantially improved performance on more complex synthetic datasets, reaching an average acceleration ratio of 72.06%. This highlights its potential effectiveness for tackling more challenging

scenarios across other companies. Overall, Algorithm 1 consistently outperforms Gurobi regarding both solution quality and computational efficiency, particularly for more difficult instances.

5.3. Ablation Study

To study the impact of different improvements over the baseline algorithm, we performed an ablation study, including the reformulations (Section 4.1) and the rounding algorithm (Algorithm 3). We are also interested in the consequences of different problem sizes and constraint configurations and the performance of our method therein.

5.3.1. Performance of different problem sizes and configurations The number of customers significantly affects the scale and complexity of the problem. To demonstrate the robustness and efficiency of our methods, we evaluate them on two practical datasets with varying numbers of customers, as shown in Table 3. We study two series of the problems identified by $\texttt{R}_1, \texttt{R}_3$, where we increase customers from 400 to 519. Additional instances with the `focb` configuration are included to assess the impact of backlogged constraints (4).

Table 3 Performance Comparison between Gurobi and Algorithm 1

	Gurobi					Algorithm 1					R_{Acc}
	$z^{\text{UB}}_{\text{Gur}}$	$z^{\text{LB}}_{\text{Gur}}$	G_{Gur}	$T_{\leq 1\%}$	T_{Gur}	$z^{\text{UB}}_{\text{PA}}$	$z^{\text{LB}}_{\text{PA}}$	G_{PA}	$G^{\star}_{\text{PA}}$	T_{PA}	
$\texttt{R}_1$/0/400/foc	6.24E+10	6.24E+10	0.00%	1009	1010.98	6.30E+10	6.25E+10	0.88%	0.84%	1180.46	-16.76%
$\texttt{R}_1$/0/519/foc	8.90E+10	8.90E+10	0.00%	1873	1874.76	8.90E+10	8.90E+10	0.01%	0.00%	1601.65	14.57%
$\texttt{R}_3$/0/400/foc	5.96E+11	5.33E+11	10.59%	-	10002.72	5.74E+11	5.72E+11	-	0.48%	529.50	94.71%
$\texttt{R}_3$/0/519/foc	7.73E+11	-3.68E+10	104.76%	-	10002.19	7.45E+11	7.40E+11	-	0.73%	1521.21	84.79%
$\texttt{R}_1$/0/400/focb	2.82E+11	2.82E+11	0.00%	9869	9871.26	2.85E+11	2.82E+11	1.11%	1.09%	2711.07	72.54%
$\texttt{R}_1$/0/519/focb	3.94E+11	3.94E+11	0.00%	4546	4548.22	4.26E+11	3.94E+11	8.16%	8.15%	2115.53	53.49%
$\texttt{R}_3$/0/400/focb	3.58E+12	9.63E+10	97.31%	-	10001.65	3.42E+12	3.41E+12	-	0.32%	1028.23	89.72%
$\texttt{R}_3$/0/519/focb	4.65E+12	1.45E+11	96.89%	-	10002.96	4.41E+12	4.40E+12	-	0.35%	6644.71	33.57%

The results in Table 3 indicate that as the number of customers increases and backlogged constraints are introduced, the Gurobi solver struggles to achieve optimality within the specified time limits. In contrast, our proposed algorithm (Algorithm 1) demonstrates robust performance across different configurations and scales, showing significant improvements in computational efficiency and solution quality in 7 out of 8 instances. While the Gurobi solver remains a suitable option for simpler cases, such as $\texttt{R}_1$/0/400/foc, these results underscore our algorithm's clear advantage in handling more complex large-scale real-world problems.

5.3.2. Performance of different reformulations In the following table, the "$z_{\text{PA}}^{\text{LB}}$" and "$z_{\text{FA}}^{\text{LB}}$" columns represent the Lagrangian relaxation bound obtained by the two reformulations. The "$T_{\text{PA}}^{\text{RMP}}$", "$T_{\text{PA}}^{\text{Pricing}}$", and "$T_{\text{PA}}$" columns denote the total time needed for solving the restricted master problem, pricing problem, and the overall column generation procedure in Algorithm 1. Ones with "FA" subscripts have corresponding meanings.

Table 4 Comparison of PA and FA Performance Metrics

No	PA					FA				
	$z_{\text{PA}}^{\text{LB}}$	$G_{\text{PA}}^{\star}$	$T_{\text{PA}}^{\text{RMP}}$	$T_{\text{PA}}^{\text{Pricing}}$	T_{PA}	$z_{\text{FA}}^{\text{LB}}$	$G_{\text{FA}}^{\star}$	$T_{\text{FA}}^{\text{RMP}}$	$T_{\text{FA}}^{\text{Pricing}}$	T_{FA}
$\texttt{R}_1\texttt{/0/50/foc}$	9.12E+08	0.07%	3.13	8.36	34.65	9.12E+08	0.10%	1.65	308.24	412.93
$\texttt{R}_1\texttt{/0/100/foc}$	2.12E+09	0.10%	45.24	12.76	153.16	2.05E+09	3.20%	2.43	1069.25	1317.76
$\texttt{R}_2\texttt{/0/50/foc}$	9.12E+08	0.08%	3.22	12.21	39.4	9.12E+08	0.13%	1.52	322.53	429.81
$\texttt{R}_2\texttt{/0/100/foc}$	2.17E+09	0.06%	33.68	13.95	139.37	2.05E+09	5.71%	2.71	1006.51	1234.03

The comparison between PA (Preparation-Allocation) and FA (Full Allocation) in Table 4 shows that PA consistently achieves better relative gap values and significantly fewer running times in all instances. Not surprisingly, PA is more efficient in solving the pricing problem due to effective reformulation, which shifts its bottleneck to solving a harder RMP. The results suggest that PA is more suitable for a formulation used in Algorithm 1 for large-scale problems.

REMARK 1. As we solve the PA reformulation directly as a generic linear programming problem (using an interior-point method), a natural question is whether we design a specialized method for problem (29). We leave this as a future research direction.

5.3.3. Performance with different enhancements We next illustrate the consequences of techniques employed in Algorithm 1 over the basic algorithm:

(1) "Algorithm 1 (Basic)": This reflects the performance of the baseline method with PA reformulations. After the column generation phase (termination with rRMP), we collect the relaxed solution and solve an integral RMP (20) to produce an integral solution.

(2) "+ RMP Tuning": In this strategy, we use myopic rules to select the columns so as to reduce the dimension of λ_c. Based on a frequency-based rule, we count the historical weights of a column and delete those that have not been selected in the last step once every k iteration. After some simple calibration, we select $k = 3$ in our setup. Meanwhile, we find that this strategy outperforms the approach of deleting columns that have not been selected for r steps, where $r \leq k$.

(3) "+ Rounding": This illustrates the difference between directly solving the integral RMP (20) and applying Algorithm 3 to query the integer feasible solutions.

Table 5 Performance Comparison of Different Algorithms

	Algorithm 1 (Basic)			+ RMP Tuning			+ Rounding (Algorithm 3)		
	$G^{\star}_{\text{PA}}$	G_{PA}	T_{PA}	$G^{\star}_{\text{PA}}$	G_{PA}	T_{PA}	$G^{\star}_{\text{PA}}$	G_{PA}	T_{PA}
$\texttt{S}_1/\texttt{1}/\texttt{519}/\texttt{foc}$	0.21%	-	4352.60	2.04%	-	4605.70	0.63%	-	2030.31
$\texttt{S}_1/\texttt{2}/\texttt{519}/\texttt{foc}$	4.36%	-	4987.20	0.42%	-	5719.10	0.64%	-	2898.67
$\texttt{S}_1/\texttt{3}/\texttt{519}/\texttt{foc}$	0.44%	-	5757.30	0.21%	-	5452.60	0.25%	-	3132.17
$\texttt{S}_1/\texttt{4}/\texttt{519}/\texttt{foc}$	0.53%	0.53%	4695.50	0.63%	0.63%	4363.90	0.68%	0.68%	2073.50
$\texttt{S}_1/\texttt{5}/\texttt{519}/\texttt{foc}$	0.42%	-	6430.10	0.18%	-	3449.70	0.27%	-	2327.36
$\texttt{S}_2/\texttt{1}/\texttt{519}/\texttt{foc}$	0.73%	0.73%	4747.30	3.01%	3.02%	3238.60	0.43%	0.43%	1088.31
$\texttt{S}_2/\texttt{2}/\texttt{519}/\texttt{foc}$	0.61%	0.61%	4799.30	1.42%	1.43%	4914.30	0.29%	0.29%	2594.77
$\texttt{S}_2/\texttt{3}/\texttt{519}/\texttt{foc}$	10.50%	10.50%	9977.60	1.89%	1.89%	3411.90	0.38%	0.38%	1530.78
$\texttt{S}_2/\texttt{4}/\texttt{519}/\texttt{foc}$	5.41%	5.41%	8951.60	1.76%	1.76%	3295.40	0.51%	0.51%	1572.30
$\texttt{S}_2/\texttt{5}/\texttt{519}/\texttt{foc}$	0.31%	0.31%	5131.40	1.94%	1.94%	3186.90	0.42%	0.42%	1382.01
$\texttt{R}_1/\texttt{0}/\texttt{519}/\texttt{foc}$	0.36%	0.16%	3412.30	0.16%	0.18%	1456.80	0.00%	0.01%	1601.65
$\texttt{R}_2/\texttt{0}/\texttt{519}/\texttt{foc}$	0.44%	0.44%	3722.40	0.00%	0.01%	2520.00	2.88%	2.89%	1941.19
$\texttt{R}_3/\texttt{0}/\texttt{519}/\texttt{foc}$	0.47%	-	10067.60	0.50%	-	2327.40	0.73%	-	1521.21

The results in Table 5 indicate that the basic method (Algorithm 1) is readily more competitive over Gurobi (cf. Table 2). As the problem difficulty increases, incorporating RMP tuning and rounding strategies remarkably improves performance. RMP tuning often leads to less time, tighter bounds, and better solutions. The rounding strategy further enhances these benefits, consistently achieving the lowest gaps and fastest solution times. We can highlight the average performance metrics in Table 6 to underscore the improvements achieved by each strategy.

Table 6 Improvement of Different Strategies

	Evaluation Metrics		
	$G^{\star}_{\text{PA}}$	G_{PA}	T_{PA}
Gurobi	38.29%	38.29%	8040.32
Algorithm 1 (Basic)	9.13%	2.34%	5151.13
+ RMP Tuning	1.06%	1.36%	3508.83
+ Rounding (Algorithm 3)	0.62%	0.70%	1976.48

5.3.4. Tuning the rounding strategies In this subsection, we will evaluate the impact of penalty parameter ρ_λ (cf. (36)). Let us recall the penalized objective used in the rounding algorithm (36):

$$\varphi(Q^n) := \min\ v_0(\mathbf{x}_0) + \rho_\lambda \cdot \sum_{i \in Q^n} \sum_{\nu=1}^{n} v_i^\nu \cdot (1 - \lambda_i^\nu)$$

We set the baseline $\rho_\lambda = 2$, which set the overall cost to 2 times the original cost (of a column). A simple tuning step indicates putting a more significant penalty is beneficial to the convergence speed of Algorithm 3. The results in Table 7 incorporate only the rounding heuristic algorithm into the basic algorithm.

Table 7 Impact of the Penalty Term ρ_λ

	$0.5\rho_\lambda$			ρ_λ			$2\rho_\lambda$		
Instance	$G^\star_{\text{PA}}$	G_{PA}	T^{R}_{PA}	$G^\star_{\text{PA}}$	G_{PA}	T^{R}_{PA}	$G^\star_{\text{PA}}$	G_{PA}	T^{R}_{PA}
$\texttt{S}_1/\texttt{1/519/foc}$	0.16%	-	1094.92	0.84%	-	218.97	0.00%	-	23.39
$\texttt{S}_1/\texttt{2/519/foc}$	0.35%	-	838.84	0.57%	-	265.16	0.00%	-	27.07
$\texttt{S}_1/\texttt{3/519/foc}$	0.42%	-	831.91	0.22%	-	206.95	0.00%	-	22.73
$\texttt{S}_1/\texttt{4/519/foc}$	0.92%	0.13%	382.43	0.70%	0.70%	163.88	0.00%	0.49%	20.71
$\texttt{S}_1/\texttt{5/519/foc}$	1.26%	-	522.54	0.57%	-	359.33	0.18%	-	53.06
$\texttt{S}_2/\texttt{1/519/foc}$	0.22%	0.22%	615.62	0.41%	0.41%	127.99	0.04%	0.51%	27.56
$\texttt{S}_2/\texttt{2/519/foc}$	0.21%	0.41%	1120.14	0.00%	0.40%	16.12	0.02%	0.42%	42.26
$\texttt{S}_2/\texttt{3/519/foc}$	0.39%	0.49%	2232.23	0.21%	0.22%	216.24	1.04%	0.61%	44.08
$\texttt{S}_2/\texttt{4/519/foc}$	0.02%	1.03%	200.83	0.12%	0.13%	257.92	0.00%	0.90%	10.57
$\texttt{S}_2/\texttt{5/519/foc}$	0.31%	0.32%	982.12	0.14%	0.14%	224.37	0.00%	0.54%	45.38

To close out the numerical experiments, we make the following remark.

REMARK 2. The lesson learned by setting different penalties opens the possibility of an adaptive scheme of tuning the penalty parameter ρ_λ on the go. We can periodically increase the penalty term whenever the progress ($|Q^n| - |Q^{n+1}|$) is small; this is an analog of penalty methods in nonlinear programming (Luenberger and Ye 2021).

6. Conclusion

Inspired by our collaboration with VX Logistics, we address an integrated model that encompasses strategic, tactical, and operational decisions over a finite time horizon. We reformulate the problem into a preparation-allocation structure and develop a distributed decomposition acceleration framework incorporating restricted master problem tuning, column initialization, rounding heuristics,

and valid inequalities for efficient problem-solving. Our experiments on real-world and synthetic datasets demonstrate that our proposed algorithm consistently outperforms state-of-the-art methods and more naive reformulation approaches. By generalizing the instance generation process and evaluating our approach across different scales and configurations, we establish its versatility and applicability beyond the specific context of our partner company.

The main challenge of this framework is solving the RMPs in the column generation framework. A recent vein of research has addressed advancements in machine learning (ML) to accelerate column generation. For example, employ machine learning to estimate upper bounds for pricing problems, select promising columns, and predict optimal dual variables, improving the efficiency of column generation (Václavík et al. 2018, Morabit et al. 2023, Kraul et al. 2023). Our future goal is to design machine learning techniques to speed up the convergence of the column generation process. Another possible direction is to explore stochastic demand from the customers and online adjustments.

Acknowledgments

We would like to express our sincere gratitude to Tianhao Liu and Qiushuang Gao for their invaluable discussions. We sincerely thank our industry partner, Suri Zang, for motivating this research.

References

Abadi M (2016) TensorFlow: learning functions at scale. *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, 1, ICFP 2016 (New York, NY, USA: Association for Computing Machinery), ISBN 978-1-4503-4219-3, URL http://dx.doi.org/10.1145/2951913.2976746.

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network flows: theory, algorithms, and applications* (USA: Prentice-Hall, Inc.), ISBN 978-0-13-617549-0.

Balakrishnan A, Li G, Mirchandani P (2017) Optimal network design with end-to-end service requirements. *Operations Research* 65(3):729–750.

Barnhart C, Boland NL, Clarke LW, Johnson EL, Nemhauser GL, Shenoi RG (1998a) Flight string models for aircraft fleeting and routing. *Transportation science* 32(3):208–220, publisher: INFORMS.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH (1998b) Branch-and-price: Column generation for solving huge integer programs. *Operations research* 46(3):316–329, publisher: INFORMS.

Carey N (2015) Third time's a charm for UPS at Christmas, but FedEx stumbles. *Reuters* URL https://www.reuters.com/article/business/third-times-a-charm-for-ups-at-christmas-but-fedex-stumbles-idUSKBN0UC1L2/.

Chen AI, Graves SC (2021) Item aggregation and column generation for online-retail inventory placement. *Manufacturing & Service Operations Management* 23(5):1062–1076.

Christofides N, Mingozzi A, Toth P (1981) Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming* 20(1):255–282, ISSN 1436-4646, URL http://dx.doi.org/10.1007/BF01589353.

Conforti M, Cornuéjols G, Zambelli G, others (2014) *Integer programming*, volume 271 (Springer).

Desrosiers J, Soumis F, Desrochers M (1984) Routing with time windows by column generation. *Networks* 14(4):545–565, ISSN 1097-0037, URL http://dx.doi.org/10.1002/net.3230140406, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230140406.

Federgruen A, Tzur M (1991) A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with n Periods in 0(n log n) or 0(n) Time. *Management Science* 37(8):909–925, ISSN 0025-1909, URL http://dx.doi.org/10.1287/mnsc.37.8.909, publisher: INFORMS.

Fukasawa R, Longo H, Lysgaard J, Aragão MPd, Reis M, Uchoa E, Werneck RF (2006) Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming* 106(3):491–511, ISSN 1436-4646, URL http://dx.doi.org/10.1007/s10107-005-0644-x.

Ge D, Huangfu Q, Wang Z, Wu J, Ye Y (2022) Cardinal Optimizer (COPT) User Guide. URL http://dx.doi.org/10.48550/arXiv.2208.14314, arXiv:2208.14314 [cs, math].

Gudapati NV, Malaguti E, Monaci M (2022a) Network design with service requirements: Scaling-up the size of solvable problems. *INFORMS Journal on Computing* 34(5):2571–2582.

Gudapati NVC, Malaguti E, Monaci M (2022b) Network Design with Service Requirements: Scaling-up the Size of Solvable Problems. *INFORMS Journal on Computing* 34(5):2571–2582, ISSN 1091-9856, URL http://dx.doi.org/10.1287/ijoc.2022.1200, publisher: INFORMS.

Gurobi Optimization, LLC (2024) Gurobi Optimizer Reference Manual. URL https://www.gurobi.com.

Huang L, Song JS, Tong J (2016) Supply Chain Planning for Random Demand Surges: Reactive Capacity and Safety Stock. *Manufacturing & Service Operations Management* 18(4):509–524, ISSN 1523-4614, 1526-5498, URL http://dx.doi.org/10.1287/msom.2016.0583.

Kang N, Shen H, Xu Y (2022) JD.com Improves Delivery Networks by a Multiperiod Facility Location Model. *INFORMS Journal on Applied Analytics* 52(2):133–148, ISSN 2644-0865, 2644-0873, URL http://dx.doi.org/10.1287/inte.2021.1077.

Kraul S, Seizinger M, Brunner JO (2023) Machine learning–supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation. *INFORMS Journal on Computing* 35(3):692–709, URL http://dx.doi.org/10.1287/ijoc.2023.1277.

Kumar N (2023) Southern USA 3-Level Supply Chain. URL https://www.kaggle.com/datasets/nitish09kr/supply-chain-demand.

Liu J (2016) Development of china's logistics market. Liu Bl, Wang L, Lee Sj, Liu J, Qin F, Jiao Zl, eds., *Contemporary Logistics in China: Proliferation and Internationalization*, 33–50 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-662-47721-2, URL http://dx.doi.org/10.1007/978-3-662-47721-2_3.

Luenberger DG, Ye Y (2021) *Linear and Nonlinear Programming*, volume 228 of *International Series in Operations Research & Management Science* (Cham: Springer International Publishing), ISBN 978-3-030-85449-2 978-3-030-85450-8, URL http://dx.doi.org/10.1007/978-3-030-85450-8.

Mirchandani PB, Francis RL, eds. (1990) *Discrete location theory*. Wiley-Interscience series in discrete mathematics and optimization (New York: Wiley), ISBN 978-0-471-89233-5.

Morabit M, Desaulniers G, Lodi A (2023) Machine-learning–based arc selection for constrained shortest path problems in column generation. *INFORMS Journal on Optimization* 5(2):191–210.

Moritz P, Nishihara R, Wang S, Tumanov A, Liaw R, Liang E, Elibol M, Yang Z, Paul W, Jordan MI, et al. (2018) Ray: A distributed framework for emerging {AI} applications. *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, 561–577.

Parmentier A, Martinelli R, Vidal T (2023) Electric Vehicle Fleets: Scalable Route and Recharge Scheduling Through Column Generation. *Transportation Science* 57(3):631–646, ISSN 0041-1655, URL http://dx.doi.org/10.1287/trsc.2023.1199, publisher: INFORMS.

Pochet Y, Wolsey LA (2006) *Production planning by mixed integer programming*. Springer series in operations research and financial engineering (New York ; Berlin: Springer), ISBN 978-0-387-29959-4.

Rao L (2015) Alibaba Rings Up a Record $14.3 Billion in Sales for Singles Day. *Fortune* URL https://fortune.com/2015/11/11/alibaba-singlesday-record-2015/.

Song JM, Zhao Y (2022) Supply Chain Coordination for E-Commerce: Risk Penalty vs. Flat Rate. *Manufacturing & Service Operations Management* 24(2):1110–1127, ISSN 1523-4614, 1526-5498, URL http://dx.doi.org/10.1287/msom.2021.1008.

Stevens L, Kapner S, Banjo S (2014) UPS, FedEx Want Retailers to Get Real on Holiday Shipping. *Wall Street Journal* ISSN 0099-9660, URL http://online.wsj.com/articles/ups-fedex-want-retailers-to-get-real-on-holiday-shipping-1412273998.

Sugishita N, Grothey A, McKinnon K (2024) Use of Machine Learning Models to Warmstart Column Generation for Unit Commitment. *INFORMS Journal on Computing* 36(4):1129–1146, ISSN 1091-9856, URL http://dx.doi.org/10.1287/ijoc.2022.0140, publisher: INFORMS.

Van Roy TJ, Wolsey LA (1986a) Valid inequalities for mixed 0–1 programs. *Discrete Applied Mathematics* 14(2):199–213, ISSN 0166-218X, URL http://dx.doi.org/10.1016/0166-218X(86)90061-2.

Van Roy TJ, Wolsey LA (1986b) Valid inequalities for mixed 0–1 programs. *Discrete Applied Mathematics* 14(2):199–213, ISSN 0166-218X, URL http://dx.doi.org/10.1016/0166-218X(86)90061-2.

Van Roy TJ, Wolsey LA (1987) Solving Mixed Integer Programming Problems Using Automatic Reformulation. *Operations Research* 35(1):45–57, ISSN 0030-364X, URL http://dx.doi.org/10.1287/opre.35.1.45, publisher: INFORMS.

Václavík R, Novák A, Šůcha P, Hanzálek Z (2018) Accelerating the Branch-and-Price Algorithm Using Machine Learning. *European Journal of Operational Research* 271(3):1055–1069, ISSN 03772217, URL http://dx.doi.org/10.1016/j.ejor.2018.05.046.

Wagner HM, Whitin TM (1958) Dynamic Version of the Economic Lot Size Model. *Management Science* 5(1):89–96, ISSN 0025-1909, URL http://dx.doi.org/10.1287/mnsc.5.1.89, publisher: INFORMS.

Wang S, Delage E (2024) A Column Generation Scheme for Distributionally Robust Multi-Item Newsvendor Problems. *INFORMS Journal on Computing* 36(3):849–867, ISSN 1091-9856, URL http://dx.doi.org/10.1287/ijoc.2022.0010, publisher: INFORMS.

Wang T, Hong LJ (2023) Large-scale inventory optimization: A recurrent neural networks–inspired simulation approach. *INFORMS Journal on Computing* 35(1):196–215.

Wolsey L (2020) *Integer Programming* (Wiley), 1 edition, ISBN 978-1-119-60653-6 978-1-119-60647-5, URL http://dx.doi.org/10.1002/9781119606475.

Electronic Companion

EC.1. Proof of Main Results

EC.1.1. Proof of Proposition 1

Proof At each iteration n, we consider the flow conservation constraints in the ν-th column $\mathbf{x}_\nu^i$ for an arbitrary $i \in \mathcal{C}$. The flow conservation constraints can be rewritten by the columns:

$$q_{j,t,k}^{[i,\nu]} = \sum_{e \in \delta_j^-} x_{e,t,k}^{[i,\nu]}, \qquad \forall j \in \mathcal{S}_i, k \in \mathcal{K}_i, t \in [T], \tag{EC.1}$$

$$I_{j,t,k}^{[i,\nu]} = I_{j,t-1,k}^{[i,\nu]} - \sum_{e \in \delta_j^-} x_{e,t,k}^{[i,\nu]} + \sum_{e \in \delta_j^+} x_{e,t,k}^{[i,\nu]}, \qquad \forall j \in \mathcal{W}_i, k \in \mathcal{K}_i, t \in [T], \tag{EC.2}$$

$$\sum_{e \in \delta_j^+} x_{e,t,k}^{[i,\nu]} + s_{i,t,k}^{[i,\nu]} = s_{i,t-1,k}^{[i,\nu]} + D_{i,t,k}, \qquad \forall k \in \mathcal{K}_i, t \in [T], \tag{EC.3}$$

where $\mathcal{S}_i$ and $\mathcal{W}_i$ denote the factories and warehouses associated with the customer i (in $\mathcal{G}_i$), respectively, and $\mathcal{K}_i$ denotes the commodities needed by the customer i.

Let $[n] = \{1, \ldots, n\}$, we have:

$$\sum_{i \in \mathcal{C}} \sum_{\nu \in [n]} \lambda_i^\nu \cdot \left(\sum_{e \in \delta_j^-} x_{e,t,k}^{[i,\nu]} - q_{j,t,k}^{[i,\nu]} \right) = 0, \qquad \forall j \in \mathcal{S}, k \in \mathcal{K}, t \in [T]$$

$$\sum_{i \in \mathcal{C}} \sum_{\nu \in [n]} \lambda_i^\nu \cdot \left(I_{j,t,k}^{[i,\nu]} - I_{j,t-1,k}^{[i,\nu]} + \sum_{e \in \delta_j^-} x_{e,t,k}^{[i,\nu]} - \sum_{e \in \delta_j^+} x_{e,t,k}^{[i,\nu]} \right) = 0, \qquad \forall j \in \mathcal{W}, k \in \mathcal{K}, t \in [T]$$

$$\sum_{\nu \in [n]} \lambda_i^\nu \cdot \left(\sum_{e \in \delta_i^+} x_{e,t,k}^{[i,\nu]} + s_{j,t,k}^{[i,\nu]} - s_{j,t-1,k}^{[i,\nu]} \right) = \sum_{\nu \in [n]} \lambda_i^\nu \cdot D_{i,t,k} = D_{i,t,k}, \qquad \forall i \in \mathcal{C}, k \in \mathcal{K}, t \in [T].$$

the last equation holds since $\lambda_i^T \mathbf{1} = 1, \forall i \in \mathcal{C}$. The rest of the proof follows by taking $\sum_{i \in \mathcal{C}} \sum_{\nu \in [n]} \lambda_i^\nu x_{e,t,k}^{[i,\nu]} = x_{e,t,k}$. This completes the proof. ■

EC.1.2. Proof of Proposition 2

Proof When disregarding layout decisions and setting the lower bound $L_e = 0, \forall e \in E$, we can construct the restricted master problem as follows:

$$\text{(RMP)} \quad v_{\text{CG}}^n := \min_{\lambda_i} \sum_{i \in \mathcal{C}} \sum_{\nu \in [n]} \lambda_i^\nu \cdot v_i(\mathbf{x}_i^\nu) \tag{EC.4a}$$

$$\text{s.t.} \quad \mathbf{0} \le \lambda_i \le \mathbf{1}, \forall i \in \mathcal{C}, \tag{EC.4b}$$

$$\lambda_i^T \mathbf{1} = 1, \forall i \in \mathcal{C}, \tag{EC.4c}$$

$$\sum_{i \in \mathcal{C}} \tilde{A}_c \lambda_i \leq d. \tag{EC.4d}$$

Note again the right-hand side $d \in \mathbb{R}^m$ and $\tilde{A}_c = [A_c \mathbf{x}_i^1, ..., A_c \mathbf{x}_i^n]$ form the coupling constraint.

As shown in Proposition 1, the flow conservation constraints automatically hold from a linear combination of the columns; this indicates that (EC.4d) consists of (12) and (8)-(10). To the fact that capacities are sufficiently large, and $L_e = 0, \forall e \in E$, we conclude that the dual variable $(\theta^n, \pi^n) \in \mathbb{R}^{|\mathcal{C}|} \times \mathbb{R}_+^m$ must be zero (complentary slackness). Therefore, the dual problem of RMP simplifies as follows:

$$\begin{aligned} \text{(dRMP)} \quad v_{\text{Dual}}^n := \max_{\theta_i} \quad & \mathbf{1}^T \theta^n \\ \text{s.t.} \quad & v_i(\mathbf{x}_i) - \theta_i^n \mathbf{1} \geq 0, \forall i \in \mathcal{C}, \mathbf{x}_i \in \mathcal{X}_i^n. \end{aligned} \tag{EC.5}$$

Accordingly, the pricing problem for each customer $i \in \mathcal{C}$ at iteration n becomes:

$$\text{(P)} \quad \mathbf{x}_i^\star := \text{argmin}_{\mathbf{x}_i^n \in \mathcal{X}_i} \; r(\mathbf{x}_i^n) := v_i(\mathbf{x}_i^n) - \theta_i^n. \tag{EC.6}$$

At $n = 0$, we initialize $\{\mathbf{x}_i^0\}$ with the costs $\{v_i^0(\mathbf{x}_i^0)\}$ for every customer $i \in \mathcal{C}$. This may be done, e.g., based on the sweeping algorithm (Algorithm 2). By solving dRMP (EC.4), only two scenarios are possible for a given $i \in \mathcal{C}$:

(1) If $\mathbf{x}_i^\star = \mathbf{x}_i^0$, then $v_i^\star(\mathbf{x}_i^\star) = v_i(\mathbf{x}_i^0) \geq \theta_i^0$, making the reduced cost $r(\mathbf{x}_i^\star)$ non-negative. The column generation process for customer c terminates at $n = 0$.

(2) If $\mathbf{x}_i^\star \neq \mathbf{x}_i^0$, then we have $v_i^\star(\mathbf{x}_i^\star) < v_i(\mathbf{x}_i^0)$.

   (a) If $v_i^\star(\mathbf{x}_i^\star) - \theta_i^0 \geq 0$, making the reduced cost $r(\mathbf{x}_i^\star)$ non-negative and the column generation process terminates for customer i at $n = 0$.

   (b) If $v_i^\star(\mathbf{x}_i^\star) - \theta_i^0 < 0$, the reduced cost $r(\mathbf{x}_i^\star)$ is negative and the new column $\mathbf{x}_i^1 = \mathbf{x}_i^\star$ is added to $\mathcal{X}_i^1$. Then at $n = 1$, we solve the reduced dual RMP (dRMP') and have $v_i^0(\mathbf{x}_i^0) \geq \theta_i^1, v_i^1(\mathbf{x}_i^1) \geq \theta_i^1$. When solving the pricing problem at $n = 1$, it is apparent that the pricing problem is independent with θ_i^1, and the optimal solution of the pricing problem is still $\mathbf{x}_i^1$. Hence, the optimal reduced cost $r(\mathbf{x}_i^\star) = v_i^1(\mathbf{x}_i^1) - \theta_i^1 \geq 0$. The column generation process terminates for customer c at $n = 1$.

To sum up, we can claim that the column generation process will terminate at $n \leq 1$. ■

EC.1.3. Proof of Proposition 3

Proof To show the correspondence to Van Roy and Wolsey (1987), we first reconstruct (27)-(28) in the backlogged setting :

$$(I_{i,t-1,k}+\sum_{e\in\delta_i^+\setminus E_{\mathcal{C}}}x_{e,t,k})-(I_{i,t,k}+\sum_{e\in\delta_i^-\setminus E_{\mathcal{C}}}x_{e,t,k})=z_{j,t,k},\qquad \forall j\in\mathcal{W},k\in\mathcal{K}, \tag{EC.7}$$

$$\sum_{e\in\delta_i^+\setminus E_{\mathcal{C}}}q_{i,t,k}-\sum_{e\in\delta_i^-\setminus E_{\mathcal{C}}}x_{e,t,k}=0,\qquad \forall i\in\mathcal{S},k\in\mathcal{K}, \tag{EC.8}$$

$$z_{j,t,k}=\sum_{c\in\delta_i^-\bigcap\mathcal{C}}x_{i,c,t,k}\le\sum_{t'\in[t]}\sum_{i\in\mathcal{C}}d_{i,t',k},\qquad \forall j\in\mathcal{W},k\in\mathcal{K}. \tag{EC.9}$$

By introducing artificial edges and a surrogate operator $\ell(\cdot)$, we can reformate the flow conservation constraint as follows:

$$\sum_{e\in\delta_i^+\setminus E_{\mathcal{C}}}\ell_{e,t,k}(x)-\sum_{e\in\delta_i^-\setminus E_{\mathcal{C}}}\ell_{e,t,k}(x)\le\sum_{t'\in[t]}\sum_{i\in\mathcal{C}}d_{i,t',k},\qquad \forall j\in\mathcal{W},k\in\mathcal{K}, \tag{EC.10}$$

$$\sum_{e\in\delta_i^+\setminus E_{\mathcal{C}}}\ell_{e,t,k}(x)-\sum_{e\in\delta_i^-\setminus E_{\mathcal{C}}}\ell_{e,t,k}(x)\le 0,\qquad \forall i\in\mathcal{S},k\in\mathcal{K}. \tag{EC.11}$$

Similarly, we can rewrite the capacity constraints (8)-(10) and edge lower bound constraints (12) as follows:

$$L_e\ell_{e,t,k}(y)\le\ell_{e,t,k}(x)\le C_e\ell_{e,t,k}(y),\qquad \forall e\in E,\forall k\in\mathcal{K}, \tag{EC.12}$$

$$\ell_{e,t,k}(y)\in\mathbb{B},\ell_{e,t,k}(x)\ge 0,\qquad \forall e\in(\delta_i^+\cup\delta_i^-)\setminus E_{\mathcal{C}},\forall k\in\mathcal{K}. \tag{EC.13}$$

We see the inequality is valid by Van Roy and Wolsey (1986a, Corollary 3). ■

EC.2. The separation problem in Proposition 4

We provide the details of solving the separation problem for the generalized flow cover inequalities (34). Van Roy and Wolsey (1986b) identifies the following procedure. First introduce $0-1$ variables α_e,β_e as indicators for $e\in C_1\setminus R, C_1\cap R$, respectively; γ_e,ϕ_e as defined accordingly for C_2:

$$A=\left\{\begin{array}{l}\alpha_e\in\{0,1\},\beta_e\in\{0,1\}\\ \gamma_e\in\{0,1\},\phi_e\in\{0,1\}\end{array}\;\middle|\;\begin{array}{l}\sum_{e\in\delta_i^+}(C_e\alpha_e+L_e\beta_e)-\sum_{e\in\delta_i^-}(L_e\gamma_e+C_e\phi_e)-Q_{i,t,k}=\Lambda,\\ \alpha_e+\beta_e\le 1,\ e\in\delta_i^+,\quad \gamma_e+\phi_e\le 1,\ e\in\delta_i^-,\\ \alpha_e\in\{0,1\},\beta_e\in\{0,1\},\gamma_e\in\{0,1\},\phi_e\in\{0,1\}\end{array}\right\} \tag{EC.14}$$

Then we can express the cut coefficients of (34) by $\alpha_e, \beta_e, \gamma_e, \phi_e$.

$$\begin{aligned}\underset{(\alpha_e,\beta_e,\gamma_e,\phi_e)\in A}{\text{Maximize}}\ s := & \sum_{e\in\delta_i^+} \alpha_e \left[\ell_{e,t,k}(x) + \max\{C_e-\Lambda,0\}\left(1-\ell_{e,t,k}(y)\right)\right] - Q_{i,t,k} \\ & + \sum_{e\in\delta_i^+} \beta_e \left[\max\{L_e-\Lambda,0\} + \min\left(L_e,\Lambda\right)\ell_{e,t,k}(y)\right] \\ & - \sum_{e\in\delta_i^-} \gamma_e \left[\ell_{e,t,k}(x) + L_e\left(1-\ell_{e,t,k}(y)\right)\right] - \sum_{e\in\delta_i^-} \phi_e C_e \\ & - \sum_{e\in\delta_i^-} (1-\gamma_e-\phi_e)\min\left\{\ell_{e,t,k}(x) - \max\{L_e-\Lambda,0\}\ell_{e,t,k}(y), \min\left(C_e,\Lambda\right)\ell_{e,t,k}(y), \ell_{e,t,k}(x)\right\}\end{aligned} \tag{EC.15}$$

Problem (EC.15) is now a combinatorial bilinear problem that can be efficiently addressed using heuristics and optimization solvers. If the optimal value is strictly positive, the desired inequality is identified.

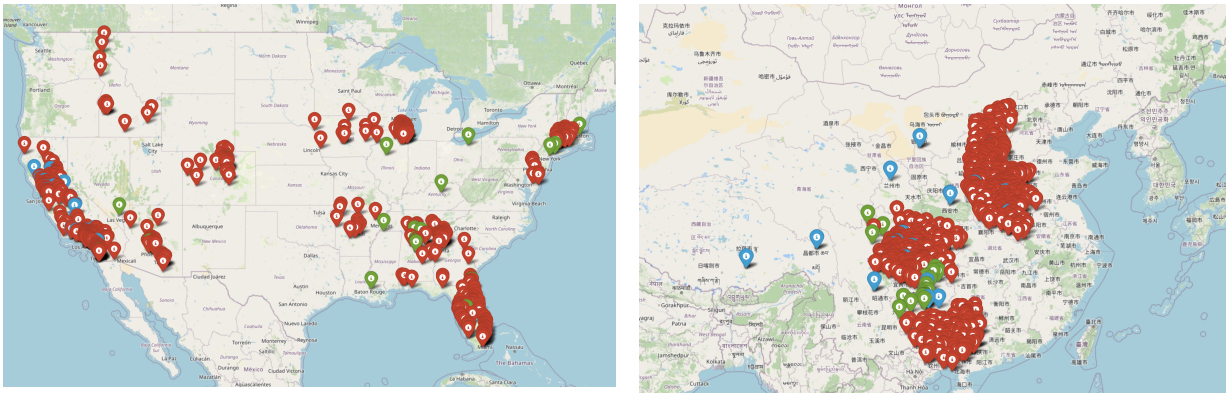
EC.3. Synthetic Dataset Generation Details

We present a data generation process to create random instances. We generate the synthetic dataset based on two main selection criteria:

(1) Basic network structure. United States: $\mathtt{S}_1$, where we conducted the southern United States supply chain network based on the fundamental nodes provided by Kumar (2023) and China: $\mathtt{S}_2$, where we are inspired by practical datasets and developed the supply chain network focusing primarily on the eastern provinces of China.

Figure EC.1 Supply Chain Structure.

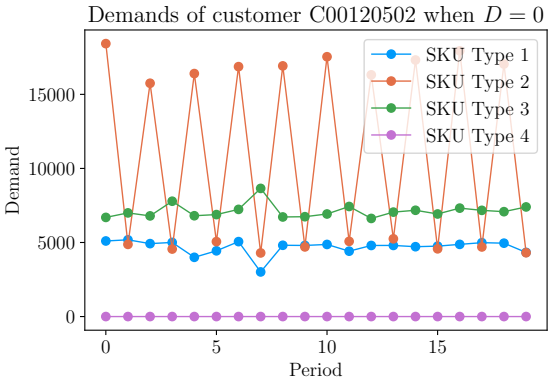
(a) Figure 1: South America. (b) Figure 2: China.



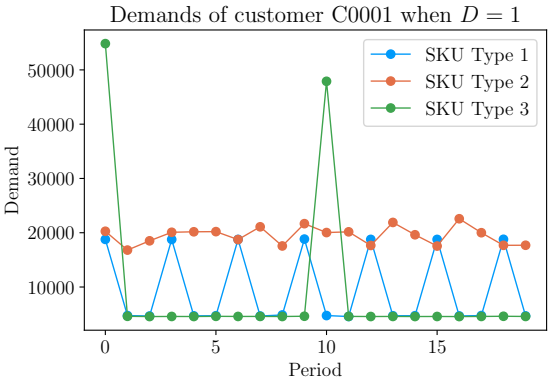
Note. The blue points indicate the locations of the warehouses, the green points represent the plants, and the red points show the customers' positions.

(2) Demand type. We have defined five distinct demand categories for the synthetic dataset, ranging from 1 to 5 while labeling real-world realistic demands as 0. To illustrate different demand

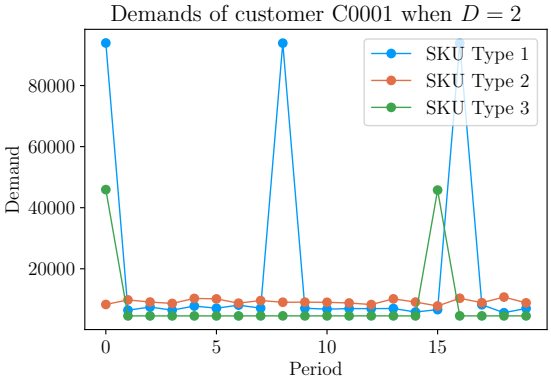
types, we cluster similar commodities and demonstrate the total demands of these similar commodities for a specific customer in the VX Logistics supply chain network EC.2a and American supply chain EC.2b-EC.2f respectively. The general characteristics of these five



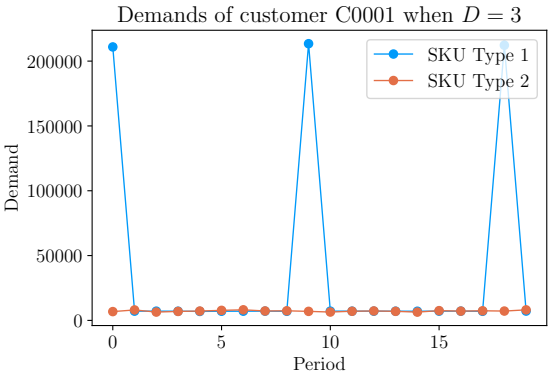
(a) Figure 1: Demand Type 0



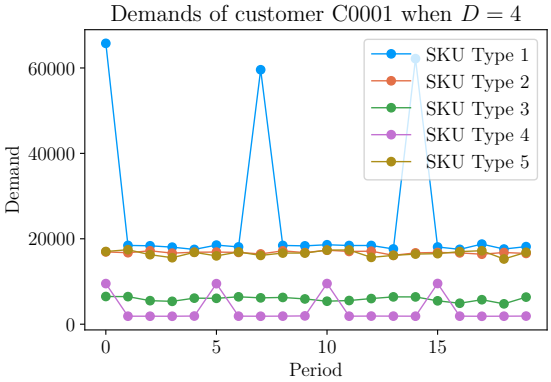
(b) Figure 2: Demand Type 1



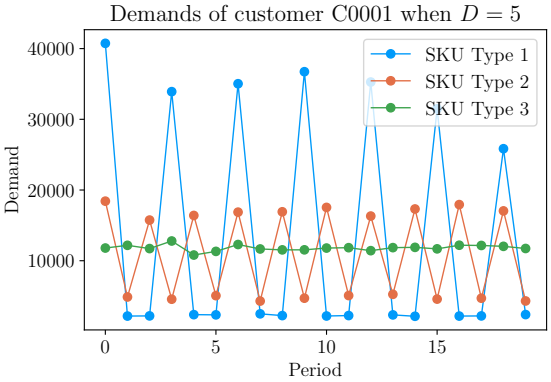
(c) Figure 3: Demand Type 2



(d) Figure 4: Demand Type 3



(e) Figure 5: Demand Type 4



(f) Figure 6: Demand Type 5

Figure EC.2 Visualized demands.

synthetic demand types can be described as follows.

- Demand Type 1: SKU Type 1 and SKU Type 3 exhibit cyclical demand patterns, reflecting their seasonal nature every 3 and 10 periods, respectively. Meanwhile, SKU Type 2 maintains a stable demand throughout the period, ranging between 18,000 and 20,000 units.
- Demand Type 2: SKU Type 1 and SKU Type 3 exhibit cyclical demand patterns, reflecting their seasonal nature every 3 and 10 periods, respectively. Meanwhile, SKU Type 2 maintains a stable demand throughout the period, ranging between 18,000 and 20,000 units.
- Demand Type 3: SKU Type 2 maintains steady, low demand throughout the periods, consistently ranging from 200 to 500 units, while SKU Type 1 exhibits significant changes every nine periods.
- Demand Type 4: This type is characterized by spikes in demand, particularly for seasonal SKUs such as Type 1 and Type 4. SKU Types 2, 3, and 5 display a steady trend, with SKU Type 3 showing the lowest demand among them.
- Demand Type 5: In this type, each SKU exhibits unique demand patterns that vary significantly across periods, reflecting individualized needs for each customer. SKU Type 1 and 2 show alternating high and low demand, while SKU Type 3 maintains a consistent, lower demand level.

After establishing distinct topology structures and demand patterns, the data generation process can be introduced and outlined as follows:

(1) Load all facility information based on the selected network structure, including node locations (plants, warehouses, and customers) and related attributes such as capacities, fixed costs, and unit costs.

(2) Generate specific demands for each customer and commodity in each period according to the selected demand type.

(3) Generate edges based on the number of warehouses:
- If the number of warehouses is less than 1000, generate a fully linked network structure connecting warehouses, plants, and customers.
- If the number exceeds 1000, generate edges only between each plant or customer and their nearest 500 warehouses, with in-transit edges between warehouses randomly generated in quantities ranging from 0 to 2000.

(4) Assign transportation costs to each edge based on distance data and randomly calculate their lower bounds and capacities according to the attributes of their start and end points.

(5) Set other customized constraint parameters for nodes and edges, such as distance limits, upper bounds, and node lower bounds.

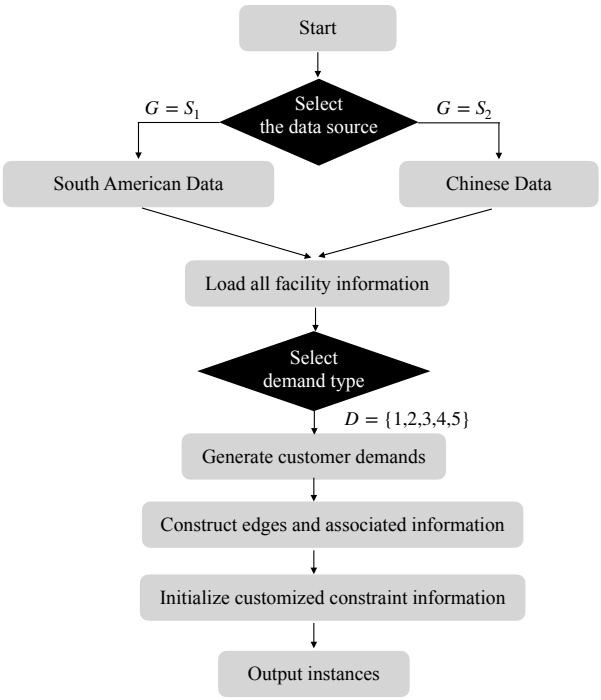


Figure EC.3 Template generation framework