7-2024

Optimal policies and heuristics to match supply with demand for online retailing

Qiyuan DENG
Chinese University of Hong Kong, Shenzhen

Xiaobo LI
National University of Singapore

Yun Fong LIM
Singapore Management University, yflim@smu.edu.sg

Fang LIU
Durham University

Optimal Policies and Heuristics To Match Supply With Demand For Online Retailing

Qiyuan Deng[a,b] • Xiaobo Li[c,*] • Yun Fong Lim[d] • Fang Liu[e]

[a]School of Management and Economics, The Chinese University of Hong Kong, Shenzhen, Guangdong 518172, China

[b]Shenzhen Finance Institute, Shenzhen, Guangdong 518000, China

[c]Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117576

[d]Lee Kong Chian School of Business, Singapore Management University, Singapore 178899

[e]Durham University Business School, Durham University, Durham DH1 3LB, United Kingdom

dengqiyuan@cuhk.edu.cn • iselix@nus.edu.sg • yflim@smu.edu.sg • fang.liu@durham.ac.uk

[*]Corresponding author

Abstract

Problem definition: We consider an online retailer selling multiple products to different zones over a finite horizon with multiple periods. At the start of the horizon, the retailer orders the products from a single supplier and stores them at multiple warehouses. The retailer determines the products' order quantities and their storage quantities at each warehouse subject to its capacity constraint. At the end of each period, after random demands in the period are realized, the retailer chooses the retrieval quantities from each warehouse to fulfill the demands of each zone. The objective is to maximize the retailer's expected profit over the finite horizon.

Methodology/results: For the single-zone case, we show that the multi-period problem is equivalent to a single-period problem and the optimal retrieval decisions follow a greedy policy that retrieves products from the lowest-cost warehouse. We design a non-greedy algorithm to find the optimal storage policy, which preserves a nested property: Among all non-empty warehouses, a smaller-index warehouse contains all the products stored in a larger-index warehouse. We also analytically characterize the optimal ordering policy. The multi-zone case is unfortunately intractable analytically and we propose an efficient heuristic to solve it, which involves a non-trivial hybrid of three approximations. This hybrid heuristic outperforms two conventional benchmarks by up to 22.5% and 3.5% in our numerical experiments with various horizon lengths, fulfillment frequencies, warehouse capacities, demand variations, and demand correlations.

Managerial implications: A case study based on data from a major fashion online retailer in Asia confirms the superiority of the hybrid heuristic. With delicate optimization, the heuristic improves the average profit by up to 16% compared to a dedicated policy adopted by the retailer. The hybrid heuristic continues to outperform the benchmarks for larger networks with various structures.

Keywords: online seasonal sales, product ordering, inventory allocation, order fulfillment, multiple periods

1 Introduction

In 2023, global retail e-commerce sales reached $5.78 trillion, making up 19.5% of the total retail sales worldwide (Insider Intelligence, 2023). The revenues of e-retailing are projected to grow to $8.03 trillion in 2027, which is equivalent to 23% of the total retail sales worldwide. The strong growth in sales makes online retailing a significant industry to study. As more people shop online, more online businesses emerge, resulting in fierce competitions.

Online seasonal sales have recently become one of the most popular online promotional events. For example, Alibaba generated more than $84.54 billion of revenue on the "Singles Day" in 2021, whereas JD.com reported a revenue of $54.60 billion in the same period (Kharpal, 2021). Online seasonal sales also occur in the online fashion sample sales industry (Ferreira et al., 2016), where retailers offer limited-time discounts, also known as "flash sales", on a collection of products. For example, Rue La La sells products through "events", each of which might represent a collection of products from the same designer or a collection of women's accessaries. Other examples include Zulily, Gilt Groupe, and VIP.com. See Local Offer Network (2011), Ostapenko (2013), and Wolverson (2012) for more examples of sample sales industries.

To offer a seasonal sale, one of the biggest challenges for online retailers is to operate a responsive supply network. For each selling season, the online retailers need to rapidly fulfill random demands of different zones (such as countries or regions). To study the challenge, consider an online retailer selling multiple products to multiple zones over a finite horizon (the selling season). At the start of the horizon, the retailer replenishes the products from a supplier and stores them in warehouses at different locations. Demands are realized in each period of the horizon. At the end of each period, the retailer ships the products from the warehouses to the different zones to fulfill their demands. To better match supply with demand, the retailer has to strategically determine how much of each product to replenish from the supplier (ordering decisions) and how much of each product to store in each warehouse (storage decisions) at the start of the horizon. At the end of each period, after knowing the demands in the period, the retailer has to determine how much of each product to retrieve from each warehouse to fulfill the demands of the different zones (retrieval decisions). The retailer's objective is to maximize the expected profit over the finite horizon.

The operations of an online retailer for a seasonal sale have the following features:

1. Anticipative ordering and storage decisions. At the start of the selling season (horizon), the online retailer makes *anticipative* ordering and storage decisions before knowing demand. Once the products are stored in the warehouses, the retailer typically does not reshuffle the inventory

among the warehouses.

2. Limited warehouse capacity. Each warehouse has a finite capacity, which restricts the retrieval quantity of each product from the warehouse. Furthermore, the total capacity of all the warehouses sets an upper limit on the total order quantity of all the products.

3. Flexible fulfillment. The online retailer has the flexibility to fulfill the demands of a zone from multiple warehouses. Thus, if the nearest warehouse to the zone is out of stock for a certain product, then the retailer can choose another, farther, warehouse to satisfy the demand (this is also known as *demand spillover* (Acimovic and Graves, 2017)). Note that although the fulfillment flexibility can increase service levels, it may also incur a higher outbound transportation cost for the retailer because the products may be shipped from farther warehouses. Furthermore, it also complicates the storage and the ordering decisions.

If each warehouse serves only one zone and each zone is served by a dedicated warehouse, then the above problem can be separated into independent capacitated multi-product inventory management problems. However, since each warehouse can flexibly serve multiple zones in practice, the retailer should take demand-pooling effect into account when making the ordering decisions. Likewise, given that each zone can be served by multiple warehouses, the online retailer needs to consider the demand-spillover effect when making the storage decisions. The problem is especially challenging because the ordering and storage decisions are made in an anticipative manner before knowing the actual demands of the selling season with limited warehouse capacity. Furthermore, if each warehouse can serve multiple zones and each zone can be served by multiple warehouses, the retrieval decisions are not straightforward in a multi-period setting.

We formulate a stochastic optimization model for an online retailer selling multiple products to different demand zones over a finite horizon with multiple periods. At the start of the horizon, the retailer orders the products from a single supplier and stores them at multiple warehouses subject to their capacity constraints. At the end of each period, after random demands in the period are realized, the retailer retrieves products from the warehouses to fulfill the demands of the different zones. Any unmet demand in each period is lost. The retailer jointly optimizes the ordering, storage, and retrieval quantities to maximize the expected profit over the finite horizon.

We make the following contributions in this paper:

1. For the *single-zone problem*, we show that the multi-period problem is equivalent to a single-period problem and characterize the structural properties of its optimal retrieval, storage, and ordering policies. Specifically, the optimal retrieval decisions follow a greedy policy. To find a storage policy, we design a non-greedy algorithm (Algorithm 1) that allocates products to the warehouses iteratively according to each warehouse's target stockout probability. Under this algorithm, the product assortment in the warehouses preserves a nested property: Among all non-empty ware-

houses, a smaller-index warehouse contains all the products stored in a larger-index warehouse. We show that Algorithm 1 produces an ϵ-optimal storage policy in polynomial time. Algorithm 1 significantly outperforms conventional methods in our numerical experiments. We also provide sufficient conditions to find the optimal ordering policy.

2. The *multi-zone problem* is unfortunately intractable analytically because the optimal retrieval policy cannot be expressed in closed form. We develop a hybrid heuristic in Section 5.4 to determine the order and the storage quantities. This heuristic involves a non-trivial hybrid of three relatively special cases where either the optimal inventory solution or the objective function can be efficiently computed. Although conceptually simple, the hybrid heuristic significantly outperforms state-of-the-art approaches in our numerical experiments based on both synthetic data and real data from a major fashion online retailer in Asia.

After reviewing the related literature in Section 2, we formulate the finite-horizon problem in Section 3. Section 4 analyzes the single-zone problem. Section 5 introduces the hybrid heuristic for the multi-zone problem. Section 6 benchmarks the hybrid heuristic against conventional methods based on realistic problem settings. Section 7 evaluates the policies using data from the fashion online retailer and Section 8 concludes the paper. All proofs are presented in Appendix A.

2 Related literature

This paper is related to two streams of literature: Online retail operations and allocation problems.

Online retail operations

Several papers analyze the fulfillment problem for an online retailer to select warehouses to satisfy customer orders from different zones. Due to the fulfillment flexibility, a customer order can be assigned to one or multiple warehouses. Since orders are typically batched before items are physically picked, Xu et al. (2009) investigate the benefits of re-assigning some orders to different warehouses during the batching period. The authors propose near-optimal heuristics for re-assigning orders to warehouses to minimize the total number of shipments. Acimovic and Graves (2015) determine the warehouses and shipping modes to fulfill orders arriving from different zones such that the expected outbound shipping cost of an online retailer is minimized. They develop a heuristic that outperforms a myopic policy. Jasin and Sinha (2015) solve a similar problem by approximating a stochastic control formulation with a deterministic linear program (DLP). They develop a heuristic by modifying the DLP solution through a correlated rounding scheme among the decision variables and provide its theoretical performance guarantee. These papers address an important problem of how to fulfill a multi-item order from multiple warehouses (that is, whether through split shipping or through order consolidation).

Lei et al. (2018) consider a multi-product, multi-period setting where orders from different zones are fulfilled by multiple warehouses. They maximize an online retailer's expected profit by dynamically pricing the products and determining the warehouse for each order jointly. The authors propose two heuristics to solve the problem and prove that they perform well compared to a benchmark. Harsha et al. (2019) consider an omni-channel retailer selling a product with no replenishments in a finite horizon. In each zone, the retailer has a brick-and-mortar store (brick store) that fulfills walk-in customers' demand. The zone's online demand can be fulfilled by a warehouse, the zone's brick store, or another zone's brick store. The retailer charges the same online price but different brick-store prices for different zones. The retailer maximizes her profit over the horizon by optimizing the prices and fulfillment decisions. Ferreira et al. (2016) study a multi-product pricing problem and conduct a field experiment with an online retailer. Miao et al. (2022) develop an asymptotically optimal heuristic to find a multi-period fulfillment policy based on predicted demand.

Some researchers study the problem of ordering the products and allocating them to different warehouses. Acimovic and Graves (2017) study demand spillover: A stockout at a warehouse leads to demand spilling over to another warehouse. They propose a heuristic to allocate inventory to the warehouses, which considers possible demand spillover during the replenishment lead time. Their simulation results suggest that the heuristic captures over 90% of the possible improvement by a pseudo-optimal policy. Zhong et al. (2018) consider a manufacturer fulfilling multiple e-distributors for a product with a centralized pool of inventory over one period. Each e-distributor faces random demand and a fill-rate requirement. The manufacturer first chooses the centralized pool's inventory level. After the demands are realized, the manufacturer allocates some quantity to each e-distributor. The authors obtain necessary and sufficient conditions for the minimum inventory level, and develop an allocation policy to satisfy the e-distributors' fill rates. In contrast, we allocate the inventory to different warehouses *before* the demands are realized.

Our paper complements the above work by jointly determining the ordering, storage, and retrieval decisions for an online retailer. Lim et al. (2020) study a similar problem. They model the product demands using uncertainty sets and develop heuristics based on robust optimization techniques to solve the problem numerically. In contrast, we assume the demands follow a distribution function and characterize the structural properties of the optimal ordering, storage, and retrieval policies for the single-zone problem. For the multi-zone problem, we develop an efficient heuristic that can handle more than 10,000 products (see Section 7), whereas the heuristics by Lim et al. (2020) can only handle around 1,000 products for the same supply chain setting. In addition, our heuristic is data driven, which uses demand samples as inputs to solve the problem, whereas the robust formulation by Lim et al. (2020) can only incorporate the demand support information.

Allocation problems

The storage stage of our single-zone problem is closely related to papers studying the theory and algorithms for general allocation problems, see Zipkin (1980), Eppen and Schrage (1981), Federgruen and Groenevelt (1986), Federgruen et al. (1986), Groenevelt (1991), Ando et al. (1995), Vidal et al. (2014), Liu (2017), Vidal et al. (2019), and Wu et al. (2021). In contrast to Groenevelt (1991), Ando et al. (1995), Vidal et al. (2014), Vidal et al. (2019), and Wu et al. (2021), who consider separable objective functions, our objective function of the single-zone storage problem is not separable. Zipkin (1980) studies a simple algorithm for allocating one resource with a nonlinear-additive separable objective function. In contrast, our model in the storage stage can handle multiple resources (products) and a more general convex objective function compared to separable objective functions. Federgruen and Groenevelt (1986) propose a greedy algorithm to compute an allocation policy, and find sufficient conditions for the greedy algorithm to be optimal. Liu (2017) proposes a greedy algorithm to solve the general transportation problem. Unfortunately, their algorithms cannot be applied to our setting because the constraints in our model do not satisfy the submodularity condition in their papers. In fact, the optimal storage quantities in our paper are not monotone in the order quantities, and so greedy algorithms are not optimal. Instead, we propose a non-greedy algorithm that stores the products to the warehouses iteratively. In each iteration, our algorithm identifies a warehouse to store the products and then reallocates the products' quantities among a subset of warehouses. We prove that the non-greedy algorithm finds an ϵ-optimal storage policy in polynomial time.

3 Problem formulation

Consider an online retailer selling products $i = 1, \ldots, I$ to zones $k = 1, \ldots, K$ over a time horizon of T periods. Each zone k represents a geographical area (such as a country or region). For each product i, let ρ_i be the *unit purchase cost* and p_i $(> \rho_i)$ be the *unit selling price*, which are independent of the zones. At the start of the time horizon, the retailer orders the products from a single supplier and stores them in warehouses $j = 1, \ldots, J$ at different locations. Each warehouse j has a limited storage capacity c_j. To store a unit of any product from the supplier to warehouse j, a *unit storage cost* s_j is incurred, which includes the transportation and the handling costs. The inventory of the same product may be stored in multiple warehouses. In each period t, the demand for product i of zone k is a random variable $\tilde{d}_{ik}^t$ distributed on $\left[0, \bar{d}_{ik}^t\right]$, where $\bar{d}_{ik}^t$ (> 0) can be infinity. Let $\hat{d}_{ik}^t$ and d_{ik}^t denote the mean and the realization, respectively, of the demand $\tilde{d}_{ik}^t$. Define $\tilde{\boldsymbol{d}}^t = \left(\tilde{d}_{ik}^t\right)_{I\times K}$ and $\boldsymbol{d}^t = \left(d_{ik}^t\right)_{I\times K}$. Let $f_{ik}^t(\cdot)$ and $F_{ik}^t(\cdot)$ denote the p.d.f. and c.d.f., respectively, of the aggregate demand $\sum_{\tau=1}^t \tilde{d}_{ik}^\tau$. Assume $F_{ik}^t(\cdot)$ is continuous and differentiable. Define $\bar{F}_{ik}^t(\cdot) = 1 - F_{ik}^t(\cdot)$. Define $\left(\bar{F}_{ik}^t\right)^{-1}(\cdot)$ as the inverse function of $\bar{F}_{ik}^t(\cdot)$ with

$\left(\bar{F}_{ik}^{t}\right)^{-1}(\cdot)=\sum_{\tau=1}^{t}\bar{d}_{ik}^{\tau}$ if $\cdot<0$, and $\left(\bar{F}_{ik}^{t}\right)^{-1}(\cdot)=0$ if $\cdot>1$. To ensure that the inverse function $\left(\bar{F}_{ik}^{t}\right)^{-1}(\cdot)$ is well defined, we assume that $f_{ik}^{t}(x)>0$, for $x\in\left(0,\sum_{\tau=1}^{t}\bar{d}_{ik}^{\tau}\right)$.

Due to fulfillment flexibility, the demands of each zone can be fulfilled by all the warehouses. A *unit retrieval cost* r_{jk} is incurred when a unit of any product is shipped from warehouse j to satisfy the demand of zone k. The unit storage costs and the unit retrieval costs are independent of the products. This assumption generally holds for products belonging to the same category. To keep a unit of product i, a *unit holding cost per period* h_i is incurred. Any unsatisfied demands are lost. For convenience, define $\mathcal{I}=\{1,...,I\}$, $\mathcal{J}=\{1,...,J\}$, $\mathcal{J}^{-}=\{1,...,J-1\}$, $\mathcal{K}=\{1,...,K\}$, $\mathcal{T}=\{1,...,T\}$, $\mathcal{T}^{-}=\{1,...,T-1\}$, and $s_0=r_{j0}=r_{0k}=0$, for $j\in\mathcal{J},k\in\mathcal{K}$. Assume $p_i+h_iT>\max_{j\in\mathcal{J},k\in\mathcal{K}}r_{jk}$, for $i\in\mathcal{I}$, and each warehouse $j\in\mathcal{J}$ is initially empty. Let Π denote a set of all feasible policies. Under any policy $\pi\in\Pi$, let q_i^{π} denote the order quantity for product i, $x_{ij}^{\pi t}$ denote the quantity of product i stored in warehouse j at the start of period t, and $y_{ijk}^{\pi t}$ denote the quantity of product i retrieved from warehouse j to fulfill the demand of zone k in period t. We call q_i^{π}, $x_{ij}^{\pi 1}$, and $y_{ijk}^{\pi t}$ the ordering, storage, and retrieval decisions respectively. Note that we only need to determine the storage quantities $x_{ij}^{\pi t}$ for $t=1$. Figure 1 shows the sequence of events.

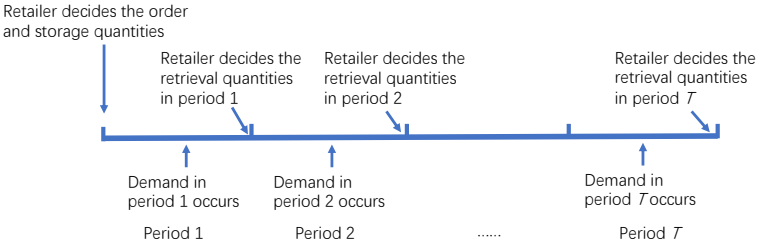


Figure 1: The sequence of events over T periods

Define $\boldsymbol{\rho}=(\rho_i)_{I\times 1}$, $\boldsymbol{p}=(p_i)_{I\times 1}$, $\boldsymbol{c}=(c_j)_{J\times 1}$, $\boldsymbol{s}=(s_j)_{J\times 1}$, $\boldsymbol{r}=(r_{jk})_{J\times K}$, $\boldsymbol{r}_k=(r_{jk})_{J\times 1}$, $\boldsymbol{q}=(q_i)_{I\times 1}$, $\boldsymbol{x}^{\pi t}=\left(x_{ij}^{\pi t}\right)_{I\times J}$, $\boldsymbol{x}_i^{\pi t}=\left(x_{ij}^{\pi t}\right)_{1\times J}$, $\boldsymbol{x}_j^{\pi t}=\left(x_{ij}^{\pi t}\right)_{I\times 1}$, $\boldsymbol{y}^{\pi t}=\left(y_{ijk}^{\pi t}\right)_{I\times J\times K}$, and $\boldsymbol{y}_k^{\pi t}=\left(y_{ijk}^{\pi t}\right)_{I\times J\times 1}$. Let $\boldsymbol{e}$ be a column vector with all its entries equal 1 and its dimension changes according to where it is used. Let $\boldsymbol{X}'$ be the transpose of a matrix $\boldsymbol{X}$. For any matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, let $\boldsymbol{X}\wedge\boldsymbol{Y}$ ($\boldsymbol{X}\vee\boldsymbol{Y}$) denote their entry-wise minimum (maximum) matrix. Let $\boldsymbol{X}^{+}=\boldsymbol{X}\vee\boldsymbol{0}$. Similarly, $\boldsymbol{X}\leq\boldsymbol{Y}$ ($\boldsymbol{X}\geq\boldsymbol{Y}$) implies $X_{ij}\leq Y_{ij}$ ($X_{ij}\geq Y_{ij}$), for $i\in\mathcal{I},j\in\mathcal{J}$. Define feasible sets $\mathcal{Q}=\{\boldsymbol{q}\mid\sum_{i=1}^{I}q_i\leq\sum_{j=1}^{J}c_j;\boldsymbol{q}\geq\boldsymbol{0}\}$ and $\mathcal{X}=\{\boldsymbol{x}\mid\sum_{j=1}^{J}x_{ij}=q_i,i\in\mathcal{I};\sum_{i=1}^{I}x_{ij}\leq c_j,j\in\mathcal{J};\boldsymbol{x}\geq\boldsymbol{0}\}$.

The retailer's objective is to maximize her expected profit over the T periods subject to various constraints. Since $q_i^{\pi}=\sum_{j=1}^{J}x_{ij}^{\pi 1}$, for $i\in\mathcal{I},\pi\in\Pi$, we can substitute q_i^{π} by $\sum_{j=1}^{J}x_{ij}^{\pi 1}$. The retailer optimizes the decisions $\boldsymbol{x}^{\pi}=(\boldsymbol{x}^{\pi 1},...,\boldsymbol{x}^{\pi T})$ and $\boldsymbol{y}^{\pi}=(\boldsymbol{y}^{\pi 1},...,\boldsymbol{y}^{\pi T})$ as follows:

$$\max_{\pi\in\Pi}\quad u(\boldsymbol{x}^{\pi},\boldsymbol{y}^{\pi})=-\sum_{i=1}^{I}\sum_{j=1}^{J}\rho_i x_{ij}^{\pi 1}-\sum_{i=1}^{I}\sum_{j=1}^{J}s_j x_{ij}^{\pi 1}+W(\boldsymbol{x}^{\pi},\boldsymbol{y}^{\pi}) \tag{1}$$

$$s.t. \qquad W(\boldsymbol{x}^\pi, \boldsymbol{y}^\pi) = \sum_{t=1}^{T}\sum_{i=1}^{I} E_{\tilde{\boldsymbol{d}}^1,\ldots,\tilde{\boldsymbol{d}}^T}\left[-h_i \sum_{j=1}^{J} x_{ij}^{\pi,t+1} + \sum_{j=1}^{J}\sum_{k=1}^{K} (p_i - r_{jk})\, y_{ijk}^{\pi t}\right]; \tag{2}$$

$$\sum_{i=1}^{I} x_{ij}^{\pi 1} \le c_j, \quad j \in \mathcal{J}; \tag{3}$$

$$x_{ij}^{\pi,t+1} = x_{ij}^{\pi t} - \sum_{k=1}^{K} y_{ijk}^{\pi t}, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}^-; \tag{4}$$

$$\sum_{j=1}^{J} y_{ijk}^{\pi t} \le \tilde{d}_{ik}^t, \quad i \in \mathcal{I}, k \in \mathcal{K}, t \in \mathcal{T}; \tag{5}$$

$$\sum_{k=1}^{K} y_{ijk}^{\pi t} \le x_{ij}^{\pi t}, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}; \tag{6}$$

$$x_{ij}^{\pi t} \ge 0, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}; \tag{7}$$

$$y_{ijk}^{\pi t} \ge 0, \quad i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}, t \in \mathcal{T}. \tag{8}$$

The first and second terms of the objective function (1) represent the total purchase cost and the total storage cost respectively. The third term represents the expected total revenue minus total holding and retrieval costs. Constraint (3) ensures that the inventory in each warehouse is within its capacity. Constraint (4) ensures that the remaining inventory in one period is carried over to the next period. Constraint (5) ensures that for each period the total quantity of product i retrieved for each zone is at most the zone's demand. Constraint (6) ensures that for each period the total quantity of product i retrieved from each warehouse does not exceed the product's inventory in the warehouse.

Problem (1) can be divided into two parts: the ordering and the storage problems at the start of period 1 and the retrieval problem in the following T periods. Given the ordering and the storage decisions, the retrieval problem is a dynamic program with an initial state $\boldsymbol{x}^{\pi t}$ and an action $\boldsymbol{y}^{\pi t}$ in each period t. If I, J, or K is large, the state space or the action space of the retrieval problem is large, making Problem (1) challenging to solve. We first study some structural properties of the single-zone problem in Section 4. Inspired by the single-zone problem, we then develop a heuristic to find a feasible ordering and storage policy for the multi-zone problem in Section 5.

4 The single-zone problem

In this section, we study a special case with only a single zone but multiple warehouses. Thus, we drop the subscript k and Problem (1) reduces to the following form:

$$\max_{\pi \in \Pi} \qquad u(\boldsymbol{x}^\pi, \boldsymbol{y}^\pi) = -\sum_{i=1}^{I}\sum_{j=1}^{J} \rho_i x_{ij}^{\pi 1} - \sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij}^{\pi 1} + W(\boldsymbol{x}^\pi, \boldsymbol{y}^\pi) \tag{9}$$

$$s.t. \qquad W(\boldsymbol{x}^\pi, \boldsymbol{y}^\pi) = \sum_{t=1}^{T}\sum_{i=1}^{I} E_{\tilde{\boldsymbol{d}}^1,\ldots,\tilde{\boldsymbol{d}}^T}\left[-h_i \sum_{j=1}^{J} x_{ij}^{\pi,t+1} + \sum_{j=1}^{J} (p_i - r_j)\, y_{ij}^{\pi t}\right]; \tag{10}$$

$$\sum_{i=1}^{I} x_{ij}^{\pi 1} \le c_j, \quad j \in \mathcal{J}; \tag{11}$$

$$x_{ij}^{\pi,t+1} = x_{ij}^{\pi t} - y_{ij}^{\pi t}, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}^{-}; \tag{12}$$

$$\sum_{j=1}^{J} y_{ij}^{\pi t} \le \tilde{d}_i^{\pi t}, \quad i \in \mathcal{I}, t \in \mathcal{T}; \tag{13}$$

$$y_{ij}^{\pi t} \le x_{ij}^{\pi t}, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}; \tag{14}$$

$$x_{ij}^{\pi t} \ge 0, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}; \tag{15}$$

$$y_{ij}^{\pi t} \ge 0, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}. \tag{16}$$

We label the warehouses so that $r_1 \le r_2 \le \ldots \le r_J$. If there exist two warehouses j and j' such that $r_j = r_{j'}$, then we set $j < j'$ if and only if $s_j < s_{j'}$. Define $\psi_j = r_j - r_{j-1}$, for $j = 1, ..., J$ with $r_0 = 0$. We assume there exist no warehouses with an identical unit storage cost and an identical unit retrieval cost because otherwise these warehouses can be combined.

Example 1 Figure 2 shows an example with $J = 5$ warehouses, each is represented by a filled circle. Each product is shipped from the supplier O to some warehouses, before it is shipped to the demand zone D. We travel along the grid lines and the travel cost between any two connected grid points is 1 unit. Warehouse j has unit retrieval and storage costs (r_j, s_j) in the subscript. ■

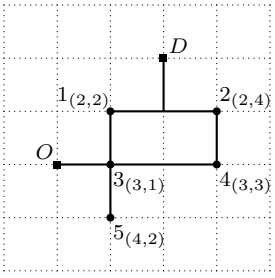


Figure 2: An online retailer's network with five warehouses

Lemma 1 shows that Problem (9) can be simplified to a single-period problem.

Lemma 1 (Equivalence to A Single-Period Problem)

1. The optimal retrieval policy is a greedy policy that, in each period t, keeps retrieving each product i from a warehouse with the smallest index that contains the product until the product is out of stock or the product's demand is fulfilled. That is, the optimal retrieval quantity is

$$y_{ij}^t = \min\left\{\sum_{\ell=1}^{j} x_{i\ell}^t, d_i^t\right\} - \min\left\{\sum_{\ell=1}^{j-1} x_{i\ell}^t, d_i^t\right\}, \qquad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}.$$

2. Under the optimal retrieval policy, Problem (9) can be simplified to a single-period problem:

$$\begin{aligned} Z^* = \quad \max \quad & u(\boldsymbol{x}) = -\sum_{i=1}^{I}\sum_{j=1}^{J}(\rho_i + s_j)x_{ij} + W(\boldsymbol{x}) \\ s.t. \quad & \sum_{i=1}^{I} x_{ij} \le c_j, \quad j \in \mathcal{J}; \\ & x_{ij} \ge 0, \qquad i \in \mathcal{I}, j \in \mathcal{J}; \end{aligned} \tag{17}$$

where

$$W(\boldsymbol{x}) = \sum_{i=1}^{I}\left[-h_i\sum_{t=1}^{T}\left[\sum_{\ell=1}^{J}x_{i\ell} - G_i^t\left(\sum_{\ell=1}^{J}x_{i\ell}\right)\right] + (p_i - r_J)G_i^T\left(\sum_{\ell=1}^{J}x_{i\ell}\right) + \sum_{j=2}^{J}\psi_j G_i^T\left(\sum_{\ell=1}^{j-1}x_{i\ell}\right)\right], \quad (18)$$

and $G_i^t(x) = E\left[\min\left(x, \sum_{\tau=1}^{t}\tilde{d}_i^\tau\right)\right]$.

4.1 Convergence and efficiency of first-order methods

Lemma 1 implies that Problem (17) is a single-period optimization problem. Since $dG_i^t(x)/dx = P\left(\sum_{\tau=1}^{t}\tilde{d}_i^\tau > x\right) = \bar{F}_i^t(x)$, as long as $\bar{F}_i^t(x)$ is known for all $i \in \mathcal{I}, t \in \mathcal{T}$, the gradient $\nabla u(\boldsymbol{x})$ can be explicitly computed without demand samples. This means that one can readily apply many first-order methods. We will show that the objective function and feasible region of Problem (17) have some nice properties that ensure the convergence and computational efficiency of these first-order methods.

Definition 1 *A twice differentiable function $g(\boldsymbol{x})$ defined on $\mathcal{X}$ has an L-Lipschitz continuous gradient (or equivalently, $g(\boldsymbol{x})$ is L-smooth) for $L > 0$ if $\lambda_{\max}\left(\nabla^2 g(\boldsymbol{x})\right) \le L, \forall \boldsymbol{x} \in \mathcal{X}$, where $\lambda_{\max}(A) = \max\left(|\lambda_1(A)|, |\lambda_2(A)|, \cdots, |\lambda_n(A)|\right)$ is the largest eigenvalue of a symmetric matrix A.*

Definition 2 *A function $g : \mathcal{X} \mapsto \mathbb{R}$ is α-strongly concave if the smallest eigenvalue of $-\nabla^2 g(\boldsymbol{x})$ is greater than or equal to α. That is, $-\boldsymbol{w}'\nabla^2 g(\boldsymbol{x})\boldsymbol{w} \ge \alpha\|\boldsymbol{w}\|^2,\ \forall \boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \forall \boldsymbol{w} \in \mathbb{R}^n$.*

Lemma 2 *The objective function (17) has an L-Lipschitz continuous gradient with $L = J(p_{\max} + h_{\max}T - r_1)f_{\max}$, where $p_{\max} = \max_{i\in\mathcal{I}} p_i$, $h_{\max} = \max_{i\in\mathcal{I}} h_i$, and $f_{\max} = \max_{i\in\mathcal{I},t\in\mathcal{T}}\max_{x\in\left[0,\sum_{\tau=1}^{T}\bar{d}_i^\tau\right]} f_i^t(x)$. Furthermore, if $\psi_j > 0$, $j \in \mathcal{J}$, and $0 < \underline{w} < f_i^t(x)$, for $0 \le x \le \sum_{j=1}^{J} c_j$, $i \in \mathcal{I}$, $t \in \mathcal{T}$, the objective function (17) is α-strongly concave with*

$$\alpha = \frac{1}{2}\underline{w}\sum_{i=1}^{I}\min\left(p_i + h_i T - r_J, \min_{j=2,\ldots,J}\psi_j\right).$$

Lemma 2 guarantees the convergence of many first-order methods. Furthermore, it implies that if $f_i^t(x)$, $i \in \mathcal{I}, t \in \mathcal{T}$, follows a truncated normal, log-normal, or exponential distribution, or if it follows a uniform distribution in $[0, U]$ for a sufficiently large U, then $u(\boldsymbol{x})$ is strongly concave, which ensures faster convergence.

Besides having nice properties of the objective function, Problem (17) has a feasible region that can be efficiently projected onto. This feasible region is the intersection of J separable simplices $\Delta_j \triangleq \{\sum_{i\in\mathcal{I}} x_{ij} \le c_j, x_{ij} \ge 0, i \in \mathcal{I}\}$, $j \in \mathcal{J}$. We just need to project $\boldsymbol{x}_j$ onto the simplex Δ_j separately for each warehouse $j \in \mathcal{J}$. Many efficient algorithms can achieve this, which is much faster than projecting it onto a general polytope. Section 5 develops a first-order method to approximately solve the multi-zone problem based on the theoretical properties in Lemma 2. Section 6 investigates the performance of this heuristic in various numerical experiments.

4.2 Structural properties of the optimal storage policy

In this section, we further analyze the structure of the storage decisions $\boldsymbol{x}^*(\boldsymbol{q})$ given the order quantities $\boldsymbol{q}$. Through this analysis, we develop an algorithm given $\boldsymbol{q}$ that can find a near optimal storage policy $\boldsymbol{x}^*(\boldsymbol{q})$ in polynomial time, which is significantly faster than conventional first-order algorithms in convex optimization (see Appendix E for details).

Problem (17) can be reformulated as two subproblems. We drop the superscript T from $f_i^T(x)$, $F_i^T(x)$, $\bar{F}_i^T(x)$, and $G_i^T(x)$ for convenience. In the first subproblem, given the ordering decisions $\boldsymbol{q}$, the retailer finds the storage decisions $\boldsymbol{x}^*(\boldsymbol{q})$ by solving the *storage problem*:

$$
\begin{aligned}
V(\boldsymbol{q}) = \quad \max \quad & G(\boldsymbol{x}) = -\sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij} + \sum_{i=1}^{I}\sum_{j=2}^{J} \psi_j G_i\left(\sum_{\ell=1}^{j-1} x_{i\ell}\right) \\
s.t. \quad & \sum_{j=1}^{J} x_{ij} = q_i, \quad i \in \mathcal{I}; \\
& \sum_{i=1}^{I} x_{ij} \le c_j, \quad j \in \mathcal{J}; \\
& x_{ij} \ge 0, \qquad i \in \mathcal{I}, j \in \mathcal{J}.
\end{aligned} \tag{19}
$$

The retailer then determines the optimal ordering decisions $\boldsymbol{q}^*$ by solving the *ordering problem*:

$$
\begin{aligned}
\max \quad & -\sum_{i=1}^{I}\left[(\rho_i + h_i T)q_i - (p_i - r_J)\, G_i\left(q_i\right) - h_i \sum_{t=1}^{T} G_i^t\left(q_i\right)\right] + V(\boldsymbol{q}) \\
s.t. \quad & \sum_{i=1}^{I} q_i \le \sum_{j=1}^{J} c_j; \\
& q_i \ge 0, \qquad i \in \mathcal{I}.
\end{aligned} \tag{20}
$$

4.2.1 Solving the storage problem

Since $\boldsymbol{q}$ is fixed in the storage problem (19), we drop it from the notation and write $\boldsymbol{x}^*(\boldsymbol{q})$ as $\boldsymbol{x}^*$. We assume $q_i \le \sum_{t=1}^{T} \bar{d}_i^t$, for $i \in \mathcal{I}$. We introduce Algorithm 1 to find $\boldsymbol{x}^*$. The main idea of Algorithm 1 is to iteratively produce *intermediate storage matrices* of the form $\boldsymbol{z}^* = \left(z_{ij}^*\right)_{I\times J}$, where z_{ij}^* represents the storage quantity of product i in warehouse j for one iteration. Specifically, for each iteration, starting with an initial storage matrix $\boldsymbol{v}$, Algorithm 1 produces a subsequent intermediate storage matrix $\boldsymbol{z}^*$ that satisfies a certain structured property and stores more products than $\boldsymbol{v}$. After that, $\boldsymbol{v}$ is updated to $\boldsymbol{z}^*$ and the algorithm proceeds to the next iteration. When Algorithm 1 terminates, all the inventory $\boldsymbol{q}$ is properly stored, producing an optimal storage matrix. Now, we illustrate Algorithm 1 in detail. Given $\boldsymbol{q}$, $\boldsymbol{c}$, and an initial storage matrix $\boldsymbol{v}$, define the feasible set of $\boldsymbol{z}^*$ as $\mathcal{Z}(\boldsymbol{v}) = \left\{\boldsymbol{z} \middle| \sum_{j=1}^{J} v_{ij} \le \sum_{j=1}^{J} z_{ij} \le q_i, i \in \mathcal{I}; \sum_{i=1}^{I} v_{ij} \le \sum_{i=1}^{I} z_{ij} \le c_j, j \in \mathcal{J}\right\}$. The first set of inequalities in $\mathcal{Z}(\boldsymbol{v})$ ensures that the total amount of product i allocated to different warehouses is non-decreasing, but does not exceed its order quantity. The second set of inequalities requires that the total amount of all the products in warehouse j is also non-decreasing, but does

not exceed the warehouse's capacity. Let $\mathcal{P}$ be a set of products with remaining quantities to be stored and Γ be a set of non-full warehouses. Let $\mathcal{A} = \{(i,j)|i \in \mathcal{P}, j \in \Gamma\}$. Each pair (i,j) in $\mathcal{A}$ indicates that product i can be stored to warehouse j. N

Algorithm 1 (Storage Algorithm)
Given order quantities $\boldsymbol{q}$, initialize $\boldsymbol{v} = \boldsymbol{0}$ and $\hat{\boldsymbol{q}} = \boldsymbol{q}$.
1. Set $\mathcal{P} = \{i|\hat{q}_i > 0, i \in \mathcal{I}\}$, set $\Gamma = \Gamma(\boldsymbol{v})$, and set $\mathcal{A} = \{(i,j)|i \in \mathcal{P}, j \in \Gamma\}$.
2. If $\mathcal{A} = \emptyset$, then terminate and return $\boldsymbol{x} = \boldsymbol{v}$; otherwise, call Algorithm 3 to compute $\boldsymbol{z}^* \in \mathcal{Z}(\boldsymbol{v})$.
3. Set $\boldsymbol{v} \leftarrow \boldsymbol{z}^*$, $\hat{\boldsymbol{q}} \leftarrow \boldsymbol{q} - \boldsymbol{z}^*\boldsymbol{e}$, and go to step 1.

Step 1 of Algorithm 1 updates the sets $\mathcal{P}$, Γ, and $\mathcal{A}$. Section 4.2.2 describes how to select the set $\Gamma(\boldsymbol{v})$ of non-full warehouses. Step 2 chooses an itermediate storage matrix $\boldsymbol{z}^*$ by calling Algorithm 3 in Appendix B. Section 4.2.3 provides the details of finding $\boldsymbol{z}^*$. Step 3 updates the inventory levels $\boldsymbol{v}$. Algorithm 1 terminates and outputs a storage policy $\boldsymbol{x}$ when $\mathcal{A}$ becomes empty. In each iteration, Algorithm 1 increases the product quantities in the warehouses. As required by $\mathcal{Z}(\boldsymbol{v})$, the algorithm ensures that $\sum_{j=1}^{J} z_{ij}^* \geq \sum_{j=1}^{J} v_{ij}, i \in \mathcal{I}$ and $\sum_{i=1}^{I} z_{ij}^* \geq \sum_{i=1}^{I} v_{ij}, j \in \mathcal{J}$. However, it is possible to have $z_{ij}^* < v_{ij}$ for some i, j. This *non-monotonicity property* of z_{ij}^* differentiates Algorithm 1 from the greedy algorithms by Lovász (1983), Federgruen and Groenevelt (1986), and Liu (2017). Thus, Algorithm 1 is a non-greedy algorithm.

4.2.2 Selecting warehouses for storage: Pareto frontier

The optimality properties in the following theorem help us to construct $\Gamma(\boldsymbol{v})$ in Algorithm 1.

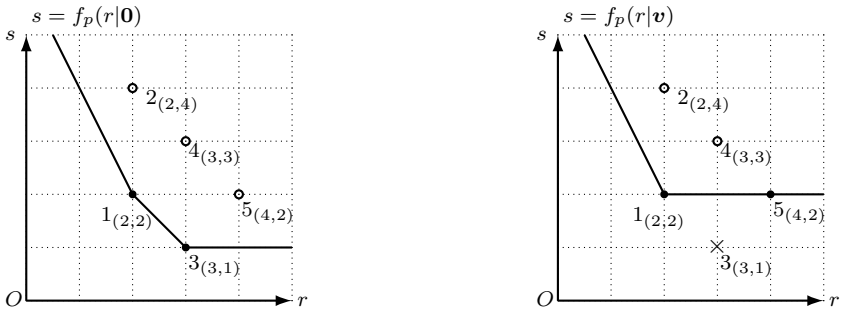
Theorem 1 (Prioritizing Warehouses)
For any two non-full warehouses $j < j'$ ($r_j \leq r_{j'}$), if $s_j < s_{j'}$, then it is optimal to fill up warehouse j before filling warehouse j'. In this situation, we say warehouse j dominates warehouse j'.

Theorem 1 implies that we should fill up warehouses with lower unit retrieval and storage costs before filling warehouses with higher unit retrieval and storage costs.

Definition 3 *A function $s = f_p(r|\boldsymbol{v})$ on an r-s plane is a Pareto frontier if it is a lower envelope of the set $\left\{(r_j, s_j) \,|\, \sum_{i=1}^{I} v_{ij} < c_j, j \in \mathcal{J}\right\}$ and is piecewise-linear non-increasing convex.*

We choose $\Gamma(\boldsymbol{v}) = \left\{j|f_p\left(r_j|\boldsymbol{v}\right) = s_j, \sum_{i=1}^{I} v_{ij} < c_j, j \in \mathcal{J}\right\}$ as the set of warehouses on the Pareto frontier, which are not dominated by other warehouses. According to Theorem 1, in each iteration of Algorithm 1, it is optimal to select the warehouses on the Pareto frontier for storage.

Example 2 Figure 3 plots the warehouse indices of Figure 2 on an r-s plane. Suppose $\boldsymbol{v} = \boldsymbol{0}$, the Pareto frontier $s = f_p(r|\boldsymbol{0})$ is a solid line in Figure 3(a). We have $\Gamma(\boldsymbol{0}) = \{1, 3\}$. Now suppose warehouse 3 is completely filled, the inventory levels become $\boldsymbol{v} \geq 0$. The Pareto frontier $s = f_p(r|\boldsymbol{v})$ changes to a solid line in Figure 3(b). We now have $\Gamma(\boldsymbol{v}) = \{1, 5\}$. ■



(a) All warehouses are empty. (b) Warehouse 3 is full.

Figure 3: Warehouses on the Pareto frontier are selected for storage

As we run Algorithm 1, the warehouses on the Pareto frontier are gradually filled up. The warehouses below the frontier are fully filled, while the warehouses above it remain empty.

4.2.3 Determining the itermediate storage matrix $\boldsymbol{z}^*$

In step 2 of each iteration of Algorithm 1, our goal is to construct an intermediate storage matrix $\boldsymbol{z}^*$ that satisfies the Karush–Kuhn–Tucker (KKT) conditions of the storage problem (19) with order quantities $\boldsymbol{z}^*\boldsymbol{e}$. We will show that if $\boldsymbol{z}^*$ is a *structured* storage matrix defined in Definition 4 below, then it will also satisfy these KKT conditions. Furthermore, at the start of each iteration of Algorithm 1, if $\boldsymbol{v}$ is structured, then the algorithm will generate the next storage matrix $\boldsymbol{z}^*$ that is also structured (as will be shown in Lemma 3 below). This structured matrix $\boldsymbol{z}^*$ will become the storage matrix at the start of the next iteration. Since in the first iteration, $\boldsymbol{v} = \mathbf{0}$ is structured, the final output of Algorithm 1 must also be structured and thus satisfy the KKT conditions. Therefore, Algorithm 1 generates an optimal storage matrix (subject to numerical errors).

To define a structured storage matrix, we need the following definitions. Recall that $G(\boldsymbol{x})$ denotes the objective function (19). Given $\boldsymbol{v}$, define the marginal cost of storing a unit of product i to warehouse j as $D_{ij}(\boldsymbol{v}) = -\left.\frac{\partial G(\boldsymbol{x})}{\partial x_{ij}}\right|_{\boldsymbol{x}=\boldsymbol{v}} = s_j - \sum_{u=j+1}^{J} \psi_u \bar{F}_i\left(\sum_{\ell=1}^{u-1} v_{i\ell}\right)$. Since the KKT conditions involve the marginal costs $D_{ij}(\boldsymbol{v})$, which are expressed in stockout probabilities $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right)$, it is more straightforward to use the stockout probabilities than the storage matrix $\boldsymbol{v}$ when analyzing the KKT conditions. Define $i^* = \arg\min_{i\in\mathcal{I}} \ \bar{F}_i(q_i)$ and $j_v^* = \max\left\{j | j = \arg\min_{\ell\in\Gamma(\boldsymbol{v})} D_{i^*\ell}(\boldsymbol{v})\right\}$. A *target warehouse* j_v^* is the largest-index warehouse to store product i^* with the least marginal cost. Part 1 of Lemma 1 shows that it is optimal to retrieve a product from the smallest-index warehouse to the largest-index warehouse containing the product. Based on this optimal retrieval policy, for a given $\boldsymbol{v}$, define a *target stockout probability* for warehouse j as $\chi_j(\boldsymbol{v}) = \bar{F}_{i^*}\left(\sum_{\ell=1}^{j} v_{i^*\ell}\right)$. Let $\boldsymbol{\chi}(\boldsymbol{v}) = (\chi_1(\boldsymbol{v}), \chi_2(\boldsymbol{v}), \cdots, \chi_J(\boldsymbol{v}))$ denote a target stockout probability vector. For convenience, let $\chi_0(\cdot) = 1$. To solve the set of equations corresponding to the KKT conditions of the storage problem (19), we will first solve for $\boldsymbol{\chi}(\boldsymbol{v})$. Then, we use the target stockout probability $\chi_j(\boldsymbol{v})$ to determine the storage quantity of product i in warehouse j, for all $i \in \mathcal{I}$. Note that j_v^* and $\boldsymbol{\chi}(\boldsymbol{v})$

are uniquely determined by $\boldsymbol{v}$ in each iteration of Algorithm 1.

Definition 4 *A storage matrix* $\boldsymbol{v}$ *is* structured *if it satisfies the following property:*

1. $G(\boldsymbol{v}) \geq G(\hat{\boldsymbol{z}})$ *for all* $\hat{\boldsymbol{z}} \in \left\{ \boldsymbol{z} \left| \sum_{j=1}^{J} z_{ij} = \sum_{j=1}^{J} v_{ij} \leq q_i, i \in \mathcal{I}; \sum_{i=1}^{I} z_{ij} \leq c_j, j \in \mathcal{J}; \boldsymbol{z} \geq \boldsymbol{0} \right. \right\}$.

2. For $i \in \mathcal{P}$, $j_v^* = \max\left\{ j | j = \arg\min_{\ell \in \Gamma(\boldsymbol{v})} D_{i\ell}(\boldsymbol{v}) \right\}$.

3. For $i \in \mathcal{P}$, $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) = \chi_j(\boldsymbol{v})$ *for* $j \in \mathcal{J}$*; for* $i \notin \mathcal{P}$, $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) \geq \chi_j(\boldsymbol{v})$ *for* $j \in \mathcal{J}$.

4. $v_{ij} = \min\left\{ \bar{F}_i^{-1}\left(\chi_j(\boldsymbol{v})\right), q_i \right\} - \min\left\{ \bar{F}_i^{-1}\left(\chi_{j-1}(\boldsymbol{v})\right), q_i \right\}, i \in \mathcal{I}, j \in \mathcal{J}$.

For $i \in \mathcal{I}$*, there exists a critical warehouse* $j_c(i) \in \mathcal{J}$ *such that*

for $j < j_c(i)$*, we have* $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) = \chi_j(\boldsymbol{v})$ *and* $v_{ij} \geq 0$*;*

for $j = j_c(i)$*, we have* $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) \geq \chi_j(\boldsymbol{v})$ *and* $v_{ij} \geq 0$*; and*

for $j > j_c(i)$*, we have* $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) > \chi_j(\boldsymbol{v})$ *and* $v_{ij} = 0$.

5. Each warehouse $j > j_v^*$ *is either empty or completely filled.*

Remark 1: Part 1 of Definition 4 implies that $\boldsymbol{v}$ is iteration-wise optimal. Part 2 ensures that the target warehouse j_v^* is the largest-index warehouse with the least marginal cost for all product $i \in \mathcal{P}$. Part 3 implies that the products that have not been stored completely meet the target stockout probability for all the warehouses, whereas the products that have been stored completely may result in a larger stockout probability for some warehouses. Part 4 shows an optimality condition for $\boldsymbol{v}$. It also implies that for any product i, there exists a critical warehouse $j_c(i)$ such that the target stockout probability can be met at warehouse $j < j_c(i)$. For the critical warehouse $j_c(i)$, its target stockout probability may not be met because we have exhausted all the quantity of product i. For warehouse $j > j_c(i)$, we no longer have any product i. Part 5 ensures that each warehouse $j > j_v^*$ is *not* partially filled.

Recall that Algorithm 1 iteratively generates a sequence of structured storage matrices $\boldsymbol{v}^0, \boldsymbol{v}^1, ..., \boldsymbol{v}^{\hat{n}}$ with $\boldsymbol{v}^0 = \boldsymbol{0}$ and $\boldsymbol{v}^{\hat{n}}$ satisfying $\sum_{j=1}^{J} v_{ij}^{\hat{n}} = q_i, i \in \mathcal{I}$. Due to part 1 of Definition 4, $\boldsymbol{v}^{\hat{n}}$ is an optimal storage matrix for the storage problem. We now describe the procedure of finding the next structured storage matrix given the current one, which is the main goal of Algorithm 3 called in step 2 of Algorithm 1 (the details of Algorithm 3 are provided in Appendix B).

Specifically, given $\boldsymbol{v}^n$ for $n < \hat{n}$, our goal is to allocate products to the target warehouse $j_{v^n}^*$, where the marginal cost is the smallest among the non-full warehouses. When allocating products to the target warehouse, we need to maintain the structured property in Definition 4. To ensure that the new storage matrix $\boldsymbol{z}^*$ adheres to parts 3 to 5 of Definition 4, we impose the following conditions on $\boldsymbol{z}^*$:

(i) The inventory levels of warehouse $j < j_{v^n}^*$ remain unchanged.

(ii) For the target warehouse $j_{v^n}^*$, $\sum_{i=1}^{I} z_{ij_{v^n}^*}^* > \sum_{i=1}^{I} v_{ij_{v^n}^*}^n$.

Remark 2: If $\boldsymbol{v}$ is a structured storage matrix, then it has a *nested property*: *If a warehouse stores a certain product, this product is also stored in all the non-empty warehouses with a smaller index (with a lower unit retrieval cost in general). In other words, among all the non-empty warehouses, a smaller-index warehouse contains all the products stored in a larger-index warehouse.* Figure 4 illustrates the nested property. Note that the nested property is non-trivial as we should consider both the unit storage and retrieval costs when making the storage decisions (for example, if one warehouse has a smaller unit retrieval cost but a larger unit storage cost than another). The unit storage costs are sunk costs when the products are replenished, whereas the unit retrieval costs are incurred only if the products are demanded. See Lemma 5 in Appendix A for details.

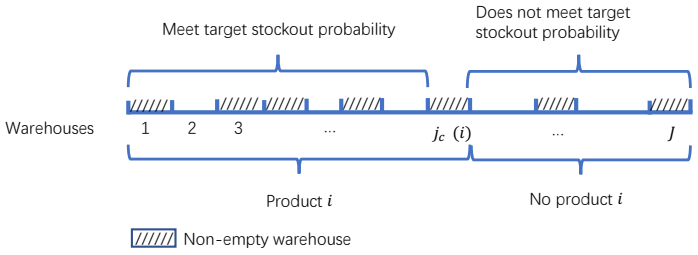


Figure 4: Nested property

(iii) For $j > j^*_{v^n}$, $\chi_j(\boldsymbol{z}^*)$ satisfies $\sum_{i=1}^{I}\left[\min\left\{\bar{F}_i^{-1}\left(\chi_j(\boldsymbol{z}^*)\right), q_i\right\} - \min\left\{\bar{F}_i^{-1}\left(\chi_{j-1}(\boldsymbol{z}^*)\right), q_i\right\}\right] = \sum_{i=1}^{I} v_{ij}^n$.
(iv) $z^*_{ij} = \min\left\{\bar{F}_i^{-1}\left(\chi_j(\boldsymbol{z}^*)\right), q_i\right\} - \min\left\{\bar{F}_i^{-1}\left(\chi_{j-1}(\boldsymbol{z}^*)\right), q_i\right\}$, for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$.

To make $\hat{n}$ as small as possible, in each iteration of Algorithm 1, we aim to fill as much as possible the target warehouse $j^*_{v^n}$ that satisfies part 2 of Definition 4. Three scenarios may arise:

1. The total volume of the remaining unallocated products *equals or exceeds* a threshold. In this case, we fill the target warehouse to its capacity while ensuring the storage matrix $\boldsymbol{z}^*$ satisfies parts 3 to 5 of Definition 4. This is executed by the first part of step 2 of Algorithm 3 in Appendix B. Furthermore, if the marginal cost at the target warehouse remains the least among the non-full warehouses, then step 3a of Algorithm 3 returns $\boldsymbol{z}^*$.

2. The total volume of the remaining unallocated products is *smaller* than the threshold. The second part of step 2 of Algorithm 3 addresses this situation. Furthermore, if the marginal cost at the target warehouse remains the least among the non-full warehouses, then step 3a of Algorithm 3 returns $\boldsymbol{z}^*$.

3. After partially filling the target warehouse, the marginal cost at the target warehouse is no longer the least among the non-full warehouses. This scenario is called *crossing*, prompting a new iteration of Algorithm 1 with a *new* target warehouse. Figure 13 in Appendix B illustrates this scenario. Step 3a of Algorithm 3 checks whether crossing happens. If so, a binary search in step 3b locates the exact crossing point and returns the corresponding

storage matrix $\boldsymbol{z}^*$.

The above principle underpins the methodology of Algorithm 3. The following lemma rigorously shows that Algorithm 1 preserves the structured property in Definition 4.

Lemma 3 *For each iteration of Algorithm 1, if the initial storage matrix $\boldsymbol{v}$ is structured, then the intermediate storage matrix $\boldsymbol{z}^*$ is also structured, and $\boldsymbol{z}^*$ satisfies the KKT conditions of the storage problem (19) with order quantities $\boldsymbol{q} = \boldsymbol{z}^*\boldsymbol{e}$.*

Furthermore, we can show that the total number of iterations in Algorithm 1 is no more than $2J-1$. We call $\boldsymbol{x}$ an *ϵ-optimal storage matrix* if there exists an optimal storage solution $\boldsymbol{x}^*$ such that $\max_{i\in\mathcal{I},j\in\mathcal{J}} |x_{ij} - x^*_{ij}| \leq \epsilon$. The following theorem shows that by setting the accuracy ϵ of the binary search in Algorithm 3, Algorithm 1 finds an ϵ-optimal storage matrix in polynomial time.

Theorem 2 *Given $\boldsymbol{q}$ and $\epsilon > 0$, Algorithm 1 obtains an ϵ-optimal storage matrix in no more than $2J-1$ iterations. Furthermore, its computational cost is at most $O\left(IJ^2T\ \log^2\left(2\bar{L}/\epsilon\right) C\left(\bar{F}^{-1}\right)\right)$, where $C\left(\bar{F}^{-1}\right)$ is the maximum computational effort to call the function $\bar{F}_i^{-1}(\cdot)$, and $\bar{L}$ is the maximum Lipschitz constant of $\bar{F}_i^{-1}(\cdot)$.*

Note that I is generally large for online retailing. Fortunately, Theorem 2 shows that the run time of Algorithm 1 is linear in I. The total run time of Algorithm 1 is polynomial if $\bar{F}_i^{-1}(\cdot)$ can be evaluated in polynomial time. Appendix E demonstrates that Algorithm 1 numerically outperforms the first-order methods in finding $\boldsymbol{x}^*$. Our analysis of the storage problem (19) not only identifies the structure of $\boldsymbol{x}^*$ (the nested property in Figure 4), but also provides a computationally efficient algorithm to compute $\boldsymbol{x}^*$ given $\boldsymbol{q}$.

The following example demonstrates the storage procedure described in Algorithms 1 and 3.

Example 3 Consider a retailer with two warehouses. We set $r_1 = 1$, $r_2 = 9$, $s_1 = 4$, $s_2 = 2$, and $c_1 = c_2 = 20$. Assume two products $i \in \{A, B\}$ and $\tilde{d}_i$, $i = A, B$, follow a uniform distribution on $[0, 20]$. Suppose $q_A = 15$ and $q_B = 16$. In the first iteration of Algorithm 1, the Pareto frontier defined in Section 4.2.2 implies $\Gamma = \{1, 2\}$. Since $\boldsymbol{v} = \boldsymbol{0}$, we have the target stockout probability $\chi_1(\boldsymbol{v}) = \chi_2(\boldsymbol{v}) = 1$. The marginal cost of storing a unit of product i to warehouses 1 and 2 is $D_{i1}(\boldsymbol{0}) = -4$ and $D_{i2}(\boldsymbol{0}) = 2$, for $i = A, B$, respectively. By definition, $j^*_v = 1$. Note that when $z_{i1} = 5$ and $z_{i2} = 0$, we have $D_{i1}(\boldsymbol{z}) = D_{i2}(\boldsymbol{z}) = -4.75$, for $i = A, B$. Thus, the crossing occurs and Algorithm 3 returns $z^*_{i1} = 5$ and $z^*_{i2} = 0$, for $i = A, B$.

In the second iteration of Algorithm 1, $v_{i1} = 5$ and $v_{i2} = 0$, for $i = A, B$. The target stockout probability for warehouses 1 and 2 is $\chi_1(\boldsymbol{v}) = \chi_2(\boldsymbol{v}) = 0.75$. The Pareto frontier implies $\Gamma = \{1, 2\}$. Since $D_{i1}(\boldsymbol{v}) = D_{i2}(\boldsymbol{v}) = -4.75$, for $i = A, B$, by definition, $j^*_v = 2$. Note that $D_{i2}(\boldsymbol{z})$ is always less

than $D_{i1}(\boldsymbol{z})$, for $i = A, B$, when filling up warehouse 2. Thus, according to Algorithm 3, $z^*_{i1} = 5$ and $z^*_{i2} = 10$, for $i = A, B$, and warehouse 2 is now full.

In the third iteration of Algorithm 1, $v_{i1} = 5$ and $v_{i2} = 10$, for $i = A, B$. The target stockout probability for warehouses 1 and 2 is $\chi_1(\boldsymbol{v}) = 0.75$ and $\chi_2(\boldsymbol{v}) = 0.25$. The Pareto frontier implies $\Gamma = \{1\}$ and hence $j^*_v = 1$. Note that the output of Algorithm 3 needs to balance the stockout probabilities of all the products in warehouse 1 and satisfies the structured optimality in Definition 4. Therefore, $z^*_{A1} = z^*_{B1} = 5.5$, $z^*_{A2} = 9.5$, and $z^*_{B2} = 10.5$. The final output of Algorithm 1 is $x^*_{A1} = x^*_{B1} = 5.5$, $x^*_{A2} = 9.5$, and $x^*_{B2} = 10.5$. ■

Example 3 demonstrates that a product's quantity allocated to a warehouse may decrease in the course of Algorithm 1. For instance, in the second iteration, we allocate 10 units of A to warehouse 2, but this amount reduces to 9.5 in the third iteration. This non-monotonicity property differentiates Algorithm 1 from the greedy algorithms by Lovász (1983), Federgruen and Groenevelt (1986), and Liu (2017). It also implies that the sub-modularity property of the rank function discussed in Lovász (1983), Federgruen and Groenevelt (1986), and Liu (2017) is important for the conventional greedy algorithms to be optimal. The above non-monotonicity property also implies that x^*_{ij} is not always increasing in q_i. Algorithm 1 also exhibits an irreversibility property: Once a warehouse is used in an iteration, it will be used for all the future iterations, even though the products allocated to the warehouse may change over the iterations.

4.3 The structure of the optimal ordering policy

Now, we study the ordering problem (20). For each product $i \in \mathcal{I}$, define $\underline{j}(i) = \max\left\{j | x^*_{ij} > 0, j \in \mathcal{J}\right\}$ and $\bar{j}(i) = \max\left\{j | x^*_{ij} > 0, \sum_{\ell=1}^{I} x^*_{\ell j} < c_j, j \in \mathcal{J}\right\}$. Note that $\bar{j}(i)$ is the largest index of a non-full warehouse storing product i. If such a warehouse does not exist, then we set $\bar{j}(i) = 0$. The following theorem characterizes the structure of $\boldsymbol{q}^*$.

Theorem 3 *The optimal order quantities are characterized as follows:*

$$\mu^*_i(\boldsymbol{q}^*) = \rho_i + h_i T - (p_i - r_J)\bar{F}_i(q^*_i) - h_i \sum_{t=1}^{T} Pr\left(q^*_i > \sum_{\tau=1}^{t} \tilde{d}_{i\tau}\right) + K, \tag{21}$$

where $K \geq 0$ *satisfies* $K\left(\sum_{i=1}^{I} q^*_i - \sum_{j=1}^{J} c_j\right) = 0$, *and*

$$\mu^*_i(\boldsymbol{q}) \quad = \quad \begin{cases} -D_{i\bar{j}(i)}(\boldsymbol{x}^*(\boldsymbol{q})), & \text{if } \bar{j}(i) > 0, \\ D_{i^*\underline{j}(i)}(\boldsymbol{x}^*(\boldsymbol{q})) - D_{i\underline{j}(i)}(\boldsymbol{x}^*(\boldsymbol{q})) - D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*(\boldsymbol{q})), & \text{if } \bar{j}(i) = 0. \end{cases}$$

$\mu^*_i(\boldsymbol{q})$ *is non-increasing in* q_ℓ, *for* $\ell \in \mathcal{I}$.

Equation (21) characterizes the optimal ordering policy. The value of K is unique from the KKT conditions. Note that the optimal ordering policy is different from the newsvendor-type

ordering policy because of the holding cost incurred in each period of the retrieval stage. If the holding cost is zero, then Equation (21) degenerates to a newsvendor-type ordering policy. In that case, $\rho_i - \mu_i^*(\boldsymbol{q}^*) + K$ represents the overall cost of ordering one more unit of product i and $p_i - r_J$ represents the smallest effective revenue from the additional unit of product i.

4.4 Special case: The single-warehouse single-zone problem

We consider a special case with $J = 1$ warehouse. Since $\boldsymbol{q} = \boldsymbol{x}$, the optimal ordering and storage policies are equivalent. Problem (17) can be simplified to

$$\begin{aligned} Z^* = \quad \max \quad & \sum_{i=1}^{I} \omega_i(q_i) \\ s.t. \quad & \sum_{i=1}^{I} q_i \le c; \\ & q_i \ge 0, \qquad i \in \mathcal{I}; \end{aligned} \tag{22}$$

where $\omega_i(q) = -(\rho_i + s)q - h_i \sum_{t=1}^{T} \left[q - G_i^t(q)\right] + (p_i - r)G_i^T(q)$. Algorithm 6 in Appendix B describes a procedure to find the optimal order quantities. Furthermore, Theorem 5 in Appendix B shows that Algorithm 6 finds an ϵ-optimal ordering policy within polynomial time. The single-warehouse single-zone problem is quite common in practice. We will apply Algorithm 6 to find the order quantities for the single-warehouse single-zone problem in Sections 6 and 7.

5 The multi-zone problem

Problem (1) with multiple zones is extremely challenging because of its complex network structure and multi-period dynamic feature. To solve the multi-zone problem, we first approximate Problem (1) using three different approximation methods presented in Sections 5.1–5.3. After that we design a *hybrid heuristic* in Section 5.4 that combines the three approximation methods. This hybrid heuristic *only optimizes the ordering and storage decisions*. We will compare the ordering and storage decisions produced by the hybrid heuristic against other benchmark heuristics using a common evaluation method in Section 6.

5.1 LP-Mean approximation

Under the first approximation method, we replace the random demands in (1)–(8) with their means $\hat{d}_{ik}^t$ and jointly optimize the storage and retrieval decisions. This results in a linear program:

$$\begin{aligned} \max \quad & -\sum_{i=1}^{I}\sum_{j=1}^{J} \rho_i x_{ij}^1 - \sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij}^1 + \sum_{t=1}^{T}\sum_{i=1}^{I} \left[-h_i \sum_{j=1}^{J} x_{ij}^{t+1} + \sum_{j=1}^{J}\sum_{k=1}^{K} (p_i - r_{jk})\, y_{ijk}^t \right] \\ s.t. \quad & \sum_{i=1}^{I} x_{ij}^1 \le c_j, \quad j \in \mathcal{J}; \end{aligned} \tag{23}$$

$$x_{ij}^{t+1} = x_{ij}^t - \sum_{k=1}^{K} y_{ijk}^t, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}^-;$$

$$\sum_{j=1}^{J} y_{ijk}^t \le \hat{d}_{ik}^t, \quad i \in \mathcal{I}, k \in \mathcal{K}, t \in \mathcal{T};$$

$$\sum_{k=1}^{K} y_{ijk}^t \le x_{ij}^t, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T};$$

$$x_{ij}^t \ge 0, \quad i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T};$$

$$y_{ijk}^t \ge 0, \quad i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}, t \in \mathcal{T}.$$

We call Problem (23) the *LP-Mean approximation* to Problem (1). By Jensen's inequality, the optimal objective value of Problem (23) serves as an upper bound on the objective of Problem (1) and can be used as a benchmark for our hybrid heuristic.

5.2 Single-zone approximation

We can approximate the multi-zone problem by the single-zone problem studied in Section 4. Specifically, we approximate the objective function $u(\cdot)$ of Problem (1) by aggregating the demands for product i of all the zones to a *virtual zone* with a total demand $\tilde{d}_i^t = \sum_{k\in\mathcal{K}} \tilde{d}_{ik}^t$, for $i \in \mathcal{I}, t \in \mathcal{T}$. We then use a sample-based approach to construct a continuous distribution for each aggregated demand to approximate its empirical distribution.

Algorithm 2 (Continuous Distribution Approximation)
Given N samples $d^{(1)}, d^{(2)}, \ldots, d^{(N)}$ of a non-negative random demand $\tilde{d}$, relabel these samples such that $d^{(1)} < d^{(2)} < \cdots < d^{(N)}$.
1. Construct $N+1$ points $b^{(1)} < b^{(2)} < \cdots < b^{(N+1)}$, where $b^{(1)} = \max\left(0, d^{(1)} - \frac{1}{2}\left(d^{(2)} - d^{(1)}\right)\right)$, $b^{(n)} = \frac{1}{2}\left(d^{(n-1)} + d^{(n)}\right), n = 2, ..., N$, and $b^{(N+1)} = d^{(N)} + \frac{1}{2}\left(d^{(N)} - d^{(N-1)}\right)$.
2. Approximate the c.d.f. of $\tilde{d}$ as

$$F(x) \triangleq P\left(\tilde{d} \le x\right) = \begin{cases} 0, & \text{if } x \le b^{(1)}; \\ \frac{n-1+\left(x-b^{(n)}\right)/\left(b^{(n+1)}-b^{(n)}\right)}{N}, & \text{if } b^{(n)} < x \le b^{(n+1)}, n = 1, \ldots, N; \\ 1, & \text{if } x > b^{(N+1)}. \end{cases} \tag{24}$$

Return the c.d.f. $F(x)$.

Note that we consider all the inequalities $d^{(1)} < d^{(2)} < \cdots < d^{(N)}$ are strict for simplicity. Algorithm 2 can easily be adapted to the case where there are ties. In Algorithm 2, we assume that $\tilde{d}$ is uniformly distributed in each interval $\left(b^{(n)}, b^{(n+1)}\right]$ with a total probability mass $1/N$. There are three advantages of using this algorithm. First, under this approximation, a Lipschitz continuous gradient is guaranteed for $E\left[\min\left(x, \tilde{d}\right)\right]$ for $x \in \left[b^{(1)}, b^{(N+1)}\right]$. Second, it requires only the demand samples without knowing the true demand distributions. Third, computing the c.d.f. of a point x is equivalent to finding which interval x belongs to. Thus, the time complexity is $O(\log N)$ using a binary search. Given N demand samples $\boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, \ldots, \boldsymbol{d}^{(N)}$ of $\tilde{\boldsymbol{d}}^t$, we can first obtain N samples of the aggregated demand $\sum_{\tau=1}^{t} \tilde{d}_i^\tau$ by simple arithmetic calculation and then we can obtain a continuous approximation of the distributions of these random variables.

We define a virtual unit retrieval cost $\hat{r}_{ij}$ as a weighted average of r_{jk} as follows:

$$\hat{r}_{ij} = \frac{\sum_{t=1}^{T}\sum_{k=1}^{K} r_{jk}\hat{d}_{ik}^{t}}{\sum_{t=1}^{T}\sum_{k=1}^{K} \hat{d}_{ik}^{t}}, \quad i \in \mathcal{I}, j \in \mathcal{J}. \tag{25}$$

For each product $i \in \mathcal{I}$, let $\sigma_i(\cdot)$ be a permutation function such that $\hat{r}_{i,\sigma_i(1)} \leq \hat{r}_{i,\sigma_i(2)} \leq \cdots \leq \hat{r}_{i,\sigma_i(J)}$. Define $\psi_{ij} = \hat{r}_{i,\sigma_i(j)} - \hat{r}_{i,\sigma_i(j-1)}$, for $j = 2, \ldots, J$. Following the closed-form formula in Lemma 1, the single-zone approximation $\Phi_{1Z}(\boldsymbol{x})$ of the second-stage expected profit $W(\cdot)$ in (1) is

$$\Phi_{1Z}(\boldsymbol{x}) \triangleq \sum_{i=1}^{I}\left[-h_i\sum_{t=1}^{T}\left[\sum_{\ell=1}^{J} x_{i\ell} - G_i^t\left(\sum_{\ell=1}^{J} x_{i\ell}\right)\right] + (p_i - \hat{r}_{i,\sigma_i(J)})G_i^T\left(\sum_{\ell=1}^{J} x_{i\ell}\right) + \sum_{j=2}^{J}\psi_{ij}G_i^T\left(\sum_{\ell=1}^{j-1} x_{i\ell}\right)\right]. \tag{26}$$

The gradient of $\Phi_{1Z}(\boldsymbol{x})$ has a closed-form expression in terms of the approximate c.d.f. and can be easily computed.

5.3 Single-warehouse approximation

We also approximate Problem (1) by a single-warehouse problem. Specifically, we aggregate each product's storage quantities to a *virtual warehouse* that can fulfill the demands of all the different zones. Define a virtual unit retrieval cost $\tilde{r}_k$ to zone k as the average of r_{jk}: $\tilde{r}_k = \sum_{j=1}^{J} r_{jk}/J, \quad k \in \mathcal{K}$. We reindex the zones such that $\tilde{r}_1 \leq \tilde{r}_2 \leq \cdots \leq \tilde{r}_K$. If two zones have the same virtual unit retrieval cost, then we break the tie arbitrarily. Define $\tilde{\psi}_k = \tilde{r}_k - \tilde{r}_{k-1}$, for $k = 2, ..., K$.

Appendix C studies some properties of the single-warehouse problem. In particular, we derive a closed-form upper bound on the objective function. The gradient of the upper bound requires the knowledge of the c.d.f. of $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$ and we apply Algorithm 2 to obtain their continuous approximation. We construct a single-warehouse approximation using the following upper bound on the second-stage expected profit $W(\cdot)$ in (1):

$$\Phi_{1W}(\boldsymbol{x}) = \sum_{i=1}^{I}\left[-h_i T x_i + \sum_{t=1}^{T} h_i \breve{G}_{i,K}^{t}(x_i) + (p_i - \tilde{r}_K)\breve{G}_{i,K}^{T}(x_i) + \sum_{k=2}^{K}\tilde{\psi}_k \breve{G}_{i,k-1}^{T}(x_i)\right], \tag{27}$$

where $\breve{G}_{i,k}^{t}(x) = E\left[\min\left(x, \sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}\right)\right]$. The gradient of $\Phi_{1W}(\boldsymbol{x})$ has a closed-form expression in terms of the c.d.f. of $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$ and can be easily computed.

5.4 Hybrid heuristic

We propose a hybrid heuristic based on the above approximations to solve Problem (1) with J warehouses, K zones, and T periods. We first calculate the weighted sum of the single-zone and the single-warehouse approximations to approximate the second-stage expected profit $W(\cdot)$ in (1):

$$\Phi_{sp}(\boldsymbol{x}) = \frac{J-1}{J+K-2}\Phi_{1Z}(\boldsymbol{x}) + \frac{K-1}{J+K-2}\Phi_{1W}(\boldsymbol{x}). \tag{28}$$

Note that $\Phi_{sp}(\boldsymbol{x}) = \Phi_{1W}(\boldsymbol{x})$ if $J = 1$, and $\Phi_{sp}(\boldsymbol{x}) = \Phi_{1Z}(\boldsymbol{x})$ if $K = 1$. Then, we solve the following deterministic convex program as an approximation to Problem (1):

$$\max \quad u_{sp}(\boldsymbol{x}) = -\sum_{i=1}^{I}\sum_{j=1}^{J}(\rho_i + s_j)x_{ij} + \Phi_{sp}(\boldsymbol{x}) \tag{29}$$

$$
\begin{aligned}
s.t. \quad & \sum_{i=1}^{I} x_{ij} \leq c_j, \quad j \in \mathcal{J}; \\
& x_{ij} \geq 0, \qquad\quad i \in \mathcal{I}, j \in \mathcal{J}.
\end{aligned}
$$

To solve Problem (29), we adopt the *Fast Iterative Shrinkage-Thresholding Algorithm* (FISTA) proposed by Chambolle and Dossal (2015) (see Appendix D for the details of this algorithm). We call $\hat{\boldsymbol{x}}$ an ϵ-optimal solution if $\hat{\boldsymbol{x}}$ satisfies the constraints in Problem (29) and $u_{sp}(\hat{\boldsymbol{x}}^*) - u_{sp}(\hat{\boldsymbol{x}}) \leq \epsilon$, where $\hat{\boldsymbol{x}}^*$ represents the optimal solution to Problem (29). The following theorem shows the computational complexity of FISTA for solving Problem (29).

Theorem 4 *Under appropriate parameter settings, FISTA produces an ϵ-optimal solution to Problem (29) and its computational cost is at most $O\left(I\left[\left(J^2+JT+K\right)C\left(\bar{F}\right)+J\log\left(\frac{1}{\epsilon}\right)\right]\frac{1}{\sqrt{\epsilon}}\right)$, where $C\left(\bar{F}\right)$ is the maximum computational effort to determine the c.d.f. of the demands in the single-zone approximation and the single-warehouse approximation.*

Theorem 4 implies that we can efficiently obtain a near-optimal solution to Problem (29). Furthermore, if $\Phi_{1Z}(\boldsymbol{x})$ and $\Phi_{1W}(\boldsymbol{x})$ are strongly concave, then the largest computational cost reduces to $O\left(I\left[\left(J^2+JT+K\right)C\left(\bar{F}\right)+J\log\left(\frac{1}{\epsilon}\right)\right]\log\left(\frac{1}{\epsilon}\right)\right)$.

Finally, we compute the weighted average $\boldsymbol{x}^H = \alpha(T)\boldsymbol{x}^{sp} + (1-\alpha(T))\boldsymbol{x}^{lp}$, where $\boldsymbol{x}^{sp}$ and $\boldsymbol{x}^{lp}$ represent the optimal solutions to Problems (29) and (23) respectively. Specifically, the weight $\alpha(T)$ follows a sigmoid function:

$$
\alpha(T) = \frac{1}{\exp\left((T-\mu_T)/\sigma_T\right)+1}, \tag{30}
$$

where μ_T and σ_T are problem-specific parameters. Our hybrid heuristic yields the solution $\boldsymbol{x}^H$. The rationale behind using the above convex combination is that both the single-zone and the single-warehouse approximations aggregate multiple periods into a single period. Consequently, they fail to capture the nuanced relationships between demands in different periods. On the other hand, while the LP-Mean approximation adeptly captures the demands over different periods, it neglects demand stochasticity. Therefore, we employ the sigmoid function to establish the weights between the solutions to Problem (29) and the LP-Mean approximation. Specifically, if T is large, emphasizing the time dependence becomes more crucial than the demand stochasticity, making the LP-Mean approximation more effective. This leads us to opt for a smaller $\alpha(T)$. Conversely, if T is small, the opposite is true.

6 Numerical experiments

We conduct numerical experiments to demonstrate the efficacy of our hybrid heuristic, investigate the impact of different parameters, and identify situations where the benefit of using the hybrid

heuristic is the largest. Section 6.1 introduces two benchmark heuristics and an evaluation method. Section 6.2 compares the hybrid heuristic with these benchmarks by varying a single parameter in each experiment. Furthermore, Appendix E focuses on the storage problem for the single-zone case in Section 4.2. For this special case, we benchmark Algorithm 1 against two asymptotically optimal algorithms. All the numerical experiments (including those in Section 7) were run on a shared server with Intel(R) Xeon(R) Silver 4116 CPU on Linux 64bit.

6.1 Benchmarks and an evaluation method

We compare the hybrid heuristic in Section 5.4 with two benchmarks. The first benchmark is the LP-Mean approximation defined in Section 5.1. The second benchmark is called *LP-Sample approximation*, which approximates the multi-period problem with a single-period problem. Specifically, the LP-Sample approximation computes a storage matrix by ignoring the holding costs and aggregating the demand samples in different periods. Given N demand sample paths $d_{i,k}^{t,(n)}, i \in \mathcal{I}, k \in \mathcal{K}, t \in \mathcal{T}, n = 1, ..., N$, the LP-Sample approximation is formulated as follows:

$$\max_{\boldsymbol{x},\boldsymbol{y}} \quad -\sum_{i=1}^{I}\sum_{j=1}^{J}\rho_i x_{ij} - \sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij} + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{I}\breve{W}_i^{(n)}(\boldsymbol{x}_i),$$

where

$$\begin{aligned}
\breve{W}_i^{(n)}(\boldsymbol{x}_i) = \quad \max \quad & \sum_{j=1}^{J}\sum_{k=1}^{K}(p_i - r_{jk})y_{ijk} \\
s.t. \quad & \sum_{j=1}^{J} y_{ijk} \le \sum_{t=1}^{T} d_{i,k}^{t,(n)}, \quad k \in \mathcal{K}; \\
& \sum_{k=1}^{K} y_{ijk} \le x_{ij}, \qquad j \in \mathcal{J}; \\
& y_{ijk} \ge 0, \qquad j \in \mathcal{J}, k \in \mathcal{K}.
\end{aligned}$$

The LP-Sample approximation is inspired by Lemma 1, which shows that if there is only a single zone, the multi-period problem can be reduced to a single-period problem. However, the LP-Sample approximation is different from the single-period problem in Lemma 1 by ignoring the holding costs. This is because, for the multi-zone setting, the simple structure of the optimal retrieval policy in Lemma 1 breaks down. Thus, determining the holding costs requires solving a multi-period dynamic program for each sample path, which is computationally intractable. Furthermore, as we will see in a real-world case in Section 7, the holding costs are relatively insignificant under all the heuristics considered (see Table 2). Therefore, we ignore the holding costs under the LP-Sample approximation. We set the sample size $N = 30$.

To evaluate the ordering and the storage decisions by each heuristic, we generate 1,000 new sample paths from a given demand distribution. For each period, we assume that demand fulfillment is done myopically. In particular, for each product i and period τ, given inventory levels

$x_{ij}^{\tau}, j \in \mathcal{J}$, and realized demands $d_{ik}^{\tau}, k \in \mathcal{K}$, we find a retrieval policy $y_{ijk}^{\tau}, j \in \mathcal{J}, k \in \mathcal{K}$, by solving the following linear program:

$$\begin{aligned} \max \quad & \sum_{j=1}^{J}\sum_{k=1}^{K}(p_i - r_{jk})\, y_{ijk}^{\tau} \\ s.t. \quad & \sum_{j=1}^{J} y_{ijk}^{\tau} \le d_{ik}^{\tau}, \quad k \in \mathcal{K}; \\ & \sum_{k=1}^{K} y_{ijk}^{\tau} \le x_{ij}^{\tau}, \quad j \in \mathcal{J}; \\ & y_{ijk}^{\tau} \ge 0, \qquad j \in \mathcal{J}, k \in \mathcal{K}. \end{aligned}$$

Since this linear program is solved independently for each product i and period τ, we can obtain the retrieval decisions efficiently even if I and T are moderately large. Unless specified otherwise, the profits in Sections 6–7 refer to the out-of-sample profits obtained by the above evaluation method.

6.2 Comparing the hybrid heuristic with the benchmarks

We first study two network settings: $J = 3, K = 5$ and $J = 4, K = 6$ for Problem (1). To scrutinize the impact of each parameter including the planning horizon's length, fulfillment frequency, warehouse capacity, demand variation, and demand correlation, we consider $I = 50$ products. Section 7 solves a real-world problem that has more than 10,000 products. Each period corresponds to a day. We consider two demand distributions for each $\tilde{d}_{ik}^{t}$, $i \in \mathcal{I}$, $k \in \mathcal{K}$, $t \in \mathcal{T}$: (i) a uniform distribution on $(0, a)$ with a uniformly chosen from $[0.5, 2]$ and (ii) an exponential distribution with mean μ uniformly chosen from $[0.5, 1.5]$. We set each warehouse's capacity as $c = IKT/(5J)$ for the uniform demand distribution and $c = 2IKT/(5J)$ for the exponential demand distribution. Note that the warehouse capacity grows linearly with T. The products' unit selling price is uniformly distributed in $[5, 15]$, their unit purchase cost is half of the price, and their annual holding cost per unit is 100% of their unit purchase cost such that $h_i = \rho_i/365$. The unit storage costs and unit retrieval costs are randomly selected from a uniform distribution on $[0, 1]$ and $[0, 5]$, respectively. Let *LP-Mean*, *LP-Sample*, and *Hybrid* denote the profit of the LP-Mean and LP-Sample approximations and hybrid heuristic, respectively. We evaluate the LP-Mean and LP-Sample approximations in terms of the relative performance *LP-Mean/Hybrid*×100% and *LP-Sample/Hybrid*×100%, respectively.

Impact of the planning horizon's length

We examine the impact of the planning horizon's length T. Figure 5 shows the performance of the two benchmarks relative to the hybrid heuristic with μ_T=150 and σ_T=40. Each point in the figure represents the average performance over five instances with different cost structures. For most of the instances, the hybrid heuristic outperforms the two benchmarks. This is because the hybrid heuristic captures demand fluctuation, which is missing from the LP-Mean approximation, and also captures the holding costs, avoiding the drawback of the LP-Sample approximation. As

T increases, *LP-Mean* gradually improves and approaches *Hybrid*. This is because the average demand for each product in each zone converges to the mean demand as the planning horizon gets longer. In contrast, *LP-Sample* relative to *Hybrid* tends to become worse when T gets large because the former ignores the holding costs, which are more significant when T is large. The gap between *Hybrid* and *LP-Mean* can be up to 6.5% when T is small, whereas the largest gap between *Hybrid* and *LP-Sample* can be 3.5% when T is large.

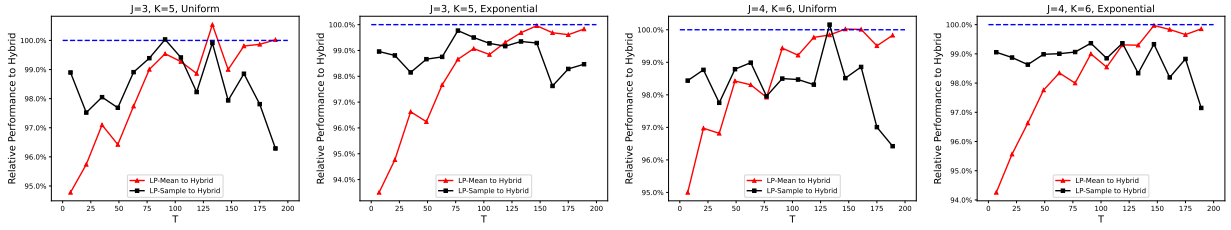


Figure 5: Relative performance under various planning-horizon lengths

Impact of fulfillment frequency

To investigate the impact of fulfillment (retrieval) frequency, we consider a problem of 64 days and divide the planning horizon into T periods, where $T \in \{2, 4, 8, 16, 32, 64\}$. Each period here corresponds to $64/T$ days and $h_i = \frac{\rho_i}{365} \times \frac{64}{T}$. We set μ_T=130 and σ_T=60 for the hybrid heuristic. Figure 6 suggests that each policy attains the highest profit when the fulfillment frequency is the lowest, and the profit drops as the frequency increases. This is because the retailer has more actual demand information to make retrieval decisions when the fulfillment frequency is lower (when each period is longer). Figure 7 shows the profit of the LP-Mean and LP-Sample approximations

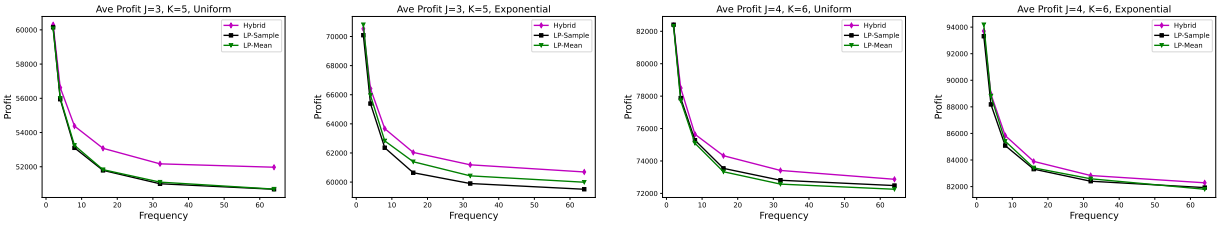


Figure 6: Performance of different policies under various fulfillment frequencies

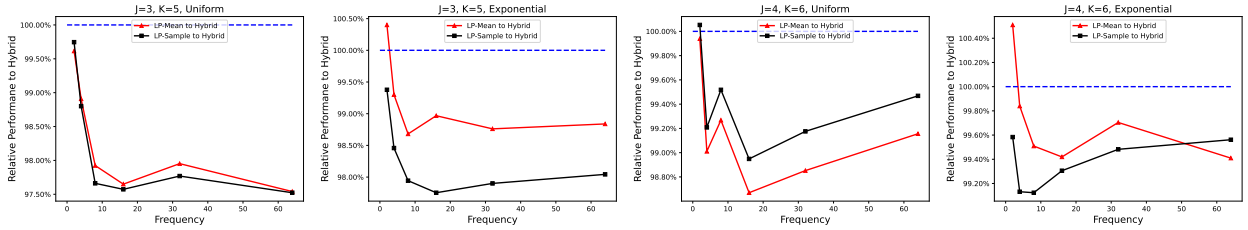


Figure 7: Relative performance under various fulfillment frequencies

relative to *Hybrid* as fulfillment becomes more frequent. The largest gap between *Hybrid* and the two benchmarks is 2.5% when the fulfillment frequency is high.

Impact of warehouse capacity

To focus on non-trivial situations, we set warehouse capacities and other parameters such that the capacity constraints are tight in a base problem instance. Figure 8 shows *LP-Mean* and *LP-Sample* relative to *Hybrid* for various warehouse capacity levels. Each warehouse's capacity is set as κ times of its original capacity in the base problem instance. We assume the demand $\tilde{d}_{ik}^t$ is uniformly distributed in an interval $[\hat{d}_{ik}^t - IL, \hat{d}_{ik}^t + IL]$, where the parameter IL is chosen such that the lower bound $\hat{d}_{ik}^t - IL \geq 0$. We set $IL = 0.3$, 0.5, and 0.7 from left to right in Figure 8, corresponding to low, medium, and high demand variation respectively. Figure 8 suggests that when the warehouse capacities are small, the two benchmarks are close to *Hybrid*. In contrast, as the warehouse capacities increase, the hybrid heuristic dominates the two benchmarks. The relative profit exhibits a U-shape curve with its lowest value occurs at some intermediate capacity level. This is because the relative advantage of using the hybrid heuristic is less significant when the warehouse capacity constraints are too tight or too loose. The largest gaps from *Hybrid* to *LP-Mean* and *LP-Sample* are 3.5% and 2%, respectively, when $\kappa = 1.25$.

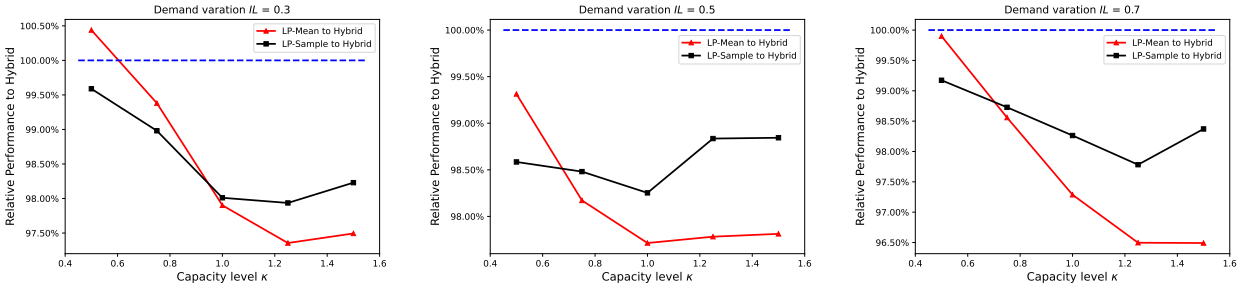


Figure 8: Relative performance for various warehouse capacity levels with J=3 and K=5

Impact of demand variation

We also assess the effect of demand variation on the policies' performance. Figure 9 suggests that when demand variation IL is small, *LP-Mean* and *LP-Sample* are close to *Hybrid*. This is because when demand variation is small, the actual demands are close to their mean value and the total holding cost is less significant. The former favors *LP-Mean* and the latter favors *LP-Sample*. However, as demand variation increases, the superiority of the hybrid heuristic becomes clear. The largest gaps from *Hybrid* to *LP-Mean* and *LP-Sample* are 3.5% and 2%, respectively, when demand variation is moderate. As demand variation further increases, these gaps tend to reduce as all the policies become less efficient. Since the LP-Mean approximation assumes deterministic demands, it is consistently worse than the other policies as demand variation increases.

Impact of demand correlation

To evaluate the impact of demand correlation, we assume demand is generated through an auto-regression process based on an $AR(p)$ model. For the first p periods, the demands are assumed to equal a random variable following a uniform or exponential distribution, plus a standard normal

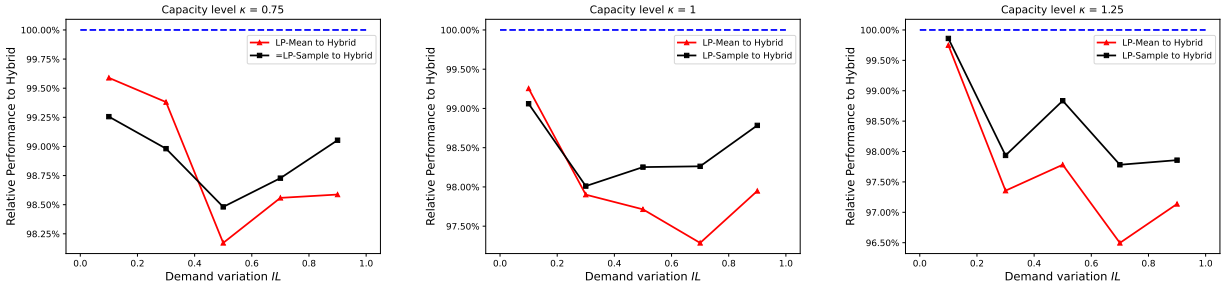


Figure 9: Relative performance for various demand variation levels with J=3 and K=5

noise. For $t > p$, the demand for each product i of zone k in period t is $\tilde{d}_{ik}^t = \sum_{\tau=1}^{p} \alpha_p[\tau]\ \tilde{d}_{ik}^{t-\tau} + \tilde{\epsilon}_{ik}^t$, where $\tilde{\epsilon}_{ik}^t$ is an independent and identically distributed standard normal random variable, and $\alpha_p[\tau]$ is a weighting vector defined as $\alpha_p[\tau] = (p - \tau + 1)/\sum_{\tau'=1}^{p} \tau'$, if $\tau \leq p$; and 0, if $\tau > p$. Clearly, $\sum_{\tau=1}^{p} \alpha_p[\tau] = 1$. For instance, if $p = 2$, $\alpha_2[1] = 2/3$ and $\alpha_2[2] = 1/3$. Any negative demand values are replaced by 0. Figure 10 shows the policies' relative profit for various p values. The hybrid heuristic consistently outperforms the two benchmarks partly because the positive demand correlation causes slower convergence of sample averages, lowering *LP-Mean* and *LP-Sample*. The largest gaps from *Hybrid* to *LP-Mean* and *LP-Sample* are 22.5% and 2.5% respectively. This highlights the hybrid heuristic's advantage in managing demand correlation across periods, which is partly due to the optimality of demand aggregation for the single-zone case shown in Lemma 1.

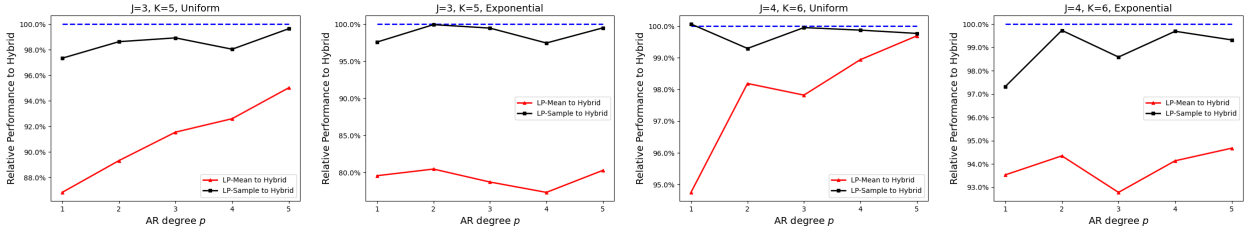


Figure 10: Relative performance under the effect of demand correlation

The superior performance of the hybrid heuristic underscores its capability to effectively deploy inventory for uncertain demand. These findings establish a foundation for the case study in Section 7, where we validate the hybrid heuristic's robustness in a real-world online retailing context and demonstrate its promising potential for practical, larger-scale networks.

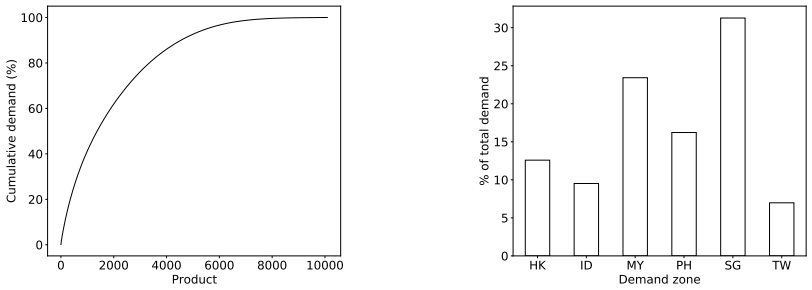
7 A case study using real data from an online retailer

We examine our hybrid heuristic using data from a major fashion (apparel) online retailer in Asia.

7.1 Data description

The retailer orders products from a single supplier in Guangzhou, China, and sells the products to six zones: Hong Kong (HK), Indonesia (ID), Malaysia (MY), the Philippines (PH), Singapore

(SG), and Taiwan (TW). Before the products are ordered by the customers, they are stored in three warehouses: one in Jakarta, ID; one in Kuala Lumpur, MY; and one in Manila, PH. Thus, we have J=3 and K=6. We have collected daily demand data for I=10,074 products, containing the actual demand of each zone for each product in each day for six months (Jan 1–Jun 30, 2017). Figure 11(a) shows the cumulative demand for the top i products, $i \in \mathcal{I}$. Figure 11(b) shows the total demand for all the products of each zone. The unit selling prices lie in $[6, 51]$. The unit purchase cost is half of the unit selling price for each product.



(a) (b)

Figure 11: Demand distributions across the products (a) and the zones (b)

The retailer replenishes the inventory at the start of the planning horizon and then fulfills the demands daily. To account for demand fluctuation and seasonality within each week, we pre-process the data to obtain a ground-truth demand distribution as follows. We first group all the days into seven clusters so that all Mondays are in a cluster, all Tuesdays are in a cluster, and so on. For each cluster, suppose the actual demand for product i of zone k in period (day) t is d. We assume demand $\tilde{d}_{ik}^t$ follows a uniform distribution in $[d, d+1]$ if $d > 0$, and $[0, 0.5]$ if $d = 0$. Specifically, for product i and zone k, suppose there are S actual demands in a cluster: $d_{ik}^{(s)}, s = 1, \ldots, S$. We assume the demand in each period t in this cluster is $\tilde{d}_{ik}^t = \tilde{d}_{ik}^s$ with probability $1/S$, where $\tilde{d}_{ik}^s$ follows $U[d_{ik}^{(s)}, d_{ik}^{(s)} + 1]$ if $d_{ik}^{(s)} > 0$, and $U[0, 0.5]$ if $d_{ik}^{(s)} = 0$. To determine the unit storage costs, we choose the cheapest rate for shipping a 1-Kg parcel from Guangzhou to each warehouse from the DHL, UPS, and Fedex websites[1]. After scaling the shipping rate with respect to the unit purchase costs of the products, we set the unit storage cost from the supplier to each warehouse at US$0.436. Likewise, we use the cheapest rate for shipping a 1-Kg parcel from each warehouse to each zone from the above websites to set the corresponding unit retrieval cost. Table 1 shows the unit retrieval costs after scaling the shipping rates with respect to the unit purchase costs.

Under the retailer's current policy, the Jakarta warehouse only serves ID, the Kuala Lumpur warehouse serves HK, MY, SG, and TW, and the Manila warehouse only serves PH. We deter-

[1]Sources: `https://www.dhl.com/us-en/home/get-a-quote/one-time-shipment-quote.html`, `https://www.ups.com/mobile/ratetnthome?loc=en_us`, `https://www.fedex.com/ratefinder/home?cc=US&language=en&locId=express`

Table 1: Unit retrieval cost from each warehouse to each demand zone (US$)

	Zones					
Warehouses	**HK**	**ID**	**MY**	**PH**	**SG**	**TW**
Jakarta	4.986	0.335	4.986	4.986	3.757	5.829
Kuala Lumpur	3.527	3.527	1.625	3.527	2.193	3.527
Manila	1.893	2.453	2.453	2.505	1.893	1.893

mine the order quantities for the Jakarta and Manila warehouses by solving the single-warehouse single-zone problem with Algorithm 6 in Appendix B, whereas the order quantities for the Kuala Lumpur warehouse can be found by solving the single-warehouse multi-zone problem (see Appendix C). Similar to Section 6, we generate 1,000 new sample paths from the ground-truth demand distributions and compute the total revenue, holding cost, and retrieval cost. We use the objective function value of the LP-Mean approximation as an upper bound on the profit.

7.2 Comparing the different policies

Table 2 shows the costs, revenue, profit, and computational time of the hybrid heuristic, LP-Mean approximation, LP-Sample approximation with 30 samples (in consistent with Section 6), and retailer's current dedicated policy. We consider planning horizons of one week, two weeks, and four weeks. The hybrid heuristic outperforms the LP-Sample approximation by up to 2%. This profit improvement is mainly due to the revenue generated. In terms of computational time, the LP-Sample approximation becomes unacceptable even for 30 samples, while the hybrid heuristic takes only 11 to 35 minutes. The hybrid heuristic generates more profit than the LP-Mean approximation by up to 5.7%, representing the value of handling random demand by anticipative ordering and storage decisions. Compared to the dedicated policy, the hybrid heuristic improves the profit by up to 16%, signifying the value of flexible fulfillment. Although there could be other vendor constraints or operational costs in practice that might reduce this potential profit improvement, the hybrid heuristic is still very promising given the notable profit gap.

Table 2: Performance of the different methods on the case study

One Week	**Hybrid**	**LP-Mean**	**LP-Sample**	**Dedicated**	**Upper bound**
Total purchase cost	725,368.99	737,926.03	715,674.42	635,366.66	
Total storage cost	30,745.85	30,745.85	30,745.85	30,745.85	
Total retrieval cost*	163,501.38	158,366.18	162,541.87	121,956.84	
Total holding cost*	4,822.81	5,298.36	4,523.00	4,312.88	
Revenue*	1,433,397.43	1,412,541.53	1,416,049.88	1,219,128.04	
Profit*	508,958.39	480,205.10	502,564.73	426,745.81	584,392.67
Profit relative to Hybrid	**100%**	**94.35%**	**98.74%**	**83.85%**	
Computational time (s)	682.88	10.82	44,055.71	121.67	
Time relative to Hybrid	100%	1.58%	6,451.45%	17.81%	
Two Weeks					
Total purchase cost	1,469,101.62	1,475,295.83	1,445,261.51	1,282,713.09	
Total storage cost	61,491.69	61,491.69	61,491.69	61,491.69	
Total retrieval cost*	336,397.73	328,095.02	335,699.73	248,315.27	
Total holding cost*	21,982.76	22,854.07	20,692.95	18,893.91	
Revenue*	2,914,986.86	2,862,513.43	2,869,433.82	2,501,766.52	
Profit*	1,026,013.05	974,776.82	1,006,287.92	890,352.56	1,154,980.44
Profit relative to Hybrid	**100%**	**95.01%**	**98.08%**	**86.78%**	
Computational time (s)	1,178.39	20.83	56,945.75	185.64	
Time relative to Hybrid	100%	1.76%	4,832.50%	15.75%	
Four Weeks					
Total purchase cost	2,936,064.64	2,948,354.21	2,909,749.13	2,573,341.27	
Total storage cost	122,983.39	122,983.39	122,983.39	122,983.39	
Total retrieval cost*	682,929.88	671,475.69	682,959.15	501,861.92	
Total holding cost*	90,189.72	94,590.28	88,952.68	77,445.65	
Revenue*	5,843,880.99	5,776,595.71	5,791,702.95	5,067,973.99	
Profit*	2,011,713.36	1,939,192.13	1,987,058.59	1,792,341.75	2,254,851.32
Profit relative to Hybrid	**100%**	**96.40%**	**98.77%**	**89.10%**	
Computational time (s)	2,123.77	194.33	46,783.57	310.86	
Time relative to Hybrid	100%	9.15%	2,202.85%	14.63%	

*Based on 1,000 demand sample paths.

7.3 Larger networks with various structures

We further investigate the hybrid heuristic's efficacy by considering larger networks of warehouses and zones in the Asia-Pacific region. The unit storage and retrieval costs are determined by the same method described in Section 7.1. We select a subset of 5,000 products from our data set to conduct experiments. For additional demand zones beyond the data set, we assume that these zones have demand means and variances similar to the existing zones in the data set. These experiments allow us to evaluate the scalability and robustness of the hybrid heuristic.

We compare the hybrid heuristic against the LP-Mean and LP-Sample approximations as well as a dedicated policy, under which warehouse 1, ..., $J-1$ serves only zone 1, ..., $J-1$ respectively, and warehouse J serves only the remaining $K-J+1$ zones. Likewise, we generate 1,000 new sample paths from the ground-truth demand distributions to evaluate the policies. Table 3 shows the experiment results based on a one-week planning horizon.

Table 3: Performance of the different methods on larger networks with 5,000 products

$J=4,\ K=7$	**Hybrid**	**LP-Mean**	**LP-Sample**	**Dedicated**	**Upper bound**
Profit*	244,199.52	228,124.09	240,959.37	187,020.57	296,055.09
Profit relative to Hybrid	**100%**	**93.42%**	**98.67%**	**76.59%**	
Computational time (s)	2,137.08	76.92	12,193.09	49.92	
Time relative to Hybrid	100%	3.60%	570.55%	2.34%	
$J=5,\ K=8$					
Profit*	284,598.26	263,513.76	279,280.62	114,491.47	351,327.71
Profit relative to Hybrid	**100%**	**92.59%**	**98.13%**	**40.23%**	
Computational time (s)	2,393.66	104.44	16,670.39	46.69	
Time relative to Hybrid	100%	4.36%	696.44%	1.95%	
$J=6,\ K=9$					
Profit*	342,425.97	320,155.02	335,397.47	111,451.73	418,911.93
Profit relative to Hybrid	**100%**	**93.50%**	**97.95%**	**32.55%**	
Computational time (s)	2,297.99	135.62	22,233.26	40.42	
Time relative to Hybrid	100%	5.90%	976.51%	1.76%	
$J=7,\ K=10$					
Profit*	398,579.11	377,773.14	392,727.68	98,932.82	483,413.33
Profit relative to Hybrid	**100%**	**94.78%**	**98.53%**	**24.82%**	
Computational time (s)	2,657.01	179.99	32,182.69	40.24	
Time relative to Hybrid	100%	6.77%	1,211.24%	1.51%	
$J=10,\ K=14$					
Profit*	378,215.85	356,992.58	368,077.51	87,713.82	440,760.75
Profit relative to Hybrid	**100%**	**94.39%**	**97.32%**	**23.19%**	
Computational time (s)	4,690.80	347.09	27,991.36	50.60	
Time relative to Hybrid	100%	7.40%	596.73%	1.08%	
$J=10,\ K=16$					
Profit*	470,377.39	453,887.26	464,095.06	92,244.02	567,151.64
Profit relative to Hybrid	**100%**	**96.49%**	**98.66%**	**19.61%**	
Computational time (s)	5,190.00	396.07	30,516.96	49.15	
Time relative to Hybrid	100%	7.63%	588.00%	0.95%	
$J=12,\ K=14$					
Profit*	400,502.77	393,409.59	399,099.76	145,367.11	475,359.83
Profit relative to Hybrid	**100%**	**98.23%**	**99.65%**	**36.30%**	
Computational time (s)	5,380.73	425.62	17,865.67	51.54	
Time relative to Hybrid	100%	7.91%	332.03%	0.96%	

*Based on 1,000 demand sample paths.

The results are generally consistent with that of the above case study with J=3 and K=6. The hybrid heuristic continues to deliver higher profitability than the LP-Mean and LP-Sample approximations for larger networks with $J \in [4, 12]$ and $K \in [7, 16]$, without compromising the computational efficiency. The hybrid heuristic outperforms the LP-Sample approximation by up to 2.68%. The latter spends a much longer time to find a solution than the hybrid heuristic, which takes 35 to 89 minutes. The LP-Sample approximation in turn outperforms the LP-Mean approximation. The dedicated policy is the least efficient with its profit ranging from 19.61% to 76.59% of *Hybrid*. By allowing flexible fulfillment in our model, which requires significantly more computational time in a multi-period setting, the hybrid heuristic is on average 63.81% more

profitable than the dedicated policy. Furthermore, the hybrid heuristic outperforms the LP-Mean approximation by up to 7.41%. This signifies the value of making anticipative ordering and storage decisions to deal with random demand, which substantially increases the problem complexity.

8 Conclusion

We consider an online retailer selling multiple products over a finite horizon with multiple periods. The retailer orders the products from a single supplier and stores them at multiple warehouses. At the start of the horizon, the retailer decides the products' order quantities and their storage quantities at each warehouse subject to its capacity constraint. At the end of each period, random product demands in the period are realized, the retailer decides the retrieval quantities from each warehouse to fulfill the demands. Any unmet demands in each period are lost. The objective is to maximize the retailer's expected profit over the horizon.

We first focus on a case where the retailer sells the products to a single demand zone. We prove that the multi-period problem is equivalent to a single-period problem for this case. We solve the single-period problem backward from the retrieval stage to the ordering stage. The optimal retrieval policy is greedy: Retrieve a product from a warehouse containing it with the smallest unit retrieval cost until all its demand is fulfilled or the system has run out of stock. We obtain a storage policy using non-greedy Algorithm 1, which allocates the products to the warehouses iteratively according to each warehouse's updated target stockout probability. Algorithm 1 produces an ϵ-optimal storage policy in polynomial time. Finally, we characterize the optimal ordering policy.

We have identified interesting insights for the single-zone problem. (i) The optimal storage policy can be characterized by a piecewise-linear non-increasing convex curve (Pareto frontier) on a two-dimensional space in Figure 3. Warehouses below the curve are full, above the curve are empty, and on the curve are partially filled. In each iteration, Algorithm 1 selects the warehouses on the curve for storage. (ii) Under the optimal storage and retrieval policies, each warehouse j has a target stockout probability χ_j that is identical for all the products. The target stockout probabilities allow us to separately determine each product's quantity allocated to each warehouse in each iteration of Algorithm 1, making its computational complexity linear in the number of products I (see Theorem 2). This is important as I is generally large for online retailing. (iii) The product assortment in the warehouses preserves a nested property: Among all the non-empty warehouses, a smaller-index warehouse contains all the products stored in a larger-index warehouse. (iv) A product's quantity allocated to a warehouse may decrease in the course of Algorithm 1 (see Example 3). This non-monotonicity property differentiates Algorithm 1 from the greedy algorithms by Lovász (1983) and Federgruen and Groenevelt (1986). (v) Algorithm 1 finds a significantly better solution in a much shorter time compared to SAA-LP and FISTA on the single-zone storage

problem (see Figure 14 in Appendix E). Algorithm 1 becomes more dominant as I increases, supporting our theoretical prediction in Theorem 2 that its complexity is linear in I.

We then consider a case with multiple demand zones, which is unfortunately intractable analytically because the optimal retrieval policy can no longer have a closed form. We propose an efficient hybrid heuristic and benchmark it against the LP-Mean and LP-Sample approximations. The hybrid heuristic consistently outperforms the benchmarks in our numerical experiments with various horizon lengths, fulfillment frequencies, warehouse capacities, demand variations, and demand correlations. The hybrid heuristic generates more profit than the LP-Mean and LP-Sample approximations by up to 22.5% and 3.5% respectively. A case study based on data from a major fashion online retailer in Asia confirms the hybrid heuristic's efficiency as it improves the average profit by up to 16% compared to the retailer's dedicated policy. Our hybrid heuristic is very promising given the notable profit gap. Furthermore, the hybrid heuristic continues to outperform the LP-Mean and LP-Sample approximations by up to 7.41% and 2.68%, respectively, without compromising the computational efficiency for larger networks with various structures.

Our single-supplier model can also be extended to a setting with multiple suppliers. For example, if all the suppliers have the same unit purchase cost for each product, then for each warehouse it is optimal to order all the products from the supplier with the smallest unit storage cost. Construct a virtual supplier by setting its unit storage cost to each warehouse equal to the smallest unit storage cost to the warehouse among all the suppliers. Under this setting, the multi-supplier problem is equivalent to a problem with a single, virtual supplier.

References

Acimovic J, Graves SC (2015) Making better fulfillment decisions on the fly in an online retail environment. *Manuf. Serv. Oper. Manag.* **17**(1): 34–51.

Acimovic J, Graves SC (2017) Mitigating spillover in online retailing via replenishment. *Manuf. Serv. Oper. Manag.* **19**(3): 419–436.

Ando K, Fujishige S, Naitoh T (1995) A greedy algorithm for minimizing a separable convex function over a finite jump system. *J. Oper. Res. Soc. Japan* **38**(3): 362–375.

Beck A, Teboulle M (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* **2**(1): 183–202.

Bertsekas DP (1999) *Nonlinear Programming*. Athena Scientific.

Bertsekas DP (2011) Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning* **3**: 1–38.

Boyd S, Vandenberghe L (2004) *Convex Optimization*, Cambridge University Press. Exercise 4.1, solution available at `http://see.stanford.edu/materials/lsocoee364b/hw4sol.pdf`. Accessed Dec 06, 2020.

Chambolle A, Dossal C (2015) On the convergence of the iterates of the fast iterative shrinkage/thresholding algorithm. *J. Optimization Theory and Applications* **166**(3): 968–982.

Eppen G, Schrage L (1981) Centralized ordering policies in a multi-warehouse system with lead times and random demand. *Multi-Level Production/Inventory Control Systems: Theory and Practice* **16**: 51–67.

Federgruen A, Groenevelt H (1986) The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality. *Oper. Res.* **34**(6): 909–918.

Federgruen A, Prastacos G, Zipkin PH (1986) An allocation and distribution model for perishable products. *Oper. Res.* **34**(1): 75–82.

Ferreira KJ, Lee BHA, Simchi-Levi D (2016) Analytics for an online retailer: Demand forecasting and price optimization. *Manuf. Serv. Oper. Manag.* **18**(1): 69–88.

Groenevelt H (1991) Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *Eur. J. Oper. Res.* **54**(25): 227–236.

Harsha P, Subramanian S, Uichanco J (2019) Dynamic pricing of omnichannel inventories. *Manuf. Serv. Oper. Manag.* **21**(1): 47–65.

Insider Intelligence (2023) Worldwide Ecommerce Forecast 2023. `https://www.insiderintelligence.com/content/worldwide-ecommerce-forecast-2023`. Accessed Jan 22, 2024.

Jasin S, Sinha A (2015) An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Oper. Res.* **63**(6): 1336–1351.

Lei Y, Jasin S, Sinha A (2018) Joint dynamic pricing and order fulfillment for e-commerce retailers. *Manuf. Serv. Oper. Manag.* **20**(2): 269–284.

Lim YF, Jiu S, Ang M (2020) Integrating anticipative replenishment allocation with reactive fulfillment for online retailing using robust optimization. *Manufacturing and Service Operations Management*, forthcoming, `https://doi.org/10.1287/msom.2020.0926`.

Lions PL, Mercier B (1979) Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis* **16**(6): 964–979.

Liu F (2017) A greedy algorithm for solving ordinary transportation problem with capacity constraints. *Operations Research Letters* **45**(4): 388–391.

Local Offer Network (2011) The daily deal phenomenon: A year in review. Report, Local Offer Network, Chicago.

Lovász L (1983) Submodular functions and convexity. In *Mathematical Programming: The State of the Art*, ed. Bachem et al., Springer Berlin Heidelberg, 235–257.

Miao S, Jasin S, and Chao X. (2022) Asymptotically optimal Lagrangian policies for multi-warehouse, multi-store systems with lost sales. *Operations Research* **70**(1), 141–159.

Malozemov VN, Tamasyan GS (2016) Two Fast Algorithms for Projecting a Point onto the Canonical Simplex. *Computational Mathematics and Mathematical Physics* **56**(5): 730–743.

Nesterov Y (2004) *Introductory Lectures on Convex Optimization: A Basic Course*, Vol. 87. Springer.

Ostapenko N (2013) Online discount luxury: In search of guilty customers. *Internat. J. Bus. Soc. Res.* **3**(2): 60–68.

Robbins H, Monro S (1951) A stochastic approximation method. *Annals of Mathematical Statistics* **22**: 400–407.

Kharpal A (2021) *Alibaba, JD smash Singles Day record with $139 billion of sales and focus on 'social responsibility'.* `https://www.cnbc.com/2021/11/12/china-singles-day-2021-alibaba-jd-hit-record-139-billion-of-sales.html`. Accessed Jan 22, 2024.

Shapiro A, Philpott A (2007) A tutorial on stochastic programming. `https://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/TutorialSP.pdf`. Accessed March 08, 2019.

Vidal T, Gribel D, Jaillet P (2019) Separable convex optimization with nested lower and upper constraints. *INFORMS Journal on Optimization* **1**(1): 71–90.

Vidal T, Jaillet P, Maculan N (2014) A decomposition algorithm for nested resource allocation problems. *Working Paper*, Massachusetts Institute of Technology.

Wolverson R (2012) High and low: Online flash sales go beyond fashion to survive. *Time Magazine* **180**(19): 9–12.

Wu Z, Nip K, He Q (2021) A new combinatorial algorithm for separable convex resource allocation with nested bound constraints. *INFORMS Journal on Computing*, forthcoming.

Xu PJ, Allgor R, Graves SC (2009) Benefits of reevaluating real-time order fulfillment decisions. *Manuf. Serv. Oper. Manag.* **11**(2): 340–355.

Zhong Y, Zheng Z, Chou MC, Teo C-P (2018) Resource pooling and allocation policies to deliver differentiated service. *Management Sci.* **64**(4): 1555–1573.

Zipkin PH (1980) Simple ranking methods for allocation of one resource. *Management Sci.* **26**(1): 34–43.

A Proofs

A.1 Proof of Lemma 1

Part 1. We first reformulate Problem (9) as a dynamic program. Denote $V^t\left(\boldsymbol{x}^t\right)$ as the profit-to-go function at the beginning of period t. Let $V^{T+1}=0$. The profit-to-go functions satisfy the following Bellman equation:

$$V^t\left(\boldsymbol{x}^t\right)=E_{\tilde{\boldsymbol{d}}^t}\max_{\substack{\mathbf{0}\leq \boldsymbol{y}^t\leq \boldsymbol{x}^t\\ \boldsymbol{y}^t\boldsymbol{e}\leq \tilde{\boldsymbol{d}}^t}}\left\{-\sum_{i=1}^{I}h_i\sum_{j=1}^{J}(x_{ij}^t-y_{ij}^t)+\sum_{i=1}^{I}\sum_{j=1}^{J}\left(p_i-r_j\right)y_{ij}^t+V^{t+1}\left(\boldsymbol{x}^t-\boldsymbol{y}^t\right)\right\} \tag{31}$$

Next, we show that

$$V^t(\boldsymbol{x}^t)=-(T-t+1)\sum_{i=1}^{I}h_i\sum_{j=1}^{J}x_{ij}^t+\sum_{i=1}^{I}E_{\tilde{\boldsymbol{d}}^t,...,\tilde{\boldsymbol{d}}^T}$$
$$\left[h_i\sum_{\tau=t}^{T}\min\left(\sum_{j=1}^{J}x_{ij}^t,\sum_{k=t}^{\tau}\tilde{d}_i^k\right)+\left(p_i-r_J\right)\min\left(\sum_{j=1}^{J}x_{ij}^t,\sum_{\tau=t}^{T}\tilde{d}_{i\tau}\right)+\sum_{j=2}^{J}\psi_j\min\left(\sum_{l=1}^{j-1}x_{il}^t,\sum_{\tau=t}^{T}\tilde{d}_{i\tau}\right)\right]$$

and

$$y_{ij}^{t*}=\min\left(\sum_{l=1}^{j}x_{il}^t,d_i^t\right)-\min\left(\sum_{l=1}^{j-1}x_{il}^t,d_i^t\right),\quad i=1,...,I,\ j=1,...,J,$$

for all $t=1,...,T$ by induction.

In period T,

$$V^T\left(\boldsymbol{x}^T\right)=E_{\tilde{\boldsymbol{d}}^T}\max_{\substack{\mathbf{0}\leq \boldsymbol{y}^T\leq \boldsymbol{x}^T\\ \boldsymbol{y}^T\boldsymbol{e}\leq \tilde{\boldsymbol{d}}^T}}\left\{-\sum_{i=1}^{I}h_i\sum_{j=1}^{J}(x_{ij}^T-y_{ij}^T)+\sum_{i=1}^{I}\sum_{j=1}^{J}\left(p_i-r_j\right)y_{ij}^T\right\}.$$

The optimal retrieval policy $\boldsymbol{y}^{T*}$ is to retrieve each product i from a warehouse with the smallest index that contains the inventory of a product until all units are retrieved or all demands are fulfilled. Therefore, for a given product i, we will retrieve the product from warehouse j if and only if $d_i^T>\sum_{l=1}^{j-1}x_{il}^T$. The total quantity of product i retrieved from warehouses 1 to j is $\min\left(\sum_{l=1}^{j}x_{il}^T,d_i^T\right)$. Thus, the optimal quantity of product i retrieved from warehouse j is

$$y_{ij}^{T*}=\min\left(\sum_{l=1}^{j}x_{il}^T,d_i^T\right)-\min\left(\sum_{l=1}^{j-1}x_{il}^T,d_i^T\right),$$

for $i=1,\ldots,I$ and $j=1,\ldots,J$. Based on this result, we can write

$$V^T\left(\boldsymbol{x}^T\right)\quad=\quad-\sum_{i=1}^{I}h_i\sum_{j=1}^{J}x_{ij}^T+\sum_{i=1}^{I}E_{\tilde{\boldsymbol{d}}^T}\left[\left(p_i+h_i-r_J\right)\min\left(\sum_{l=1}^{J}x_{il}^T,\tilde{d}_i^T\right)+\sum_{j=2}^{J}\psi_j\min\left(\sum_{l=1}^{j-1}x_{il}^T,\tilde{d}_i^T\right)\right].$$

Thus, the results hold for T.

Suppose the results hold for period $t+1$. In period t, by induction, the Bellman equation can be simplified to

$$\begin{aligned}V^t(\boldsymbol{x}^t)\quad&=\quad E_{\tilde{\boldsymbol{d}}^t}\max_{\substack{\mathbf{0}\leq \boldsymbol{y}^t\leq \boldsymbol{x}^t\\ \boldsymbol{y}^t\boldsymbol{e}\leq \tilde{\boldsymbol{d}}^t}}\left\{-\sum_{i=1}^{I}h_i\sum_{j=1}^{J}(x_{ij}^t-y_{ij}^t)+\sum_{i=1}^{I}\sum_{j=1}^{J}\left(p_i-r_j\right)y_{ij}^t+V^{t+1}\left(\boldsymbol{x}^t-\boldsymbol{y}^t\right)\right\}\\
&=\quad-(T-t+1)\sum_{i=1}^{I}h_i\sum_{j=1}^{J}x_{ij}^t+E_{\tilde{\boldsymbol{d}}^t}\max_{\substack{\mathbf{0}\leq \boldsymbol{y}^t\leq \boldsymbol{x}^t\\ \boldsymbol{y}^t\boldsymbol{e}\leq \tilde{\boldsymbol{d}}^t}}\left\{(T-t+1)\sum_{i=1}^{I}h_i\sum_{j=1}^{J}y_{ij}^t+\sum_{i=1}^{I}\sum_{j=1}^{J}\left(p_i-r_j\right)y_{ij}^t\right.\\
&\quad+\sum_{i=1}^{I}E_{\tilde{\boldsymbol{d}}^{t+1},...,\tilde{\boldsymbol{d}}^T}\left[h_i\sum_{\tau=t+1}^{T}\min\left(\sum_{j=1}^{J}\left(x_{ij}^t-y_{ij}^t\right),\sum_{k=t+1}^{\tau}\tilde{d}_i^k\right)\right.\\
&\quad\left.\left.+\left(p_i-r_J\right)\min\left(\sum_{j=1}^{J}\left(x_{ij}^t-y_{ij}^t\right),\sum_{\tau=t+1}^{T}\tilde{d}_{i\tau}\right)+\sum_{j=2}^{J}\psi_j\min\left(\sum_{l=1}^{j-1}\left(x_{il}^t-y_{il}^t\right),\sum_{\tau=t+1}^{T}\tilde{d}_{i\tau}\right)\right]\right\}.\end{aligned}$$

Consider the maximization problem of $V^t(\boldsymbol{x}^t)$, we obtain the derivative of y_{ij}^t over the objective function as $(T-t+1)h_i + (p_i - r_j) - h_i \sum_{\tau=t+1}^{T} Pr\left(\sum_{j=1}^{J}\left(x_{ij}^t - y_{ij}^t\right) \le \sum_{k=t+1}^{\tau} \tilde{d}_i^k\right) - (p_i - r_J)Pr\left(\sum_{j=1}^{J}\left(x_{ij}^t - y_{ij}^t\right) \le \sum_{\tau=t+1}^{T} \tilde{d}_i^k\right) - \sum_{j=2}^{J} \psi_j Pr\left(\sum_{l=1}^{j-1}\left(x_{il}^t - y_{il}^t\right) \le \sum_{\tau=t+1}^{T} \tilde{d}_i^k\right)$. We also obtain the difference between the derivatives of y_{ij-1}^t and y_{ij}^t as
$(r_j - r_{j-1})\left(1 - Pr\left(\sum_{l=1}^{j-1}\left(x_{il}^t - y_{il}^t\right) \le \sum_{\tau=t+1}^{T} \tilde{d}_i^k\right)\right) \ge 0$.Thus, for any storage matrix, retrieving a unit of product i from warehouse $j-1$ is more profitable than retrieving from warehouse j. This hence implies that the optimal retrieval policy for each product i is to retrieve from the warehouse with a smaller index that contains the product.

Therefore, for a given product i, we will retrieve the product from warehouse j if and only if $d_i^t > \sum_{l=1}^{j-1} x_{il}^t$. The total quantity of product i retrieved from warehouses 1 to j is $\min\left(\sum_{l=1}^{j} x_{il}^t, d_i^t\right)$. Thus, the optimal quantity of product i retrieved from warehouse j is

$$y_{ij}^{t*} = \min\left(\sum_{l=1}^{j} x_{il}^t, d_i^t\right) - \min\left(\sum_{l=1}^{j-1} x_{il}^t, d_i^t\right),$$

for $i = 1, \ldots, I$ and $j = 1, \ldots, J$. We substitute y_{ij}^{t*} into $V^t(\boldsymbol{x}^t)$. Note that

$$\begin{aligned}
&\min\left(\sum_{j=1}^{J}\left(x_{ij}^t - y_{ij}^{t*}\right), \sum_{k=t+1}^{\tau} d_i^k\right) = \min\left(\sum_{j=1}^{J} x_{ij}^t - \min\left(\sum_{l=1}^{J} x_{il}^t, d_i^t\right), \sum_{k=t+1}^{\tau} d_i^k\right) \\
= \;& \min\left(\sum_{j=1}^{J} x_{ij}^t, \sum_{k=t+1}^{\tau} d_i^k + \min\left(\sum_{l=1}^{J} x_{il}^t, d_i^t\right)\right) - \min\left(\sum_{l=1}^{J} x_{il}^t, d_i^t\right) = \min\left(\sum_{j=1}^{J} x_{ij}^t, \sum_{k=t}^{\tau} d_i^k\right) - \min\left(\sum_{l=1}^{J} x_{il}^t, d_i^t\right).
\end{aligned}$$

$$\begin{aligned}
V^t(\boldsymbol{x}^t) \quad &= -(T-t+1)\sum_{i=1}^{I} h_i \sum_{j=1}^{J} x_{ij}^t + \sum_{i=1}^{I} E_{\tilde{\boldsymbol{d}}^t,\ldots,\tilde{\boldsymbol{d}}^T} \\
&\left[h_i \sum_{\tau=t}^{T} \min\left(\sum_{j=1}^{J} x_{ij}^t, \sum_{k=t}^{\tau} \tilde{d}_i^k\right) + (p_i - r_J)\min\left(\sum_{j=1}^{J} x_{ij}^t, \sum_{\tau=t}^{T} \tilde{d}_{i\tau}\right) + \sum_{j=2}^{J} \psi_j \min\left(\sum_{l=1}^{j-1} x_{il}^t, \sum_{\tau=t}^{T} \tilde{d}_{i\tau}\right)\right].
\end{aligned}$$

This hence shows that the results hold for all t.

Part 2. From Part 1, the retailer's expected profit in period 1 under the optimal retrieval policy is given by

$$\begin{aligned}
V^1(\boldsymbol{x}) \quad &= \quad -T\sum_{i=1}^{I} h_i \sum_{j=1}^{J} x_{ij} + \sum_{i=1}^{I} E_{\tilde{\boldsymbol{d}}} \\
&\left[h_i \sum_{t=1}^{T} \min\left(\sum_{j=1}^{J} x_{ij}, \sum_{\tau=1}^{t} \tilde{d}_{i\tau}\right) + (p_i - r_J)\min\left(\sum_{j=1}^{J} x_{ij}, \sum_{t=1}^{T} \tilde{d}_i^t\right) + \sum_{j=2}^{J} \psi_j \min\left(\sum_{l=1}^{j-1} x_{il}, \sum_{t=1}^{T} \tilde{d}_i^t\right)\right] \\
&= \quad -\sum_{t=1}^{T}\sum_{i=1}^{I} h_i \left[\sum_{\ell=1}^{J} x_{i\ell} - G_i^t\left(\sum_{\ell=1}^{J} x_{i\ell}\right)\right] + \sum_{i=1}^{I}(p_i - r_J)G_i^T\left(\sum_{\ell=1}^{J} x_{i\ell}\right) + \sum_{i=1}^{I}\sum_{j=2}^{J} \psi_j G_i^T\left(\sum_{\ell=1}^{j-1} x_{i\ell}\right).
\end{aligned}$$

Adding back the purchase and storage costs, we obtain the expression for $u(\boldsymbol{x})$. ■

A.2 Proof of Lemma 2

Since $p_i - r_J > 0$ and $G_i^t(x)$ is a concave function of x for all $i \in \mathcal{I}, t \in \mathcal{T}$, it is clear that $u(\boldsymbol{x})$ is also a concave function. The objective function $u(\boldsymbol{x})$ is separable in i: $u(\boldsymbol{x}) = \sum_{i=1}^{I} u_i(\boldsymbol{x}_i)$, where $u_i(\boldsymbol{x}_i) = -\sum_{j=1}^{J}(\rho_i + s_j)x_{ij} - h_i \sum_{t=1}^{T}\left[q_i - G_i^t(q_i)\right] + (p_i - r_J)G_i^T(q_i) + \sum_{j=2}^{J} \psi_j G_i^T\left(\sum_{\ell=1}^{j-1} x_{i\ell}\right)$. Given that $d\bar{F}_i^t(x)/dx = -f_i^t(x)$, $i \in \mathcal{I}, t \in \mathcal{T}$, the first- and second-order partial derivatives of $u(\boldsymbol{x})$ are

$$\begin{aligned}
\frac{\partial u_i}{\partial x_{ij}} \quad &= \quad -(\rho_i + h_i T + s_j) + (p_i - r_J)\bar{F}_i^T(q_i) + h_i \sum_{t=1}^{T} \bar{F}_i^t(q_i) + \sum_{\upsilon=j+1}^{J} \psi_\upsilon \bar{F}_i^T\left(\sum_{\ell=1}^{\upsilon-1} x_{i\ell}\right), \\
\frac{\partial^2 u_i}{\partial x_{ij}\partial x_{i\ell}} \quad &= \quad -(p_i - r_J)f_i^T(q_i) - h_i \sum_{t=1}^{T} f_i^t(q_i) - \sum_{\upsilon=\max(j+1,\ell+1)}^{J} \psi_\upsilon f_i^T\left(\sum_{\ell=1}^{\upsilon-1} x_{i\ell}\right). \qquad (32)
\end{aligned}$$

It is well known that $\lambda_{\max}(A) \leq n \max_{i,j} |A_{i,j}|$ for A being $n \times n$ symmetric matrix. Therefore,

$$\lambda_{\max}\left(\nabla^2 u_i(\boldsymbol{x}_i)\right) \leq J \max_{j,l}\left((p_i - r_J)f_{\max} + h_i T f_{\max} + \sum_{\upsilon=\max(j+1,\ell+1)}^{J} \psi_\upsilon f_{\max}\right) = J(p_i + h_i T - r_1)f_{\max}.$$

Now we observe that $u(\boldsymbol{x}) = \sum_{i=1}^{I} u_i(\boldsymbol{x}_i)$ is separable. Therefore,

$$\lambda_{\max}\left(\nabla^2 u(\boldsymbol{x})\right) \leq \max_i \lambda_{\max}\left(\nabla^2 u_i(\boldsymbol{x}_i)\right) \leq J(p_{\max} + h_{\max} T - r_1)f_{\max},$$

where $p_{\max} = \max_i p_i$ and $h_{\max} = \max_i h_i$.

From the expression of the second order partial derivative, we can easily verify that

$$-\nabla^2 u_i(\boldsymbol{x}_i) = (p_i - r_J)\,\bar{f}_i^T(q_i)\,\boldsymbol{e}\boldsymbol{e}' + h_i \sum_{t=1}^{T} f_i^t(q_i)\,\boldsymbol{e}\boldsymbol{e}' + \sum_{\upsilon=2}^{J} \psi_\upsilon f_i^T\left(q_i^{(\upsilon-1)}\right)\boldsymbol{e}_{\upsilon-1}\boldsymbol{e}'_{\upsilon-1},$$

where $\boldsymbol{e}_k = (1, ..., 1, 0, ..., 0)$ is an J-dimensional vector with the first k components equal one and other components equal zero. For any vector $\boldsymbol{z}$, we have

$$-\boldsymbol{z}^T \nabla^2 u_i(\boldsymbol{x}_i)\boldsymbol{z} \quad \geq \quad \min\left((p_i - r_J)\,f_i^T(q_i) + h_i \sum_{t=1}^{T} f_i^t(q_i)\,, \min_{\upsilon=1,...,J-1} \psi_{\upsilon+1} f_i^T\left(q_i^{(\upsilon)}\right)\right) \sum_{\upsilon=1}^{J} (\boldsymbol{e}_\upsilon^T \boldsymbol{z})^2.$$

Note that

$$\|\boldsymbol{z}\|^2 = \sum_{j=1}^{J} z_j^2 = \sum_{j=1}^{J} (\boldsymbol{e}_j^T\boldsymbol{z} - \boldsymbol{e}_{j-1}^T\boldsymbol{z})^2 \leq \sum_{j=1}^{J} (\boldsymbol{e}_j^T\boldsymbol{z} - \boldsymbol{e}_{j-1}^T\boldsymbol{z})^2 \leq \sum_{j=1}^{J} (\boldsymbol{e}_j^T\boldsymbol{z})^2 + (\boldsymbol{e}_{j-1}^T\boldsymbol{z})^2 \leq 2\sum_{j=1}^{J} (\boldsymbol{e}_j^T\boldsymbol{z})^2.$$

Combining the above two inequalities, together with the fact that $0 \leq q_i^{(\upsilon)} \leq \sum_{j=1}^{J} c_j$ for all $\upsilon = 1, ..., J$, we have

$$-\boldsymbol{z}^T \nabla^2 u_i(\boldsymbol{x}_i)\boldsymbol{z} \geq \alpha_i \|\boldsymbol{z}\|^2,$$

where

$$\alpha_i = \frac{1}{2}\underline{\mathrm{w}} \min\left(p_i + h_i T - r_J, \min_{\upsilon=1,...,J-1} \psi_{\upsilon+1}\right).$$

From the problem assumption, it is clear that $\alpha_i > 0$. This suggests that $u(\boldsymbol{x}_i)$ is an α-strongly concave function where $\alpha = \sum_{i\in\mathcal{I}} \alpha_i$. ■

A.3 Proof of Theorem 1

The storage problem (19) is a concave optimization problem with linear constraints. The Lagrangian of the storage problem is

$$\Lambda(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) \quad = \quad -\sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij} + \sum_{i=1}^{I}\sum_{j=2}^{J} \psi_j G_i\left(\sum_{l=1}^{j-1} x_{il}\right) + \boldsymbol{\lambda} \odot \boldsymbol{x} - \boldsymbol{\mu} \cdot (\boldsymbol{x}\boldsymbol{e} - \boldsymbol{q}) - \boldsymbol{\nu} \cdot (\boldsymbol{x}'\boldsymbol{e} - \boldsymbol{c}), \tag{33}$$

where $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ are Lagrangian multipliers. Specifically, $\boldsymbol{\lambda} \geq \boldsymbol{0}$ is an $I \times J$ matrix, $\boldsymbol{\mu}$ is an I-dimensional column vector, $\boldsymbol{\nu} \geq \boldsymbol{0}$ is a J-dimensional column vector, and $\odot$ is the sum of component-wise products of the matrices. The following lemma identifies conditions for $\boldsymbol{x}^*$.

Lemma 4 *The optimal storage policy $\boldsymbol{x}^*$ satisfies the following KKT conditions:*

$$\psi_{j+1}\bar{F}_i\left(\sum_{k=1}^{j} x_{ik}^*\right) \quad = \quad s_j - s_{j+1} - \lambda_{ij}^* + \lambda_{i,j+1}^* + \nu_j^* - \nu_{j+1}^*, \quad j \in \mathcal{J}^-, i \in \mathcal{I}; \tag{34}$$

$$\mu_i^* - \lambda_{iJ}^* \quad = \quad -\nu_J^* - s_J, i \in \mathcal{I}; \tag{35}$$

$$\boldsymbol{\nu}, \boldsymbol{\lambda} \geq \boldsymbol{0}. \tag{36}$$

Proof : The first-order derivative of the Lagrangian in Equation (33) is $\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu})}{\partial x_{ij}} = -s_j + \sum_{k=j}^{J-1} \bar{F}_i\left(\sum_{u=1}^{k} x_{iu}\right)\psi_{k+1} + \lambda_{ij} - \mu_i - \nu_j$, for all $j \in \mathcal{J}^-, i \in \mathcal{I}$. The optimal solution satisfies the first-order condition $\partial\Lambda\left(\boldsymbol{x}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*,\boldsymbol{\nu}^*\right)/\partial x_{ij} = 0$. Take the difference between $\partial\Lambda\left(\boldsymbol{x}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*,\boldsymbol{\nu}^*\right)/\partial x_{ij} = 0$ and $\partial\Lambda\left(\boldsymbol{x}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*,\boldsymbol{\nu}^*\right)/\partial x_{i,j+1} = 0$ for all $j \in \mathcal{J}^-, i \in \mathcal{I}$, we have $\psi_{j+1}\bar{F}_i\left(\sum_{u=1}^{j} x^*_{iu}\right) = s_j - s_{j+1} - \lambda^*_{ij} + \lambda^*_{i,j+1} + \nu^*_j - \nu^*_{j+1}$. If $j = J$, the first-order condition $\partial\Lambda\left(\boldsymbol{x}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*,\boldsymbol{\nu}^*\right)/\partial x_{iJ} = 0$ implies $s_J - \lambda^*_{iJ} + \mu^*_i + \nu^*_J = 0$. ■

We are now ready to prove Theorem 1.

The left-hand side of Equation (34) is always non-negative. This implies that under the optimal storage policy, we have $s_j - s_{j+1} - \lambda^*_{ij} + \lambda^*_{i,j+1} + \nu^*_j - \nu^*_{j+1} \geq 0$, for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$. If $s_j < s_{j+1}$, we have $-\lambda^*_{ij} + \lambda^*_{i,j+1} > 0$ for all $i \in \mathcal{I}$, or $\nu^*_j - \nu^*_{j+1} > 0$. In the first case, $-\lambda^*_{ij} + \lambda^*_{i,j+1} > 0$ implies $x^*_{i,j+1} = 0$ for all $i \in \mathcal{I}$. In the second case, $\nu^*_j - \nu^*_{j+1} > 0$ implies $\sum_{i=1}^{I} x^*_{i,j} = c_j$. Combining the two cases, $s_j < s_{j+1}$ implies that the optimal storage policy is not to store any unit in warehouse $j+1$ ($x^*_{i,j+1} = 0$ for all $i \in \mathcal{I}$) or to have warehouse j full ($\sum_{i=1}^{I} x^*_{i,j} = c_j$). In other words, if $s_j < s_{j+1}$, it is always optimal to fill up warehouse j before filling warehouse $j+1$.

Summing Equation (34) from warehouse j to warehouse $j'-1$, we have $\sum_{h=j}^{j'-1} \psi_{h+1}\bar{F}_i\left(\sum_{u=1}^{h} x^*_{iu}\right) = s_j - s_{j'} - \lambda^*_{ij} + \lambda^*_{ij'} + \nu^*_j - \nu^*_{j'}$. For any non-full warehouses $j < j'$, we have $\nu^*_j = \nu^*_{j'} = 0$, and hence $\sum_{h=j}^{j'-1} \psi_{h+1}\bar{F}_i\left(\sum_{u=1}^{h} x^*_{iu}\right) = s_j - s_{j'} - \lambda^*_{ij} + \lambda^*_{ij'}$. Note that the left-hand side is non-negative. Thus, if $s_j < s_{j'}$, we have $\lambda^*_{ij'} > 0$, which implies $x^*_{ij'} = 0$. In other words, if $s_j < s_{j'}$, it is always optimal to fill up warehouse j before filling warehouse j'. ■

A.4 Nested property

Lemma 5 *If the initial storage matrix $\boldsymbol{v}$ is structured, then it has the nested property.*

Proof : We prove this by contradiction. Suppose for warehouse j, we have $v_{ij} = 0$ and $v_{i'j} > 0$ for products and i and i', and suppose $v_{ij'} > 0$, where j' is the smallest index larger than j such that warehouse j' is non-empty. Clearly, the nested property does not hold. From part 4 of Definition 4, we know that $j < j' \leq j_c(i)$. Thus, we have $\bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) = \chi_j(\boldsymbol{v})$. Also, since $v_{i'j} > 0$, from part 4 of Definition 4, we know that $j \leq j_c(i')$. Thus, we have $\bar{F}_{i'}\left(\sum_{\ell=1}^{j} v_{i'\ell}\right) \geq \chi_j(\boldsymbol{v})$.

Since $j-1$ is smaller than both $j_c(i)$ and $j_c(i')$, according to part 4 of Definition 4, we have $\bar{F}_i\left(\sum_{\ell=1}^{j-1} v_{i\ell}\right) = \bar{F}_{i'}\left(\sum_{\ell=1}^{j-1} v_{i'\ell}\right)$. Since $v_{ij} = 0$, we have $\bar{F}_i\left(\sum_{\ell=1}^{j-1} v_{i\ell}\right) = \bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) = \chi_j(\boldsymbol{v})$. Furthermore, since $v_{i'j} > 0$, we have $\bar{F}_{i'}\left(\sum_{\ell=1}^{j-1} v_{i'\ell}\right) > \bar{F}_{i'}\left(\sum_{\ell=1}^{j} v_{i'\ell}\right) \geq \chi_j(\boldsymbol{v})$. Combining the three results above, we have $\chi_j(\boldsymbol{v}) = \bar{F}_i\left(\sum_{\ell=1}^{j} v_{i\ell}\right) = \bar{F}_i\left(\sum_{\ell=1}^{j-1} v_{i\ell}\right) = \bar{F}_{i'}\left(\sum_{\ell=1}^{j-1} v_{i'\ell}\right) > \bar{F}_{i'}\left(\sum_{\ell=1}^{j} v_{i'\ell}\right) \geq \chi_j(\boldsymbol{v})$, which leads to a contradiction. This proves the lemma. ■

A.5 The marginal costs

Lemma 6 *For $j < J$,*

$$D_{ij}(\boldsymbol{v}) = \left(s_j + r_j\bar{F}_i\left(\sum_{k=1}^{j} v_{ik}\right)\right) - \sum_{u=j+1}^{J-1} r_u\left[\bar{F}_i\left(\sum_{k=1}^{u-1} v_{ik}\right) - \bar{F}_i\left(\sum_{k=1}^{u} v_{ik}\right)\right] - r_J\bar{F}_i\left(\sum_{k=1}^{J-1} v_{ik}\right); \tag{37}$$

and $D_{iJ}(\boldsymbol{v}) = s_J$.

Proof : Recall that $G(\boldsymbol{x}) = -\sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij} + \sum_{i=1}^{I}\sum_{j=2}^{J} \psi_j G_i\left(\sum_{l=1}^{j-1} x_{il}\right)$. If $j = J$ the result is obvious. For $j < J$, we have

$$D_{ij}(\boldsymbol{v}) = -\left.\frac{\partial G(\boldsymbol{x})}{\partial x_{ij}}\right|_{\boldsymbol{x}=\boldsymbol{v}} = s_j - \sum_{u=j+1}^{J} \bar{F}_i\left(\sum_{k=1}^{u-1} v_{ik}\right)\psi_u.$$

Since $\boldsymbol{v}$ represents the inventory levels in an iteration of Algorithm 1, we have $q_i \geq \sum_{k=1}^{u-1} v_{ik}$ for all $u = j+1, \cdots, J$. The above equation can be simplified as

$$D_{ij}(\boldsymbol{v}) = \left(s_j + r_j \bar{F}_i\left(\sum_{k=1}^{j} v_{ik}\right)\right) - \sum_{u=j+1}^{J-1} r_u \left[\bar{F}_i\left(\sum_{k=1}^{u-1} v_{ik}\right) - \bar{F}_i\left(\sum_{k=1}^{u} v_{ik}\right)\right] - r_J \bar{F}_i\left(\sum_{k=1}^{J-1} v_{ik}\right).$$

This completes the proof. ■

A.6 Proof of Lemma 3

If $\boldsymbol{v}$ is structured, we only need to show that $\boldsymbol{z}^*$ is also structured. Parts 2 to 4 of the structured property in Definition 4 can be verified directly from the construction of $\boldsymbol{z}^*$ through Algorithm 3. Thus, here we shall only verify part 1 and 5.

We verify part 5 first. Initially, in the first iteration, $\boldsymbol{v} = \boldsymbol{0}$ and part 5 of the structured property holds. Suppose in this iteration, it is the first iteration where in Algorithm 1 a non-empty warehouse $\tilde{j} > j_v^*$ exists and $\sum_{i=1}^{I} v_{i\tilde{j}} < c_{\tilde{j}}$.

For any i, compare $D_{i\tilde{j}}(\boldsymbol{v})$ with $D_{ij_v^*}(\boldsymbol{v})$, we have

$$D_{ij_v^*}(\boldsymbol{v}) - D_{i\tilde{j}}(\boldsymbol{v}) = \left(s_{j_v^*} - s_{\tilde{j}}\right) + r_{j_v^*}\bar{F}_i\left(\sum_{k=1}^{j_v^*} v_{ik}\right) - \sum_{u=j_v^*+1}^{\tilde{j}-1} r_u \left[\bar{F}_i\left(\sum_{k=1}^{u-1} v_{ik}\right) - \bar{F}_i\left(\sum_{k=1}^{u} v_{ik}\right)\right] - r_{\tilde{j}}\bar{F}_i\left(\sum_{k=1}^{\tilde{j}-1} v_{ik}\right),$$

which is independent of $v_{ij'}$ for $j' \geq \tilde{j}$. According to the definition of j_v^*, we know that $D_{ij_v^*}(\boldsymbol{v}) - D_{i\tilde{j}}(\boldsymbol{v}) < 0$. However, given that this is the first time warehouse j_v^* is selected for storage with its index smaller than $\tilde{j}$ after warehouse $\tilde{j}$ has been selected, all the storage quantities in the warehouses with an index smaller than $\tilde{j}$ have not been changed by the algorithm. Therefore, $\tilde{\boldsymbol{v}}$ in the iteration where warehouse $\tilde{j}$ was selected the last time has the same entry values as $\boldsymbol{v}$ for all the products in the warehouses that have an index smaller than $\tilde{j}$ (that is, $\tilde{v}_{ij} = v_{ij}, i \in \mathcal{I}, j < \tilde{j}$). Thus,

$$D_{ij_v^*}(\boldsymbol{v}) - D_{i\tilde{j}}(\boldsymbol{v}) = D_{ij_v^*}(\tilde{\boldsymbol{v}}) - D_{i\tilde{j}}(\tilde{\boldsymbol{v}}) \geq 0.$$

The last inequality holds because warehouse $\tilde{j}$ was selected under $\tilde{\boldsymbol{v}}$. This contradicts $D_{ij_v^*}(\boldsymbol{v}) - D_{i\tilde{j}}(\boldsymbol{v}) < 0$. Thus, part 5 of the structured property holds.

Now we verify part 1. To verify part 1, we will show that the KKT conditions for $\boldsymbol{z} = \boldsymbol{z}(\xi)$, $\xi < \xi^*$ hold. That is, from the modified KKT conditions given in Lemma 4, we have

$$D_{ij}(\boldsymbol{z}) = \lambda_{ij} - \mu_i - \nu_j, i \in \mathcal{I}, j \in \mathcal{J}, \tag{38}$$

and $\lambda_{ij} \geq 0$ and $\nu_j \geq 0$. We verify the KKT conditions by first solving for the Lagrangian multipliers in (38), and then verify that the solution is (i) well defined, that is, different expressions for a Lagrangian multiplier must refer to the same value, and (ii) $\lambda_{ij} \geq 0$ and $\nu_j \geq 0$.

We first solve Equations (38) and obtain the following expressions for the Lagrangian multipliers

$$\mu_i = \begin{cases} -D_{ij}(\boldsymbol{z}), & \text{if } z_{ij} > 0 \text{ and } \sum_{m=1}^{I} z_{mj} < c_j, \text{ for some } j; \\ D_{i'j}(\boldsymbol{z}) - D_{ij}(\boldsymbol{z}) + \mu_{i'}, & \text{if } z_{ik} = 0 \text{ or } \sum_{m=1}^{I} z_{mk} = c_k, \forall k; \end{cases}$$

where $z_{ij}, z_{i'j} > 0$, and $z_{i'j'} > 0$ and $\sum_{m=1}^{I} z_{mj'} < c_{j'}$, for some i', j, j', and therefore, $\mu_{i'} = -D_{i'j'}(\boldsymbol{z})$.

$$\nu_j = \begin{cases} -D_{ij}(\boldsymbol{z}) - \mu_i, & \text{if } \sum_{m=1}^{I} z_{mj} = c_j; \\ 0, & \text{if } \sum_{m=1}^{I} z_{mj} < c_j; \end{cases}$$

where $z_{ij} > 0$, for some i.

$$\lambda_{ij} = \begin{cases} D_{ij}(\boldsymbol{z}) + \mu_i + \nu_j, & \text{if } z_{ij} = 0; \\ 0, & \text{if } z_{ij} > 0. \end{cases}$$

We now verify the above expressions. First, to prove that μ_i is well defined, we need to show that the expression of μ_i is independent of j. To verify the first case of μ_i, note that if $z_{ij} > 0$ and $\sum_{m=1}^{I} z_{mj} < c_j$, for some j, then from Part 5, either $j < j_v^*$ or $j = j_v^*$. We have the following three cases.

Case (i): If $j < j_v^*$, then from the construction of $\boldsymbol{z}$ we have $z_{ik} = v_{ik}$, for all $i \in \mathcal{I}$ and $k \leq j$, and hence $D_{ij}(\boldsymbol{z}) = D_{ij}(\boldsymbol{v})$. Since $\boldsymbol{v} \in \hat{\mathcal{Z}}$, we must have $\boldsymbol{v}$ also satisfies (38). Thus, $\mu_i = -D_{ij}(\boldsymbol{z}) = -D_{ij}(\boldsymbol{v})$ is independent of j.

Case (ii): If $j = j_v^*$ and product i is all stored in the previous iteration, then again $z_{ij''} = v_{ij''}$, for all $j'' \in \mathcal{J}$, and $\mu_i = -D_{ij}(\boldsymbol{z}) = -D_{ij}(\boldsymbol{v})$ is independent of j.

Case (iii): If $j = j_v^*$ and product i is not all stored in the previous iteration, then from part 5 of the structured property, we know that the difference $D_{ij'}(\boldsymbol{z}) - D_{ij_v^*}(\boldsymbol{z})$, $j' < j_v^*$, is independent of z_{ij}, for $j \geq j_v^*$. Since $z_{ik} = v_{ik}$, for all $i \in \mathcal{I}$ and $k < j_v^*$, we have $D_{ij'}(\boldsymbol{z}) - D_{ij_v^*}(\boldsymbol{z}) = D_{ij'}(\boldsymbol{v}) - D_{ij_v^*}(\boldsymbol{v})$. Since $\boldsymbol{v}$ satisfies the KKT conditions, we must have $D_{ij'}(\boldsymbol{z}) - D_{ij_v^*}(\boldsymbol{z}) = D_{ij'}(\boldsymbol{v}) - D_{ij_v^*}(\boldsymbol{v}) = 0$, for some j' such that $\sum_{m=1}^{I} z_{mj'} < c_{j'}$.

This verifies the first case of μ_i. To verify the second case of μ_i, if for all j such that when $z_{ij} > 0$ we have $\sum_{m=1}^{I} z_{mj} = c_j$, then we must have product i stored in warehouses with an index smaller than j_v^*. This is because $\boldsymbol{z}$ is constructed so that it satisfies part 4 of the structured property. Let $\bar{j}$ be the largest warehouse index in which product i is stored. We must have $j \leq \bar{j}$. For $j \leq \bar{j}$, $D_{i'j}(\boldsymbol{z}) - D_{ij}(\boldsymbol{z}) = \sum_{u=\bar{j}}^{J-1} (r_{u+1} - r_u) \left(\bar{F}_i \left(\sum_{m=1}^{u} z_{im} \right) - \bar{F}_{i'} \left(\sum_{m=1}^{u} z_{i'm} \right) \right)$ is independent of j. Therefore, the above difference in the marginal cost is only a function of z_{ij} and $z_{i'j}$, for all $j < j_v^*$. By setting $z_{ij} = v_{ij}$, for all $i \in \mathcal{I}$ and $j < j_v^*$, we have μ_i as a function of v_{ij} and $v_{i'j}$, for $j < j_v^*$. Furthermore, since $\boldsymbol{v} \in \hat{\mathcal{Z}}$, $\boldsymbol{v}$ satisfies the KKT conditions. Thus, the second case of μ_i is verified.

Second, we need to show that ν_j is well defined and non-negative. The second case of ν_j is trivial. Thus, we only need to verify the first case of ν_j. If $z_{ij} > 0$ and $\sum_{m=1}^{I} z_{mj} = c_j$, there are only two possible cases.

Case (i): If $\{i | z_{ij} > 0, i \in \mathcal{I}\} \subseteq \{i | z_{ij_v^*} > 0, i \in \mathcal{I}\}$, then $\mu_i = -D_{ij_v^*}(\boldsymbol{z})$. We have $\nu_j = -D_{ij}(\boldsymbol{z}) + D_{ij_v^*}(\boldsymbol{z})$, which is a function of $\bar{F}_i \left(\sum_{k=j \wedge j_v^*}^{l} z_{ik} \right)$, for all $l = j \wedge j_v^*, \cdots, (j \vee j_v^*) - 1$. If $j < j_v^*$, then $\nu_j = -D_{ij}(\boldsymbol{z}) + D_{ij_v^*}(\boldsymbol{z})$ is a function of $z_{ij'}$, $j' < j_v^*$. Since $z_{ij'} = v_{ij'}$, for all $j' < j_v^*$, and $\boldsymbol{v} \in \hat{\mathcal{Z}}$, we know that parts 3 to 4 of the structured property hold for $\boldsymbol{v}$. Thus, the stockout probabilities are identical for all the possible i. Hence, ν_j is independent of i. If $j \geq j_v^*$, from part 4 of the structured property, we know that $j = j_v^*$. Thus, ν_j is independent of i.

Now we need to show that $\nu_j \geq 0$. If $j < j_v^*$, since $z_{ij'} = v_{ij'}$, for $j' < j_v^*$, we know that $\boldsymbol{z}$ and $\boldsymbol{v}$ lead to the same ν_j. Hence, from the optimality of $\boldsymbol{v}$, we know that $\nu_j \geq 0$. If $j \geq j_v^*$, from part 4 of the structured property, we know that $j = j_v^*$. Thus, $\nu_j = 0$.

Case (ii): If $\{i | z_{ij} > 0, i \in \mathcal{I}\} \not\subseteq \{i | z_{ij_v^*} > 0, i \in \mathcal{I}\}$, then we need to consider two values of μ_i. If $\mu_i = -D_{ij'}(\boldsymbol{z})$, then similar to case (i), we know that the stockout probabilities are identical. Thus, ν_j is independent of i. If $\mu_i = D_{i'j'}(\boldsymbol{z}) - D_{ij'}(\boldsymbol{z}) + \mu_{i'}$ for some i'. We can set $j' = j$ and i' satisfies $z_{i'j_v^*} > 0$, therefore $\nu_j = -D_{i'j}(\boldsymbol{z}) - \mu_{i'}$ is independent of i.

We can verify that $\nu_j \geq 0$ in a way similar to case (i).

Finally, we show that λ_{ij} is well defined and non-negative. It is trivial that if $z_{ij} > 0$ then $\lambda_{ij} = 0$. Thus, we only need to verify the first case of λ_{ij}. If $z_{ij} = 0$, there are two cases.

Case (i): If warehouse j is not full, that is, $\sum_{m=1}^{I} z_{mj} < c_j$, then from part 5 of the structured property and the construction of $\boldsymbol{z}$, we know that $j < j_v^*$ and $\lambda_{ij} = D_{ij}(\boldsymbol{z}) + \mu_i$. Now, if $\mu_i = -D_{ij'}(\boldsymbol{z})$ for some j', then given that $\boldsymbol{z}$ satisfies part 4 of the structured property and $z_{ij} = 0$, we must have $j' < j$. As a result, we have $\lambda_{ij} \geq 0$ because $z_{kj} = v_{kj}$ for all $j < j_v^*$ and $\boldsymbol{v} \in \hat{\mathcal{Z}}$. If $\mu_i = D_{i'j'}(\boldsymbol{z}) - D_{ij'}(\boldsymbol{z}) + \mu_{i'}$, then $z_{ij'} > 0$. Since we have part 4 of the structured property and $z_{ij} = 0$, we must have $j' < j$. Pick an i' such that $z_{i'j} > 0$, then $\mu_{i'} = -D_{i'j}(\boldsymbol{z})$. We have $\lambda_{ij} \geq 0$ because $z_{kj''} = v_{kj''}$ for all $j'' < j_v^*$ and $\boldsymbol{v} \in \hat{\mathcal{Z}}$.

Case (ii): If warehouse j is full, that is, $\sum_{m=1}^{I} z_{mj} = c_j$, then $\lambda_{ij} = D_{ij}(\boldsymbol{z}) + \mu_i - D_{i'j}(\boldsymbol{z}) - \mu_{i'}$, where $z_{i'j} > 0$. Now, if $\mu_i = -D_{ij'}(\boldsymbol{z})$ for some j', then from part 3 of the structured property, we must have $z_{i'j'} > 0$ and $\mu_{i'} = -D_{i'j'}(\boldsymbol{z})$. Let j' be the largest index such that $z_{ij'} > 0$, then $j' < j$ and $D_{ij}(\boldsymbol{z}) - D_{ij'}(\boldsymbol{z}) = s_j - s_{j'}$. Thus, $\lambda_{ij} = \sum_{u=j'}^{j-1} (r_{u+1} - r_u)\, \bar{F}_i \left(\sum_{k=1}^{u} z_{ik}\right) \geq 0$. If $\mu_i = D_{i''j'}(\boldsymbol{z}) - D_{ij'}(\boldsymbol{z}) + \mu_{i''}$, for some i'' and j', then because of the nested property we must have $j' < j$. We can pick $i'' = i'$ and hence $\lambda_{ij} = D_{ij}(\boldsymbol{z}) + D_{i'j'}(\boldsymbol{z}) - D_{ij'}(\boldsymbol{z}) - D_{i'j}(\boldsymbol{z})$. Now, λ_{ij} has the same expression as that under the case of $\mu_i = -D_{ij'}(\boldsymbol{z})$. Thus, we have $\lambda_{ij} \geq 0$.

Thus, we have verified part 1: $\boldsymbol{z} \in \hat{\mathcal{Z}}$.

Part 2 follows directly from the construction of $\boldsymbol{z}$ and the definition of ξ^*. ■

A.7 Proof of Theorem 2

We first prove that, subject to the error due to bisection, Algorithm 1 obtains an optimal storage matrix by calling Algorithm 3 at most $2J-1$ times. Since the structured property in Definition 4 holds for the initial storage matrix $\boldsymbol{v} = \mathbf{0}$, the structured property also holds for $\boldsymbol{z}^*$ after each iteration of Algorithm 1 according to Lemma 3. Thus, Algorithm 1 generates an ϵ-optimal storage matrix if it terminates. Now, we show that Algorithm 1 terminates in a finite number of iterations. Note that Algorithm 3 is called once in each iteration of Algorithm 1. Suppose there are n warehouses. Let m_n be the maximum possible number of times Algorithm 3 is called. We will prove that $m_J = 2J-1$ by induction. Note that $m_1 = 1$ and $m_2 = 3$. Suppose $m_n = 2n-1$ for $n < J$. When $n = J$, we have the following result. Suppose in the first iteration of Algorithm 1, warehouse j is selected as the target warehouse. After some iterations, warehouse j is filled. Each warehouse $j' < j$ is empty, whereas each warehouse $j'' > j$ is full or empty. Let b be the number of warehouses that are full. The number of empty warehouses is $J-b$. Thus, if $b < J$, then the maximum possible number of times Algorithm 3 is called, given that the first target warehouse is j, is $m_b + m_{J-b} = 2b-1+2(J-b)-1 = 2J-2$. If $b = J$, then we must have $j = 1$ and warehouse j is the last warehouse that is completely filled. In this case, we first partially fill warehouse 1, then fill warehouses 2 to J, and finally fill up warehouse 1. The number of iterations is $1+m_{J-1}+1 = 2J-1$. Combining all the possibilities, we have $m_J = \max\{2J-2, 2J-1\} = 2J-1$.

Let $\bar{L}$ represent the maximum Lipschitz constant for $\bar{F}_i^{-1}(\cdot)$. Next, we show that if we set the search accuracy of Algorithm 3 as $\epsilon/(2\bar{L})$, then the output of Algorithm 3, denoted as $\hat{z}$, satisfies the condition $\max_{i\in\mathcal{I}, j\in\mathcal{J}} |\hat{z}_{ij} - z^*_{ij}| \leq \epsilon$, where $\boldsymbol{z}^* \in \mathcal{Z}(\boldsymbol{v})$ is the intermediate storage matrix. This is because the only source of error originates from the bisection method. If the search accuracy of Algorithm 3 is $\epsilon/(2\bar{L})$, then upon termination of the algorithm, we have $\max_{j\in\mathcal{J}} |\chi_j(\boldsymbol{z}) - \chi_j(\boldsymbol{z}^*)| \leq \epsilon/(2\bar{L})$. Since $\hat{z}_{ij} = \min\left\{\bar{F}_i^{-1}(\chi_j(\boldsymbol{z})), q_i\right\} - \min\left\{\bar{F}_i^{-1}(\chi_{j+1}(\boldsymbol{z})), q_i\right\}$, we have

$$|\hat{z}_{ij} - z^*_{ij}| \quad \leq \quad 2\bar{L} \max_{j\in\mathcal{J}} |\chi_j(\boldsymbol{z}) - \chi_j(\boldsymbol{z}^*)| \leq 2\bar{L} \cdot \frac{\epsilon}{2\bar{L}} = \epsilon, \quad \forall i \in \mathcal{I}, j \in \mathcal{J}.$$

Therefore, we conclude that as Algorithm 1 terminates, an ϵ-optimal storage matrix is produced.

Given that the search accuracy is $\epsilon/(2\bar{L})$, filling each warehouse j using the binary search requires at most $\log\left(2\bar{L}/\epsilon\right)$ calculations of $\boldsymbol{z}$ based on the stockout probability, which requires $O(IT)$ calls of $\bar{F}_i^{-1}(\cdot)$ functions. For each call of Algorithm 3, if there is no crossing, we just need to call Algorithms 4 and 5 once. If there is a crossing, for each value of ξ in Algorithm 3, we need to call Algorithm 4 once. In total, we need to call Algorithm 4 at most $\log\left(2\bar{L}/\epsilon\right)$ times. Both Algorithms 4 and 5 require filling at most J warehouses and require at most $O\left(IJ\log\left(2\bar{L}/\epsilon\right)\right)$. Overall, each call of Algorithm 3 requires at most $O\left(IJT\log^2\left(2\bar{L}/\epsilon\right)\right)$ calls of $\bar{F}_i^{-1}(\cdot)$. According to the first paragraph above, the total number of calls on Algorithm 3 is at most $2J-1$, which is $O(J)$. Therefore, the total computational effort is upper bounded by $O\left(IJ^2T\log^2\left(2\bar{L}/\epsilon\right)C\left(\bar{F}^{-1}\right)\right)$. ■

A.8 Proof of Theorem 3

In order to prove Theorem 3, we need to first prove the following lemmas.

The following lemma characterizes the KKT conditions of the optimal order policy $\boldsymbol{q}^*$.

Lemma 7 *An optimal ordering policy $\boldsymbol{q}^*$ satisfies the following KKT conditions: For $i \in \mathcal{I}$,*

$$\mu_i^*(\boldsymbol{q}^*) = \rho_i + h_i T - (p_i - r_J)\bar{F}_i(q_i^*) - h_i \sum_{t=1}^{T} Pr\left(q_i^* > \sum_{\tau=1}^{t} d_{i\tau}\right) + K \tag{39}$$

or

$$\mu_i^*(\boldsymbol{q}^*) < \rho_i + h_i T - (p_i - r_J)\bar{F}_i(q_i^*) - h_i \sum_{t=1}^{T} Pr\left(q_i^* > \sum_{\tau=1}^{t} d_{i\tau}\right) + K \text{ and } q_i^* = 0, \tag{40}$$

where K is non-negative and $K > 0$ only if all the warehouses are full.

Proof : Since $V(\boldsymbol{q})$ is differentiable and jointly concave in $\boldsymbol{q}$ and $\boldsymbol{\rho}'\boldsymbol{q}$ is linear in $\boldsymbol{q}$, the objective function of the ordering problem is differentiable and jointly concave in $\boldsymbol{q}$. Since the constraints are linear, the optimal ordering policy satisfies the first-order conditions. This proves the lemma. ■

We derive $\partial G(\boldsymbol{q}, \boldsymbol{x}^*(\boldsymbol{q}))/\partial q_i$ in closed form in the following lemma.

Lemma 8 *If $\frac{\partial G(\boldsymbol{x}^*(\boldsymbol{q}))}{\partial q_i}$ exists, then*

$$\frac{\partial G(\boldsymbol{x}^*(\boldsymbol{q}))}{\partial q_i} \quad = \quad \mu_i^*(\boldsymbol{q}) \tag{41}$$

where

$$\mu_i^*(\boldsymbol{q}) \quad = \quad \begin{cases} -D_{i\bar{j}(i)}(\boldsymbol{x}^*(\boldsymbol{q})), & \text{if } \bar{j}(i) > 0, \\ D_{i^*\underline{j}(i)}(\boldsymbol{x}^*(\boldsymbol{q})) - D_{i\underline{j}(i)}(\boldsymbol{x}^*(\boldsymbol{q})) - D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*(\boldsymbol{q})), & \text{if } \bar{j}(i) = 0, \end{cases} \tag{42}$$

and $\mu_i^*(\boldsymbol{q})$ is non-increasing in q_k, for all $k \in \mathcal{I}$.

Proof : Recall that given any order quantities $\boldsymbol{q}$, the optimal storage quantities $\boldsymbol{x}^*(\boldsymbol{q})$ can be determined by solving the following problem:

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}} & G(\boldsymbol{x}) \\ s.t. & \boldsymbol{x}\boldsymbol{e} - \boldsymbol{q} = \boldsymbol{0}; \quad \boldsymbol{x}'\boldsymbol{e} - \boldsymbol{c} \le \boldsymbol{0}; \quad \boldsymbol{x} \ge \boldsymbol{0}. \end{array} \tag{43}$$

The Lagrangian of the above problem is

$$L(\boldsymbol{q}, \boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\xi}) \quad = \quad G(\boldsymbol{x}) - \boldsymbol{\mu} \cdot (\boldsymbol{x}\boldsymbol{e} - \boldsymbol{q}) + \boldsymbol{\nu} \cdot (\boldsymbol{x}'\boldsymbol{e} - \boldsymbol{c}) - \boldsymbol{\lambda} \odot \boldsymbol{x}, \tag{44}$$

where $\boldsymbol{\nu}, \boldsymbol{\lambda} \ge \boldsymbol{0}$, and $\boldsymbol{a} \odot \boldsymbol{b}$ represents the sum of all entries of the entry-wise product of the matrices $\boldsymbol{a}$ and $\boldsymbol{b}$ $\left(\text{that is, } \boldsymbol{a} \odot \boldsymbol{b} = \sum_{i,j} a_{ij} b_{ij}\right)$.

Since $\tilde{d}_i$ has positive support on $[0, U_i]$ with continuous and differentiable p.d.f. for all i, and $V(\boldsymbol{q})$ is differentiable in $\boldsymbol{q}$, $\mathcal{U}$ is continuous in warehouse capacity $\boldsymbol{c}$. Thus, we can ignore the discussions on the degenerated cases when $\sum_{i \in \mathcal{I}} x^*_{ij^*_{x^*}} = c_j$ or $q_i^* = 0$ because we can always perturb c_j or q_i^* so that these cases will not occur.

We first prove Lemma 8 when $\boldsymbol{q} \cdot \boldsymbol{e} < \boldsymbol{c} \cdot \boldsymbol{e}$. Note that when $\boldsymbol{q} \cdot \boldsymbol{e} < \boldsymbol{c} \cdot \boldsymbol{e}$, $\bar{j}(i^*) > 0$. Based on the Lagrangian of the storage problem in Equation (44) and according to the Envelope Theorem (Milgrom and Segal 2002), we have

$$\frac{\partial G(\boldsymbol{x}^*(\boldsymbol{q}))}{\partial q_i} = \frac{\partial L(\boldsymbol{q}, \boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\xi}^*)}{\partial q_i} = \mu_i^*.$$

Note that from the proof of Lemma 3 we have

$$\mu_i^* \quad = \quad \begin{cases} -D_{ij}(\boldsymbol{x}^*), & \text{if } x_{ij}^* > 0 \text{ and } \sum_{m=1}^{I} x_{mj}^* < c_j, \text{ for some } j; \\ D_{i^*j}(\boldsymbol{x}^*) - D_{ij}(\boldsymbol{x}^*) + \mu_{i^*}^*, & \text{if } x_{ik}^* = 0 \text{ or } \sum_{m=1}^{I} x_{mk}^* = c_k, \forall k; \end{cases}$$

where $x_{ij}^*, x_{i^*j}^* > 0$, and $x_{i^*\bar{j}(i^*)}^* > 0$ for some j, and therefore, $\mu_{i^*}^* = -D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*)$. Hence we have the expression of $\mu_i^*(\boldsymbol{q})$ in Equation (42).

Now we prove that μ_i^* is non-increasing in $q_{i''}$ for all $i'' \in \mathcal{I}$. For a certain $\boldsymbol{q}$, there are two possible cases: (i) $\bar{j}(i'') = 0$, and (ii) $\bar{j}(i'') > 0$. We increase $q_{i''}$ by a small amount $\epsilon > 0$ and discuss each case.

Case (i) $\bar{j}(i'') = 0$**:** Under optimality, ϵ is added to warehouse $\underline{j}(i'')$. Let $\hat{j} = \min\{j | j > \underline{j}(i''), 0 < \sum_{i\in\mathcal{I}} x^*_{ij} < c_j, j \in \mathcal{J}\}$ be the smallest warehouse index larger than $\underline{j}(i'')$ such that $\hat{j}$ is not empty but also not completely filled. Similar to part 4 of the structured property in Definition 4 , if we adjust the stockout probabilities from warehouse $\underline{j}(i'')$ to $\hat{j}$ such that the stockout probability for each product is identical or that product is all stored, and $\sum_{k=1}^{\hat{j}} x^*_{i'k}$ remains constant for all $i' \neq i''$ before and after ϵ is added to warehouse $\underline{j}(i'')$, then because of the KKT conditions the storage matrix after adjustment is optimal.

Compare μ^*_i before and after ϵ is added to warehouse $\underline{j}(i'')$, we have the following three sub-cases.

- **(a)** For product i such that $\underline{j}(i) < \underline{j}(i'')$, because $\bar{j}(i'') = 0$ we have all warehouses with $x^*_{i''j} > 0$ are full. From part 4 of the structured property, $\underline{j}(i) < \underline{j}(i'')$ implies that warehouses with $x^*_{ij} > 0$ is a subset of warehouses with $x^*_{i''j} > 0$, and hence they are all full. Therefore, $\mu^*_i = D_{i^*\underline{j}(i)}(\boldsymbol{x}^*) - D_{i\underline{j}(i)}(\boldsymbol{x}^*) - D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*)$. Because $\bar{j}(i'') = 0$, we have $\bar{j}(i^*) \geq \hat{j}$. After ϵ amount of i'' is added optimally to the warehouses, $D_{i\underline{j}(i)}(\boldsymbol{x}^*)$ remains unchanged because x^*_{ik} for all $k \in \mathcal{J}$ is unchanged. $D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*)$ remains unchanged because $\sum_{k=1}^{j} x^*_{i^*k}$ remains unchanged for all $j \geq \hat{j}$. $D_{i^*\underline{j}(i)}(\boldsymbol{x}^*)$ decreases because $\sum_{k=1}^{j} x^*_{i^*j}$ decreases for all $\underline{j}(i'') \leq j < \hat{j}$. Thus, μ_i decreases.
- **(b)** For product i such that $\hat{j} > \underline{j}(i) \geq \underline{j}(i'')$, because $\underline{j}(i) < \hat{j}$ warehouses with $x^*_{ij} > 0$ are all full. Therefore, $\mu^*_i = D_{i^*\underline{j}(i)}(\boldsymbol{x}^*) - D_{i\underline{j}(i)}(\boldsymbol{x}^*) - D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*)$. Because $\bar{j}(i'') = 0$, we have $\bar{j}(i^*) \geq \hat{j}$. $D_{i\underline{j}(i)}(\boldsymbol{x}^*)$ remains unchanged because $\sum_{k}^{j} x^*_{ik}$ is unchanged for $k > \underline{j}(i)$. Similar to sub-case (i), $D_{i^*\bar{j}(i^*)}(\boldsymbol{x}^*)$ remains unchanged whereas $D_{i^*\underline{j}(i)}(\boldsymbol{x}^*)$ decreases. Thus, μ_i decreases.
- **(c)** For product i such that $\hat{j} \leq \underline{j}(i)$, $\mu^*_i = -D_{i\hat{j}}(\boldsymbol{x}^*)$. $D_{i\hat{j}}(\boldsymbol{x}^*)$ remains unchanged because $\sum_{k=1}^{j} x^*_{ik}$ remains unchanged for all $j \geq \hat{j}$. Thus, μ_i is unchanged.

Case (ii) $\bar{j}(i'') > 0$**:** We have the following two sub-cases.

- **(a)** If (ii) occurs and $\underline{j}(i'') = \bar{j}(i'')$, then add ϵ to warehouse $\underline{j}(i'')$ is optimal and the rest of the storage quantities remain unchanged. Thus, μ^*_i remains unchanged for all $i \neq i''$ and $\mu^*_{i''}$ decreases.
- **(b)** If (ii) occurs and $\underline{j}(i'') > \bar{j}(i'')$, let $\hat{j} = \min\{j | j > \underline{j}(i''), 0 < \sum_{i\in\mathcal{I}} x^*_{ij} < c_j, j \in \mathcal{J}\}$ be the smallest warehouse index larger than $\underline{j}(i'')$ such that $\hat{j}$ is not empty but also not completely filled. If such warehouse does not exist, then let $\hat{j} = J$. From the KKT conditions, under optimality a total ϵ unit of product i'' is stored to warehouses $\bar{j}(i'')$ to $\underline{j}(i'')$ such that: (1) the stockout probability for each product is identical or that product is all stored, (2) $\sum_{k=1}^{j} x^*_{ik}$ remains unchanged for all $j \geq \hat{j}$ and $i \neq i''$. Similar to the discussion on the three sub-cases under case (i), we have μ_i is non-increasing in $q_{i''}$.

When $\sum_{i=1}^{I} q_i = \sum_{j=1}^{J} c_j$, if we increase the warehouse capacity at the last warehouse that is completely filled by Algorithm 1 by some amount, then the optimal storage quantities $\boldsymbol{x}^*$ remain the same. Therefore, the Lagrangian multiplier $\mu^*_i(\boldsymbol{q})$ for all $i \in \mathcal{I}$ also remains the same. Under this new setting, i' is the product that is last stored and $\bar{j}(i')$ is the warehouse that is last filled in Algorithm 1. Then, the problem returns to that of $\sum_{i=1}^{I} q_i < \sum_{j=1}^{J} c_j$. ■

Now, we are ready to prove Theorem 3. Substituting Equation (41) into Equation (39), we obtain Theorem 3. ■

A.9 Proof of Theorem 4

Since the c.d.f. is constructed based on (24), it is clear that all the distributions are continuous. Based on Lemma 2 and Equations (58)-(59), both $\Phi_{1Z}(\boldsymbol{x})$ and $\Phi_{1W}(\boldsymbol{x})$ have Lipschitz continuous gradient. Chambolle and Dossal (2015) show that the number of iterations required to achieve an ϵ-optimal solution is $O\left(\frac{1}{\sqrt{\epsilon}}\right)$. In each iteration of Algorithm 7, we need to compute the gradient of $u_{sp}(\boldsymbol{x})$ and project a point to the feasible region once. Computing the gradient of $\Phi_{1Z}(\boldsymbol{x})$ requires $I(J^2+JT)C\left(\bar{F}\right)$ based on (32). Computing the gradient of $\Phi_{1W}(\boldsymbol{x})$ requires $I(K+T)C\left(\bar{F}\right)$ based on (58). Projecting a point to the feasible region incurs a computational cost of $O(IJ\log(\frac{1}{\epsilon}))$ by setting the projection accuracy to a constant fraction of ϵ (see Algorithm 8 and the subsequent discussion). In total, the computational cost is upper bounded by $O\left(I\left[\left(J^2+JT+K\right)C\left(\bar{F}\right)+J\log\left(\frac{1}{\epsilon}\right)\right]\frac{1}{\sqrt{\epsilon}}\right)$. ■

B Algorithms

B.1 Finding the intermediate storage matrix $\boldsymbol{z}^*$

Figure 12 illustrates the binary search in Algorithm 3. In Algorithm 3, we first try to fill the target warehouse j_v^* as much as possible. We can achieve this by reducing ξ as much as possible to obtain a new iteration-wise feasible $\hat{\boldsymbol{z}}$. After that, we check whether $D_{i^*j_v^*}(\hat{\boldsymbol{z}})$ is still the smallest marginal cost among all the warehouses in $\Gamma(\boldsymbol{v})$. If so, $\hat{\boldsymbol{z}}$ is iteration-wise optimal and is returned to Algorithm 1. Otherwise, a phenomenon called *crossing* of the marginal costs occurs. This is because when ξ decreases, the marginal cost $D_{i^*j}(\boldsymbol{z}(\xi))$ of each warehouse $j \in \Gamma(\boldsymbol{v})$ will continuously increase. See Figure 13 for an illustration. Therefore, it is possible that there exists another warehouse j' that has a smaller marginal cost than warehouse j_v^* under $\hat{\boldsymbol{z}}$. We call this phenomenon crossing because warehouse j' has a larger marginal cost under $\boldsymbol{v}$ but has a smaller marginal cost under $\hat{\boldsymbol{z}}$ than warehouse j_v^*. We want to find ξ^* in Figure 13 so that warehouses j_v^* and j' both have the smallest marginal cost under $\boldsymbol{z}(\xi^*)$. That is, the target warehouse switches from j_v^* to j' at $\boldsymbol{z}(\xi^*)$.

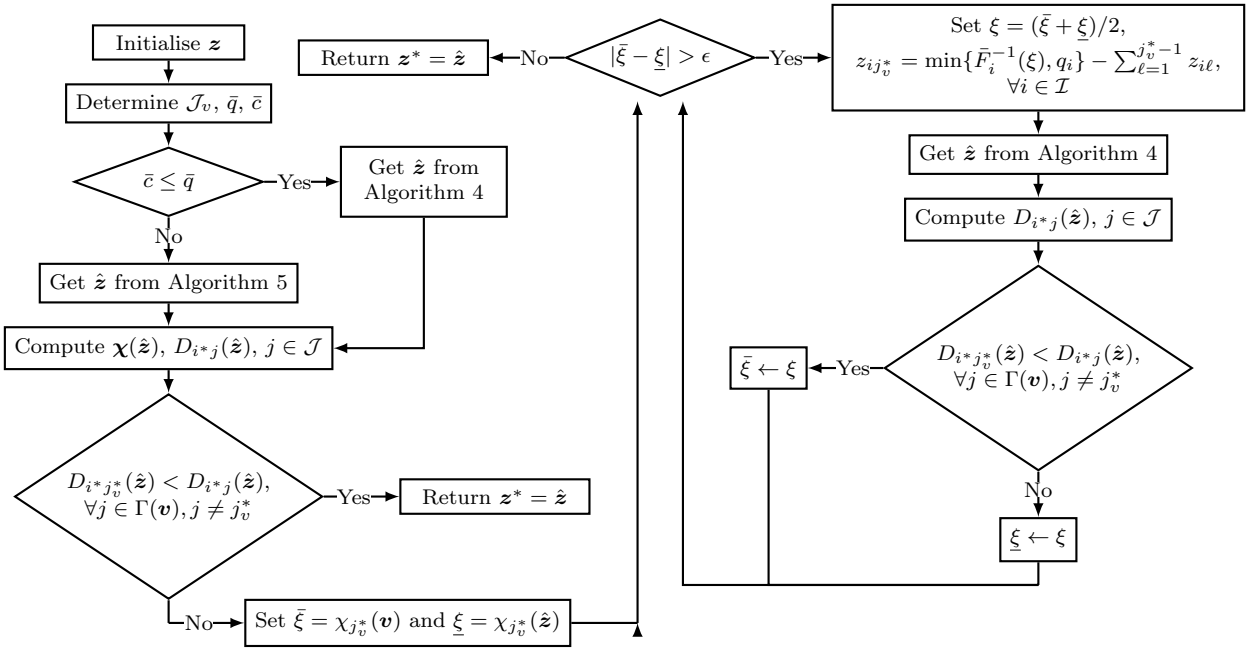


Figure 12: Illustration of Algorithm 3

The detailed procedure of computing a storage matrix $\boldsymbol{z}^*$ is described in Algorithm 3.

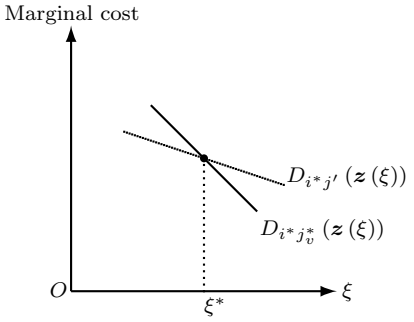


Figure 13: Illustration of crossing

Algorithm 3 (Finding $\boldsymbol{z}^* \in \mathcal{Z}(\boldsymbol{v})$)
Given initial inventory levels $\boldsymbol{v}$, determine the target warehouse $j_v^* \in \Gamma(\boldsymbol{v})$ and the target stockout probability vector $\boldsymbol{\chi}(\boldsymbol{v})$. Initialize $z_{ij} = v_{ij}$ for $j < j_v^*$, and $z_{ij} = 0$ for $j \geq j_v^*$, $i \in \mathcal{I}$.

1. Set $\mathcal{J}_v = \{j | j > j_v^*, \sum_{i\in\mathcal{I}} v_{ij} > 0, j \in \mathcal{J}\}$. Set $\bar{q} = \sum_{i\in\mathcal{I}} \left(q_i - \sum_{j\in\mathcal{J}} z_{ij}\right)$ and $\bar{c} = \sum_{j\in j_v^* \cup \mathcal{J}_v} c_j$.

2. If $\bar{c} \leq \bar{q}$, let $\hat{\boldsymbol{z}}$ be the output of Algorithm 4 with the input being $j_v^* \cup \mathcal{J}_v$ and $\boldsymbol{z}$.

Else, let $\hat{\boldsymbol{z}}$ be the output of Algorithm 5 with the input being $\mathcal{J}_v$ and $\boldsymbol{z}$.

Set $\hat{z}_{ij_v^*} = q_i - \sum_{j\neq j_v^*} \hat{z}_{ij}$ for all $i \in \mathcal{I}$.

3. Compute the target stockout probability vector $\boldsymbol{\chi}(\hat{\boldsymbol{z}})$ and the marginal costs $D_{i^*j}(\hat{\boldsymbol{z}}), j \in \mathcal{J}$.

3a. If $D_{i^*j_v^*}(\hat{\boldsymbol{z}}) < D_{i^*j}(\hat{\boldsymbol{z}})$ for all $j \in \Gamma(\boldsymbol{v}), j \neq j_v^*$, then return $\boldsymbol{z}^* = \hat{\boldsymbol{z}}$.

3b. Else, set $\bar{\xi} = \chi_{j_v^*}(\boldsymbol{v})$ and $\underline{\xi} = \chi_{j_v^*}(\hat{\boldsymbol{z}})$.

While $|\bar{\xi} - \underline{\xi}| > \epsilon$:

Set $\xi = \left(\bar{\xi} + \underline{\xi}\right)/2$ and $z_{ij_v^*} = \min\left\{\bar{F}_i^{-1}(\xi), q_i\right\} - \sum_{\ell=1}^{j_v^*-1} z_{i\ell}$, for all $i \in \mathcal{I}$.

Let $\hat{\boldsymbol{z}}$ be the output of Algorithm 4 with the input being $\mathcal{J}_v$ and $\boldsymbol{z}$.

Compute the marginal costs $D_{i^*j}(\hat{\boldsymbol{z}}), j \in \mathcal{J}$.

If $D_{i^*j_v^*}(\hat{\boldsymbol{z}}) < D_{i^*j}(\hat{\boldsymbol{z}})$ for all $j \in \Gamma(\boldsymbol{v}), j \neq j_v^*$, then set $\bar{\xi} = \xi$. Else, set $\underline{\xi} = \xi$.

Return $\boldsymbol{z}^* = \hat{\boldsymbol{z}}$.

Algorithm 3 involves two subroutines: Algorithms 4 and 5, which are presented as follows.

Algorithm 4 (Filling warehouses from the smallest index to the largest index)
Given a set of warehouses $\breve{\mathcal{J}}$ and initial inventory levels $\hat{\boldsymbol{z}}$, compute the target stockout probability vector $\boldsymbol{\chi}(\hat{\boldsymbol{z}})$. Initialize $\xi = \chi_{j_v^*}(\hat{\boldsymbol{z}})$ and $j = j_v^*$.

While $j \leq J$:

If $j \in \breve{\mathcal{J}}$:

1. Solve for η_j such that

$$\sum_{i=1}^{I} \min\left\{\bar{F}_i^{-1}(\eta_j), q_i\right\} - \sum_{i=1}^{I} \min\left\{\bar{F}_i^{-1}(\xi), q_i\right\} = c_j.$$

2. Compute $\hat{z}_{ij} = \min\left\{\bar{F}_i^{-1}(\eta_j), q_i\right\} - \min\left\{\bar{F}_i^{-1}(\xi), q_i\right\}$, for all $i \in \mathcal{I}$.

3. Set $\xi \leftarrow \eta_j$.

$j \leftarrow j + 1$.

Return $\hat{\boldsymbol{z}}$.

Algorithm 5 (Filling warehouses from the largest index to the smallest index)
Given a set of warehouses $\breve{\mathcal{J}}$ and initial inventory levels $\hat{\boldsymbol{z}}$. Initialize $\eta = 0$ and $j = J$.

While $j > 0$:

If $j \in \breve{\mathcal{J}}$:

1. Solve for ξ_j such that

$$\sum_{i=1}^{I} \min\left\{\bar{F}_i^{-1}(\eta), q_i\right\} - \sum_{i=1}^{I} \min\left\{\bar{F}_i^{-1}(\xi_j), q_i\right\} = c_j.$$

2. Compute $\hat{z}_{ij} = \min\left\{\bar{F}_i^{-1}(\eta), q_i\right\} - \min\left\{\bar{F}_i^{-1}(\xi_j), q_i\right\}$, for all $i \in \mathcal{I}$.

3. Set $\eta \leftarrow \xi_j$.

$j \leftarrow j - 1$.

Return $\hat{\boldsymbol{z}}$.

Given $\boldsymbol{v}$, Algorithm 3 first determines j_v^* and $\boldsymbol{\chi}(\boldsymbol{v})$. We then initialize $z_{ij} = v_{ij}$, $j < j_v^*$, $i \in \mathcal{I}$ based on conditions 1 and 4 for the iteration-wise feasibility. Note that $\mathcal{J}_v$ is a set of warehouses $j > j_v^*$ that are non-empty. Step 2 of Algorithm 3 fills the warehouses in $j_v^* \cup \mathcal{J}_v$ as much as possible. Note that $\bar{c}$ is the total capacity of warehouses in $j_v^* \cup \mathcal{J}_v$, and $\bar{q}$ is the total quantity of all the remaining products that are not stored in warehouses $j < j_v^*$. If $\bar{c} \leq \bar{q}$, there are sufficient products to fill the warehouses in $j_v^* \cup \mathcal{J}_v$. We call Algorithm 4 to fill these warehouses from the smallest index to the largest index. If $\bar{c} > \bar{q}$, we keep all the warehouses in $\mathcal{J}_v$ full (see part 5 of the structured property). We call Algorithm 5 to fill the warehouses in $\mathcal{J}_v$ from the largest index to the smallest index, and then store the remaining products to warehouse j_v^*. Algorithms 4 and 5 ensure that $\hat{\boldsymbol{z}}$ produced by step 2 of Algorithm 3 is an iteration-wise feasible storage matrix. Algorithm 3 returns $\hat{\boldsymbol{z}}$ if step 3a finds that $D_{i^* j_v^*}(\hat{\boldsymbol{z}})$ is the smallest marginal cost among all the warehouses in $\Gamma(\boldsymbol{v})$. Otherwise, crossing occurs and we apply a binary search to find the crossing point ξ^* in step 3b.

B.2 Ordering algorithm for the single-warehouse single-zone problem

Recall that G_i^t is continuous and $\omega_i'(q) = -\rho_i - s - h_i T + h_i \sum_{t=1}^{T} \left(\bar{F}_i^t(q)\right) + (p_i - r)\bar{F}_i^T(q)$ is a decreasing function, and thus its inverse function $(\omega_i')^{-1}(\cdot)$ exists.

Algorithm 6 (Ordering Algorithm for The Single-warehouse Single-zone Problem) *Given $\omega_i^{'}(0), i = 1, ..., I$, initialize $\boldsymbol{q} = \mathbf{0}$, $i_0 = 0$, and $z = 0$.*

1. If $i_0 = I$, then return $\boldsymbol{q}$; otherwise, set $i_0 = i_0 + 1$ and compute z such that $\sum_{i=1}^{i_0} (\omega_i^{'})^{-1}(z) = c$.

2. If $z \geq (\omega_{i_0+1}^{'})(0)$, update $q_i = (\omega_i^{'})^{-1}(z)$ for $i = 1, ..., i_0$ and return $\boldsymbol{q}$; otherwise update $q_i = (\omega_i^{'})^{-1}\left((\omega_{i_0+1}^{'})(0)\right)$ for $i = 1, ..., i_0$ and go to step 1.

Theorem 5 *We call $\boldsymbol{q}$ an* ϵ-optimal ordering policy *if $\max_{i \in \mathcal{I}} |q_i - q_i^*| \leq \epsilon$, where $\boldsymbol{q}^*$ is an optimal ordering policy. Given any accuracy $\epsilon > 0$, Algorithm 6 obtains an ϵ-optimal ordering policy and its computational cost is at most $O\left(I^2 T L \log^2(L/\epsilon)\, C\left(\bar{F}^{-1}\right)\right)$, where $C\left(\bar{F}^{-1}\right)$ is the maximum computational effort to call the function $(\bar{F}_i^t)^{-1}(\cdot)$ and $L = \max_{i,z} |(\omega_i^{''})^{-1}(z)|$.*

Proof. For the output $\boldsymbol{q}$ of Algorithm 6, there exists an integer $1 \leq i_0 \leq I$ such that $q_i > 0$ for $i \leq i_0$ and $q_i = 0$ for $i > i_0$. From Algorithm 6, there exists a non-negative constant a such that q_i satisfies $\omega_i^{'}(q_i) = a$ for all $i \leq i_0$ and $\omega_i^{'}(0) \leq a$ for all $i > i_0$. If $i_0 = I$, then $a = 0$, we have $\omega_i^{'}(q_i) = 0$ for all $i \leq I$. This solution coincides with the optimal solution when the capacity constraint is not binding. If $i_0 < I$, then $a > 0$, $q_i = 0$ for all $i > I_0$, and the capacity constraint is binding. The Lagrangian multiplier for the capacity constraint is $a > 0$ and that for each $q_i = 0$ $i > I_0$ is $0 < \omega_i^{'}(0) < a$. This implies the optimality of the output of Algorithm 6.

Given a search accuracy, $\epsilon/L > 0$ for the binary search, solving the equation in step 1 using the binary search requires at most $\log(L/\epsilon)$ candidate solutions of z, and the final z satisfies $|z - z^*| < \epsilon/L$, which means that $|q_i - q_i^*| = |(\omega_i^{'})^{-1}(z) - (\omega_i^{'})^{-1}(z^*)| < L\epsilon/L = \epsilon$.The evaluation of left-hand-side for each candidate z requires $O(I)$ calls of $(\omega_i^{'})^{-1}(\cdot)$ functions, which calls at most T times of $(\bar{F}_i^t)^{-1}(\cdot)$ function. The computational time for step 2 is upper bounded by step 1. Since the total calls of step 1 is at most I, the total computational effort is upper bounded by $O\left(I^2 T L \log^2(L/\epsilon)\, C\left(\bar{F}^{-1}\right)\right)$. ■

C The single-warehouse multi-zone problem

In this section, we study a special case with only a single warehouse but multiple demand zones. We drop the subscript j and Problem (1) becomes as follows:

$$\max_{\pi \in \Pi} \quad -\sum_{i=1}^{I} \rho_i x_i^{\pi 1} - s \sum_{i=1}^{I} x_i^{\pi 1} + \sum_{i=1}^{I} \sum_{t=1}^{T} E_{\tilde{\mathbf{d}}^1, ..., \tilde{\mathbf{d}}^T} \left[-h_i x_i^{\pi t+1} + \sum_{k=1}^{K} (p_i - r_k)\, y_{ik}^{\pi t} \right] \tag{45}$$

$$
\begin{aligned}
s.t. \quad & \sum_{i=1}^{I} x_i^{\pi 1} \le c; && (46)\\
& x_i^{\pi t} = x_i^{\pi t-1} - \sum_{k=1}^{K} y_{ik}^{\pi t-1}, t = 2, ..., T; && (47)\\
& y_{ik}^{\pi t} \le d_{ik}^{\pi t}, \quad i \in \mathcal{I}, k \in \mathcal{K}, t = 1, ..., T; && (48)\\
& \sum_{k=1}^{K} y_{ik}^{\pi t} \le x_i^{\pi t}, \quad j \in \mathcal{J}, t = 1, ..., T; && (49)\\
& x_i^{\pi t} \ge 0, \quad i \in \mathcal{I}, t = 1, ..., T; && (50)\\
& x_i^{\pi T+1} = 0, \quad i \in \mathcal{I}; && (51)\\
& y_{ik}^{\pi t} \ge 0, \quad i \in \mathcal{I}, k \in \mathcal{K}, t = 1, ..., T. && (52)
\end{aligned}
$$

An Upper Bound

Denote $d_{ik} = \sum_{t=1}^{T} d_{ik}^t$, $\tilde{d}_{ik} = \sum_{t=1}^{T} \tilde{d}_{ik}^t$, $i \in \mathcal{I}$, $k \in \mathcal{K}$, $\boldsymbol{d}_i = (d_{ik})_{1\times K}$, and $\tilde{\boldsymbol{d}}_i = (\tilde{d}_{ik})_{1\times K}$, $k \in \mathcal{K}$. For each product i, we sum over the demands from all periods into one aggregate demand and allow the retailer to choose his retrieval policy based on the aggregated demand. This provides an upper bound to Problem (45) and it can be solved via a two-stage stochastic optimization problem.

$$
\begin{aligned}
\max \quad u(\boldsymbol{x}) = \quad & -\sum_{i=1}^{I} \rho_i x_i - s\sum_{i=1}^{I} x_i - \sum_{t=1}^{T}\sum_{i=1}^{I} h_i E\left[x_i - \sum_{\tau=1}^{t}\sum_{k=1}^{K} d_{ik}^{\tau}\right]^+ + E\left[\sum_{i=1}^{I} \breve{W}_i\left(x_i, \tilde{\boldsymbol{d}}_i\right)\right] && (53)\\
s.t. \quad & \sum_{i=1}^{I} x_i \le c;\\
& x_i \ge 0, \qquad i \in \mathcal{I};
\end{aligned}
$$

where

$$
\begin{aligned}
\breve{W}_i\left(x_i, \boldsymbol{d}_i\right) = \quad \max \quad & \sum_{k=1}^{K} (p_i - r_k) y_{ik} && (54)\\
s.t. \quad & y_{ik} \le d_{ik}, \qquad k \in \mathcal{K};\\
& \sum_{k=1}^{K} y_{ik} \le x_i;\\
& y_{ik} \ge 0, \qquad k \in \mathcal{K}.
\end{aligned}
$$

We relabel the demand zones so that $r_1 \le r_2 \le \ldots \le r_K$. If there exist two warehouses k and k' such that $r_k = r_{k'}$, then we set $k < k'$ if and only if $s_k < s_{k'}$. Define $\psi_k = r_k - r_{k-1}$, for $k = 1, ..., K$, with $r_0 = 0$. Since there is only one warehouse, after the demands are realized, the optimal retrieval policy is to first fulfill the zone with the smallest index. The following lemma determines the retailer's optimal retrieval policy $\boldsymbol{y}^*$.

Theorem 6 (Optimal Retrieval Policy for The Upper Bound)
1. Given storage quantities $\boldsymbol{x}$ and realized demands $\boldsymbol{d}$, an optimal retrieval policy is

$$
y_{ik}^* = \min\left(x_i, \sum_{\ell=1}^{k} d_{i\ell}\right) - \min\left(x_i, \sum_{\ell=1}^{k-1} d_{i\ell}\right), \; i \in \mathcal{I}, \; k \in \mathcal{K}. \tag{55}
$$

The objective function (54) under the optimal retrieval policy $\boldsymbol{y}^*$ is

$$
\breve{W}_i\left(x_i, \boldsymbol{d}_i\right) \quad = \quad (p_i - r_K)\left[\min\left(x_i, \sum_{\ell=1}^{K} d_{i\ell}\right)\right] + \sum_{k=2}^{K} \psi_k \min\left(x_i, \sum_{\ell=1}^{k-1} d_{i\ell}\right). \tag{56}
$$

2. The objective function of the single-warehouse problem (53) can be written as

$$
u(\boldsymbol{x}) = -\sum_{i=1}^{I} \left(\rho_i + h_i T + s\right) x_i + \sum_{i=1}^{I}\sum_{t=1}^{T} h_i \breve{G}_{i,K}^t(x_i) + \sum_{i=1}^{I} (p_i - r_K)\breve{G}_{i,K}^T(x_i) + \sum_{i=1}^{I}\sum_{k=2}^{K} \psi_k \breve{G}_{i,k-1}^T(x_i), \tag{57}
$$

where $\breve{G}_{i,k}^t(x) = E\left[\min\left(x, \sum_{\tau=1}^{t}\sum_{\ell=1}^{k} \tilde{d}_{i\ell}^{\tau}\right)\right]$.

Proof : Part 1: Since there is only one warehouse, after the demands are realized, the optimal retrieval policy is to always fulfill the zone with the smallest index. Therefore, for a given product i, the demand of zone k will be fulfilled if and only if $x_i > \sum_{l=1}^{k-1} d_{il}$. The total quantity of product i that is retrieved for zones 1 to k is $\min\left(x_i, \sum_{l=1}^{k} d_{il}\right)$. Thus, the optimal quantity of product i retrieved for zone k is

$$y_{ik}^* = \min\left(x_i, \sum_{l=1}^{k} d_{il}\right) - \min\left(x_i, \sum_{l=1}^{k-1} d_{il}\right),$$

for $i = 1, \ldots, I$ and $k = 1, \ldots, K$. Based on this result, we can write

$$\breve{W}_i\left(x_i, \boldsymbol{d}_i\right) = \sum_{k=1}^{K}(p_i - r_k)\left[\min\left(x_i, \sum_{l=1}^{k} d_{il}\right) - \min\left(x_i, \sum_{l=1}^{k-1} d_{il}\right)\right] = (p_i - r_K)\left[\min\left(x_i, \sum_{l=1}^{K} d_{il}\right)\right] + \sum_{k=2}^{K}\psi_k \min\left(x_i, \sum_{l=1}^{k-1} d_{il}\right).$$

Part 2: Using Equation (56) and taking the expectation of $\breve{W}_i\left(x_i, \tilde{\boldsymbol{d}}_i\right)$, we can rewrite the objective function $u(\boldsymbol{x})$ of Problem (53) as Equation (57). ■

Similar to Section 4, Problem (53) becomes a one-stage optimization problem after the reformulation. All the randomness is captured by the functions $\breve{G}_{i,k}^{t}(x)$. Based on our assumption that $\tilde{d}_{ik}^{t}$ follows a continuous distribution, the distribution of $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$ is also continuous. Again, the gradient $\nabla u(\boldsymbol{x})$ can be determined if the derivative $\breve{G}_{i,k}^{t\prime}(x)$ can be determined. For random variables that follow some distributions, such as the normal distribution, the distribution of the summation of these random variables can be obtained analytically. For other distributions, the c.d.f. of $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$ may not be explicitly available. In that case, the c.d.f. of $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$ can be efficiently approximated with the help of demand samples and interpolation.

Since $p_i - r_K > 0$ and $\breve{G}_{i,k}^{t}(x)$ is a concave function of x for all i, k, and t, it is clear that $u(\boldsymbol{x})$ is also a concave function. The objective function $u(\boldsymbol{x})$ is separable in i: $u(\boldsymbol{x}) = \sum_{i=1}^{I} u_i(x_i)$ and $u_i(x) = -(\rho_i + h_i T + s)\, x_i + (p_i - r_K)\breve{G}_{i,K}^{T}(x) + h_i \sum_{t=1}^{T}\breve{G}_{i,K}^{t}(x_i) + \sum_{k=2}^{K}\psi_k \breve{G}_{i,k-1}^{T}(x)$. Let $\hat{f}_{i,k}^{t}(\cdot)$ and $\hat{F}_{i,k}^{t}(\cdot)$ denote the p.d.f. and the c.d.f., respectively, of the random variable $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$. Thus, we have $\breve{G}_{i,k}^{t\prime}(x) = P(\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{t} > x) = 1 - \hat{F}_{i,k}^{t}(x)$. Again, if the distribution of $\tilde{d}_{i\ell}^{t}$ is continuous, then the distribution of $\sum_{\tau=1}^{t}\sum_{\ell=1}^{k}\tilde{d}_{i\ell}^{\tau}$ is also continuous. That is, $\hat{F}_{i,k}^{t\prime}(x) = \hat{f}_{i,k}^{t}(x)$ is bounded. Then the first- and the second-order derivatives of $u_i(x)$ can be determined as follows:

$$u_i'(x) = -\rho_i - h_i T - s + (p_i - r_K)\left(1 - \hat{F}_{i,K}^{T}(x)\right) + h_i \sum_{t=1}^{T}\left(1 - \hat{F}_{i,K}^{t}(x)\right) + \sum_{k=2}^{K}\psi_k\left(1 - \hat{F}_{i,k-1}^{T}(x)\right); \tag{58}$$

$$u_i''(x) = -(p_i - r_K)\hat{f}_{i,K}^{T}(x) - h_i \sum_{t=1}^{T}\hat{f}_{i,K}^{t}(x) - \sum_{k=2}^{K}\psi_k \hat{f}_{i,k-1}^{T}(x). \tag{59}$$

Clearly, $u_i(x)$ has a Lipschitz continuous gradient. It is also clear that $u_i(x)$ is strongly concave if for $x < c$, there exists a k such that $\hat{f}_{i,k}(x) > \underline{w}$, for some positive constant $\underline{w}$. We can use the first-order methods mentioned in Section 4 to solve Problem (53) efficiently. Similar to the single-zone problem, Equation (59) also implies that if $\tilde{d}_{i,k}$ follows a discrete distribution for certain i and k, it is likely that the objective function would not have a Lipschitz continuous gradient. This implies that the objective function of the SAA formulation is likely to be non-smooth in $\boldsymbol{x}$ even for the single-warehouse problem.

D Solving Problem (29) using FISTA

The *Fast Iterative Shrinkage-Thresholding Algorithm* (FISTA) is first proposed by (Beck and Teboulle, 2009). Several variants of FISTA exist and they are usually called accelerating proximal gradient methods in the optimization literature (see, for example, Nesterov (2004)). Define a projection function $\text{Proj}_{\mathcal{X}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{w}\in\mathcal{X}}\{\|\boldsymbol{x} - \boldsymbol{w}\|\}$. We adopt the FISTA algorithm proposed by Chambolle and Dossal (2015) described as follows.

Algorithm 7 (FISTA)
Given $\gamma > 0$ and $d > 2$, initialize $\tau = 0, \boldsymbol{x}^{(0)} \in \mathcal{X}, \boldsymbol{x}^{(-1)} = \boldsymbol{x}^{(0)}$.

While the stopping criterion is not satisfied:

1. *Set* $a^{(\tau)} = \frac{\tau}{\tau+d}$.
2. $\hat{\boldsymbol{x}}^{(\tau)} \leftarrow \boldsymbol{x}^{(\tau)} + a^{(\tau)}\left(\boldsymbol{x}^{(\tau)} - \boldsymbol{x}^{(\tau-1)}\right)$, $\boldsymbol{x}^{(\tau+1)} \leftarrow Proj_{\mathcal{X}}\left(\hat{\boldsymbol{x}}^{(\tau)} + \gamma \nabla u\left(\hat{\boldsymbol{x}}^{(\tau)}\right)\right)$.
3. $\tau \leftarrow \tau + 1$.

Return $\boldsymbol{x}^{(\tau)}$.

Theoretically, we should set $\gamma = 1/L$, where L is a Lipschitz constant that can be computed explicitly using the Hessian formulas (32) and (59). In practice, we typically choose a small value for γ and tune it according to the numerical performance. According to Chambolle and Dossal (2015), if the objective function has a Lipschitz continuous gradient, then $\boldsymbol{x}^{(\tau)}, \tau = 1, 2, \ldots$, will converge to the optimal solution and the convergence rate is $O\left(1/M^2\right)$, where M is the number of iterations. In addition, if the objective function is α-strongly concave, the convergence rate can be improved to $O\left(\left(1 - \sqrt{\alpha\gamma}\right)^M\right)$. According to Chambolle and Dossal (2015), FISTA performs very well with $d = 50$. Thus, we set $d = 50$ for FISTA in this paper.

Each iteration of Algorithm 7 involves projecting a point $\boldsymbol{x}$ onto $\mathcal{X}$. We can simply project $\boldsymbol{x}_j$ for each warehouse j separately onto a simplex. Many algorithms can efficiently project a point onto a simplex (see, for example, Malozemov and Tamasyan (2016)). We adopt an approach by Boyd and Vandenberghe (2004) to the inequality constraints of our problem.

Algorithm 8 (Projecting A Point onto A Simplex)
Given $\boldsymbol{x}_j$, c_j, *and an accuracy* ϵ, *set* $\boldsymbol{x}_j \leftarrow \boldsymbol{x}_j \vee \boldsymbol{0}$.

1. *If* $\sum_{i=1}^{I} x_{ij} \le c_j$, *then return* $\boldsymbol{x}_j$. *Otherwise,* $\underline{\zeta} \leftarrow \left(\sum_{i=1}^{I} x_{ij} - c_j\right)/I$ *and* $\bar{\zeta} \leftarrow \max_{i\in\mathcal{I}} x_{ij} - c_j/I$.
2. *Do a binary search to find* $\zeta \in [\underline{\zeta}, \bar{\zeta}]$ *such that* $\left|\sum_{i=1}^{I}(x_{ij} - \zeta)^+ - c_j\right| < \epsilon$.
3. $x_{ij} \leftarrow (x_{ij} - \zeta)^+$, $\forall i \in \mathcal{I}$, *return* $\boldsymbol{x}_j$.

The projection to the feasible region is very efficient as the run time of Algorithm 8 is $O(I \log(1/\epsilon))$.

E Solving the storage problem with a single zone

Here, we examine the efficiency of Algorithm 1 for solving the storage problem (19) with multiple warehouses and a single zone. We consider $I = 500$ and 2,000 products and $J = 20$ warehouses. Each warehouse's capacity is uniformly distributed in $[0.4I, 0.5I]$. We randomly generate the order quantities $\boldsymbol{q}$ such that $\sum_{i\in\mathcal{I}} q_i = 0.8 \times \sum_{j\in\mathcal{J}} c_j$. Recall that Lemma 1 shows that if there is only one zone, the multi-period problem can be simplified to a single-period problem. Thus, without loss of generality, we consider the case with $T = 1$. For each problem instance, we consider three distributions for each $\tilde{d}_i$, $i \in \mathcal{I}$: (i) a triangular distribution with parameters $(0, U_i, \max(3U_i, 1.2q_i))$, where U_i is uniformly distributed in $[4, 20]$; (ii) an exponential distribution with mean uniformly distributed in $[1, 20]$; (iii) a log-normal distribution with mean uniformly distributed in $[1, 4]$ and standard deviation uniformly distributed in $[1, 5]$. We randomly generate the unit storage and unit retrieval costs of each warehouse such that crossing occurs.

We benchmark Algorithm 1 against two asymptotically optimal algorithms on the storage problem (19). The first algorithm SAA-LP is to solve for the storage matrix given the demand samples. Specifically, given N demand samples $d_i^{(n)}$, $i \in \mathcal{I}$, and $n = 1, ..., N$, the optimization formulation of SAA-LP is

$$\begin{aligned} \max_{\boldsymbol{x}\ge\boldsymbol{0}} \quad & -\sum_{i=1}^{I}\sum_{j=1}^{J} \rho_i x_{i,j} - \sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{i,j} + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{I} \breve{u}_i^{(n)}(\boldsymbol{x}) \\ s.t. \quad & \sum_{i=1}^{I} x_{i,j} \le c_j, j \in \mathcal{J}; \end{aligned}$$

where

$$\breve{u}_i^{(n)}(\boldsymbol{x}) = \max_{\boldsymbol{y}\geq \mathbf{0}} \quad \sum_{i=1}^{I}\sum_{j=1}^{J}(p_i - r_j)y_{i,j} - \sum_{i=1}^{I} h_i \sum_{j=1}^{J}(x_{i,j} - y_{i,j})$$

$$s.t. \quad \sum_{j=1}^{J} y_{i,j} \leq d_i^{(n)}, i \in \mathcal{I};$$

$$y_{i,j} \leq x_{i,j}, i \in \mathcal{I}, \quad j \in \mathcal{J}.$$

The second algorithm FISTA solves for the ϵ-optimal storage policy for Problem (19).

For each algorithm, we record the objective function value. For Algorithm 1, we vary the search accuracy of Algorithms 3–5 and record the run time. A higher accuracy leads to a better objective function value with a longer run time. We report the run time and the cumulative run time for each iteration for SAA-LP and FISTA respectively. Note that FISTA requires projecting a storage matrix to the feasible region of Problem (19), which is not separable in i. To our best knowledge, there is no specialized algorithm to address this problem and we solve it as a convex quadratic program in GUROBI. Based on a near-optimal solution, we construct an upper bound on the objective function $G(\boldsymbol{x}) = -\sum_{i=1}^{I}\sum_{j=1}^{J} s_j x_{ij} + \sum_{i=1}^{I}\sum_{j=2}^{J} \psi_j G_i\left(\sum_{\ell=1}^{j-1} x_{i\ell}\right)$. Since $G(\boldsymbol{x})$ is concave, we have $G(\boldsymbol{x}) \leq G(\hat{\boldsymbol{x}}) + \nabla G(\hat{\boldsymbol{x}}) \cdot (\boldsymbol{x} - \hat{\boldsymbol{x}})$, for $\hat{\boldsymbol{x}}, \boldsymbol{x} \in \mathcal{X}$. Given a candidate solution $\hat{\boldsymbol{x}}$, we have $G^* \triangleq \max_{\boldsymbol{x}\in\mathcal{X}} G(\boldsymbol{x}) \leq \max_{\boldsymbol{x}\in\mathcal{X}} \{G(\hat{\boldsymbol{x}}) + \nabla G(\hat{\boldsymbol{x}}) \cdot (\boldsymbol{x} - \hat{\boldsymbol{x}})\} \triangleq \bar{G}(\hat{\boldsymbol{x}})$. For any given $\hat{\boldsymbol{x}}$, $\bar{G}(\hat{\boldsymbol{x}})$ can be determined by solving a linear program. The optimal dual objective function value of this linear program serves as a valid upper bound on G^*, which we denote as $\bar{G}^*$.

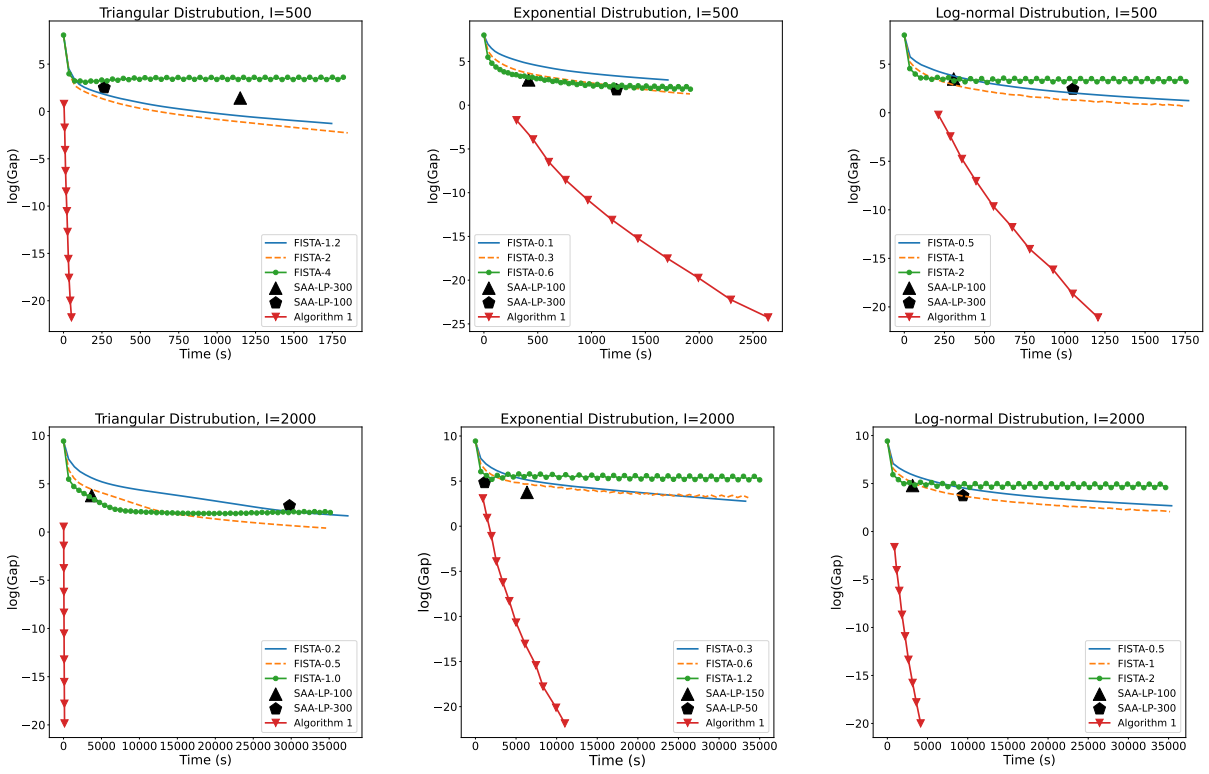


Figure 14: Performance of each heuristic on the single-zone problem

Figure 14 shows the results under the three demand distributions for $I = 500$ and 2,000. Each graph shows the results of FISTA (denoted as FISTA-γ, where γ is the step size of Algorithm 7), SAA-LP-N, where N is the sample size, and Algorithm 1. We record the run time and the gap between the upper bound $\bar{G}^*$ and the objective function value of each method. Figure 14 suggests that Algorithm 1 finds a significantly better solution in a much shorter time compared to the other methods. Furthermore, Algorithm 1 becomes more dominant as the number of products I increases from 500 to 2,000. The strong numerical performance of Algorithm 1 supports our theoretical prediction in Theorem 2 that the complexity of Algorithm 1 is linear in I.