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 $\boldsymbol{z}(\xi^*)$.

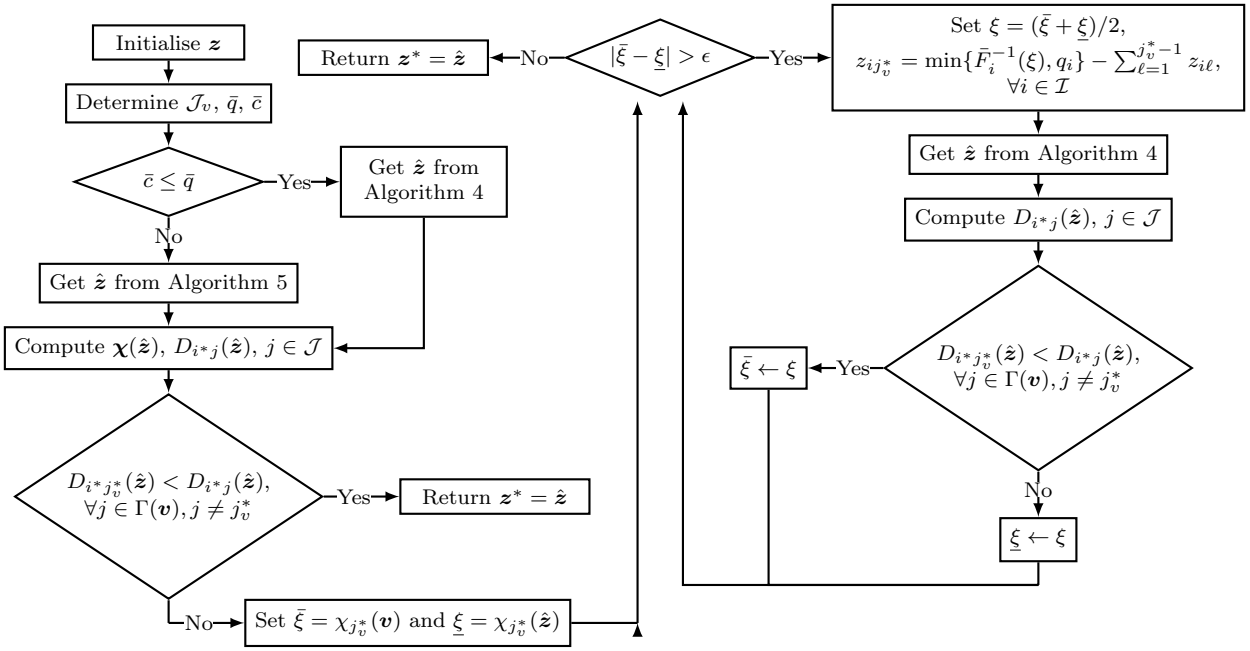

Figure 12: Illustration of Algorithm 3

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

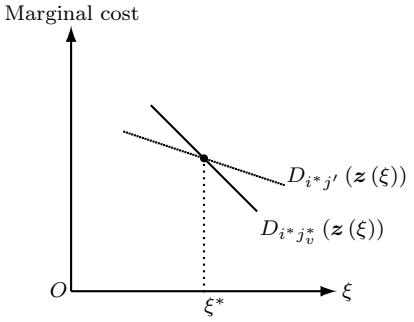

Figure 13: Illustration of crossing

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 $\bar{G}^*$.

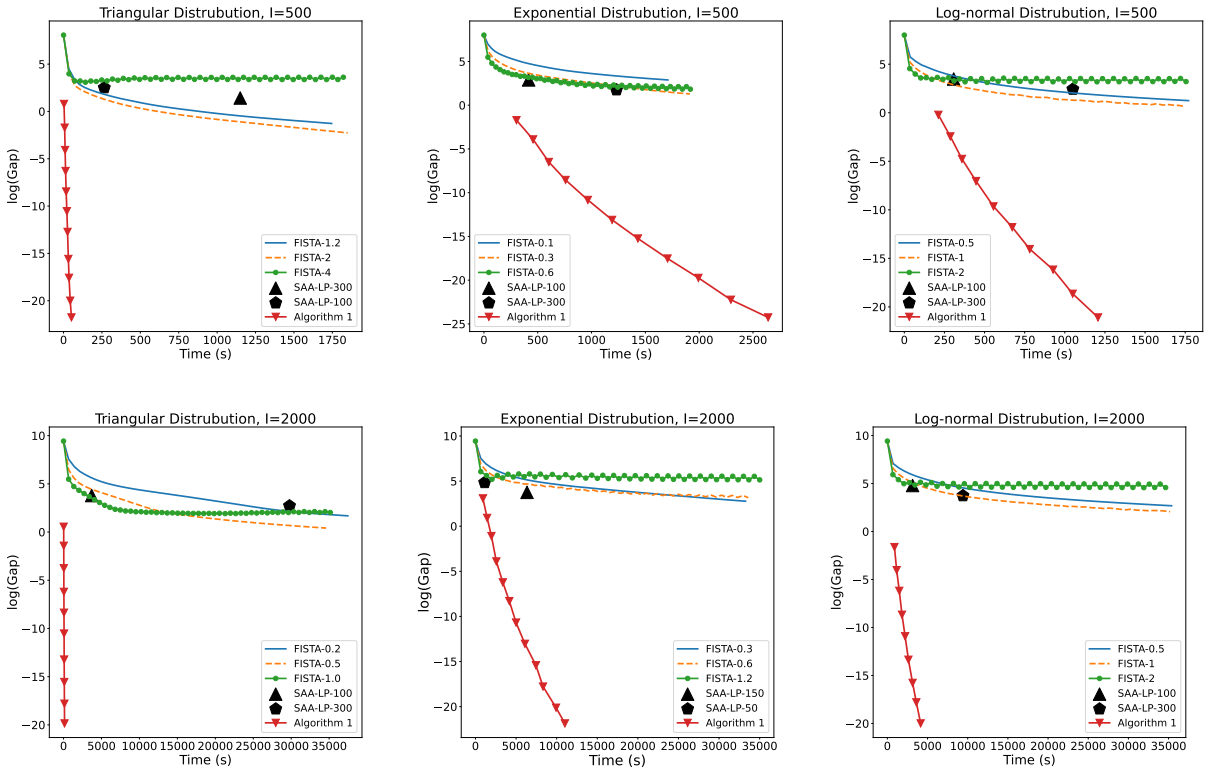

Figure 14: Performance of each heuristic on the single-zone problem

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.