Inventory Management in a Closed-Loop Supply Chain with Advanced Demand Information

Li Chen • Şafak Yücel • Kaijie Zhu

Johnson Graduate School of Management, Cornell University, Ithaca, NY 14853
Fuqua School of Business, Duke University, Durham, NC 27708
CUHK Business School, The Chinese University of Hong Kong, Shatin, Hong Kong
li.chen@cornell.edu • safak.yucel@duke.edu • kzhu@baf.msmail.cuhk.edu.hk

We study forecast and inventory control problems for rental operations in a closed-loop supply chain. In such a system (e.g., Netflix for DVD rentals), customers create online queues in a service provider’s website to indicate the next items that they would like to rent. Leveraging this advanced demand information, we propose effective forecast models for item-level returns and demands. Based on the forecast models, we formulate a multi-item inventory control problem. We prove that the \((L,U)\) policy, which was shown to be optimal in single-item settings, remains to be optimal in our multi-item setting. Moreover, we show that the problem is separable in items, so the optimal solution can be computed by solving multiple single-item problems. In practice, the service provider would also like to impose a minimum aggregate service level across all items. This aggregate service level constraint complicates the problem significantly and the \((L,U)\) policy is no longer optimal. In this case, we propose an inventory heuristic based on a single-period, multi-item formulation of our problem. This heuristic outperforms similar ones proposed in the literature and attains the optimal solution in all cases in a benchmark study.

Key words: Closed-Loop Supply Chains, Multi-Item Inventory Control, Online Rental Systems

History: Revised on October 25, 2015.
1. Introduction

Closed-loop supply chains focus on extracting additional value from the reuse of products through rentals (leasing), recycling or remanufacturing. In doing so, these supply chains employ reverse logistics to collect used products from customers. As of 2011, remanufacturing activities alone have generated more than $43B as revenue for the closed-loop supply chains in the U.S., supporting 180,000 full-time jobs (USITC 2012). In addition, these supply chains have received more attention in the last decade due largely to the legislation that holds the manufacturers responsible for the environmental footprint of their products (Ferguson and Souza 2010). Effectively managing closed-loop supply chains requires a set of non-trivial decisions at strategic (e.g., leasing versus selling), tactical (e.g., acquisition of used products) and operational (e.g., scheduling deliveries) levels (Souza 2013). In this paper we study the operational problem, i.e., inventory management of a firm that rents customers a set of non-identical products in a closed-loop supply chain. What is unique in our setting is the availability of advanced demand information as we next explain through an example.

Consider the example of online DVD rental services, which allow subscribers to rent movies or video games online and receive/return the DVD discs by mail. Notable players in this market are Netflix (for movie rentals) and GameFly (for video game and movie rentals). This kind of rental services has a unique operational process, which allows the service provider to acquire advanced demand information from its customers. The service starts when a customer creates an online rental queue on the service provider’s website. The items in the queue are then delivered to the customer by mail. The customer can keep an item for as long as desired, but there is a limit on the number of items that can be checked out at any time. To rent a new item, the customer mails a currently checked out item back to the service provider, and upon the receipt of the item, the service provider sends out the next item according to the customer’s online queue. Thus, by forecasting when a customer will return a currently checked out item, the service provider can immediately infer the following information: 1) when a new demand for the customer will occur, 2) what the new demand will be from the customer’s online queue, and 3) when the inventory of the returned item will increase by one unit. Clearly, the online queue serves as the link between the return and demand processes, and, more importantly, it provides valuable future demand information.

In this paper, we study how to leverage this advanced demand information in a closed-loop supply chain of rental services. We ask the following questions: What is the optimal inventory management policy considering the link between the return and demand processes of customers? How can practical considerations, such as an aggregate service level requirement across all items,
be incorporated into the inventory management policy?

To determine the optimal inventory control policy, we first model the return and demand processes for the closed-loop supply chain. Then, building on these processes, we formulate the inventory management problem as a multi-item, multi-period dynamic program with a finite planning horizon. The state variable is the currently held items of customers and their online queues. We find that a state-dependent \((L, U)\) policy is optimal in this setting. That is, in each period, depending on the state, there exists two thresholds, \(L\) and \(U\), for each item in the system. If the on-hand inventory for the item is less (greater, respectively) than \(L\) (\(U\), respectively), it is optimal to bring inventory level to \(L\) (\(U\), respectively); if the on-hand inventory level is in between \(L\) and \(U\), staying put is optimal. This result extends the optimality of the \((L, U)\) policy from a single-item setting (e.g., Eppen and Iyer 1997) to a multi-item setting. We further show that the problem is separable in items, so that computing the optimal policy reduces to solving multiple single-item problems.

In practice, an important consideration of a service provider is customer satisfaction, which is closely related to an aggregate service level (fill rate) across all items. To study this issue, we introduce an aggregate service level constraint into our model. This constraint complicates the problem considerably and poses two significant challenges. First, the separability result and thus the optimality of the \((L, U)\) policy may no longer hold in this more general setting. To address this challenge, we consider a single-period multi-item problem instead, which is a simplification commonly employed in the literature (e.g., Cohen et al. 1986). We observe that this formulation belongs to the class of classical knapsack problems, based on which we propose an inventory heuristic. The second challenge of the service constraint lies in its computation. In the literature, an approximate service level is considered (e.g., Cohen et al. 1989), which may be too coarse in our setting. Thus, to tackle this challenge, we propose a three-step algorithm to exactly compute the service level (see Appendix D).

The remainder of the paper is organized as follows. We review the related literature in Section 2 and introduce the model as well as our forecast and inventory control methods in Section 3. We consider a service level constraint in Section 4, followed by concluding remarks in Section 5. All proofs are presented in Appendix A.

2. Literature Review

Our paper is related to three streams of research in the literature. This first is the closed-loop supply chain literature, which is reviewed by Souza (2013). In this literature, product returns and
remanufacturing has been extensively studied. Toktay et al. (2000) adopted a closed queueing network model to study the inventory procurement problem for a closed-loop remanufacturing system. DeCroix et al. (2005) considered a multi-echelon system with product returns and showed that an echelon base-stock policy is optimal. DeCroix (2006) further characterized the optimal policy structure in a multi-echelon system with remanufacturing operations. Under certain conditions, DeCroix and Zipkin (2005) showed that an assembly system where some used components are recovered is equivalent to a series system with returns. DeCroix et al. (2009) studied a multi-item assemble-to-order system facing both demands for products and returns of components, and presented a method for computing a near-optimal base-stock policy. Our problem differs from these problems in that we consider a subscription-based rental service — it does not require remanufacturing operations, and, more importantly, the return and demand events from a subscriber are closely linked through the customer’s online queue.

The second stream constitutes the recent research on Netflix-like closed-loop supply chains. In particular, several papers took the queueing model approach and investigated inventory allocation and initial order quantity decisions for newly released items. For example, Bassamboo et al. (2009) considered the inventory stocking problem for new products in a closed rental queue. In a similar setting, Randhawa and Kumar (2008) compared the profit performance between subscription and pay-per-use. Cachon and Feldman (2011) further showed that subscription yields higher profit than pay-per-use even if subscription may cause more congestion. Besides the queueing approach, Milkman et al. (2009) investigated customer return behavior using data from an Australian online DVD rental company. More recently, Baron et al. (2011) used data for a brick-and-mortar video rental firm to study the DVD purchasing and allocation problem, and proposed several DVD demand and return forecast models. Our paper differs from the studies in this stream in two aspects. First, the traditional DVD rental operations do not have the same rich information structure as the online environment that we consider (e.g., the customer online queues). Therefore, we are able to take advantage of this rich information structure to propose new item-level demand and return forecast models that differ significantly from those in the literature (e.g., Baron et al. 2011). Second, our focus is more on in-circulation items after the new release rental peak, whereas many papers in this stream (e.g., Bassamboo et al. 2009) study the introduction of new releases.

The third stream of research related to our paper is the spare parts literature, which also deals with closed-loop supply chains. However, our forecast and inventory control methods are different than those found in the spare parts literature in three aspects. First, we leverage the unique online queue information to build a dynamic forecast model that links the item-level return and demand
processes, while simple static demand models, such as (compound) Bernoulli or (compound) Poisson, are typically used in spare parts inventory problems (e.g., Cohen et al. 1986, 1988, Janssen et al. 1998, Hopp et al. 1999, and Strijbosch et al. 2000). Second, while most papers in this stream use an approximate aggregate service level, we compute the exact service level, which gives a more accurate estimate of the realized service level. Third, we propose a solution approach based on a dynamic program, which has not been well studied in the spare parts literature.

3. Main Model and Results

We consider the daily operations of a typical closed-loop rental supply chain, as exemplified by an online DVD rental system illustrated in Figure 1. There are two levels in the supply chain: the regional shipping centers (RSCs) and the central warehouse (CWH). The RSCs fulfill demand from customers located within a geographical region, while the CWH fulfills orders from the RSCs and restocks excess inventory sent back from the RSCs.

![Diagram of daily operations of a typical online DVD rental system](image-url)

Figure 1: Illustration of daily operations of a typical online DVD rental system.

Each RSC operates according to a daily-reviewed inventory system. We assume that replenishment orders placed by the RSC in a day will arrive from the CWH the next day morning via overnight shipping. Thus, the inventory replenishment lead time is effectively zero. Besides ordering inventory from the CWH, the RSC can also send excess inventory to the CWH via overnight
shipping, so that the CWH can use them to replenish other RSCs at a later time.

The sequence of the item receiving and shipping events at the RSC can be described as follows: 1) items ordered from the CWH on the previous day are received, 2) items returned by customers are received and restocked at the RSC, 3) new rental demands from customer online queues are fulfilled with the on-hand inventory, 4) forecasts are generated for the next day customer returns and new demands, and 5) based on the updated forecasts, inventory decisions such as ordering or sending back extra copies to the CWH are determined for all items at the RSC. This five-step cycle repeats everyday.

Our main focus is on inventory control of in-circulation items. For example, in an online DVD rental system, such items correspond to the DVDs with stable demand following the peak demand of the new release period. For in-circulation items, there are usually surplus inventories in the system. Therefore, we shall assume that the CWH has ample inventory to fulfill orders from the RSCs. This is also a quite standard assumption in the inventory literature. For ease of exposition, we shall also assume that each customer can rent at most one item at any given time, with the understanding that our model can be extended to any maximum number of items to hold offered by the service provider.

Due to the multi-item, multi-customer nature of the problem, the order fulfillment process at the RSC could further complicate the analysis. Suppose that two customers request the same item from the RSC but the RSC has only one copy of this item. That is, the service provider can only fulfill the order for one customer. For the other customer, should the provider ship the next available item in the customer’s online queue, or should it expedite the order from the CWH to the customer? In the first scenario, there is no delay but the shipped item is not the customer’s top choice, while in the second scenario, the customer receives the top-choice in his online queue but may have to wait an extra day or so. Thus, there is a trade-off between speed and quality in these two scenarios. Moreover, in the first scenario, future demand of an item would depend on the on-hand inventory levels and the allocation rules of all other items. Thus, forecasting future demand under the first scenario becomes very difficult, if not intractable.

On the other hand, as will be shown below, the second scenario (i.e., the top-choice fulfillment case), which emphasizes on the quality of order fulfillment, enables us to develop a tractable demand forecast model. Moreover, under the top-choice fulfillment assumption, the resulting system service level (fill rate) also serves as a lower bound for that under the first scenario. Thus, for tractability reasons, we shall assume that order fulfillment at the RSC is based on top-choice only, i.e., if a new rental demand from a customer’s online queue cannot be met by the RSC’s on-hand inventory, the
unmet demand is expedited directly from the CWH to the customer as depicted in Figure 1.

Formally, we assume that there are a total of $I$ unique items in circulation, and there are a total of $J$ customers that are served by the RSC. Let $i$ ($i = 1, \ldots, I$) denote the index of each item and $j$ ($j = 1, \ldots, J$) the index of each customer served by the RSC. Furthermore, $W(k, j)$ corresponds to the item in the $k$-th position of customer $j$’s online queue list, where $W(0, j)$ is the item currently held by customer $j$. Also, let $p_{ij}$ represent the next-day return probability for item $i$ from customer $j$. When customer $j$ does not possess item $i$, that is, when $i \neq W(0, j)$, we set $p_{ij}$ to zero. For this key probability parameter, we provide empirical estimation methods in Appendix B.

3.1 Item-Level Return and Demand Forecast Models

In this subsection, we first present the item-level return forecast model. Building on the return model, we then derive the item-level demand forecast model.

3.1.1 Return Forecast Model

For a given item, the quantity of its returns on the next day can be calculated by considering all the possible returns from customers who possess that item. Formally, define $X_{ij}$ as a Bernoulli random variable given by

$$X_{ij} = \begin{cases} 
1 & \text{with probability } p_{ij}, \\
0 & \text{with probability } 1 - p_{ij},
\end{cases}$$

where $p_{ij}$ is the next-day return probability for item $i$ from customer $j$. Then, the total number of item $i$ to be returned on the next day, denoted by $R_i$, can be written as

$$R_i = \sum_{j=1}^{J} X_{ij}.$$  \hspace{1cm} (1)

We note that $R_i$ in the above expression is a sum of of independent but non-identical Bernoulli random variables. The following proposition gives a simple recursive procedure to compute the discrete probability distribution of $R_i$.

**Proposition 1** Let $\xi_n$ ($1 \leq n \leq N$) be an independent Bernoulli random variable with success probability $p_n$. The probability mass function of $\sum_{n=1}^{N} \xi_n$, denoted by $P_N(n)$, for $n = 0, \ldots, N$, can be computed according to the following recursive formula: for $1 \leq n \leq N$, and $1 \leq m \leq n - 1$,

$$P_n(m) = P_{n-1}(m) \cdot (1 - p_n) + P_{n-1}(m - 1) \cdot p_n,$$

$$P_n(0) = P_{n-1}(0) \cdot (1 - p_n),$$

$$P_n(n) = P_{n-1}(n - 1) \cdot p_n.$$
with the initial condition $P_0(0) = 1$.

### 3.1.2 Demand Forecast Model

Intuitively, the customer online queue provides future demand information to the service provider. We now study how to utilize this information to forecast the next-day demand for each item.

Consider customer $j$, who holds the item denoted by $W(0,j)$. If the item is returned, then it triggers a demand for the item $W(1,j)$, i.e., the top choice in customer $j$’s online queue. Thus, across all the customers, we can write the next-day total new demand for item $i$, denoted by $D_i$, as

$$D_i = \sum_{W(1,j) = i}^{W(0,j) = i} X_{W(0,j),j}.$$  \hfill (2)

The probability distribution of $D_i$ can be computed by Proposition 1 because $D_i$ is also a sum of independent but non-identical Bernoulli random variables.

We note that the above demand forecast model hinges upon the top choice fulfillment assumption. Without this assumption, future demand of an item would depend on the on-hand inventory levels and the allocation rules of all other items, making demand forecasting difficult. It is also worth commenting here that for each item $i$, the return process $R_i$ and the demand process $D_i$ are independent because the demand and return for the item $i$ come from different customers (in the online DVD rental context, a rational customer who returns an item is unlikely to put the same item in his or her next-to-watch queue). However, the return and demand for different items can be correlated due to the link between the return and demand processes. For this reason, both the item-level return and demand processes are correlated in successive periods.

### 3.2 Inventory Optimization

In this section, we present the inventory control formulation of the closed-loop supply chain under study. We consider a finite planning horizon of $T$ days and use subscript $t$ as the time index, where $t = 1, \ldots, T$. The cost structure of the inventory problem involves unit inventory holding cost denoted by $h$ and the unit shortage cost (e.g., cost of expediting from the CWH) denoted by $b$.

Furthermore, we consider two types of variable cost at the RSC. Specifically, let $c_o$ denote the per unit cost of ordering items from the CWH and $c_d$ denote the per unit cost of sending items back to the CWH. These costs represent the shipping and handling costs required for the item transfers between the RSC and the CWH.

For item $i$, we define the next-day net demand as $Z_i = D_i - R_i$, where $R_i$ and $D_i$ are defined in (1) and (2), respectively. Note that the next-day net demand can take a negative value if the
return exceeds the demand. Let \( x_{it}, y_{it} \) denote the inventory level of item \( i \) before (after, respectively) the inventory decision at the RSC in day \( t \). In addition, for the conciseness of the formulation, we use bold notation to represent the vector of associated variable across all the items and tilde notation to represent a matrix. For example, we let \( \mathbf{x}_t = (x_{1t}, \ldots, x_{It}) \) and \( \tilde{W} \) is the \( J \) by \( I + 1 \) matrix whose entries correspond to \( W(k,j) \).

We write the single-period (daily) cost function for item \( i \) as

\[
G(y_{it}, x_{it} | Z_i) = c_o \cdot (y_{it} - x_{it})^+ + c_d \cdot (x_{it} - y_{it})^+ + L(y_{it} | Z_i) = \begin{cases} 
   c_o \cdot (y_{it} - x_{it}) + L(y_{it} | Z_i) & \text{if } y_{it} \geq x_{it}, \\
   c_d \cdot (x_{it} - y_{it}) + L(y_{it} | Z_i) & \text{if } y_{it} < x_{it}, 
\end{cases}
\]

where \( L(y | Z) = h \cdot E(y - Z)^+ + b \cdot E(Z - y)^+ \), and \((z)^+ = \max\{z, 0\}\). Thus, we obtain the following formulation for the problem:

\[
V_t(\mathbf{x}_t, \tilde{W}_t) = \min_{0 \leq y_{it} \leq J} \left\{ \sum_{i=1}^{I} G(y_{it}, x_{it} | Z_i) + E[V_{t+1}(y_{it} - Z, \tilde{W}_{t+1})] \right\},
\]

with a terminal condition of \( V_{T+1} = 0 \).

We next explain the evolution of state variables. The first state variable is the on-hand inventory level, which evolves according to the realization of net demand as in the standard inventory models. The second state variable, \( \tilde{W}_t \), is the unique feature of our model as it carries the future demand information for the service provider. For each period \( t \), we define \( r_t \) as the subset of the customers who return their currently held item, i.e., \( W(0,j) \). Then, the state variable \( \tilde{W}_t \) transitions into \( \tilde{W}_{t+1} \) as follows:

\[
W_{t+1}(k,j) = W_t(k+1,j) \quad \text{for } j \in r_t \text{ and } k \geq 0,
\]

\[
W_{t+1}(k,j) = W_t(k,j) \quad \text{for } j \notin r_t \text{ and } k \geq 0.
\]

That is, we shift the items in each row (corresponding to each customer) of the \( \tilde{W}_t \) to the left by one unit for the customers that return their items, i.e., for \( j \in r_t \). The remaining rows stay unchanged.

We also note that the probability of such a transition is given as:

\[
\Pr\{X_{W(0,j),j} = 1, j \in r_t\} = \left( \prod_{j \in r_t} p_{W(0,j),j} \right) \left( \prod_{j \notin r_t} (1 - p_{W(0,j),j}) \right),
\]

where \( W(0,j) \) is the item currently held by customer \( j \), and \( \Pr\{X_{W(0,j),j} = 1, j \in r_t\} \) is the probability of observing \( r_t \) as the subset of customers who return their items. The multiplication on the right hand side represents the probability that the customers in the subset \( r_t \) return their items, whereas the customers that are not in \( r_t \) do not return theirs.
The net demand \( Z_i \) defined above has a discrete probability distribution. Thus, we shall assume the inventory decisions also take integer values throughout the paper. This integer formulation allows us to use the exact distribution of the net demand rather than an approximate continuous distribution. Also, similar to the demand pattern in spare parts systems, the daily item-level return and demand volumes are usually very low, especially for the after-peak in-circulation items that we focus on in this paper. As a result, the required inventory level for each item is also low. A continuous variable approximation thus may be too coarse. This is also the reason why many spare parts problems are formulated in integer programming (see Cohen et al. 1986, 1988, 1989, 1992).

We now present a separability result which shows that the multi-item problem presented above can be expressed as the sum of multiple single-item problems. Let

\[
V_{i,t} \left( x_{it}, \tilde{W}_t \right) = \min_{0 \leq y_{it} \leq J} \left\{ G(y_{it}, x_{it} | Z_i) + E \left[ V_{i,t+1} \left( y_{it} - Z_i, \tilde{W}_{t+1} \right) \right] \right\},
\]

with \( V_{i,T+1} = 0 \).

**Proposition 2** \( V_t \left( x_t, \tilde{W}_t \right) = \sum_{i=1}^{I} V_{i,t} \left( x_{it}, \tilde{W}_t \right) \).

Proposition 2 indicates that the multi-item problem given in (4) is separable in items, i.e., it can be represented as the sum of multiple single-item problems. This result is mainly due to the top-choice fulfillment assumption. That is, we assume that the service provider ships the top-choice item in a customer’s queue upon receiving a return from that customer regardless of the on-hand inventory. Thus, the inventory decision of an item has no impact on those of the other items, which also helps us characterize the optimal inventory policy.

**Proposition 3** A state-dependent \((L, U)\) inventory policy is optimal for the problem given in (4). That is, in day \( t \) and for item \( i \), there exists a pair of control points, \( L_{it} \) and \( U_{it} \), whose values depend on the state (i.e., \( \tilde{W}_t \)) and the optimal policy is to i) order \((L_{it} - x_{it})\) units from the CWH if \( x_{it} < L_{it} \); ii) send \((x_{it} - U_{it})\) units back to the CWH if \( x_{it} > U_{it} \); iii) do nothing otherwise.

Proposition 3 characterizes the optimal inventory control policy as an \((L, U)\) policy. We note that, in the literature, the optimality of the \((L, U)\) policy has been demonstrated for the single-item setting (see for instance Eppen and Iyer 1997). We consider a multi-item inventory control problem and show that the \((L, U)\) policy continues to be optimal. This result, together with Proposition 2, further implies that, while the demands of the items are correlated, solving the multi-item inventory-control problem under consideration reduces to solving a set of single-item problems, with the optimal policy to each being the \((L, U)\) policy. In other words, we can calculate the thresholds of the \((L, U)\) policy separately for the items over the planning horizon.
4. Service Level Constraint

In practice, to ensure a high service level, a service provider of a closed-loop supply chain often imposes an aggregate service level constraint across all items. In the literature, this issue has been considered in multi-item inventory problems (e.g., Cohen et al. 1989). In this section, we also add a service level constraint into our problem formulation. Without loss of generality, we assume that each customer in the system is an engaged user in the sense that he or she maintains a queue list with at least one item at any given time.

Based on the return and demand forecast models developed in the previous section, given an inventory level $y_i$ for each item $i$, we can define the aggregate service level (fill rate) in a period as

$$u(y) = E \left[ \sum_{i=1}^{I} \min(y_i + R_i, D_i) \left| \sum_{i=1}^{I} D_i > 0 \right. \right],$$

where $R_i$ and $D_i$ are the return and demand of the item $i$. This service level definition represents the proportion of the expected demand that can be met from the available inventory at the RSC on a given day. Let us further define

$$u_i(y_i) = E \left[ \min(y_i + R_i, D_i) \left| \sum_{i=1}^{I} D_i > 0 \right. \right],$$

which is the share of item $i$ to the aggregate service level. By linearity, we can thus express the aggregate service level as

$$u(y) = \sum_{i=1}^{I} u_i(y_i).$$

Thus, the inventory control formulation that incorporates the service level constraint is

$$V_t(x_t, \tilde{W}_t) = \min_{0 \leq y_t \leq J, u(y_t) \geq \alpha} \left\{ \sum_{i=1}^{I} G(y_{it}, x_{it}|Z_i) + E \left[ V_{t+1}\left( y_t - Z, \tilde{W}_{t+1}\right) \right] \right\},$$

where $\alpha$ is a predetermined service level (such as 95% or 99%).

This aggregate service level constraint considerably complicates the inventory control problem by posing two major challenges. First, the optimality of the $(L, U)$ policy does not carry over to this setting. This is because the separability result stated in Proposition 2 no longer holds as inventory decisions of the items are now connected via the aggregate service level constraint. To obtain baseline insights and to develop an easy-to-implement heuristic, we adopt a simplification
commonly used in the literature (e.g., Cohen et al. 1986) and solve the single-period problem:

$$\min_{0 \leq y_1, \ldots, y_I \leq J} \sum_{i=1}^{I} G(y_i, x_i|Z_i)$$

subject to $\sum_{i=1}^{I} u_i(y_i) \geq \alpha$,

$y_i$ is integer for all $1 \leq i \leq I$,

where $G(y_i, x_i|Z_i)$ and $u_i(y_i)$ are defined in (3) and (6), respectively. This single-period, multi-item formulation belongs to the class of classic knapsack problems with a real-valued constraint. To solve the problem efficiently, we define the following one-dimensional dynamic program (DP): for $0 \leq \gamma \leq \alpha$, and $1 \leq i \leq I - 1$,

$$V_i(\gamma) = \min_{0 \leq y_i \leq J} G(y_i, x_i|Z_i) + V_{i+1}(\gamma + u_i(y_i)),$$

$$V_I(\gamma) = \begin{cases} 
\min_{0 \leq y_1 \leq J} G(y_1, x_1|Z_1), \\
\text{subject to } u_I(y_I) \geq \alpha - \gamma.
\end{cases}$$

The optimal solution to the original integer program (7) can be obtained by solving for $V_1(0)$ in the above dynamic program. Because $\gamma$ is a continuous state variable, we need to discretize the range of $\gamma$ in numerical implementation. As a result, the numerical solution might not be the exact optimal solution of the original integer program. With a finer discretizing interval, we can achieve very close approximations, and the problem can be solved efficiently (i.e., in polynomial time of the discretization size).

The second challenge regarding the above formulation is that computing the service level, given in (5), is a non-trivial task due to the dependency between $\sum_{i=1}^{I} D_i$, and $D_i$ as well as $R_i$. In the literature, for similar problems, this service level is approximated by the ratio of the two expectations, i.e., $E\left[\sum_{i=1}^{I} \min(y_i + R_i, D_i)\right]/E\left[\sum_{i=1}^{I} D_i\right]$ (e.g., Cohen et al. 1989). Such an approximation might be too coarse in our setting as demand level for in-circulation items is usually low. To resolve this issue, we use the fact that the demand and return random variables are discrete and provide a three-phase algorithm to exactly compute the service level (see Appendix D). This algorithm enables us to compute the joint density of $(\sum_{i=1}^{I} D_i, R_i, D_i)$ iteratively for each item $i$ to calculate $u_i(y_i)$.

Finally, we note that the above single-period formulation allows for a more general cost structure. For example, shipments between the RSC and the CWH might incur manual labor costs for processing items during picking and restocking operations. These can be modeled as two-sided fixed costs of ordering and disposal. In this case, the cost function should be augmented to:

$$G(y_i, x_i|Z_i) = c_o \cdot (y_{it} - x_{it})^+ + c_d \cdot (x_{it} - y_{it})^+ + K_o \cdot \delta(y_i - x_i)^+ + K_d \cdot \delta(x_i - y_i)^+ + L(y_i|Z_i),$$
where $K_o$ and $K_d$ denote fixed ordering and disposal costs, respectively. Also, $\delta(z)^+ = 1$ if $z > 0$ and $\delta(z)^+ = 0$ otherwise. In the next subsection, we provide a benchmark study to test the performance of our DP based heuristic under both linear and fixed ordering/disposal costs.

### 4.1 Solution Benchmarking

In order to benchmark the performance of our dynamic program solution, we conduct numerical experiments for a small-scale problem with 5 items and 10 customers, i.e., $I = 5$ and $J = 10$. Specifically, we consider small, medium, and large parameter values for penalty-to-holding cost ratio ($b/h = 1, 5, 10$) and service level ($\alpha = 75\%, 85\%, 95\%$). In addition, we also consider three initial inventory level scenarios: 1) $x_i = 3$ for all $i$, 2) $x_i = 8$ for all $i$, and 3) $x_i = i - 1$ for all $i$. Also, we test with both linear and fixed ordering/disposal costs. Specifically, we first assume that the order and disposal costs are linear and use three levels ($c_o = c_d = 1, 10, 20$). Then, we assume that these costs are two-sided fixed costs and use the same levels as $K_o$ and $K_d$. Thus, we have a total of 162 experimental scenarios. The same randomly generated item-level return and demand probability distributions are used for each parameter scenario. Furthermore, we use a discretizing increment of 0.01 for the state variable $\gamma$ in the dynamic program.

To compute the exact optimal solution, we first evaluate the value of the objective function for all combinations of possible inventory levels. Then, we select the least cost solution that satisfies the service level constraint as the optimal solution. In addition to our DP based approach, there are other possible heuristic solutions for the original integer program. In the spare parts literature, Cohen et al. (1989) proposed a near-optimal greedy heuristic for a similar inventory optimization problem. This heuristic can be adapted into our setting with the following approach: start with a solution by minimizing (3) for each individual item without the service level constraint, and then, iteratively, increase the inventory of the item yielding the most “bang for the buck” by one unit until the service constraint is met. The most “bang for the buck” item in each iteration is the one with the least cost-to-benefit ratio, where cost-to-benefit ratio for an item refers to the ratio of the incremental cost and the incremental service level by increasing the item’s inventory by one unit.

To compare the DP based heuristic and the greedy heuristic, we define cost gap percentage as the percentage by which the cost of the approximate solution exceeds the optimal cost. The average/min/max cost gap percentages for each heuristic are given in Table 1, where each row reports the average/min/max among 27 scenarios (i.e., three $b/h$ scenarios by three initial inventory level scenarios by three ordering and disposal costs). From the table, we observe that our dynamic program solution is superior to the greedy heuristic, attaining the optimal solution in all scenarios.
Table 1: Cost gap percentage comparison of the approximate solutions.

<table>
<thead>
<tr>
<th></th>
<th>DP Based Heuristic</th>
<th>Greedy Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average Minimum Maximum</td>
<td>Average Minimum Maximum</td>
</tr>
<tr>
<td>Linear</td>
<td>75% 0.00% 0.00% 0.00%</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>Cost</td>
<td>85% 0.00% 0.00% 0.00%</td>
<td>0.02% 0.00% 0.14%</td>
</tr>
<tr>
<td>Case</td>
<td>95% 0.00% 0.00% 0.00%</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>Two-Sided</td>
<td>75% 0.00% 0.00% 0.00%</td>
<td>0.00% 0.00% 0.00%</td>
</tr>
<tr>
<td>Fixed Cost</td>
<td>85% 0.00% 0.00% 0.00%</td>
<td>0.45% 0.00% 2.88%</td>
</tr>
<tr>
<td>Case</td>
<td>95% 0.00% 0.00% 0.00%</td>
<td>2.27% 0.00% 6.59%</td>
</tr>
</tbody>
</table>

Although the greedy heuristic performs reasonably well in the linear order/disposal cost case, its performance drops significantly under the two-sided fixed costs. This performance drop is more prevalent as the problem size increases. Specifically, we provide another benchmark study in Appendix C, where we consider 10 items and 20 customers. In this case, the greedy heuristic can result in a cost increase from the optimal performance by up to 13.5%, whereas the DP based heuristic still attains the optimal solution in all cases. Intuitively, under two-sided fixed costs, any suboptimal inventory decision results in a higher performance gap, leading our DP based heuristic to significantly outperform the greedy heuristic.

5. Concluding Remarks

In this paper, we have studied the rental operations in a closed-loop supply chain. To summarize, we make the following contributions to the literature. First, we propose dynamic forecast models for item-level returns and demands in a closed-loop supply chain. To our knowledge, our forecast models are the first in the literature to leverage the advanced demand information available in such a system. Second, based on these forecast models, we establish the optimality of the \((L,U)\) policy in the multi-item inventory management problem of the closed-loop supply chain. Moreover, the optimal solution is separable in items, allowing one to solve multiple single-item problems to determine the \(L\) and \(U\) levels. Third, we incorporate an aggregate service level constraint across all items into the inventory control problem. We propose a heuristic that relies on the single-period formulation of the same problem, and unlike previous literature that relied on approximate service levels, we develop a three-phase algorithm to compute the exact service level. Finally, we conduct extensive simulation studies to evaluate the performance of our proposed forecast and inventory control methods. Our simulation results show that our dynamic forecast models are the key driver for cost performance improvement, suggesting the importance of future demand information in our
problem. The details are omitted here for brevity, but are available from the authors. Although we have focused on the forecast and inventory control for in-circulation items, our dynamic forecast model can also be applied to newly released items. Moreover, our inventory control heuristic can also be adapted for the new release inventory problems by introducing additional inventory capacity constraints for the new release items, which can be done by adjusting the feasible range of the decision variables in the problem formulation.

Acknowledgements

The authors would like to thank Jeff Hong, Xin Chen, and Huseyin Topaloglu for their insightful comments and suggestions. Thanks also go to seminar participants at the INFORMS Conference.

References


Online Appendix

A. Proofs

Proof (Proposition 1) Let \( Y_n = \sum_{i=1}^{n} \xi_i \). It is straightforward to verify that for any \( 1 \leq n \leq N \) and \( 1 \leq m \leq n - 1 \),

\[
P_n(m) = \Pr(Y_n = m) = \Pr(Y_{n-1} = m) \cdot \Pr(\xi_n = 0) + \Pr(Y_{n-1} = m-1) \cdot \Pr(\xi_n = 1)
\]

\[
= P_{n-1}(m) \cdot (1 - p_n) + P_{n-1}(m - 1) \cdot p_n.
\]

When \( m = 0 \), we have \( P_n(0) = \Pr(Y_{n-1} = 0) \cdot \Pr(\xi_n = 0) = P_{n-1}(0) \cdot (1 - p_n) \). Similarly, when \( m = n \), we have \( P_n(n) = \Pr(Y_{n-1} = n - 1) \cdot \Pr(\xi_n = 1) = P_{n-1}(n - 1) \cdot p_n \). Also, for this proposition, we provide a computational procedure in Algorithm 1. □

Proof (Proposition 2) We prove the result by induction. The result holds trivially for \( T + 1 \). Now we assume it holds for day \( t + 1 \). That is, \( V_{t+1}(x_{t+1}, \bar{W}_{t+1}) = \sum_{i=1}^{I} V_{i,t+1}(x_{it+1}, \bar{W}_{t+1}) \).

Next, consider day \( t \):

\[
V_t(x_t, \bar{W}_t) = \min_{0 \leq y_t \leq J} \left\{ \sum_{i=1}^{I} G(y_{it}, x_{it}|Z_i) + E \left[ V_{t+1}(y_t - Z, \bar{W}_{t+1}) \right] \right\}
\]

\[
= \min_{0 \leq y_t \leq J} \left\{ \sum_{i=1}^{I} \left( G(y_{it}, x_{it}|Z_i) + E \left[ V_{i,t+1}(y_{it} - Z_i, \bar{W}_{t+1}) \right] \right) \right\}
\]

\[
= \sum_{i=1}^{I} \min_{0 \leq y_t \leq J} \left\{ G(y_{it}, x_{it}|Z_i) + E \left[ V_{i,t+1}(y_{it} - Z_i, \bar{W}_{t+1}) \right] \right\}
\]

\[
= \sum_{i=1}^{I} V_{i,t}(x_{it}, \bar{W}_t),
\]

where the first equality is the definition of \( V_t(x_t, \bar{W}_t) \), and the second is due to the induction hypothesis. In the third equality, summation and minimization can be interchanged because the realization of the net demands \( Z_i \)'s (or equivalently, the returns \( R_i \)'s and demands \( D_i \)'s) is independent of the inventory decisions \( y_{it} \)'s. Hence, the result holds. □

Proof (Proposition 3) We claim that \( V_{it}(\cdot, \bar{W}_t) \) is convex. We can prove the claim by induction. We see the claim holds trivially for \( t = T + 1 \). Now we assume the claim is true for \( t + 1 \), that
is, \( V_{i,t+1}(\cdot, \tilde{W}_{t+1}) \) is convex in \( x_{i,t+1} \), and proceed to analyzing \( V_{it}(x_{it}, \tilde{W}_t) \). We substitute the expression of \( G(y_{it}, x_{it}|Z_i) \), given in (3), into \( V_{it}(x_{it}, \tilde{W}_t) \) and obtain

\[
V_{it}(x_{it}, \tilde{W}_t) = \min_{0 \leq y_{it} \leq 1} \left\{ \min_{y_{it} \geq x_{it}} \left\{ c_o(y_{it} - x_{it}) + L(y_{it}|Z_i) + E\left[V_{i,t+1}\left(y_{it} - Z_i, \tilde{W}_{t+1}\right)\right]\right\}, \right. \\
\min_{y_{it} \leq x_{it}} \left\{ c_d(x_{it} - y_{it}) + L(y_{it}|Z_i) + E\left[V_{i,t+1}\left(y_{it} - Z_i, \tilde{W}_{t+1}\right)\right]\right\}.
\]

Due to the convexity of \( L(\cdot|Z_i) \) and \( V_{i,t+1}(\cdot, \tilde{W}_{t+1}) \), we can see that the objective functions of the second and the third minimization above are both convex in \( y_{it} \). We define

\[
L_{it}(\tilde{W}_t) = \arg \min_{0 \leq y_{it} \leq 1} \left\{ c_o y_{it} + L(y_{it}|Z_i) + E\left[V_{i,t+1}\left(y_{it} - Z_i, \tilde{W}_{t+1}\right)\right]\right\}, \\
U_{it}(\tilde{W}_t) = \arg \min_{0 \leq y_{it} \leq 1} \left\{ -c_d y_{it} + L(y_{it}|Z_i) + E\left[V_{i,t+1}\left(y_{it} - Z_i, \tilde{W}_{t+1}\right)\right]\right\},
\]

where it is straightforward that \( L_{it}(\tilde{W}_t) \leq U_{it}(\tilde{W}_t) \). Thus, we can obtain

\[
V_{it}(x_{it}, \tilde{W}_t) = \begin{cases} 
    c_d(x_{it} - U_{it}(\tilde{W}_t)) + L(U_{it}(\tilde{W}_t)|Z_i) + E\left[V_{i,t+1}\left(U_{it}(\tilde{W}_t) - Z_i, \tilde{W}_{t+1}\right)\right], & \text{if } x_{ij} \geq U_{it}(\tilde{W}_t), \\
    L(x_{it}|Z_i) + E\left[V_{i,t+1}\left(x_{it} - Z_i, \tilde{W}_{t+1}\right)\right], & \text{if } L_{it}(\tilde{W}_t) < x_{ij} < U_{it}(\tilde{W}_t), \\
    c_o(L_{it}(\tilde{W}_t) - x_{it}) + L(L_{it}(\tilde{W}_t)|Z_i) + E\left[V_{i,t+1}\left(L_{it}(\tilde{W}_t) - Z_i, \tilde{W}_{t+1}\right)\right], & \text{if } x_{ij} \leq L_{it}(\tilde{W}_t).
\end{cases}
\]

Then, we can show \( \Delta V_{it}(x_{it}, \tilde{W}_t) \) is increasing in \( x_{it} \), which indicates that \( V_{it}(\cdot, \tilde{W}_t) \) is convex and thus completes the proof of the claim.

Notice that, for the formulation given in (A.1), a state-dependent \((L, U)\) policy is optimal. This is because if \( x_{ij} \geq U_{it}(\tilde{W}_t) \), \( y^*_{it} = U_{it}(\tilde{W}_t) \) and if \( x_{ij} \leq L_{it}(\tilde{W}_t) \), \( y^*_{it} = L_{it}(\tilde{W}_t) \). Finally, if \( L_{it} < x_{ij} < U_{it} \), then \( y^*_{it} = x_{it} \), proving the optimality of the \((L, U)\) policy.

\[\square\]

### B. Parameter Estimation

Our return and demand forecast models are hinged upon the model parameter \( p_{ij} \). We now discuss how to estimate \( p_{ij} \) based on customer \( j \)'s historical rental data. This probability is affected by two main factors: 1) How long it has been since customer \( j \) received item \( i \), and 2) How long on average the customer keeps an item after receiving it. Both of these factors can be estimated from the rental history of the customer.
Formally, let us assume that item $i$ has been out with customer $j$ for $\tau$ days. From customer $j$’s historical rental data, we can compute $\hat{p}_j(\tau)$, which denotes the percentage of items that were returned on the $\tau$-th day after the shipping day. This percentage is essentially an empirical estimate of the return probability on the $\tau$-th day. Thus, conditional on item $i$ not having been returned in the last $\tau$ days, the probability of customer $j$ returning it on the next day is

$$\hat{p}_{ij} = \Pr\{\text{Return on day } \tau + 1 | \text{Not returned by day } \tau\} = \frac{\hat{p}_j(\tau + 1)}{\sum_{\tau' = \tau + 1}^{\infty} \hat{p}_j(\tau')}.$$  

The above empirical estimate $\hat{p}_{ij}$ can then be substituted into our item-level return and demand forecast models to obtain the next-day return and demand forecasts for each item.

We note that the above estimation procedure is customer specific in the sense that it allows for customer heterogeneity in usage characteristics. This estimation method can be further extended to account for day-of-week effects, particularly relevant for online DVD rentals. For example, an item shipped on Monday might stay a few extra days with a customer because the customer might not watch it until the weekend and the item is more likely to be returned after the weekend. On the other hand, an item shipped on Thursday or Friday might be returned right after the weekend.

To account for this effect, we can first group customer historical rental data based on the day of the week shipping occurred, and then estimate the empirical return probability $\hat{p}_j(\tau|a)$ based on the day of week $a$ (e.g., $a =$ Monday, Tuesday, ..., Sunday). Finally, conditional on item $i$ having been out with customer $j$ for $\tau$ days since shipping day $a$, we can estimate the next-day return probability as $\hat{p}_{ij} = \hat{p}_j(\tau + 1|a)/\sum_{\tau' = \tau + 1}^{\infty} \hat{p}_j(\tau'|a)$. In addition to the day-of-week effects, we can also include additional features into the estimation. For example, for online DVD rentals, the genre information can be incorporated into the model. Instead of considering how long it takes for a customer to return his/her checked out items on average, we can estimate how long it usually takes for a customer to return items that belong to the same genre.

C. Benchmark Study Results for the Heuristics

In this appendix, we extend our benchmark study and consider 10 items and 20 customers. We use the same parameter scenarios as described in Section 4.1 but we only consider two-sided fixed costs. To compute the exact optimal solution for the integer program, we can no longer enumerate all possible set of inventory levels as this set is prohibitively large. Instead, we decompose the problem based on the three possible actions for each item, i.e., order from the CWH, send back to the CWH, and do nothing. As a result, we obtain $3^I$ subproblems. The optimal solution is the one that attains the lowest cost among all subproblems. With this computation strategy, we can
avoid the discontinuity issue caused by the two-sided fixed costs. Moreover, to facilitate an efficient search for the optimal solution, we interpolate the objective function and the service constraint with polynomial splines (using the spline function in Matlab). We code and solve the problem using the Tomlab Optimization tool in Matlab. It takes an average of 314 CPU hours to solve the integer program for each experimental scenario. In contrast, it takes less than a second on average for the DP based heuristic to complete each scenario.

In Table 2, we compare the performance of the DP based heuristic and the greedy heuristic. We observe that the DP based heuristic attains the optimal solution in all cases, significantly outperforming the greedy heuristic. We also note that the performance of the original greedy heuristic in our experiment is comparable to what was reported in a spare parts problem by Cohen et al. (1989, p. 112 Table III): They found the maximum cost gap percentage to be 16% among all numerical experiments, whereas our maximum cost gap is 13.5%.

<table>
<thead>
<tr>
<th>α</th>
<th>$K_1 = K_2$</th>
<th>DP Based Heuristic</th>
<th>Greedy Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Average</td>
<td>Minimum</td>
</tr>
<tr>
<td>75%</td>
<td>1</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>75%</td>
<td>10</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>75%</td>
<td>20</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>85%</td>
<td>1</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>85%</td>
<td>10</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>85%</td>
<td>20</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>95%</td>
<td>1</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>95%</td>
<td>10</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>95%</td>
<td>20</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

D. Case Study

In this appendix, we describe the evaluation of the service level by introducing a case study from the online DVD rentals context. We first define the following two algorithms to provide a computational procedure for Proposition 1.

D.1 Algorithms for Proposition 1

It is easy to verify that the recursive formula of Proposition 1 can be implemented by Algorithm 1 given below.
Algorithm 1 Algorithm for Proposition 1

Step 1: Set \( n \leftarrow 1 \), \( P_0(0) \leftarrow 1 \).
Step 2: Initialize. For \( 0 \leq t \leq n \), set \( P_n(t) \leftarrow 0 \).
Step 3: Update distribution. For \( 0 \leq t \leq (n - 1) \) and \( 0 \leq k \leq 1 \),
\[
P_n(t + k) \leftarrow P_n(t + k) + P_{n-1}(t) \cdot \Pr(\xi_n = k).
\]
Step 4: Set \( n \leftarrow n + 1 \). If \( n \leq N \) go to Step 2, Else Output \( P_N(t) \).

Next, we consider a rental operation in which every customer is allowed to rent \( K = 3 \) items at once. To handle this case, Algorithm 1 can be extended to the following more general format. Let \( \xi_i \) be independent Bernoulli random variables for \( 1 \leq i \leq KN \). We can express the index as \( i = K(n - 1) + m \), with \( 1 \leq n \leq N - 1 \) and \( 1 \leq m \leq K \). Then, we have \( \sum_{i=1}^{KN} \xi_i = \sum_{n=1}^{N} \sum_{m=1}^{K} \xi_{K(n-1)+m} \). Thus, the probability mass function of \( \sum_{i=1}^{KN} \xi_i \), denoted by \( P_{KN}(t) \), for \( t = 0, ..., KN \), can be computed according to the following algorithm. We note that Algorithm 1 is a special case of Algorithm 2 with \( K = 1 \).

Algorithm 2 Extension of the Algorithm for Proposition 1

Step 1: Set \( n \leftarrow 1 \), \( P_0(0) \leftarrow 1 \).
Step 2: Initialize. For \( 0 \leq t \leq Kn \), set \( P_{Kn}(t) \leftarrow 0 \).
Step 3: Update distribution. For \( 0 \leq t \leq K(n - 1) \) and \( 0 \leq k \leq K \),
\[
P_{Kn}(t + k) \leftarrow P_{Kn}(t + k) + P_{Kn-1}(t) \cdot \Pr\left(\sum_{m=1}^{K} \xi_{K(n-1)+m} = k\right).
\]
Step 4: Set \( n \leftarrow n + 1 \). If \( n \leq N \) go to Step 2, Else Output \( P_{KN}(t) \).

D.2 An Example Based on an Online DVD Rental Service

For illustrative purposes, consider a simple example of an online DVD rental system with eight items (i.e., \( I = 8 \)) indexed from \( i = 1 \) to \( 8 \) sequentially: “Avatar,” “Flywheel,” “Gladiator,” “Hugo,” “Inception,” “Snatch,” “Wall-E,” and “Up.” There are three customers (i.e., \( J = 3 \)): Emily, John, and William. In Table 3, we list the items rented by all three customers and the associated next-day return probabilities \( p_{ij} \). Each customer is allowed to rent at most three items at any given time. For instance, Emily \( (j = 1) \) keeps items of “Hugo” \( (i = 4) \), “Inception” \( (i = 5) \), and “Up” \( (i = 8) \) and she is expected to return them in the next day with probabilities of \( p_{41} = 0.12 \), \( p_{51} = 0.15 \) and \( p_{81} = 0.07 \), respectively.

Recall that in the online DVD rental system, the customers reveal the items that they want to rent next in their online queues. Let us assume the online queue lists of the three customers are as shown in Table 4 (note that only the top three positions of the queues are relevant to the demand forecast on the next day because a customer can rent at most three items at any given time). For instance, according to John’s queue list, the top three items that he wants to rent next are “Up,”
Table 3: Items Rented by Customers and Their Next-Day Return Probabilities

<table>
<thead>
<tr>
<th></th>
<th>Emily ((j = 1))</th>
<th>John ((j = 2))</th>
<th>William ((j = 3))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Index ((i))</td>
<td>(p_{ij})</td>
<td>Title</td>
</tr>
<tr>
<td>Hugo</td>
<td>4</td>
<td>0.12</td>
<td>Avatar</td>
</tr>
<tr>
<td>Inception</td>
<td>5</td>
<td>0.15</td>
<td>Inception</td>
</tr>
<tr>
<td>Up</td>
<td>8</td>
<td>0.07</td>
<td>Wall-E</td>
</tr>
</tbody>
</table>

“Hugo,” and “Snatch.”

Table 4: Customer Online Queue Lists

<table>
<thead>
<tr>
<th></th>
<th>Emily</th>
<th>John</th>
<th>William</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avatar</td>
<td>Up</td>
<td>Inception</td>
<td></td>
</tr>
<tr>
<td>Gladiator</td>
<td>Hugo</td>
<td>Wall-E</td>
<td></td>
</tr>
<tr>
<td>Flywheel</td>
<td>Snatch</td>
<td>Avatar</td>
<td></td>
</tr>
</tbody>
</table>

We can then use this mini-example and illustrate how to compute the exact service level.

D.3 Evaluation of Service Level

We now discuss how to evaluate \(u_i(y_i)\) given in (6). Let \(T = \sum_{i=1}^{I} D_i\). The challenge here is that both \(D_i\) and \(R_i\) are correlated with \(T\). Thus, we need to first derive the joint probability distribution of \((T, R_i, D_i)\) for each item \(i\). Without loss of generality, we focus on an item \(i\) below and suppress the item index \(i\) whenever no confusion arises. For a given item \(i\), we classify the entire customer base into three mutually exclusive sets:

\[
\Omega_1 = \{j: \text{customer } j \text{ who neither holds item } i \text{ nor lists it in the top three queue position}\},
\]

\[
\Omega_2 = \{j: \text{customer } j \text{ who holds item } i\},
\]

\[
\Omega_3 = \{j: \text{customer } j \text{ who lists item } i \text{ in the top three queue position}\}.
\]

It is clear that only customers in \(\Omega_2\) (in \(\Omega_3\), respectively) may contribute to the return \(R_i\) (the demand \(D_i\), respectively) of item \(i\). Let the number of customers in \(\Omega_k\) be \(N_k\), with \(k = 1, 2, 3\). Thus, \(N_1 + N_2 + N_3 = J\), because the three sets are mutually exclusive and collectively exhaustive subsets of the entire customer base. Since each customer holds three items, the total demand \(T\) in a period can be written as a sum of \(3J\) Bernoulli random variables, with each Bernoulli random variable representing a possible item return from a customer that triggers a new demand. For
instance, in the illustrative example discussed above, according to the data provided in Tables 3 and 4, if we consider the item of “Wall-E,” then, \( \Omega_1 = \{\text{Emily}\} \), \( \Omega_2 = \{\text{John}\} \), \( \Omega_3 = \{\text{William}\} \), with \( N_1 + N_2 + N_3 = J = 3 \). The total possible demand in the system is \( 3N_1 + 3N_2 + 3N_3 = 3J = 9 \).

In what follows, we present a three-phase algorithm to compute the joint distribution of \((T, R_i, D_i)\) in an iterative manner by considering the customers in the three sets defined above. The final output of this algorithm is the probability mass function of \((T, R_i, D_i)\), given as \( P_{3J}(t, r, d) \), where \( t, r \) and \( d \) are the realizations of \( T, R_i \) and \( D_i \), respectively, with \( 0 \leq t \leq 3J \), \( 0 \leq r \leq N_2 \), and \( 0 \leq d \leq N_3 \). We use the subscript of \( P_{3J} \) to indicate the number of Bernoulli random variables that have been considered to compute the joint probability.

In Phase 1, we initialize the probability distribution and consider the customers in \( \Omega_1 \). In the case of “Wall-E,” this set only contains Emily and she neither keeps “Wall-E” nor places it in her top three queue list. Thus, her returns can only affect the distribution of the total demand \( T \), but not the return and demand for “Wall-E.”

Formally, customer \( j \in \Omega_1 \) returns \( k \) many items with probability \( \Pr\left(\sum_{m=1}^{I_i} X_{mj} = k\right) \) for \( 0 \leq k \leq 3 \), increasing the total demand realization \( t \) by \( k \) units without changing return \( r \) or demand \( d \) for item \( i \). As there are \( N_1 \) customers in \( \Omega_1 \), in this phase, we process \( 3N_1 \) Bernoulli random variables, obtaining an intermediate probability mass function given as \( P_{3N_1}(t,0,0) \) for \( 0 \leq t \leq 3N_1 \). The detailed computation steps are defined in Algorithm 3. This algorithm is essentially the same as Algorithm 2 (with \( K = 3 \) and \( N = N_1 \)) given above, where we apply Proposition 1 for the \( 3N_1 \) Bernoulli random variables stemming from the possible returns of each of the three items kept by each of the \( N_1 \) customers in \( \Omega_1 \).

**Algorithm 3** Algorithm for Phase 1

1. **Step 1:** Set \( n \leftarrow 1 \), \( P_0(0,0,0) \leftarrow 1 \). Let \( j_1, \ldots, j_{N_1} \) be the index of the customers in \( \Omega_1 \).
2. **Step 2:** Initialize. For \( 0 \leq t \leq 3n \), set \( P_{3n}(t,0,0) \leftarrow 0 \).
3. **Step 3:** Update distribution. For \( 0 \leq t \leq 3(n-1) \) and \( 0 \leq k \leq 3 \),
   
   \[ \text{Increase } P_{3n}(t+k,0,0) \text{ by } P_{3(n-1)}(t,0,0) \cdot \Pr\left(\sum_{m=1}^{I_i} X_{mj} = k\right). \]

4. **Step 4:** Set \( n \leftarrow n + 1 \). If \( n \leq N_1 \) go to Step 2, Else Output \( P_{3N_1}(t,0,0) \).

For instance, for the item “Wall-E” \((i = 7)\), this Phase 1 algorithm yields a joint probability distribution for \((T, R_7, D_7)\) as follows: \( P_3(0,0,0) = 0.69564 \), \( P_3(1,0,0) = 0.26998 \), \( P_3(2,0,0) = 0.03312 \), and \( P_3(3,0,0) = 0.00126 \).

In Phase 2, we consider the customers in \( \Omega_2 \), namely, John in the case of “Wall-E.” Because John keeps a copy of “Wall-E,” there are two possibilities. First, John can return his two items other than “Wall-E” (i.e., “Avatar” and “Inception”) and contribute only to the total demand.

A7
Second, John can return his copy of “Wall-E” and contribute not only to the total demand but also the return of “Wall-E.” Hence, we need to consider these two possibilities separately.

We present a two-part algorithm for Phase 2. In the first part, we take the output of Phase 1, i.e., $P_{3N_1}(t,0,0)$, and extend it to $P_{3N_1+2N_2}(t,0,0)$ by considering $2N_2$ Bernoulli random variables associated with the returns of two non-$i$ items for each customer in $\Omega_2$. Specifically, customer $j \in \Omega_2$ returns $k$ many non-$i$ items with probability $\Pr \left( \sum_{m=1, m \neq i}^I X_{mj} = k \right)$ where $k$ ranges between 0 and 2. Consequently, the total demand, $t$ increases by $k$ units; both $r$ and $d$ remain unchanged at zero. Thus, the outcome of the first part is $P_{3N_1+2N_2}(t,0,0)$ where $0 \leq t \leq 3N_1 + 2N_2$. We note that this part is essentially the same as Algorithm 2 (with $K = 2$ and $N = N_2$) given above, where we apply Proposition 1 for the $2N_2$ Bernoulli random variables stemming from each of the two non-$i$ item returns from each of the $N_2$ customers in $\Omega_2$.

Then, in the second part, we extend the output of the first part to $P_{3N_1+3N_2}(t,r,0)$ where $0 \leq t \leq 3N_1 + 3N_2$ and $0 \leq r \leq N_2$ by considering the returns of item $i$ from the customers in $\Omega_2$. Specifically, we consider two scenarios: First, customer $j \in \Omega_2$ does not return item $i$ with probability $1 - p_{ij}$ and $t$, $r$ and $d$ all remain unchanged. Second, customer $j$ returns item $i$ with probability $p_{ij}$ causing both $t$ and $r$ to increase by one unit without changing $d$. This algorithm is given in detail in Algorithm 4.

**Algorithm 4 Algorithm for Phase 2**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Let $j_1, ..., j_{N_2}$ be the index of the customers in $\Omega_2$.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Consider returns of non-$i$ items from customers in $\Omega_2$ and set $n \leftarrow 1$.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Initialize. For $0 \leq t \leq 3N_1 + 2n$ set $P_{3N_1+2n}(t,0,0) \leftarrow 0$.</td>
</tr>
<tr>
<td>Step 4</td>
<td>Update distribution. For $0 \leq t \leq 3N_1 + 2(n-1)$ and $0 \leq k \leq 2$, [ \text{Increase } P_{3N_1+2n}(t+k,0,0) \text{ by } P_{3N_1+2(n-1)}(t,0,0) \cdot \Pr \left( \sum_{m=1, m \neq i}^I X_{mj} = k \right). ]</td>
</tr>
<tr>
<td>Step 5</td>
<td>Set $n \leftarrow n + 1$. If $n \leq N_2$ go to Step 3. Else go to Step 6 with $P_{3N_1+2N_2}(t,0,0)$.</td>
</tr>
<tr>
<td>Step 6</td>
<td>Consider returns of item $i$ from customers in $\Omega_2$ and set $n \leftarrow 1$.</td>
</tr>
<tr>
<td>Step 7</td>
<td>Initialize. For $0 \leq t \leq 3N_1 + 2N_2 + n$ and $0 \leq r \leq n$, set $P_{3N_1+2N_2+n}(t,r,0) \leftarrow 0$.</td>
</tr>
<tr>
<td>Step 8</td>
<td>Update distribution. For $0 \leq t \leq 3N_1 + 2N_2 + (n-1)$ and $0 \leq r \leq n-1$, [ \text{Increase } P_{3N_1+2N_2+n}(t,r,0) \text{ by } P_{3N_1+2N_2+(n-1)}(t,r,0) \cdot (1 - p_{ij}); ] [ \text{Increase } P_{3N_1+2N_2+n}(t+1,r+1,0) \text{ by } P_{3N_1+2N_2+(n-1)}(t,r,0) \cdot p_{ij}. ]</td>
</tr>
<tr>
<td>Step 9</td>
<td>Set $n \leftarrow n + 1$. If $n \leq N_2$ go to Step 7. Else Output $P_{3N_1+3N_2}(t,r,0)$.</td>
</tr>
</tbody>
</table>

In the case of “Wall-E,” following the Phase 2 algorithm, we can obtain $P_6(0,0,0) = 0.44988$, $P_6(1,0,0) = 0.36031$, ..., $P_6(5,1,0) = 7.7112 \times 10^{-5}$, and $P_6(6,1,0) = 2.1168 \times 10^{-6}$.

Finally, in Phase 3, we analyze $\Omega_3$. In the case of “Wall-E,” this set consists of William who places “Wall-E” to the second position in his queue list. Thus, if William returns at least two of the items that he currently keeps, he generates a unit of demand for “Wall-E” and also, he increases
the total demand by one unit. If he returns less than two units, his action only affects the total demand, but not the demand for “Wall-E.”

In the Phase 3 algorithm, we build upon the output of Algorithm 4 and extend the probability distribution to its final form of $P_{3,l}(t, r, d)$ for $0 \leq t \leq 3J$, $0 \leq r \leq N_2$ and $0 \leq d \leq N_3$. In general, consider customer $j$ in $\Omega_3$, who places item $i$ in the $l(i, j)$-th position in his/her online queue list. Whether the demand for item $i$ is triggered or not depends on whether the quantity of returns from the customer is at least $l(i, j)$ units. Thus, if the customer returns $k$ units for $l(i, j) \leq k \leq 3$ with probability $\Pr \left( \sum_{m=1}^{I} X_{mj} = k \right)$, $t$ and $d$ are increased by $k$ and $1$ units, respectively, while $r$ remains the same. Otherwise if the customer returns $k$ units for $0 \leq k \leq l(i, j) - 1$ with probability $\Pr \left( \sum_{m=1}^{I} X_{mj} = k \right)$, only $t$ is increased by $k$, while $r$ and $d$ remains unchanged. Note that under both cases, $r$ does not change. Thus, we will fix $r$ value first and iterate through $t$ and $d$. The detailed computation steps are defined in Algorithm 5.

**Algorithm 5 Algorithm for Phase 3**

1. Set $j_1, \ldots, j_{N_3}$ be the index of the customers in $\Omega_3$ and set $n \leftarrow 1$ and $r \leftarrow 0$.
2. Initialize. For $0 \leq t \leq 3N_1 + 3N_2 + 3n$ and $0 \leq d \leq n$, set $P_{3N_1+3N_2+3n}(t, r, d) \leftarrow 0$.
3. Update distribution. For $0 \leq t \leq 3N_1 + 3N_2 + 3(n - 1)$, $0 \leq d \leq n - 1$ and $0 \leq k \leq 3$, if $k \leq l(i, j_n) - 1$

   Increase $P_{3N_1+3N_2+3n}(t + k, r, d)$ by $P_{3N_1+3N_2+3(n-1)}(t, r, d) \cdot \Pr \left( \sum_{m=1}^{I} X_{mj_n} = k \right)$;

   Else

   Increase $P_{3N_1+3N_2+3n}(t + k, r, d + 1)$ by $P_{3N_1+3N_2+3(n-1)}(t, r, d) \cdot \Pr \left( \sum_{m=1}^{I} X_{mj_n} = k \right)$.

4. Set $r \leftarrow r + 1$. If $r \leq N_2$ go to Step 2, Else set $r \leftarrow 0$ and go to Step 5.
5. Set $n \leftarrow n + 1$. If $n \leq N_3$ go to Step 2, Else Output $P_{3N_1+3N_2+3N_3}(t, r, d)$.

Finally, by performing the three-phase algorithm above, we can compute the joint probability distribution of $(T, R_i, D_i)$ for each item $i$. Then, service level share of each item $i$, $u_i(y_i)$ can be computed as follows:

$$u_i(y_i) = \sum_{t=1}^{3J} \sum_{r=0}^{N_2} \sum_{d=0}^{N_3} \frac{\min(y_i + r, d)}{t} \cdot \frac{P_{3,l}(t, r, d)}{1 - P_{3,l}(0, 0, 0)}.$$ 

For example, in the case of “Wall-E” ($i = 7$), we can obtain the joint probability distribution as $P_3(0, 0, 0) = 0.31290$, $P_3(1, 0, 0) = 0.37218$, ..., $P_3(8, 1, 1) = 1.68473 \times 10^{-7}$, and $P_3(9, 1, 1) = 2.70950 \times 10^{-9}$. Given an inventory level $y_7 = 1$ for the item, the resulting service level share from “Wall-E” is given by $u_7(y_7 = 1) = 0.31713$. 

A9