Hybrid Replenishment and Dispatching Policy with Deteriorating Items for VMI: Analytical Model and OSJCA Approach

Shao-Fu Du *, Liang Liang, Jian Yang and Hao Qiu
School of Management, University of Science and Technology of China, 96, Jinzhai Road, Hefei, Anhui 230026, P. R. China
Accepted in May 2006
Available online

Abstract

This paper focuses on stock replenishment and shipment scheduling with deteriorating item for vendor-managed inventory systems. In consideration that demand process follows a typical Poisson process, hybrid replenishment and dispatching model with correction for VMI is proposed, and a combined solving approach named OSJCA is presented.

As approximation of expected dispatching number per replenishment cycle is involved, a varying correction factor in the left half-open interval of $(0, 1]$ is introduced into the model. Therefore, a series of near-optimal hybrid policies rather than an exactly optimal one, under which average long-run cost is fairly close to the minimum, are obtained by solving a series of programming problems. In order to pick the nearest optimal one out, simuPolicy, a special simulation program for tracking the effect of any hybrid policy for any number of replenishment cycles, is developed in the environment of MATLAB. According to the suggested discriminant rules, the nearest optimal policy can be obtained by comparing results obtained through optimization and simulation one by one. Moreover, a numerical example has shown that the disparity between results from model and simulation under the nearest optimal policy is small or even negligible, thus the validity of our model and approach is confirmed.

Keywords: Supply chain; Vendor-managed inventory (VMI); Deteriorating item; Replenishment and dispatching policy; Shipment consolidation; Poisson process; Optimization-simulation jointly correcting approach (OSJCA).

1. Introduction

In the circumstance of supply chain, it is difficult to control and coordinate production, inventory and shipment. In a traditional supply chain, partners usually make decisions separately because of their independency and quasi-autonomy. They are only interested in the maximization of their own profits and/or minimization of their own costs, which results in Bullwhip effect (Lee et al., 1997), demonstrated by MIT beer game (Kimbrough et al., 2002). For lack of control and coordination, a good many supply chain run in an inefficient way. Consequently, supply chain coordination has drawn more attention in recent years.

**Vendor-managed inventory** (VMI) suggests a new paradigm to resolve problem mentioned above. In many literatures (e.g., Valentini and Zavanella, 2003; Hung et al., 1995; Fagel, 1996), VMI is known as **consignment inventory**. In a VMI-consignment system, upstream members of supply chain, the “consignors”, deliver their goods to their direct downstream members, the “consignees”, for use or sale, with payment to the consignors only after the actual use/sale. VMI implies that vendors must take on extra responsibility to maintain inventory on retailers’ side (Dong and Xu, 2002; Aviv and Federgruen, 1998) and make appropriate continuous replenishment plan according to demand forecast. Therefore, vendors are confronted with both stock replenishment problem and shipment scheduling problem simultaneously. Higginson and Bookbinder (1995) have shown that significant savings can be achieved by properly incorporating shipment consolidation with stock replenishment decision in a VMI system. However, how to make joint decision of these two aspects to achieve global optimum?

Before VMI was proposed, the problem of shipment consolidation has come into researchers’ sight (e.g., Hall, 1987; Higginson and Bookbinder 1995). The so-called shipment consolidation refers to the delivery decision of consolidating several smaller shipments or orders into a larger load in order to achieve economy of scale on shipment. That is, demands are not satisfied immediately, but rather are shipped in batches of consolidated loads (Çetinkaya and Lee, 2000, p. 220). Higginson and Bookbinder (1995) categorize shipment consolidation into two distinct types: quantity-based and time-based dispatching policies. A quantity-based policy dispatches whenever the size of
an accumulated load reaches a predetermined quantity (planned dispatching load), while a time-based policy dispatches in every period of a predetermined time interval (planned dispatching period). Both types imply that each decision of dispatching must satisfy/clear all the outstanding demands. Çetinkaya and Lee (2000) is the pioneering paper on joint decision of stock replenishment and shipment scheduling in a VMI system. Under the idealized assumption that inventory replenishment lead time is negligible (Çetinkaya and Lee 2000, p. 221, para. 4, line 3), they have developed an integrated time-based replenishment and dispatching model with stochastic demand for a VMI vendor. This model focuses on the problem of how to determine the optimal pair of order-up-to point Q and planned dispatching period T to minimize average long-run cost. Çetinkaya and Bookbinder (2003) have discussed both time- and quantity-based dispatching models in either way of private carriage or common carriage respectively. Ching and Tai (2005) have analyzed quantity-time-based/hybrid replenishment and dispatching policy in a VMI system. They formulate the upper and lower bounds of expected dispatching number per replenishment cycle, and set the lower bound as its approximate value. From this approximate value, they derive the optimal combination of order-up-to point Q, planned dispatching load q and planned dispatching period T. Chen, Wang and Xu (2005) have made comparison among time-based, quantity-based and hybrid policies with cost-effectiveness concerned. The result shows that quantity-based model often outperforms its time-based counterpart, and hybrid policy is superior to the time-based while inferior to the quantity-based. However, survey by Jackson (1985) has shown that the adoption ratio of time-based policy is much larger. Çetinkaya and Lee (2000) have explained such phenomenon. They deem that time-based policy seems superior to quantity-based policy in the sense of several strategic considerations, such as customer satisfaction (CS), quick response (QR), and efficient customer response (ECR), etc. It's for these reasons that a term of Time-Definite Delivery (TDD) or Delivery Time Guarantee (DTG) often appears in a transport contract between vendor and third party logistics supplier. Hybrid policy is a tradeoff between the time-based and the quantity-based. The advantage of hybrid policy is that both cost-effectiveness and the aforementioned strategic performances are taken into account simultaneously.

All the above literatures proceed on an explicit idealized assumption of zero replenishment lead time, and an implicit assumption, goods do not deteriorate all the time. In fact, there are often deteriorating items in bills of material (BOM). Manna and Chaudhuri (2006) defined deterioration as “decay or damage such that the item cannot be used for its original purpose”. “Food items, photographic films, drugs, pharmaceuticals, chemicals, electronic components and radioactive substances are some examples of items in which sufficient deterioration may occur during the normal storage period of the units and consequently this loss must be taken into account while analyzing the inventory system (Chakrabarty et al., 1998)”. The influence of deterioration on production or inventory management is remarkable (Ghare and Schrader, 1993; Yang and Wee, 2000, 2002; Katagiri et al., 2003; Kim et al., 2003). Therefore, stock replenishment and shipment scheduling problem for deteriorating items under VMI is fairly worth discussing, which is an unexplored area.

This research will follow the assumption of zero lead time and discuss hybrid replenishment and dispatching policy with deteriorating items in a VMI system. Here, it's necessary to clarify that maintaining a certain level of inventory is essential (similar comments can be found in Ganeshan, 1999; Barbaro, 2004; Mitra and Chatterjee, 2004 etc.) regardless of zero replenishment lead time, because the fixed setup cost of replenishment in the problem concerned makes the vendor to pursue replenishment scale economy. Researches on EOQ have shown that it's a “trade-off between the constant setup costs and the variable holding costs (Berman and Perry, 2006, p. 255)”. Moreover, this assumption is also widely adopted by researches on deteriorating items, e.g. Chakrabarty et al. (1998), Chang (2004), Chung and Lin (2001), Manna and Chaudhuri (2006), Moon et al. (2005), etc.

However, it is somewhat too difficult to derive an explicit and precise expression by cost analysis, because when deterioration is taken into account there must be some transcendental equations in the model, which make it much too complex to reduce some key middle formulas involving operations like k-fold convolution, such as Equation (6) in Çetinkaya and Lee (2000). In this paper, the expected dispatching number per replenishment cycle is estimated approximately, and a correction factor is introduced to correct this approximation. Based on such corrected approximation, the corresponding hybrid replenishment and dispatching model is proposed. Moreover, a new approach, OSJCA (optimization-simulation jointly correcting approach), is developed to find one of the nearest optimal policies. OSJCA is the combination of model-based optimization and policy-based simulation.

The rest of the paper is organized as follows. In Section 2, the problem we are concerned is characterize. In Section 3, assumptions and notations involved in our analytical model are specified. In Section 4, hybrid replenishment and dispatching model with deteriorating items for VMI is proposed with a cost analysis, and then OSJCA approach is introduced. Section 5 presents a numerical example. In Section 6, sensitivity analysis is provided. And, concluding remarks will be given in Section 7.

2. Problem Characteristics

A typical VMI system considered in stock replenish-
The lead time of inventory

\[ \text{The time interval between the} \] (\text{Fixed setup cost of replenishment.} \]

\[ \text{The dispatching number per replenishment cycle. It} \]

\[ \text{Fixed setup cost of dispatching.} \]

\[ \text{The deterioration rate of in-stock inventory,} \]

\[ \text{order-up-to point,} \]

\[ \text{time interval} \]

\[ \text{son process, i.e. the size of accumulated demand in any} \]

\[ \text{ations, (1) Demand accumulating process is a typical Pois-} \]

\[ \text{son process.} \]

\[ \text{3. Assumptions and Notations} \]

Our model is established on the following assumptions, (1) Demand accumulating process is a typical Poisson process, i.e. the size of accumulated demand in any time interval \( t \) is only related to the size of \( t \) while independent of the start instant, and follows the Poisson distribution with mean \( \lambda t \). (2) The lead time of inventory replenishment is negligible, i.e. the replenishment order can be satisfied by the manufacturer instantaneously. (3) Items in stock are deteriorating gradually. The deterioration rate is denoted by \( \theta \), \( I(t) = -\theta I(t) \). Deteriorated items have no residual value.

The main parameters and notations involved in the analytical model are specified as follows.

\[ N(t) : \text{The size of accumulated demand in time interval} \]

\[ N(t) \sim P(\lambda t), \text{ where } N(t) : t \geq 0 \text{ is a typical Poisson process.} \]

\[ I(t) : \text{Undeteriorated in-stock inventory level on the ven-} \]

\[ \text{dor’s side at time point} \]

\[ \theta : \text{The deterioration rate of in-stock inventory,} \]

\[ I(t) = -\theta I(t) \]

\[ K : \text{The dispatching number per replenishment cycle. It} \]

\[ \text{is not a deterministic but a random variable.} \]

\[ t_i : \text{The instant of the} i^{th} \text{dispatching in a replenishment} \]

\[ \text{cycle for } i = 1, 2, \ldots, K. \]

\[ h : \text{Inventory holding cost per unit per unit time.} \]

\[ s_D : \text{Fixed setup cost of dispatching.} \]

\[ c_D : \text{Unit transportation cost.} \]

\[ s_R : \text{Fixed setup cost of replenishment.} \]

\[ c_R : \text{Unit replenishment cost.} \]

\[ w : \text{Customer waiting cost per unit per unit time.} \]

\[ (Q, q, T) : \text{General hybrid replenishment and dispatching} \]

\[ \text{policy, where} \]

\[ Q \text{ denotes the order-up-to point,} \]

\[ q \text{ is the planned dispatching load,} \]

\[ T \text{ is the planned dispatching period. The typical process under such a policy can be described as follows.} \]

\[ V \text{ regularly makes a dispatching decision every} \]

\[ \text{planned dispatching period} \]

\[ \text{But if the outstanding demands are accumulated up to the level of the} \]

\[ \text{planned dispatching load} \]

\[ \text{q before the end of a} \]

\[ \text{planned dispatching period,} \]

\[ V \text{ dispatches a load immediately.} \]

\[ \text{On the verge of dispatching,} \]

\[ V \text{ must determine} \]

\[ \text{whether its inventory should be replenished or not. If} \]

\[ \text{replenishment is necessary, what lot size is appropri-} \]

\[ \text{ate? A replenishment order is issued only if the out-} \]

\[ \text{standing demands cannot be cleared with the undete-} \]

\[ \text{riorated in-stock inventory. The lot size of replenish-} \]

\[ \text{ment will be able to clear the shortage and bring the} \]

\[ \text{inventory level back to} \]

\[ Q. \]
\[ I(t) = I(t_{i-1})e^{-\theta(t-t_{i-1})} \]  

From Equation (1), the segmented function of inventory level in the whole replenishment cycle can be derived as follows.

\[ I(t) = \begin{cases} 
Qe^{-\theta t}, & 0 \leq t < t_i \\
Qe^{-\theta t_i} - \sum_{i=1}^{K} N_i e^{-\theta(t_{i-1})}, & t_i \leq t < t_{i+1} \\
Qe^{-\theta t_{i+1}} - \sum_{i=1}^{K} N_i e^{-\theta(t_{i+1}-t_{i})} - \sum_{j=1}^{i-1} e^{-\theta(t_{j+1}-t_{j})}, & t_{i+1} \leq t < t_k 
\end{cases} \]  

Let \( t_k \) denote the critical time point just before the \( K^{th} \) dispatching, then we have

\[ I(t_k) = Qe^{-\theta t_k} - \sum_{i=1}^{K} N_i e^{-\theta(t_{i-1})} \]  

and thus

\[ K = \inf \left\{ k \mid Qe^{-\theta t_k} - \sum_{i=1}^{K} N_i e^{-\theta(t_{i-1})} < N_k \right\} \]

\[ = \inf \left\{ k \mid \sum_{i=1}^{k} N_i e^{\theta t_i} > Q \right\} \]  

Let \( p_T \) be the probability of dispatching a load according to the planned period, i.e. the time interval of dispatching is \( T \). Accordingly, the probability of dispatching according to the planned load must be \( (1 - p_T) \). We have

\[ p_T = P(\Delta t = T) = P(N(T) < q) = \sum_{i=0}^{\left\lfloor \frac{T}{\theta} \right\rfloor} \frac{(\lambda T)^i}{i!} e^{-\lambda T} \]  

By definition, \( N_i \) denotes the outstanding demands accumulated within the \( i^{th} \) dispatching period, and the dispatching load must clear all these accumulated demands. Since \( \tilde{q} \) is the expected lot size of dispatching, \( \tilde{q} \) must be equal to the mathematical expectation of \( N_i \), that is

\[ \tilde{q} = E(N_i) = q(1 - p_T) + E[N(T)]p_T \]

\[ = q(1 - p_T) + \lambda T p_T \]  

\( \{N(t) : t \geq 0\} \) is a typical Poisson process according to assumption, as is equivalent to that the arrival time interval per unit demand follows exponential distribution with parameter \( \lambda \), whose mean equals \( \lambda^{-1} \). Consequently, the expected time interval of dispatching is

\[ \overline{T} = E(\Delta t) = \tilde{q} \lambda^{-1} \]

Meanwhile, \( t_i = \sum_{j=1}^{i} \Delta t_j \), and \( \Delta t_j (j = 1, \cdots, K) \) are independent and identically distributed (iid) random variables, so we have

\[ E(t_i) = \sum_{j=1}^{i} E(\Delta t_j) = i\tilde{q} \lambda^{-1} \]

If \( i = K \), \( t_k \) represents a whole replenishment cycle, i.e. \( T_{RC} = t_k \), thus

\[ E(T_{RC}) = E(t_k) = E(K) \times \overline{T} = \tilde{q} \lambda^{-1} E(K) \]  

Now, turn to the problem of formulating the expected dispatching number per replenishment cycle, \( E(K) \).

From Equation (4), \( \sum_{i=1}^{K} N_i e^{\theta t_i} > Q \geq \sum_{i=1}^{K} N_i e^{\theta t_i} \) is got. In the long run, it’s reasonable to substitute \( K, N_i \) and \( t_i \) with their expected values respectively. \( E(N_i) = \tilde{q} \) and Equation (8) is known. Therefore \( \tilde{q} \sum_{i=1}^{K} e^{\theta t_{i+1}} > Q \geq \tilde{q} \sum_{i=1}^{K} e^{\theta t_i} \) is held in the expected case. So the bounds of \( E(K) \) can be given by

\[ \ln \left[ \frac{(Q + \tilde{q} - Q e^{\theta t_{K+1}})}{\tilde{q} \lambda^{-1}} \right] - \ln \tilde{q} + 1 \geq E(K) > \ln \left[ \frac{(Q + \tilde{q} - Q e^{\theta t_1})}{\tilde{q} \lambda^{-1}} \right] - \ln \tilde{q} \]

or equivalently,

\[ E(K) = \frac{\ln \left[ \frac{(Q + \tilde{q} - Q e^{\theta t_{K+1}})}{\tilde{q} \lambda^{-1}} \right] - \ln \tilde{q} + \varepsilon}{\tilde{q} \lambda^{-1}} \]

where \( \varepsilon \) is a correction factor, \( \varepsilon \in (0, 1] \).

4.2 Cost Analysis

In this section, an approximate average long-run cost in a VMI system under the hybrid replenishment and dispatching policy \((Q, q, T)\) with deterioration considered is analyzed. It is known that a typical Poisson process is a renewal process as well, so the renewal theory can be applied in the following analysis. In terms of the renewal reward theorem, the average long-run cost is given by

\[ C(Q, q, T) = E\left( \frac{C_{RC}}{E(T_{RC})} \right) = \frac{E(H_{RC} + R_{RC} + D_{RC} + W_{RC})}{\tilde{q} \lambda^{-1} E(K)} \]
Now, the mathematical expectations of inventory holding cost, replenishment cost, dispatching cost and customer waiting cost per replenishment cycle can be formulated respectively as follows.

1. From the segmented function of inventory level in Equation (2), the expected inventory holding cost per replenishment cycle is given by

\[
E(H_{RC}) = \sum_{j=0}^{K-1} \int h Q e^{-\theta E(t_j)} - \tilde{q} \sum_{j=1}^{K-1} e^{-\theta E((j-1)\tilde{q})} \int e^{E(t_j)} e^{-\theta [t_j-E(t_j)]} dt
\]

\[
= \theta^{-1} [1 - e^{-\theta \tilde{q} \epsilon}] \left\{ \sum_{j=1}^{K-1} \tilde{q} e^{-\theta \tilde{q} \epsilon(j-1)} - \sum_{j=1}^{K-1} \tilde{q} e^{-\theta \tilde{q} \epsilon(j-1)} \right\}
\]

\[
= \theta^{-1} [Q - \tilde{q} E(K) + \tilde{q} \frac{1 - e^{-\theta \tilde{q} \epsilon}}{1 - e^{-\theta \tilde{q} \epsilon}}]
\]

(13)

2. Only one replenishment decision is made in every single replenishment cycle, and the inventory level would be replenished up to \(Q\). Thereby the lot size per replenishment must be

\[
Q_R = Q - I(\epsilon) + N = Q(1 - e^{-\theta \epsilon}) + \sum_{j=1}^{K} \epsilon N e^{-\theta (\epsilon - \epsilon)}
\]

(14)

whose expected value is

\[
E(Q_R) = Q + \tilde{q} \frac{1 - e^{-\theta \tilde{q} \epsilon}}{1 - e^{-\theta \tilde{q} \epsilon}}
\]

(15)

Consequently, the expected replenishment cost per replenishment cycle is given by

\[
E(R_{RC}) = s_R + c_R (Q + \tilde{q} \frac{1 - e^{-\theta \tilde{q} \epsilon}}{1 - e^{-\theta \tilde{q} \epsilon}})
\]

(16)

3. The expected dispatching cost per replenishment cycle is

\[
E(D_{RC}) = (s_D + c_P \tilde{q}) E(K)
\]

(17)

4. In order to formulate the expected customer waiting cost per replenishment, the following expected situation is considered. As mentioned above, the expected value of the arrival time interval per unit demand, the dispatching load and the time interval of dispatching can be denoted by \(\lambda^{-1}\), \(\tilde{q}\), and \(\tilde{T} = \tilde{q} \lambda^{-1}\) respectively, and a replenishment cycle consists of \(K\) such dispatching periods. That is to say, if considered in an expected case, a replenishment cycle can be partitioned into \(K\) dispatching periods, and a dispatching period can also be partitioned into \(\tilde{q}\) time spans, at the end of which one unit demand arrives. Therefore, the expected customer waiting cost per replenishment cycle is given by

\[
E(W_{RC}) = E(K) \cdot \sum_{i=0}^{K-1} \frac{\tilde{q}(\tilde{q} - 1) E(K)}{2\lambda}
\]

(18)

Substituting Equations (13), (16), (17) and (18) into Equation (12), we have

\[
C(Q, \tilde{q}) = \frac{\theta s_R + (h + b \tilde{q})}{\ln (Q + \tilde{q})} \left( \frac{1 - e^{-\theta \tilde{q} \epsilon}}{1 - e^{-\theta \tilde{q} \epsilon}} \right)
\]

(19)

4.3 OSJCA: Optimization-simulation Jointly Correcting Approach

Strictly to say, the correction factor \(\epsilon\) is a function of \(Q\) and \(\tilde{q}\). Anyway, no matter what values are assigned to \(Q\) and \(\tilde{q}\), \(\epsilon\) always lies in the left half-open interval \((0, 1]\). In other words, The combined influence of \(Q\) and \(\tilde{q}\) on \(\epsilon\) is finite, and the contribution of \(\epsilon\) to \(E(K)\) is small as well. Therefore, it’s feasible to consider \(\epsilon\) as a constant when \(E(K)\) is much larger than \(\epsilon\). Under such condition, \(C(Q, \tilde{q})\) is strictly convex for positive \(Q\) and \(\tilde{q}\), hence there exists a unique global optimal pair \((Q', \tilde{q}')\) to minimize the objective cost function. Theoretically speaking, \((Q', \tilde{q}')\) can be obtained by solving the first-order conditions of Equation (19), yet they both are transcendental functions therefore difficult to be solved by using general mathematical approach. Alternatively, computer is used to solve a set of programming problems: \(\{\min C(Q, \tilde{q}) \mid Q \geq 0, \tilde{q} \geq 0\}\) with \(\epsilon\) varying in \((0, 1]\). In theory, any value in the interval can be assigned to \(\epsilon\), however it is infeasible and somewhat unnecessary. Generally, a sequence of discrete values is selected in terms of a certain step size, which has a negative correlation with the precision of result. For instant, let the step size be 0.1, then 10 programming problems need solving with elements of \(\{0.1, 0.2, 0.3, ..., 1\}\) assigned to \(\epsilon\) one by one. The less the step size is set, the more programming problems arise, hence the final result would be closer to optimum. There are only two decision variables in these programming problems, and the solutions are certain to be existing and unique, so it’s unnecessary for us to develop a new algorithm to solve these problems. They can be solved by the existing tools, such as EXCEL, Mathematica and MATLAB etc., in polynomial time. Then, a near-optimal set of \((Q', \tilde{q}')\) with the same number as the selected discrete values of \(\epsilon\) is obtained.

For each \((Q, \tilde{q})\) in the near-optimal set, all the hybrid policies \((Q, q, T)\) that follow Equation (5) and Equation (6) are equivalent in the sense of cost-effectiveness. But how can a choice be made from these equivalents? If partial shipment is allowable (which is feasible to the industrial practices), a good choice is to set \(q = \tilde{q}\), and accordingly the corresponding \(T = \tilde{q} \lambda^{-1}\), in respect that in this circumstance,

1. The expected dispatching time point according to planned dispatching period is coincident with that
according to planned dispatching load;

2. No matter what value \( p_T \) is equal to, Equation (6) is always satisfied.

Moreover, we note that
\[
\lim_{p_T \to +1} T = \lim_{p_T \to +1} \frac{\tilde{q} - q(1 - p_T)}{\lambda p_T} = \tilde{q} \lambda^{-1}
\]
\[
\lim_{p_T \to +0} q = \lim_{p_T \to +0} \frac{\tilde{q} - AT p_T}{1 - p_T} = \tilde{q}
\]

As \( p_T \) denotes the probability of dispatching a load according to the planned period, \( p_T \to 0 \) means that the hybrid policy is reduced to a quantity-based policy \((Q, q) = (Q, \tilde{q})\), whereas \( p_T \to 1 \) means that it’s reduced to a time-based policy \((Q, T) = (Q, \tilde{q} \lambda^{-1})\). Therefore, time-based and quantity-based models can be regarded as special cases of hybrid model, which discloses the essential connection among them.

Now, each \((Q', \tilde{q}')\) in the aforementioned near-optimal set has a corresponding near-optimal policy \((Q', \tilde{q}' , T') = (Q', \tilde{q}' , \tilde{q}' \lambda^{-1})\), among which the policy corresponding to the best correction factor is certain to be the nearest optimal.

By using process simulation methodology, the nearest optimal policy can be distinguished from the aforementioned near-optimal set, and accordingly the corresponding \( \varepsilon \) must be the best correction to \( E(K) \). Besides, the validity of our model can be demonstrated by comparing the results obtained through simulation and optimization. A simulation program named simuPolicy in MATLAB language (source code: see Appendix) is developed, which is designed to track the process of replenishment and dispatching under any hybrid policy for any number of replenishment cycles.

The approach discussed in this section is named Optimization-Simulation Jointly Correcting Approach (OSJCA). It’s summarized as follows.

**Step 1:** Choose an appropriate step size of correction so as to define a correction factor set.

**Step 2:** Iteratively solve the programming problems \( \{ \min C(Q, q) | Q \geq 0, q \geq 0 \} \) for each correction factor by using the existing solving tools and obtain a near-optimal set of \((Q', \tilde{q}')\).

**Step 3:** For each \((Q', \tilde{q}')\), set \( q = \tilde{q} \), \( T = \tilde{q} \lambda^{-1} \). Then the corresponding near-optimal policy set of \((Q', q', T')\) is obtained.

**Step 4:** Select a positive integer which is large enough as the number of replenishment cycles to be simulated. Call procedure simuPolicy (source code: see Appendix) to simulate each policy in the near-optimal set, then \( \bar{K} \) and \( C(Q, q, T) \) (the average values) would be obtained. Compare the results from simulation with those from optimization one by one so as to identify the nearest optimal one, which would simultaneously satisfy the following two propositions.

1. Among the results from simulation, \( C(Q, q, T) \) corresponding to the nearest optimal policy is the smallest;

2. In general, when the nearest optimal policy is adopted, \( E(K) \) and \( C(Q, q, T) \) obtained through optimization is very close to \( \bar{K} \) and \( C(Q, q, T) \) through simulation and closer than any other policy in the near-optimal policy set.

These propositions can also play the role of discriminant rules to screen out the nearest optimal policy.

**5. Numerical Example**

In this section, a numerical example is provided to show the application of the model and OSJCA approach developed in this research. The base values of input parameters are given in Table 1.

OSJCA approach can be applied to this example.

**Step 1:** Set the step size of correction to 0.1, i.e. the correction factor set is \{0.1, 0.2, 0.3, \ldots, 1\}.

**Step 2:** Substitute each correction factor into Equation (19) and solve the corresponding programming problem \( \{ \min C(Q, q) | Q \geq 0, q \geq 0 \} \) iteratively. The optimization results are shown in Table 2. The 3rd and 4th columns indicate the near-optimal set of \((Q', q')\).

**Step 3:** For each \((Q', q')\), derive the corresponding \((Q', q', T')\) by setting \( q' = \tilde{q} \), \( T' = \tilde{q}' \lambda^{-1} \), as shown in Table 3.

**Table 1. Base Values of Input Parameters**

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>( \lambda )</th>
<th>( h )</th>
<th>( s_0 )</th>
<th>( s_0 )</th>
<th>( c_s )</th>
<th>( c_0 )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>25</td>
<td>2</td>
<td>200</td>
<td>10</td>
<td>1</td>
<td>0.5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table 2. Results through Optimization**

<table>
<thead>
<tr>
<th>( \varepsilon )</th>
<th>( E(K) )</th>
<th>( Q^* )</th>
<th>( q^* )</th>
<th>( C(Q^<em>, q^</em>) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>7.740</td>
<td>70.824</td>
<td>8.600</td>
<td>239.600</td>
</tr>
<tr>
<td>0.2</td>
<td>7.487</td>
<td>69.845</td>
<td>8.897</td>
<td>237.670</td>
</tr>
<tr>
<td>0.3</td>
<td>7.226</td>
<td>68.793</td>
<td>9.226</td>
<td>235.675</td>
</tr>
<tr>
<td>0.4</td>
<td>6.955</td>
<td>67.658</td>
<td>9.594</td>
<td>233.606</td>
</tr>
<tr>
<td>0.5</td>
<td>6.674</td>
<td>66.421</td>
<td>10.007</td>
<td>231.456</td>
</tr>
<tr>
<td>0.6</td>
<td>6.381</td>
<td>65.064</td>
<td>10.478</td>
<td>229.213</td>
</tr>
<tr>
<td>0.7</td>
<td>6.074</td>
<td>63.558</td>
<td>11.018</td>
<td>226.864</td>
</tr>
<tr>
<td>0.8</td>
<td>5.752</td>
<td>61.865</td>
<td>11.649</td>
<td>224.394</td>
</tr>
<tr>
<td>0.9</td>
<td>5.413</td>
<td>59.933</td>
<td>12.397</td>
<td>221.781</td>
</tr>
<tr>
<td>1.0</td>
<td>5.052</td>
<td>57.863</td>
<td>13.303</td>
<td>218.998</td>
</tr>
</tbody>
</table>
Table 3. Near-optimal Policy Set

<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>$Q^*$</th>
<th>$q^*$</th>
<th>$T^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>70.824</td>
<td>8.600</td>
<td>0.3440</td>
</tr>
<tr>
<td>0.2</td>
<td>69.845</td>
<td>8.897</td>
<td>0.3559</td>
</tr>
<tr>
<td>0.3</td>
<td>68.793</td>
<td>9.226</td>
<td>0.3691</td>
</tr>
<tr>
<td>0.4</td>
<td>67.658</td>
<td>9.594</td>
<td>0.3838</td>
</tr>
<tr>
<td>0.5</td>
<td>66.421</td>
<td>10.007</td>
<td>0.4003</td>
</tr>
<tr>
<td>0.6</td>
<td>65.064</td>
<td>10.478</td>
<td>0.4191</td>
</tr>
<tr>
<td>0.7</td>
<td>63.558</td>
<td>11.018</td>
<td>0.4407</td>
</tr>
<tr>
<td>0.8</td>
<td>61.865</td>
<td>11.649</td>
<td>0.4660</td>
</tr>
<tr>
<td>0.9</td>
<td>59.933</td>
<td>12.397</td>
<td>0.4959</td>
</tr>
<tr>
<td>1.0</td>
<td>57.683</td>
<td>13.303</td>
<td>0.5321</td>
</tr>
</tbody>
</table>

Step 4: Call procedure simuPolicy to simulate each policy in Table 3 for 2000 replenishment cycles. Table 4 shows the average values of $K$ and $C(Q,q,T)$ obtained through simulation.

Table 4. Results through Simulation

<table>
<thead>
<tr>
<th>$Q^*$</th>
<th>$q^*$</th>
<th>$T^*$</th>
<th>$K$</th>
<th>$(C(Q,q,T))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>70.824</td>
<td>8.600</td>
<td>0.3440</td>
<td>8.057</td>
<td>235.1566</td>
</tr>
<tr>
<td>69.845</td>
<td>8.897</td>
<td>0.3559</td>
<td>7.813</td>
<td>234.3400</td>
</tr>
<tr>
<td>68.793</td>
<td>9.226</td>
<td>0.3691</td>
<td>7.111</td>
<td>233.7209</td>
</tr>
<tr>
<td>67.658</td>
<td>9.594</td>
<td>0.3838</td>
<td>6.823</td>
<td>233.5194</td>
</tr>
<tr>
<td>66.421</td>
<td>10.007</td>
<td>0.4003</td>
<td>6.207</td>
<td>233.7464</td>
</tr>
<tr>
<td>65.064</td>
<td>10.478</td>
<td>0.4191</td>
<td>5.926</td>
<td>233.7662</td>
</tr>
<tr>
<td>63.558</td>
<td>11.018</td>
<td>0.4407</td>
<td>5.897</td>
<td>233.8944</td>
</tr>
<tr>
<td>61.865</td>
<td>11.649</td>
<td>0.4660</td>
<td>5.031</td>
<td>234.0897</td>
</tr>
<tr>
<td>59.933</td>
<td>12.397</td>
<td>0.4959</td>
<td>4.413</td>
<td>234.8947</td>
</tr>
<tr>
<td>57.683</td>
<td>13.303</td>
<td>0.5321</td>
<td>3.971</td>
<td>234.9478</td>
</tr>
</tbody>
</table>

Figure 3. Trendline of Average Long-run Cost under Policy $(67.658,9.594,0.3838)$, 2000 Replenishment Cycles Tracked

Let’s observe Table 2 and Table 4, and make comparisons row by row. According to the discriminant rules suggested in Section 4, it’s clear that the nearest optimal policy is $(Q,q,T) = (67.658,9.594,0.3838)$ and $\varepsilon = 0.4$ is the best correction to $E(K)$ (see the underlined rows of Table 2 and Table 4). The trendline of average long-run cost under this policy is depicted in Figure 3, which is generated by simuPolicy in the environment of MATLAB.

6. Sensitivity Analysis

Let’s vary one parameter at a time while keep others at base values, and apply OSJCA to it. The qualitative effect of increasing each parameter on the resulting nearest optimal policy and average long-run cost is shown in Table 5.

The following conclusions can be made from Table 5,

1. All the input parameters have increasing effect on the resulting nearest optimal policy and average long-run cost;
2. As $\theta$ increases, $Q^*$ should be set smaller, whereas $q^*$ and $T^*$ should be set larger;
3. As $\lambda$ increases, $Q^*$ and $q^*$ should be enlarged, whereas $T^*$ should be shorten;
4. Parameter $h$ and $sk$ have decreasing and increasing effects on $Q^*$ respectively;
5. The effect of $sD$ on the resulting nearest optimal policy is same as $\theta$, while $w$ is just the reverse;
6. Parameter $cr$ has decreasing effect on $Q^*$, whereas $CD$ has no obvious effect on $Q^*$;
7. The effect of $h$, $sk$, $cr$, $CD$ on $q^*$ and $T^*$ is not remarkable.

These conclusions are just coincident with our intuition.

7. Concluding remarks and further research

The argument of this research focuses on stock replenishment and shipment scheduling with deteriorating item for vendor-managed inventory systems. Considering demand process follows a typical Poisson process, hybrid replenishment and dispatching model with correction for
VMI is proposed, and a combined solving approach named OSJCA is suggested. As approximation of expected dispatching number per replenishment cycle is involved, a correction factor varying in the left half-open interval of \((0, 1]\) is introduced into our model. Therefore, a series of nearly-optimal hybrid policies rather than an exactly optimal one, under which average long-run cost is fairly close to the minimum, are obtained by solving a series of \(\varepsilon\)-corrected programming problems. In order to screen the nearest optimal one out, \textit{simuPolicy}, a special simulation program for tracking the effect of any hybrid policy for any number of replenishment cycles, is developed in the environment of MATLAB. According to the discriminant rules suggested in Section 4, the nearest optimal policy can be obtained by comparing results obtained through optimization and simulation one by one. Moreover, a numerical example has shown that the disparity between results through model and simulation under the nearest optimal policy is small or even negligible. Thus the validity of our model and OSJCA approach is confirmed.

For ease of discussion, the effect of inventory replenishment lead time is not taken into account in this paper, which is also held as an important idealized assumption by the existing literatures on such issue. The further work will focus on the replenishment and dispatching policy with stochastic replenishment lead time considered.

Acknowledgments

Supported by National Natural Science Foundation of China (NSFC, 70525001, 70371023), Doctoral Fund of Ministry of Education of China (20030358052).

References

nal of Production Economics, 81-82(1), 215-224.


Appendix. Source Code of simuPolicy in MATLAB

function avgCost=simuPolicy( input, policy, times)
% simuPolicy is used to track the effect of any hybrid policy for any replenishment cycles.
% input: A 1×8 matrix consisting of θ, λ, h, sR, sD, cR, cD, w;
% policy: A 1×3 matrix used to denote the policy to be tracked;
% times: The number of replenishment cycles to be tracked.

% initialize relevant parameters
% input parameters
theta=input(1,1); lambda=input(1,2); h=input(1,3); sR=input(1,4);
sD=input(1,5); cR=input(1,6); cD=input(1,7); w=input(1,8);

% policy-related parameters
Q=policy(1,1); q=policy(1,2); T=policy(1,3);

%% declare a 1×times matrix to permanently record average cost of every replenishment cycle.
avgCost_Series=zeros(1,times);

% initialize relevant parameters
% for the latest dispatching
It=Q; % to temporarily record the inventory level just after the latest dispatching
Cost=0; % to temporarily record the cost at the end of each replenishment cycle
interval=exprnd(1/lambda, 1,1);
% generate an exponential random number to simulate the time interval after which arrives the next unit demand.
counter=1;
while (counter<=times)
    elapsed_time=0; % the elapsed time since the latest dispatching
    N=0; % accumulated demand since the latest dispatching
    W=0; % customer waiting cost happened in the new dispatching period
    while (N<q & elapsed_time+interval<=T)
        % the elapsed time since the latest dispatching
        N=N+1;
        % accumulated demand since the latest dispatching
        W=W + N. * w * interval;
        % customer waiting cost happened in the new dispatching period
        N=N+1;
        % interval
        interval= exprnd(1/lambda, 1,1);
    end
    D= sD+ cd*N; % dispatching cost happened in current dispatching period
    if (elapsed_time + interval > T)
        H= (1-exp(-theta*T))*It*h/theta;
        % holding cost happened in current dispatching period
        W=W+N.*w*(T-elapsed_time);
        % customer waiting cost happened in current dispatching period
        t=t+T;
        It=It*exp(-theta*T);
        if (It>=N) % do not replenish
            It = It - N;
            Cost = Cost + D + H + W;
        else % replenish
            R=sR+cr*(Q+Nt-It); % replenishment cost
            It=Q;
            Cost=Cost + R + D + H + W;
            avgCost_Series(1,counter)= Cost/t;
            counter=counter+1;
        end
        interval = elapsed_time + interval - T;
    else % replenish
        H= (1-exp(-theta*elapsed_time))*It*h/theta;
        t=t+elapsed_time;
        It=It*exp(-theta*elapsed_time);
        if (It>=N) % do not replenish
            It = It - N;
            Cost = Cost + D + H + W;
        else % replenish
            R=sR+cr*(Q+Nt-It); % replenishment cost
            It=Q;
            Cost=Cost + R + D + H + W;
            avgCost_Series(1,counter)= Cost/t;
            counter=counter+1;
        end
    end
end
% return average cost
avgCost = avgCost_Series(1,times);
% output trendline of average long-run cost
plot([1:times], avgCost_Series, 'r-');