IIE Transactions (2000) **32**, 785–795

Control of a manufacturing system with random product yield and downward substitutability

IZAK DUENYAS and CHI-YANG TSAI

School of Business, The University of Michigan, Ann Arbor, MI 48109-1234, USA
E-mail: duenyas@engin.umich.edu

Received January 1998 and accepted October 1999

We consider a manufacturing system where the quality of the end product is uncertain and is graded into one of several quality levels after production. We assume stochastic demand for each quality level, stochastic production times, and random quality yields. We also assume downward substitutability (i.e., customers who require a given product will be satisfied by a higher quality product at the same price). The firm produces to stock and has the option to refuse satisfying customers even when it has items in stock. We formulate this problem as a Markov Decision Process in the context of a simple M/M/1 make-to-stock queue with multiple customer classes to gain insight into the following questions: (i) how does the firm decide when to produce more units (i.e., what is the optimal production policy?) and (ii) how does the firm decide when to accept/reject orders and when to satisfy customers demanding lower quality products using higher quality products? In the case of two product classes, we completely characterize the structure of the optimal production and acceptance/substitution policies. However, the structure of the optimal policy is complicated and we therefore develop a simple heuristic policy for any number of classes which performs very well. We finally extend our heuristic to the system where production occurs in batches of size of larger than one, the system where there is a setup cost for initiating production, and the case where processing time distribution is Erlang.

1. Introduction

In many manufacturing environments, the quality of the end product is uncertain. This leads many manufacturers to sell their different quality products to different customers. Examples are in apparel manufacturing where firms will sell products with slight defects at large discounts, and electronics manufacturing where the clock speeds of chips produced by the same process is uncertain. In most of these environments, customers demanding the lower quality product will be satisfied receiving a higher quality product so there is downward substitutability. For example, many apparel companies have outlet stores where they sell slightly lower quality products. Customers of these stores usually travel from major metropolitan areas for the discounts and will not pay full price as the stores are known to generally carry products with slight defects. However, if the manufacturer actually runs out of clothes with defects, these stores still need to receive inventory and the manufacturer will sometimes sell them first quality products at second quality prices. Several researchers (Bitran and Dasu, 1992; Bitran and Leong, 1992; Nahmias and Moinzadeh, 1997) describe the similar problem in electronics manufacturing where customers demanding slower chips can be satisfied by faster chips.

In all the problems described above, the manufacturer faces the following questions: (i) how does one decide when to start/stop production (i.e., what inventory levels are appropriate?) and (ii) when is it optimal to satisfy customers demanding the lower quality products using the higher quality ones? In this paper, we address these problems first in the context of a simple M/M/1 queueing system with multiple customer classes to gain insight into the structure of the optimal solutions to these problems. As the optimal structure can be rather complicated, we then develop and test a simple and effective heuristic. We then extend our formulations and heuristics to the more realistic case where processing times are assumed to be Erlang distributions.

The literature on the optimal control of systems with random quality yields and substitution is rather limited. (For a review of the literature on problems where the quantity yield is uncertain, see Yano and Lee (1995)). Bitran and Dasu (1992), Bitran and Leong (1992) and Bitran and Gilbert (1994) are some of the first papers to address this problem. These papers assume that demand is deterministic. They formulate the problem as a large stochastic program. Our paper differs from theirs in several ways: (i) we allow random production times and demands; and (ii) we focus on characterizing the structure of the optimal policy. Nahmias and Moin-

zadeh (1997) focus on a system with products of two quality classes and constant demands. They build a continuous review EOQ-type model. Our paper differs in that we have stochastic production times and we also consider the difference in profits generated by different classes of products. Furthermore, our paper focuses on characterizing the structure of the optimal policy and comparing developed heuristics to the optimal policy whereas Nahmias and Moinzadeh focus on performance evaluation of a specific policy. Gerchak *et al.* (1996) and Hsu and Bassok (1999) also analyze a similar problem in a single period context where an order is placed once and random quantities of products of each class are received as a result. In contrast, our formulation is infinite-horizon and we take into account queueing effects by focusing on a production system where production of each unit (or batch) takes a random amount of time.

In a recent paper, Ha (1997a) addresses the issue of inventory rationing in a make-to-stock production system with several demand classes and lost sales. In his model, all demands from the different classes are satisfied by the same inventory. The optimal policy is characterized by a stock rationing level for each class at or below which it is optimal to start rejecting the demand from that class. Our analysis and proof approaches follow the same lines as those in Ha (1997a) and Ha (1997b). However, our model is different in that we have separate inventories for each class, we have uncertain yield and also substitution is allowed in only one direction.

The rest of this paper is organized as follows. In Section 2, we formulate the problem with two quality classes and characterize the structure of the optimal policy. The structure of the optimal policy is rather complicated for problems with more than two quality classes and therefore in Section 3, we develop a simple heuristic policy. In Section 4, we extend our formulation to the case where production occurs in batches and the case with setup cost. We also develop simple heuristics for these cases. In Section 5, we explore the system with Erlang processing times and show how the heuristic we develop in Section 3 can be easily modified to be used in this case. In Section 6, we test our heuristics on a variety of test problems and the results show that the heuristics perform very well. The paper concludes in Section 7.

2. Two quality classes

In this section, we limit ourselves to problems with two quality classes. We formulate the problem in the context of a simple make-to-stock M/M/1 queueing system with two classes of items to gain insight into the nature of the optimal policy. The system produces items with rate μ and the items are classified either as low quality (class 1) or high quality (class 2) when they are produced. With probability p_1, the outcome of a production is a class 1 item. With probability p_2, (since there are only two classes, $p_2 = 1 - p_1$) the result is a class 2 item. The price for class 1 items is R_1 and the price for class 2 items is $R_2 > R_1$. Demand for low-quality item arrives with rate λ_1 and demand for high-quality item arrives with rate λ_2. Class 2 customers will only accept class 2 items, and therefore if there is no class 2 item in stock, the firm loses the sale. However, when class 1 customers arrive, they will be satisfied if they receive a class 2 item at price R_1 as well. Therefore, when a class 1 customer arrives, the firm can decide between selling the customer a class 1 or class 2 item or if the firm has no class 1 item, it can also reject the customer. Let n_i denote the number of class i units in stock; we also assume that holding costs are charged at the rate of h per unit time (It is reasonable to assume that the holding costs for the two classes of items are the same since they are made of the same raw materials, go through the same processes and thus have the same production costs).

A control policy specifies the actions taken by the firm at each decision epoch upon the arrivals of customers or the completion of production. When class 1 customers arrive, the firm can take the following actions: satisfy the demands with class 1 items, meet the demands with class 2 items, or when class 1 items are out of stock, reject the orders. Also, the firm can decide whether to start a production at any decision epoch when the workstation is idle. Our goal is to find an optimal policy that maximizes the long-run average profit. The problem can be formulated as a Markov Decision Process. Let $v(n_1, n_2)$ be the relative value function of being in state (n_1, n_2) and g denote the average profit per transition (where transitions occur with rate $\Lambda = \lambda_1 + \lambda_2 + \mu$ and therefore the average profit per unit time is $g\Lambda$). Then, using uniformization as in Lippman (1975) we can write:

$$
\begin{aligned}
&g + v(n_1, n_2) \\
&= \frac{1}{\Lambda}\Bigg\{ -h(n_1 + n_2) + \lambda_2[(v(n_1, n_2 - 1) + R_2)I_{n_2>0} \\
&\quad + v(n_1, 0)I_{n_2=0}] \\
&\quad + \lambda_1 \max\begin{pmatrix} (v(n_1 - 1, n_2) + R_1)I_{n_1>0} + v(0, n_2)I_{n_1=0} \\ (v(n_1, n_2 - 1) + R_1)I_{n_2>0} + v(n_1, 0)I_{n_2=0} \end{pmatrix} \\
&\quad + \mu \max\begin{pmatrix} p_1 v(n_1 + 1, n_2) + p_2 v(n_1, n_2 + 1) \\ v(n_1, n_2) \end{pmatrix} \Bigg\}, \\
&\qquad\qquad n_1, n_2 = 0, 1, 2, \ldots, \quad (1)
\end{aligned}
$$

where $I_{(\cdot)}$ denotes the indicator function.

In (1), the terms multiplied by h represent the holding costs associated with the units in stock. The terms multiplied by λ_2 represent the transitions and revenues generated by the arrival of a class 2 customer. Uniformization as in Lippman, see also Ha (1997a) and Carr and Duenyas (1999), allows us to view both customer arrivals and production "opportunities" as events that

occur. The probability λ_2/Λ can be interpreted as the probability that the next event is a class 2 customer's arrival. In this case, if we have a class 2 item in stock, we get revenue R_2 and n_2 decreases by one, and if not we get no revenue and the state of the system remains the same. Similarly, λ_1/Λ represents the probability that the next event is the arrival of a class 1 customer. When a class 1 customer arrives, we can decide between satisfying his demand with a class 1 or class 2 item. If no class 1 item is in stock, then our choice is between not satisfying the customer's demand and satisfying it with a class 2 item. (Note that it is in fact obvious that when $n_1 > 0$, it can not be optimal to reject a class 1 customer. Class 1 items can only be used to satisfy class 1 customers' demands and therefore there is no point in turning away a class 1 customer when $n_1 > 0$). Finally, with probability μ/Λ, the next event is a production completion "opportunity". In this case, production will lead to another class 1 or class 2 item and not taking advantage of this opportunity (i.e., idling) will result in the same state.

Figure 1 shows an example of the transition rate diagram for this problem assuming that the following actions are optimal in the displayed states: (i) it is optimal to produce; (ii) it is optimal to satisfy class 1 customer's orders using only class 1 items. For example, in state (0,1), if the next event is the arrival of a class 1 customer, (with rate λ_1), the state does not change because there are no class 1 items in stock. If the next event is a production completion, that takes the system either to state $(1,1)$ or to state $(0,2)$. Finally, arrival of a demand from a class 2 customer takes the system to state $(0,0)$.

Having formulated the problem, we can next characterize the structure of the optimal policy in the following

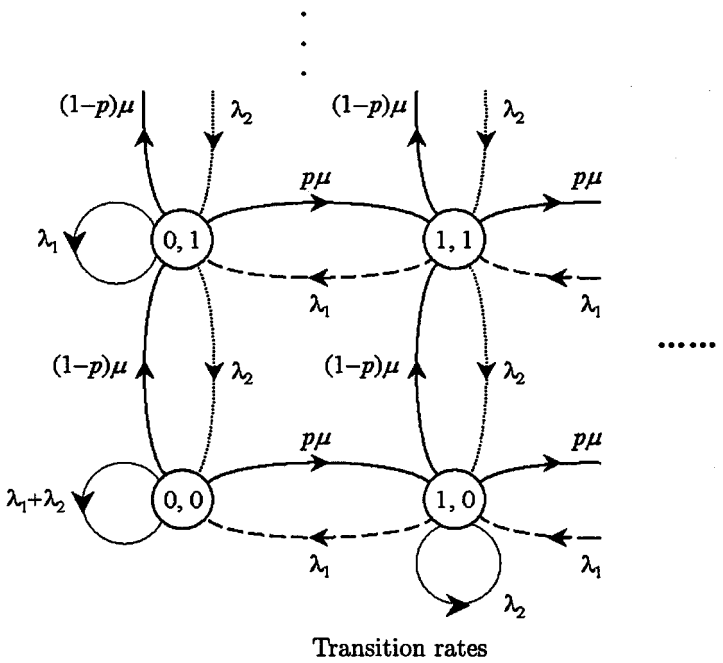

Fig. 1. Example transition diagram for two class problem.

Theorem 1. *The optimal policy for the system described by* (1) *has the following structure.*

(i) *The optimal production policy is defined by a switching curve* $d(n_1)$ *such that for* $n_2 \geq d(n_1)$, *the optimal policy is to idle*; *for* $n_2 < d(n_1)$, *the optimal policy is to produce. Furthermore,* $d(n_1)$ *is nonincreasing in* n_1.

(ii) *If there are class* 1 *items on hand* $(n_1 > 0)$, *we always sell class* 1 *items to class* 1 *customers. If the inventory level of class* 1 *items is zero* $(n_1 = 0)$, *then there is a threshold for selling class* 2 *items to class* 1 *customers which is defined by an integer* S *such that for* $n_2 \geq S$, *the optimal policy is to sell*; *for* $n_2 < S$, *the optimal policy is not to sell.*

Proof: See Appendix.

Figure 2 illustrates the production switching curve described in (i) of Theorem 1. Increases in the inventory of either item makes it less desirable to produce. Furthermore, Theorem 1 indicates that we only sell class 2 items to class 1 customers when $n_1 = 0$ and n_2 is sufficiently high.

Having characterized the structure of the optimal policy, we note that it is straightforward to show that if sales for the higher quality item (but not the lower quality item) can be backordered instead of lost that the structure of the optimal policy remains exactly the same. The proof for this result follows the same lines as Theorem 1 and is therefore omitted.

The structure of the optimal policy gets extremely complicated for problems with more than two quality classes. This is because the decision to produce now depends on the number of items of each class on hand and the resulting policy is in the form of N-dimensional switching planes which are neither easy to compute nor

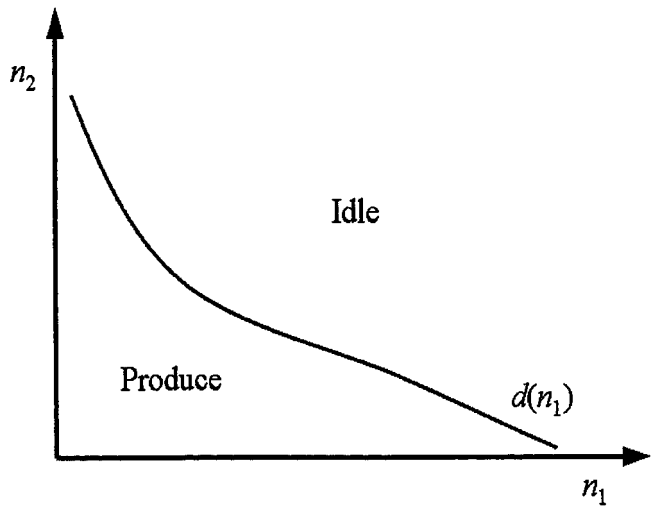

Fig. 2. The structure of the optimal production policy for the two product case.

easy to describe even if one is able to compute them. Furthermore, even if we limit ourselves to a maximum of 50 items for each class, for a four-class problem, the MDP we would have to solve would have over six million states. Therefore, in the next section, we focus on simple heuristic policies which are easy to compute and easy to implement and perform well.

3. A heuristic policy

In this section, we first propose a simple, easily implementable heuristic for the problem with two classes. We then generalize the heuristic to any number of classes.

In the problem with two classes, there are two decisions to be made:

(a) When the workstation is idle, do we start producing or keep the workstation idle?
(b) When a class 1 item is out of stock, do we sell class 2 items to class 1 customers?

We replace the more complicated optimal policy by a simpler heuristic policy defined by two variables, Q and S, corresponding to the thresholds for each of these decisions.

Heuristic for the problem with two classes

Step 1. Produce if $n_1 + n_2 < Q$ and do not produce otherwise.

Step 2. When a class 1 item is out of stock (i.e., $n_1 = 0$) if $n_2 \geq S$, we sell a class 2 item to a class 1 customer and we reject class 1 customers otherwise.

From the structure of the optimal production policy described in Theorem 1, we know that Decision (a) has to be made based on the inventory levels of both class 1 and class 2 items. We simplify the problem by combining the two classes of items into a single aggregate class and the two classes of customers into a single class of customers. We do this by solving the following MDP:

$$g + v(n) = \frac{1}{\lambda + \mu}\left\{ -hn + \lambda[(v(n-1) + R)I_{n>0} + v(0)I_{n=0}] + \mu \max\begin{pmatrix} v(n+1) \\ v(n) \end{pmatrix}\right\}, \quad n = 0, 1, 2, \ldots, \quad (2)$$

where $\lambda = \lambda_1 + \min\{\lambda_2, p_2\mu\}$, and

$$R = \frac{\min\{\lambda_2, p_2\mu\}R_2 + (\min\{\lambda_1 + \lambda_2, \mu\} - \min\{\lambda_2, p_2\mu\})R_1}{\min\{\lambda_1 + \lambda_2, \mu\}}.$$

Note that since the average rate of production of class 2 items is $p_2\mu$, the rate with which we can satisfy class 2 demands is at most the minimum of class 2 demand λ_2 and class 2 production. In creating an aggregate item, we therefore replace λ_2 by $\min\{\lambda_2, p_2\mu\}$. Similarly, we take a weighted average of the revenue generated by each class. As a result, the MDP is reduced to a single dimension and the value of Q can be determined by solving this simple MDP and choosing the first value of n for which $v(n) > v(n+1)$.

For Decision (b), we simplify (1) by taking out n_1 so that it becomes a system with only class 2 items but still with two classes of customers. This is because this decision only comes up when $n_1 = 0$. Then we have:

$$g + v(n_2) = \frac{1}{\Lambda}\left\{ -hn_2 + \lambda_2[(v(n_2 - 1) + R_2)I_{n_2>0} + v(0)I_{n_2=0}] + \lambda_1 \max\begin{pmatrix} (v(n_2 - 1) + R_1)I_{n_2>0} + v(0)I_{n_2=0} \\ v(n_2) \end{pmatrix} + p_2\mu \max\begin{pmatrix} v(n_2 + 1) \\ v(n_2) \end{pmatrix}\right\}, \quad n_2 = 0, 1, 2, \ldots, \quad (3)$$

where $\Lambda = \lambda_1 + \lambda_2 + p_2\mu$.

Again, the state space of the MDP is drastically reduced and solving it gives us S, the value of the threshold for selling class 2 items to class 1 customers. The value of S is obtained by inspecting the solution to the above MDP and choosing the first value of n_2 for which $(v(n_2 - 1) + R_1)I_{n_2>0} + v(0)I_{n_2=0} > v(n_2)$. Solving for Q and S takes less than a second on a Pentium computer.

We use the same ideas to extend the heuristic to problems with any number of classes.

Heuristic for the problem with k classes

Step 1. Production policy: produce when $n_1 + n_2 + \cdots + n_k < Q$ and do not produce otherwise.

Step 2. Sales policy: when $n_j = n_{j+1} = \cdots = n_{i-1} = 0$, sell a class i item to a class j customer if $n_i \geq S_{i,j}$, otherwise do not sell, $i \in \{2, \ldots, k\}$, $j \in \{1, 2, \ldots, i-1\}$.

We note that in a problem with k classes, our heuristic policy is defined by $1 + k(k-1)/2$ thresholds. There is one threshold for deciding to produce or idle and the remaining thresholds define when we would use a higher quality item to satisfy demand from a lower quality class.

To compute Q, the threshold for producing or idling, we once again aggregate all classes of items to a single average class by taking the sum of the demand rates and a weighted average of the revenues and solve the following MDP:

$$g + v(n) = \frac{1}{\lambda + \mu}\left\{ -hn + \lambda[(v(n-1) + R)I_{n>0} + v(0)I_{n=0}] + \mu \max\begin{pmatrix} v(n+1) \\ v(n) \end{pmatrix}\right\}, \quad n = 0, 1, 2, \ldots,$$

where $\lambda = \lambda_1 + \lambda_2' + \cdots + \lambda_k'$, $\lambda_i' = \min\{\lambda_i, \mu \sum_{j=i}^{k} p_j\}$ and

$$R = \frac{x_1 R_1 + x_2 R_2 + \cdots + (z - x_1 - x_2 - \cdots - x_{k-1}) R_k}{z},$$

$$x_i = \min\{\lambda_i, p_i \mu\} \text{ and } z = \min\{\lambda, \mu\}.$$

To compute the values of $S_{i,j}$ ($j = 1, \ldots, i-1$), we solve a total of $k-1$ single dimensional MDPs.

$$\begin{aligned} g + v(n_i) &= \frac{1}{\Lambda}\Bigg\{ - hn_i + \lambda_i[(v(n_i - 1) + R_i) I_{n_i > 0} + v(0) I_{n_i = 0}] \\ &\quad + \lambda_{i-1} \max\begin{pmatrix} (v(n_i - 1) + R_{i-1}) I_{n_i > 0} + v(0) I_{n_i = 0} \\ v(n_i) \end{pmatrix} \\ &\quad + \lambda_{i-2} \max\begin{pmatrix} (v(n_i - 1) + R_{i-2}) I_{n_i > 0} + v(0) I_{n_i = 0} \\ v(n_i) \end{pmatrix} \\ &\quad \vdots \\ &\quad + \lambda_1 \max\begin{pmatrix} (v(n_i - 1) + R_1 I_{n_i > 0} + v(0) I_{n_i = 0} \\ v(n_i) \end{pmatrix} \\ &\quad + p_i \mu \max\begin{pmatrix} v(n_i + 1) \\ v(n_i) \end{pmatrix} \Bigg\}, \quad n_i = 0, 1, 2, \ldots, \end{aligned}$$

where $\Lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_i + p_i \mu$.

Each of the thresholds for $S_{i,j}$ can then be obtained by inspecting the solutions to the above MDPs. $S_{i,j}$ is the first value of n_i for which $(v(n_i - 1) + R_j) I_{n_i > 0} + v(0) I_{n_i = 0} > v(n_i)$. We would like to emphasize that the computational load of the heuristic is minimal. For example, whereas in a problem with four quality classes where each class was limited to at most 100 units in inventory, one would have to solve an MDP with 10^8 states, our heuristic would only require solving four MDPs each with a state space of 100. Therefore, even for problems with very large number of classes, our heuristic is easy to compute and gives results in under a second. We compare its performance to that of the optimal policy in Section 6.

4. Extensions of the basic model

In this section, we explore two extensions of the basic model. The previous model is rather simple as it assumes that each unit is produced individually and inspected and that decisions are made after the production of each unit. On the other hand, in most manufacturing environments, production occurs in batches. This may be due to the fact that producing units individually may not be feasible or economical. For example, computer chips are produced in the form of wafers that contain many chips. For this reason, in this section, we extend our formulation to take into account these effects from batch production.

First, assume that items are always produced in batches of a given size (for example, how many chips would be on a wafer would define the batch size) and then the whole batch is inspected to find out how many items of each class were actually produced. The numbers of class 1 items and class 2 items in a batch are random. Let b be the batch size and Z, a discrete random variable between zero and b, be the number of class 1 items in a batch. Then (1) can be modified as:

$$\begin{aligned} g + v(n_1, n_2) &= \frac{1}{\Lambda}\Bigg\{ - h(n_1 + n_2) + \lambda_2[(v(n_1, n_2 - 1) + R_2) I_{n_2 > 0} \\ &\quad + v(n_1, 0) I_{n_2 = 0}] \\ &\quad + \lambda_1 \max\begin{pmatrix} (v(n_1 - 1, n_2) + R_1) I_{n_1 > 0} + v(0, n_2) I_{n_1 = 0} \\ (v(n_1, n_2 - 1) + R_1) I_{n_2 > 0} + v(n_1, 0) I_{n_2 = 0} \end{pmatrix} \\ &\quad + \mu \max\begin{pmatrix} E[v(n_1 + Z, n_2 + b - Z)] \\ v(n_1, n_2) \end{pmatrix} \Bigg\}, \\ &\qquad n_1, n_2 = 0, 1, 2, \ldots, \end{aligned} \tag{4}$$

where $\Lambda = \lambda_1 + \lambda_2 + \mu$ and $I_{(\cdot)}$ denotes the indicator function.

The only modification is in the terms multiplied by μ where after the completion of the production of a batch, the state changes from (n_1, n_2) to $(n_1 + Z, n_2 + b - Z)$. We conjecture that the results of Theorem 1 apply to this case as well with no changes although we have not been able to prove this result to date. We have not been able to find a numerical example where the structural results described in Theorem 1 do not hold for the case with batch production.

Our heuristic policy can easily be adapted to this problem by changing (2) and (3) such that when the decision to produce is made, the state changes from (n) to $(n + b)$, and adjusting the aggregated demand rate, λ, for computing the value of Q. For example, if we assume that the probability that each individual item in the batch is of class i is p_i (independent of the class of other items in the batch), then in the case of two classes, Equation (2) becomes

$$\begin{aligned} g + v(n) = \frac{1}{\lambda + \mu}\Bigg\{ & - hn + \lambda[(v(n - 1) + R) I_{n > 0} \\ & + v(0) I_{n = 0}] + \mu \max\begin{pmatrix} v(n + b) \\ v(n) \end{pmatrix} \Bigg\}, \\ & n = 0, 1, 2, \ldots, \end{aligned}$$

where $\lambda = \lambda_1 + \min\{\lambda_2, p_2 b \mu\}$. Equation (3) can be adjusted in the same manner.

Next, assume that there is a setup cost of K every time the machine is turned on after a period of time that it idles. Let $V(n_1, n_2, 0)$ ($V(n_1, n_2, 1)$) denote the profit when the system contains n_1 class 1 items and n_2 class 2 items, and the machine is idle (busy). Then we can revise the original formulation of the problem to:

$$
\begin{aligned}
&g+v(n_1,n_2,c)\\
&=\frac{1}{\Lambda}\Bigg\{-h(n_1+n_2)\\
&\quad+\lambda_2[(v(n_1,n_2-1,c)+R_2)I_{n_2>0}+v(n_1,0,c)I_{n_2=0}]\\
&\quad+\lambda_1\max\begin{pmatrix}(v(n_1-1,n_2,c)+R_1)I_{n_1>0}+v(0,n_2,c)I_{n_1=0}\\(v(n_1,n_2-1,c)+R_1)I_{n_2>0}+v(n_1,0,c)I_{n_2=0}\end{pmatrix}\\
&\quad+\mu\max\begin{pmatrix}K(c-1)+p_1v(n_1+1,n_2,1)+p_2v(n_1,n_2+1,1)\\v(n_1,n_2,0)\end{pmatrix}\Bigg\},\\
&\qquad\qquad n_1,n_2=0,1,2,\ldots;\ c=0,1, \quad (5)
\end{aligned}
$$

where $I_{(\cdot)}$ denotes the indicator function.

In the state description, the variable c indicates whether the workstation is idle ($c=0$) or busy ($c=1$). For the transitions related to sales (the terms multiplied by λ_1 and λ_2), c does not change after transitions are completed, no matter which decisions are made. For the transitions related to production policy (the terms multiplied by μ), $c=0$ if the production decision is to idle and $c=1$ if the production decision is to produce. Whether a setup cost, K, occurs is determined by the term $K(c-1)$ (i.e., a setup cost is incurred whenever the machine is turned on after it's off).

In this case, the optimal production and sales policies are more complicated. Based on all the numerical examples we have run, we conjecture that the optimal production policy can be characterized by two switching curves $d_0(n_1)$ and $d_1(n_1)\geq d_0(n_1)$ such that when the machine is already on, it is optimal to continue producing so long as $n_2<d_1(n_1)$ and to shut the machine off otherwise. Similarly, when the machine is off, it is optimal to start producing when $n_2<d_0(n_1)$. Similarly, there exist two thresholds S_0 and S_1 that depend on the state of the machine, for selling class 2 items to class 1 customers.

We can similarly adjust our original heuristic for use in this case. Our heuristic computes two sets of values (Q_0,S_0) and (Q_1,S_1). When the machine is on, it keeps producing as long as the sum of all classes of items in stock is less than Q_1. A class 2 item is sold to a class 1 customer if $n_1=0$ and $n_2>S_1$. However, when the machine is idle, the machine only starts producing again if total inventory falls below Q_0. Finally, when the machine is idle, a class 2 item is sold to a class 1 customer only if $n_1=0$ and $n_2>S_0$. We can compute the values of Q_c by solving

$$
\begin{aligned}
&g+v(n,c)\\
&\quad=\frac{1}{\lambda+\mu}\Bigg\{-hn+\lambda[(v(n-1,c)+R)I_{n>0}+v(0,c)I_{n=0}]\\
&\qquad+\mu\max\begin{pmatrix}K(c-1)+v(n+1,1)\\v(n,0)\end{pmatrix}\Bigg\}\\
&\qquad\qquad n=0,1,2,\ldots;\quad c=0,1 \quad (6)
\end{aligned}
$$

where $\lambda=\lambda_1+\min\{\lambda_2,p_2\mu\}$, and

$$
R=\frac{\min\{\lambda_2,p_2\mu\}R_2+(\min\{\lambda_1+\lambda_2,\mu\}-\min\{\lambda_2,p_2\mu\})R_1}{\min\{\lambda_1+\lambda_2,\mu\}}.
$$

Then, $Q_1=\min\{n: v(n,0)\geq v(n_1+1,1)\}$ and $Q_0=\max\{n:-K+v(n_1+1,1)\geq v(n,0)\}$. Finally, our heuristic computes the values of S_c by solving

$$
\begin{aligned}
&g+v(n_2,c)\\
&\quad=\frac{1}{\Lambda}\Bigg\{-hn_2+\lambda_2[(v(n_2-1,c)+R_2)I_{n_2>0}+v(0,c)I_{n_2=0}].\\
&\qquad+\lambda_1\max\begin{pmatrix}(v(n_2-1,c)+R_1)I_{n_2>0}+v(0,c)I_{n_2=0}\\v(n_2,c)\end{pmatrix}\\
&\qquad+p_2\mu\max\begin{pmatrix}K(c-1)+v(n_2+1,1)\\v(n_2,0)\end{pmatrix}\Bigg\},\\
&\qquad\qquad n_2=0,1,2,\ldots;\quad c=0,1,
\end{aligned}
$$

where $\Lambda=\lambda_1+\lambda_2+p_2\mu$.

S_c is the first value of n_2 for which $(v(n_2-1,c)+R_1)I_{n_2>0}+v(0,c)I_{n_2=0}\geq v(n_2,c)$, $c=0,1$.

So far all our formulations have assumed exponential processing time distributions which is not very practical. Therefore, we explore the system with Erlang production in the next section.

5. Systems with Erlang processing distributions

In this section, we explore the case where processing times are allowed to have an Erlang-z distribution with rate μ. Since an Erlang-z distribution can be regarded as the convolution of z exponential stages, we can once again use uniformization to write an MDP for deciding when to produce and whether to provide class 2 items to class 1 customers. In this case, the state of the system consists of (n_1,n_2,k) where k denotes the number of exponential stages (with rate $z\mu$) that have been completed on the item that is currently being processed. Then the completion time of each stage is exponentially distributed with rate $z\mu$. The decision whether to produce another unit or not is made only at production completion times (i.e., $k=0$). However, our formulation assumes that the decision maker can "observe" the number of exponential stages when making the decision to sell a class 2 item to a class 1 customer. Of course, in practice, this would not be the case. However, the decision maker can still solve the formulation we give below and then use only the solutions at $k=0$ as an approximation.

Let $v(n_1,n_2,k)$ be the relative value function of being in state (n_1,n_2,k) where k is the current stage of the item being processed. Then we have:

$$
\begin{aligned}
&g+v(n_1,n_2,k)\\
&\quad=\frac{1}{\Lambda}\Bigg\{-h(n_1+n_2)+\lambda_2[(v(n_1,n_2-1,k)+R_2)I_{n_2>0}\\
&\qquad+v(n_1,0,k)I_{n_2=0}]
\end{aligned}
$$

$$+\lambda_1 \max\begin{pmatrix}(v(n_1-1,n_2,k)+R_1)I_{n_1>0}+v(0,n_2,k)I_{n_1=0}\\(v(n_1,n_2-1,k)+R_1)I_{n_2>0}+v(n_1,0,k)I_{n_2=0}\end{pmatrix}$$
$$+z\mu W(n_1,n_2,k)\Big\}, \tag{7}$$

where $\Lambda = \lambda_1+\lambda_2+z\mu$ and

$$W(n_1,n_2,k) = \begin{cases}\max[v(n_1,n_2,1),v(n_1,n_2,0)] & \text{if } k=0,\\ v(n_1,n_2,k+1) & \text{if } 0<k<z-1,\\ p_1v(n_1+1,n_2,0)+p_2v(n_1,n_2+1,0) & \text{if } k=z-1.\end{cases}$$

$W(n_1,n_2,k)$ represents the transitions generated by a production stage completion opportunity. When the workstation is ready for processing ($k=0$), the firm needs to decide whether to start production or keep the workstation idle. After the completion of stage k, where $0<k<z-1$, the production advances to the next stage, $k+1$. At the end of the last stage, $z-1$, one unit of either item 1 or item 2 is produced.

We conjecture that the optimal policy for the system described by (7) has similar properties to those in Theorem 1. However, the policy for selling class 2 items to class 1 customers when $n_1=0$ is now defined by, z thresholds, $S_k, k=0,\ldots,z-1$ depending on the stage of the item in production. Furthermore, the relationship among these thresholds has the following property: $0\le S_k-S_l\le 1$, $l>k$. The interpretation of this property is that, when $n_1=0$, we are more willing to meet demands of class 1 customers with class 2 items if the item currently being processed is closer to completion. Similarly, we conjecture that the optimal production policy has exactly the same properties as in Theorem 1.

We can easily extend our heuristic for the original problem to the case where processing times are Erlang. In particular, the MDP for computing Q can be modified to the following:

$$g+v(n,k)=\frac{1}{\lambda+z\mu}\{-hn+\lambda[(v(n-1,k)+R)I_{n>0}+v(0,k)I_{n=0}]+z\mu W_Q(n,k)\},$$
$$n=0,1,2,\ldots;\quad k=0,\ldots,z-1, \tag{8}$$

where $\lambda=\lambda_1+\min\{\lambda_2,p_2\mu\}$,

$$R=\frac{\min\{\lambda_2,p_2\mu\}R_2+(\min\{\lambda_1+\lambda_2,\mu\}-\min\{\lambda_2,p_2\mu\})R_1}{\min\{\lambda_1+\lambda_2,\mu\}},$$

and

$$W_Q(n,k)=\begin{cases}\max[v(n,1),v(n,0)] & \text{if } k=0,\\ v(n,k+1) & \text{if } 0<k<z-1,\\ v(n+1,0) & \text{if } k=z-1.\end{cases}$$

The value of Q is the smallest value of n for which $v(n,0)>v(n,1)$.

The value of S can be obtained by solving the following MDP:

$$v(n_2,k)=\frac{1}{\Lambda}\Big\{-hn_2+\lambda_2[(v(n_2-1,k)+R_2)I_{n_2>0}+v(0,k)I_{n_2=0}]+\lambda_1\max\begin{pmatrix}(v(n_2-1,k)+R_1)I_{n_2>0}+v(0,k)I_{n_2=0}\\v(n_2,k)\end{pmatrix}+p_2z\mu W_S(n_2,k)\Big\},\quad n_2=0,1,2,\ldots;\ k=0,\ldots,z-1, \tag{9}$$

where $\Lambda=\lambda_1+\lambda_2+p_2z\mu$ and

$$W_S(n_2,k)=\begin{cases}\max[v(n_2,1),v(n_2,0)] & \text{if } k=0,\\ v(n_2,k+1) & \text{if } 0<k<z-1,\\ v(n_2+1,0) & \text{if } k=z-1.\end{cases}$$

Recall that (9) describes a system which only yields class 2 items. The terms multiplied by λ_1 represent the transitions generated by the arrivals of class 1 customers who are only willing to pay the lower price, R_1, for class 2 items. We have the option to accept or reject the orders depending on the outcome of the maximal function. By inspecting the solutions to (9), S_k, the threshold when production is in stage k, can be obtained such that S_k is the first value of $n_2>0$ for which $(v(n_2-1,k)+R_1)>v(n_2,k)$. Furthermore, it is easy to show that all S_k values differ at most by one.

The value of S to be used by the heuristic can be obtained by taking the average of the S_k values. As we described above, in practice, the user will not know the exponential stage in which the current item being produced is at. Therefore, it makes more sense to use an S value that does not depend on k. We therefore set S to be the average of all the S_k values. Having obtained both Q and S, the implementation of our heuristic is exactly as in the exponential processing time distribution case.

Finally, we note that the state spaces for the problems we have explored are vary large. As a result, it will not be practical to obtain optimal solutions for realistic sized problems. However, our heuristics result in fairly small state spaces, and thus provide results very quickly. In the next section, we report the results of the experiments we performed to test our heuristics.

6. Numerical results

We tested our heuristics, which we introduced in the previous three sections, on a variety of numerical examples. The results are compared to the performance of the optimal policies to show how well the heuristics perform.

Table 1 illustrates the results for the 22 examples we used to test our heuristic in the case of two quality classes. We use Example 1 as the base case. From Example 2 to 15, exactly one parameter is either increased or decreased in each example. Examples 16 to 22 cover highly unbal-

Table 1. Comparison of optimal and heuristic policies for the two class cases

Example	λ_1	λ_2	μ	p	h	R_1	R_2	*Optimal*	Q	S	*Heuristic*	*Percentage difference (%)*
1	0.2	0.2	0.3	0.4	5	500	1000	297.70	16	6	297.56	0.04
2	0.2	0.2	0.3	0.4	1	500	1000	329.75	67	17	329.75	0.00
3	0.2	0.2	0.3	0.4	10	500	1000	268.26	9	4	267.71	0.21
4	0.2	0.2	0.1	0.4	5	500	1000	153.21	26	15	153.21	0.00
5	0.2	0.2	0.5	0.4	5	500	1000	280.20	7	3	275.56	1.66
6	0.1	0.2	0.3	0.4	5	500	1000	256.12	8	6	255.12	0.39
7	0.3	0.2	0.3	0.4	5	500	1000	267.43	30	6	267.43	0.00
8	0.2	0.1	0.3	0.4	5	500	1000	259.61	8	3	257.34	0.88
9	0.2	0.3	0.3	0.4	5	500	1000	281.26	16	13	281.16	0.03
10	0.2	0.2	0.3	0.4	5	200	1000	241.35	14	8	241.31	0.02
11	0.2	0.2	0.3	0.4	5	800	1000	357.55	18	4	357.00	0.15
12	0.2	0.2	0.3	0.4	5	500	500	188.33	11	1	188.14	0.10
13	0.2	0.2	0.3	0.4	5	500	1500	417.73	21	10	417.46	0.06
14	0.2	0.2	0.3	0.2	5	500	1000	310.98	19	4	310.92	0.02
15	0.2	0.2	0.3	0.6	5	500	1000	246.80	9	10	245.06	0.70
16	0.2	0.2	0.3	0	5	500	1000	316.16	19	3	316.07	0.03
17	0.2	0.2	0.3	0.1	5	500	1000	313.91	19	4	313.82	0.03
18	0.2	0.2	0.3	0.9	5	500	1000	137.89	5	18	137.85	0.03
19	0.2	0.2	0.3	1	5	500	1000	112.32	4	21	112.32	0.00
20	1	0.1	0.3	0.4	5	500	1000	131.46	107	3	131.46	0.00
21	0.1	1	0.3	0.4	5	500	1000	117.95	8	83	117.08	0.74
22	0.2	0.2	0.3	0.4	5	500	5000	1300.19	54	26	1300.16	0.00

anced cases where the problem parameters for one class are significantly different than problem parameters for the other class. In Examples 16 to 19, p is set to either zero or one so that producing only one of the classes is possible. When $p_1 = 0$, the workstation only produces class 2 items. However, the firm still has the option of selling class 2 items to class 1 customers at the lower price, R_1. When $p_1 = 1$, the workstation only produces class 1 items and the firm is not able to satisfy demands from class 2 items. Notice that our heuristic is optimal when $p_1 = 1$. In Example 20, the arrival rate of class 1 customers is 10 times larger than that of class 2 customers, while in Example 21, $\lambda_2 = 10\lambda_1$. In the last example, $R_2 = 10R_1$. For each example, we report the value of the objective function achieved by the optimal policy as well as that achieved by our heuristic policy. We also report the (Q, S) values suggested by our heuristic. As it can be seen in Table 1, our heuristic performed extremely well and the average percentage difference in profit between the heuristic and the optimal policy was 0.23%.

We next tested our heuristic on problems with three quality classes. (As the size of the MDPs become extremely large as the number of quality classes are increased, three was the largest number of classes for which we could test our heuristics against the optimal policy). Once again, we used Example 1 as the base case and changed one parameter in each example. The average difference between the cost of the heuristic and the optimal cost was slightly larger in this case but the heuristic still performed very well with an average difference of 1.52%. Compared to the performance of the heuristic for the system with two quality classes, the average percentage difference increases. Since the policy for the system with three quality classes is more complicated, the heuristic policy is defined by four thresholds. The increase in percentage difference is understandable.

We next tested our heuristics in systems where production occurs in batches, or there is a setup cost or processing has an Erlang distribution. We used the same numerical examples as in Table 1, and we report the average and maximum percentage difference for the 22 examples in each case in Table 2. In the case where production occurs in batches, we assumed binomial yield (that is, the probability that each item in the batch is of class 1 is given by p_1). We tested our heuristics in examples where production occurs in batches of two or three. (We adjusted μ in Table 1 to $\mu/2$ in examples with batches of two and similarly to $\mu/3$ in examples with batches of three to keep the same overall production rate). The average percentage difference between our heuristics for cases with batch sizes of two or three and the cost of the optimal policy (which we obtained by solving the MDP) was less than 0.3%, and the maximum difference was below 2%.

We next tested our heuristic in the case where there is a setup cost to turn the machine on. We used costs of \$500, \$2000, and \$6000 for the setup cost, and used all of the examples in Table 1. Figure 3 displays the numerically

Table 2. Comparison of optimal and heuristic policies for systems with batch production, setup costs and Erlang distributions

	Average percentage difference (%)	*Maximum percentage difference (%)*
Producing in batches with batch size = 2	0.23	1.33
Producing in batches with batch size = 3	0.29	1.60
With setup cost $K = 500$	0.21	1.23
With setup cost $K = 2000$	0.26	1.90
With setup cost $K = 6000$	0.30	2.07
With Erlang-2 processing time distribution	0.29	2.85
With Erlang-4 processing time distribution	0.28	2.51
With Erlang-4 processing time distribution, setup cost $K = 2000$ and producing in batches with batch size = 3	0.37	1.83

obtained optimal policy (by solving the MDP) and our heuristic for Example 12 from Table 1 with a setup cost of \$500 to turn the machine on after it is turned off. It is interesting to note that our heuristic policy is very close to the optimal policy in most of the policy space. Notice that, our straight lines defined by Q_0 and Q_1 (which we compute using (6)) are very close to the optimal switching curves d_0 and d_1 except in cases where n_1 is very large. Once again, the average difference between the heuristic policy and the optimal policy in this case was less than 0.3% demonstrating that the performance is not very sensitive to slight shifts in the switching curves used. In this case, our heuristic obtained $S_0 = 1$ and $S_1 = 1$ to decide when to sell class 2 items to class 1 customers which were also the values used by the optimal policy.

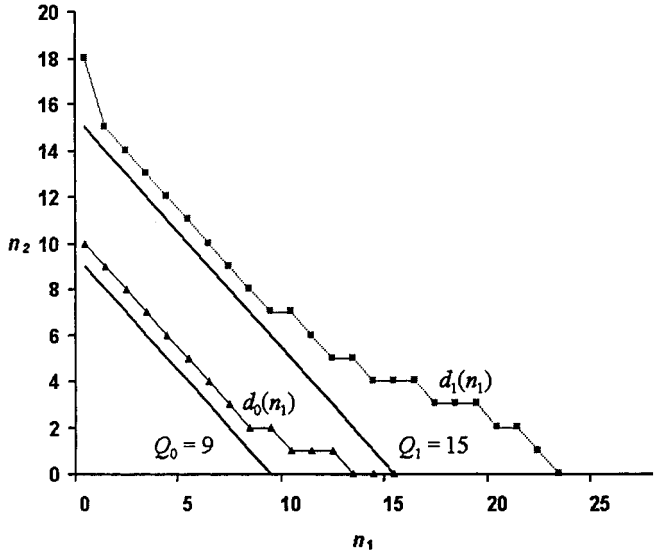

Fig. 3. The optimal production switching curves and heuristic Qs for the case with setup cost.

Finally, we tested our heuristic in the case of Erlang processing distributions. In this case, we are comparing the optimal value obtained by (7) to the heuristic defined in (8) and (9). Notice however that the optimal MDP assumes that the stage of the item being processed is observable whereas the implementation of our heuristic does not. Despite this, the difference between the performance of the heuristic and the values obtained by the MDP in (7) was very small. The average difference was once again below 0.3%.

All these examples clearly demonstrate that our heuristic is easily usable in a wide variety of practical situations. Our heuristic can accommodate production in batches, Erlang distributions as well as setup costs. In fact, as a final exercise, we tested the case where processing time distributions are Erlang-4, production occurs in batches of three and there is a setup cost of \$2000 to turn the machine on after it has been off. (We do not provide the formulations because they can easily be derived in the same manner). We again used the same examples as in Table 1 (with μ reduced by one-third) in order to keep the same yield rate in terms of the number of items produced. As reported in Table 2, the average percentage difference between the heuristic policy and the optimal policy was less than 0.4% and the maximum difference was below 2%.

7. Conclusions

In this paper, we considered a manufacturing system with uncertain product yields and downward substitutable demands. We were able to characterize the structure of the optimal policy for the problem with two quality classes of items by formulating the problem as a Markov Decision Process. Since the structure of the optimal policy is complicated, and computation or implementation of the optimal policy is not practical for more than three classes, we developed a simple heuristic policy. We also explored three extensions: the case where items are produced in batches, the case where there is a setup cost for starting production and the case with Erlang processing time distribution, and showed how the heuristic policy can be modified for those cases. We finally tested our heuristic against the optimal policy and the results showed that the heuristic performs very well. The fact that the heuristic policy can be computed and described easily makes it a good candidate for practical implementation.

Many extensions remain to be explored. In particular, characterizing the structure of optimal policies for the cases with batch production and setup times remains open. We have provided a conjecture for the case with

batch production but the proof of the result remains an open problem. Also, we assumed that the probability that a unit is of a given class is independent of the class that previous units belonged to. It would be interesting to extend our analysis to cases where yield distributions of consecutive units are allowed to be dependent.

References

Bitran, G.R. and Dasu, S. (1992) Ordering policies in an environment of stochastic yields and substitutable demands. *Operations Research*, **40**, 999–1017.

Bitran, G.R. and Gilbert, S.M. (1994) Co-production processes with random yields in the semiconductor industry. *Operations Research*, **42**, 476–491.

Bitran, G.R. and Leong, T. (1992) Deterministic approximations to co-production problems with service constraints and random yields. *Management Science*, **38**, 724–742.

Carr, S. and Duenyas, I. (1999) Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Operations Research*, (in press).

Gerchak, Y., Tripathy, A. and Wang, K. (1996) Co-production models with random functionality yields. *IIE Transactions*, **28**, 391–403.

Ha, A.Y. (1997a) Inventory rationing in a make-to-stock production system with several demand classes and lost sales. *Management Science*, **43**, 1093–1103.

Ha, A.Y. (1997b) Optimal dynamic scheduling policy for a make-to-stock production system. *Operations Research*, **45**, 42–53.

Hsu, A. and Bassok, Y. (1999) Random yield and random demand in a production system with downward substitution. *Operations Research*, **47**, 227–290.

Lippman, S. (1975) Applying a new device in the optimization of exponential queueing systems. *Operations Research*, **23**, 687–710.

Nahmias, S. and Moinzadeh, K. (1997) Lot sizing with randomly graded yields. *Operations Research*, **45**, 974–986.

Puterman, M.L. (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York.

Yano, C.A. and Lee, H.L. (1995) Lot sizing with random yields: a review. *Operations Research*, **43**, 311–334.

Appendix

Proof of Theorem 1

Our proof follows the framework of Ha (1997) in proving the required submodularity conditions required for the result. We also use the same notation.

For any real valued function t on the state space S, define the following:

$$T_1t(n_1,n_2) = (t(n_1,n_2-1)+R_2)I_{n_2>0} + t(n_1,0)I_{n_2=0},$$

$$T_2t(n_1,n_2) = \max[(t(n_1-1,n_2)+R_1)I_{n_1>0} + t(0,n_2)I_{n_1=0}, (t(n_1,n_2-1)+R_1)I_{n_2>0} + t(n_1,0)I_{n_2=0}],$$

$$T_3t(n_1,n_2) = \max[p_1t(n_1+1,n_2)+p_2t(n_1,n_2+1), t(n_1,n_2)],$$

$$Tt(n_1,n_2) = 1/\Lambda[-h(n_1+n_2)+\lambda_2T_1t(n_1,n_2)+\lambda_1T_2t(n_1,n_2)+\mu T_3t(n_1,n_2)],$$

$$D_1t(n_1,n_2) = p_1t(n_1+1,n_2)+p_2t(n_1,n_2+1)-t(n_1,n_2),$$

$$D_2t(n_1,n_2) = t(n_1+1,n_2)-t(n_1,n_2),$$

$$D_3t(n_1,n_2) = t(n_1+1,n_2)-t(n_1,n_2+1),$$

$$D_4t(n_1,n_2) = t(n_1,n_2+1)-t(n_1,n_2).$$

Let V be the set of functions on S such that if $t \in V$ then for every $R_2 \geq R_1 \geq 0$:

(i) $D_1t(n_1,n_2)$ is nonincreasing in n_1, n_2 and $\leq R_2$.
(ii) $D_2t(n_1,n_2)$ is nonincreasing in n_1 and $\leq R_1$.
(iii) $D_3t(n_1,n_2)$ is nonincreasing in n_1, nondecreasing in n_2, and ≤ 0.
(iv) $D_4t(n_1,n_2)$ is nonincreasing in n_2, and $\leq R_2$.

Lemma 1. *If $t \in V$ then T_1t, T_2t, T_3t, $Tt \in V$.*

Proof. For brevity, we only provide the proof that D_2 applied to the various function in nonincreasing in n_1. Proofs of other conditions are similar, and thus omitted.

1. We first show that $D_2T_1t(n_1,n_2)$ is nonincreasing in n_1. For any n_1, when $n_2=0$, $D_2T_1t(n_1,0) = D_2t(n_1,0)$, which is nonincreasing in n_1 because $t \in V$. For $n_2>0$, $D_2T_1t(n_1,n_2) = D_2t(n_1,n_2-1)$ which is also nonincreasing in n_1 followed by that $t \in V$.
2. We now show that $D_2T_2t(n_1,n_2)$ is nonincreasing in n_1. When $n_1=0$, $n_2=0$, we need to show that $D_2T_2t(0,0) \geq D_2T_2t(1,0)$.

$$D_2T_2t(0,0) = T_2t(1,0)-T_2t(0,0) = t(0,0)+R_1-t(0,0) = R_1,$$

$$D_2T_2t(1,0) = T_2t(2,0)-T_2t(1,0) = D_2t(1,0).$$

Therefore, $D_2T_2t(0,0) \geq D_2T_2t(1,0)$. When $n_2>0$ and $n_1=0$, we have,

$$D_2T_2t(0,n_2) = T_2t(1,n_2)-T_2t(0,n_2) = t(0,n_2)+R_1-\max[t(0,n_2-1)+R_1,t(0,n_2)].$$

There are two possible outcomes, $D_4t(0,n_2-1)$ and R_1. Also, note that

$$D_2T_2t(1,n_2) = T_2t(2,n_2)-T_2t(1,n_2) = t(1,n_2)+R_1-t(0,n_2)-R_1 = D_2t(0,n_2).$$

Therefore, we have one of the two following outcomes:

(i) $D_2T_2t(0,n_2)-D_2T_2t(1,n_2)$
$= D_4t(0,n_2-1)-D_2t(0,n_2),$
$\geq D_4t(0,n_2)-D_2t(0,n_2),$
$= t(0,n_2+1)-t(0,n_2)-t(1,n_2)+t(0,n_2),$
$= -D_3t(0,n_2) \geq 0.$

(ii) $D_2T_2t(0,n_2)-D_2T_2t(1,n_2) = R_1-D_2t(0,n_2) \geq 0.$

Therefore, $D_2T_2t(0,n_2) \geq D_2T_2t(1,n_2)$.
When $n_1>0$,

$$D_2T_2t(n_1,n_2) = D_2t(n_1-1,n_2),$$

which is nonincreasing in n_1.

3. We now show that $D_2T_3t(n_1,n_2)$ is nonincreasing in n_1. Since

$$\begin{aligned} &D_2T_3t(n_1,n_2)\\ &= \max[p_1t(n_1+2,n_2)+p_2t(n_1+1,n_2+1),t(n_1+1,n_2)]\\ &\quad -\max[p_1t(n_1+1,n_2)+p_2t(n_1,n_2+1),t(n_1,n_2)]. \end{aligned}$$

One of the four possible outcomes, $p_1t(n_1+2,n_2)+p_2t(n_1+1,n_2+1)-t(n_1,n_2)$, violates that D_1t is nonincreasing in n_1. The other three outcomes are $p_1D_2t(n_1+1,n_2)+p_2D_2t(n_1,n_2+1)$, $p_2D_3t(n_1,n_2)$, and $D_2t(n_1,n_2)$, and they are all nonincreasing in n_1.

4. Finally, we show that $D_2Tt(n_1,n_2)$ is nonincreasing in n_1. It can be easily seen that holding costs are nonincreasing in n_1 and Tt is otherwise constructed by addition and multiplication of positive constants with constituent functions (D_2T_1t, D_2T_2t, and D_2T_3t) which are all nonincreasing in n_1. ■

Now consider a value iteration algorithm to solve for the optimal policy in (1), in which $v_0(n_1,n_2)=0$ for every state n_1 and n_2 and $v_{k+1}(n_1,n_2)=Tv_k(n_1,n_2)$. Here, $v_k(n_1,n_2)$ can be viewed as the optimal value function when the problem is terminated after k transitions. We can now state the following

Lemma 2. *There exists an integer J, a constant g and a function v such that $v_{kJ+r}(n_1,n_2)-(kJ+r)g \to v(n_1,n_2)$ for all $r=0,1,\ldots,J-1$ as $k\to\infty$.*

Proof. We first note that, without loss of optimality, we can add the constraint to the original problem that we will never produce another unit when $h(n_1+n_2)/\Lambda > R_2$. This is because when this equation holds true, it implies that there is already so much stock in inventory that the cost of holding it until the next event is greater than any revenue that one could earn by selling the inventory. Therefore, the original problem can be converted to one with finite state space. The existence of finite action spaces and the fact that the model is unichain is sufficient for the lemma to hold by Theorem 8.4.5 of Puterman (1994). ■

To complete the proof of Theorem 1, we first note that by Lemmas 1 and 2, $v \in V$. Conditions (i) through (iv) are sufficient for the structural requirements on the optimal policy to hold as follows: The fact that $D_1v(n_1,n_2)$ is nonincreasing in n_1 and n_2 implies the existence and monotonicity of the production switching curve described in Property (i) of Theorem 1. The fact that we will always sell class 1 items to class 1 customers when $n_1>0$ follows from the fact that $D_3v(n_1,n_2)\le 0$. Finally, the existence of a threshold for selling class 2 items to class 1 customers when $n_1=0$ follows from the fact that $D_4v(n_1,n_2)$ is nonincreasing in n_2. ■

Biographies

Izak Duenyas is an Associate Professor of Operations Management at the University of Michigan. His research interests are in the optimal control and performance evaluation of complex manufacturing systems. He serves on the editorial boards of *Management Science*, *IIE Transactions on Scheduling and Logistics*, *IJFMS*, and *MSOM*.

Chi-Yang Tsai is a doctoral student in Industrial and Operations Engineering at the University of Michigan. He is interested in optimal control and scheduling of stochastic manufacturing systems.