Look-back policies for two-stage, pull-type production/inventory systems*

M. Baykal-Gürsoy, T. Altiok and H. Danhong
Department of Industrial Engineering, Rutgers University, Piscataway, NJ 08854, USA

We consider a two-stage, pull-type production/inventory system with a known service mechanism at the first stage. Set-ups and start-ups are involved in the operation of the second stage. We develop a production control policy for the second stage, within the class of (R, r) continuous-review policies, that minimizes the long run average total cost. We use a semi-Markov decision model to obtain an optimal policy for the operation of the second stage. The structure of the optimal policy suggests the use of a suboptimal look-back policy that delays the set-up at the second stage if the buffer lacks sufficient raw material. The performance of the system and the average total cost under the suboptimal policy can be obtained approximately using a decomposition algorithm. We show examples justifying the use of this suboptimal policy.

1. Introduction

This paper is concerned with the operation of a two-stage, pull-type production/inventory system as shown in fig. 1. It is assumed that the first stage has always raw material to process. Between the stages, there is an intermediate storage of work-in-process inventory that operates with the well-known base-stock policy. That is, the first stage produces as long as there is space in the buffer. The second stage produces to store in the finished-product warehouse. The demand for the finished product arrives at the warehouse on a random basis. It is assumed that a set-up cost is incurred every time production starts at stage 2. Moreover, a start-up charge is incurred when stage 2 becomes idle during a production cycle. Thus, a fresh start requires a set-up, and an intermediate starvation requires a start-up. As a result, stage 2 does not respond to every demand arrival at the warehouse. Rather, it initiates a production cycle when the inventory level in the warehouse drops to a certain value. Production at stage 2 continues until the inventory level in the warehouse reaches a target value, implying a continuous-review (R, r) policy

* This research is supported by the NSF Grant No. NSF-NCR-9110105, NSF Grant No. NSF-DDM-9014868 and by the North Atlantic Treaty Organization Grant No. NATO-CRG-900580.

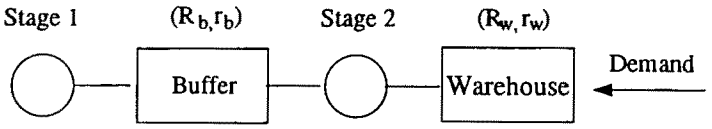

Fig. 1. A two-stage, pull-type production/inventory system.

where R is the target value and r is the reorder level. Furthermore, the production at stage 2 starts only if there is a sufficient amount of material in the intermediate buffer. This is what we refer to as the look-back policy. In the case of the lack of sufficient material in the buffer, the look-back policy forces stage 2 to stay idle until a certain amount accumulates in the buffer.

In pull-type systems, the downstream, in the hierarchy of production control, being closer to the market, has the final authority on how many units to produce. The production schedule is usually not set a priori, but the process is triggered as the finished inventory level reaches a critical point. Thus, in multi-stage, pull-type production systems, each stage produces as much as the immediate downstream stage requests. The contribution of this paper is to incorporate the state of readiness (to produce) of the upstream stages into the production decision at any stage in a pull-type system. We show through the cost minimization arguments that within a general continuous-review inventory policy, due to the set-up and start-up costs, it may be beneficial to delay the production at stage 2 until there is a sufficient amount of material in the upstream buffer.

Pull-type systems have been investigated in the literature. Most of the studies appear to be in the context of kanban systems. Kimura and Terada [18] study the effect of fluctuations in demand on the system performance measures. Karmarkar [17] conjectures about the control of pull systems and the impact of the variability of the inventory levels on the congestion in pull systems. Bitran and Chang [3], and Bard and Golany [5] extend the work by Kimura and Terada to optimize the number of kanbans for a deterministic future demand for a period of time.

So and Pinault [26], Buzacott [6], and Altiok and Ranjan [2] study multi-stage production/inventory systems where production at each stage is triggered by the demand coming from the immediate downstream stage. Zipkin [28] incorporates base-stock policies in a two-stage kanban-like production system and analyzes simple models to evaluate the average performance measures for various cases. Additional analytical models of kanban-like systems include Mitra and Mitrani [19], and Hopp and Spearman [16] focusing on the estimation of the average performance measures of tandem production systems.

Our objective in this study is to show that in pull-type systems, it may be cost effective to delay the production, triggered by the downstream demand, at the stage where there is a lack of work-in-process inventory in the upstream buffer. The gain in delaying the production results from the reduced number of set-ups and start-ups per unit time and consequently from the reduced average set-up and start-up costs

per unit time. In a two-stage system, for instance, this saving may be substantial to drive a policy of "looking back" that checks the buffer before a production cycle starts at stage 2. To be able to implement our objective in this research, we have modeled the above problem as a semi-Markov decision problem. We present an algorithm to obtain the optimal policy for stage 2. The structure of the optimal policy is quite complicated, it depends on the number of items both in the buffer and in the warehouse. When the set-up cost is high, the optimal policy is a type of look-back policy. For every warehouse state there is a buffer threshold such that stage 2 starts working if the buffer level is higher than this threshold. The threshold decreases with the decreasing inventory level in the warehouse. We present some examples which show that the optimal policy does not have this monotone structure in general. The structure of the optimal policy suggests the use of a look-back type suboptimal policy with the same threshold for every inventory level in the warehouse. We provide examples to justify the use of this suboptimal policy. We analyze the system approximately, under the suboptimal policy using a decomposition scheme. We give an algorithm to obtain the necessary performance measures for each buffer threshold and show the accuracy of the approximation method. The cost-minimizing buffer threshold can be obtained using the approximation method.

2. Model description

We consider the two-stage, single-product production/inventory system shown in fig. 1. All jobs are processed first in stage 1 and then in stage 2 and placed in the warehouse. Demand for the finished product is assumed to be governed by an independent time-homogeneous Poisson process with rate λ. Upon arrival, a demand is satisfied immediately if there are units available in the warehouse. Otherwise, the customer leaves, incurring a lost sale cost. Let X_i denote the random variable representing the processing time at stage $i = 1, 2$. X_i's are independent and exponentially distributed with rate μ_i. There is a fixed set-up cost for starting the production at stage 2. Later, during the production cycle, a start-up cost is charged if stage 2 becomes idle due to the lack of material in the buffer. A holding cost is charged per unit time for the inventory both in the buffer and the warehouse. The production at stage 1 and the inventory in the buffer are controlled by an $(R_b, R_b - 1)$ continuous-review policy implying that stage 1 should produce as long as the buffer is not full. A continuous-review (R_w, r_w) inventory control policy is used to control the inventory in the warehouse as well as the production at stage 2. This policy implies that if the inventory is below or equal to the reorder level r_w, then a request for production is placed at stage 2. The production continues until the warehouse inventory reaches R_w. The optimality of such policies in the presence of the start-up cost is shown for single-stage systems by Heyman [14], Sobel [25] and Bell [4].

The decision to start the production at stage 2 may depend on the sufficiency of the inventory in the intermediate buffer. At the moment of a production request

at stage 2, if there are not enough units in the buffer, stage 2 may delay its production until the inventory in the buffer reaches a threshold level. Below, we present a semi-Markov decision approach to identify the optimal production policy for stage 2.

2.1. SEMI-MARKOV DECISION MODEL

We observe the system at the following instances: arrival to buffer (a_b), arrival to warehouse (a_w) and demand arrival (d). The state of the system at any given epoch is determined by the inventory levels at the buffer and the warehouse and the state of stage 2, which may be in one of the following states: *working, waiting* and *forced-idle*. It is waiting to be activated in the *waiting* state, and is starved due to the lack of items in the buffer in the *forced-idle* state. These states are represented by $\{0, 1, 2\}$, respectively. If stage 2 is in the *forced-idle* state, and an item becomes available in the buffer, stage 2 goes through a *start-up*.

At each transition epoch, after observing the state of the system, an action is chosen from a set of available actions. At each decision epoch, as long as stage 2 is not working, it may either start working or wait. Thus, the action space for these states, is $\mathscr{A} = \{1, 0\}$, with 1 representing the *start working* action and 0 representing the *wait* action. Due to the (R_w, r_w) policy, once it starts working, incurring a set-up cost, it will work until the inventory reaches R_w. After reaching R_w, it remains idle until the inventory drops to r_w. Thus, for some states the action space will include only one action, so as to keep producing in the active states and remain idle in the idle states. We represent this action by 0, meaning that the action does not change the state of stage 2. If stage 2 is in the *forced-idle* state, as soon as an item enters the buffer, stage 2 will start working immediately. For those states, there is also only one action to choose, that is the action to start working. We denote this action by 1, meaning that this action changes the state of stage 2.

A policy is defined as a sequence of decision rules $\pi = (\pi_0, \pi_1, \ldots)$, where π_m is the vector of probability distribution on the actions available for every state at the decision epoch m. A policy is said to be stationary if it has the same decision rule at every epoch m and this decision rule depends only on the present state. A stationary policy is called deterministic if there is no randomization between the actions and only one action is chosen in any state.

By the above assumptions, given that the present state of the system is $\underline{i}$ and action a is taken, the probability that the next state is $\underline{j}$, i.e. the transition probability $P_{\underline{i}aj}$, is known. The time spent in each state is exponential with rates depending on the present state and the action taken. It is well known that this is a semi-Markov decision process (see, e.g., Ross [21]). In fact, it is a special type of semi-Markov decision process, which is called the continuous-time Markov decision chain [15]. It is known that under a stationary policy, the state process is a continuous-time Markov chain, thus, at the transition epochs it has an embedded Markov chain.

Let $I_b(t)$ and $I_w(t)$ be the inventory level in the buffer and in the warehouse, respectively, at time t given the initial state and the policy. For notational simplicity

we suppress the dependence of the state variables on the initial state and the policy. Let $S_2(t)$ denote the state of stage 2 at time t.

2.2. THE OPTIMIZATION PROBLEM

The long-run average total cost includes the holding cost with rate h, the one time charge of set-up cost k per set-up, the one time charge of penalty cost p per start-up and the lost sale cost l per lost customer. Thus, given that the initial state of the system is z and the policy π is applied, the average total cost per unit time becomes

$$V_\pi(z) = \limsup_{t\to\infty} \frac{1}{t} E_\pi\left[\int_0^t h[I_b(\tau) + I_w(\tau)]\,d\tau + kK(t) + pR(t) + lL(t)\right],$$

where E_π denotes the expectation operator with respect to policy π, $K(t)$ denotes the total number of set-ups, $R(t)$ denotes the total number of start-ups and $L(t)$ denotes the total number of customers with unsatisfied demand by time t. Notice that the buffer and the warehouse holding rates are assumed to be the same.

A policy π^* is said to be optimal if it attains the minimum cost value among all possible policies, i.e.

$$V_{\pi^*}(z) = \inf_\pi V_\pi(z) \quad \text{for all } z.$$

2.3. SYSTEM DYNAMICS

To locate the optimal policy we define a Markov chain embedded at each decision epoch m, with the following state representation

$$\underline{I}(m) = (I_b(m), I_w(m), S_2(m), E(m)),$$

where $E(m)$ is the event type taking values from $\{a_b, a_w, d\}$. The state space is therefore

$$\mathcal{S} = \{0, 1, \ldots, R_b\} \times \{0, 1, \ldots, R_w\} \times \{0, 1, 2\} \times \{a_b, a_w, d\}.$$

Note that the first three dimensions of the state representation is the state of the Markov chain just before the transition denoted by the event type. With this state description, the total number of states and the number of action states in the system are given by

$$\text{The total number of states} = 5R_w(R_b + 1) + 2R_b - r_w,$$

$$\text{The number of action states} = (2r_w + 3)(R_b - 1).$$

This state description is similar to the one used in Gopal and Stern [13]. The state description in Ross and Tsang [22] could also be used without decreasing the number of state–action pairs.

At each decision epoch m, given that the process is in state $\underline{i}$ and action a is chosen, we have

$$\begin{aligned}\tau(\underline{i}, a) &\triangleq \text{the expected time until the next epoch,}\\ c(\underline{i}, a) &\triangleq \text{the expected cost until the next epoch,}\\ P_{\underline{i}a\underline{j}} &\triangleq \text{the probability that the next state is } \underline{j}\\ &= P\{\underline{I}(m+1) = \underline{j} \mid \underline{I}(m) = \underline{i}, A(m) = a\}.\end{aligned}$$

To avoid the deadlock in the system, we assume that the production at stage 2 starts immediately when the buffer is full and the inventory in the warehouse reaches the reorder level. Since we have assumed the (R_w, r_w) policy for stage 2, the states where stage 2 is working will have only one action available, that is to continue to work, denoted by action $a = 0$ until $I_w = R_w$. For example, for the states where an arrival of a finished item occurs at the warehouse, $\underline{i} = (i_b, i_w, 1, a_w)$ with $0 < i_b < R_b$, and $0 \le i_w < R_w - 1$, we have

$$\tau(\underline{i}, 0) = \frac{1}{\lambda + \mu_1 + \mu_2}, \tag{1}$$

$$P_{\underline{i}0\underline{j}} = \begin{cases} \lambda\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b - 1, i_w + 1, 1, d),\\ \mu_1\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b - 1, i_w + 1, 1, a_b),\\ \mu_2\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b - 1, i_w + 1, 1, a_w), \end{cases} \tag{2}$$

$$c(\underline{i}, 0) = h(i_b + i_w)\tau(\underline{i}, 0), \tag{3}$$

while for the demand arrival states $\underline{i} = (i_b, i_w, 1, d)$, with $0 \le i_w < R_w$, and $i_b < R_b$, we have

$$\tau(\underline{i}, 0) = \frac{1}{\lambda + \mu_1 + \mu_2}, \tag{4}$$

$$P_{\underline{i}0\underline{j}} = \begin{cases} \lambda\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b, [i_w - 1]^+, 1, d),\\ \mu_1\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b, [i_w - 1]^+, 1, a_b),\\ \mu_2\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b, [i_w - 1]^+, 1, a_w), \end{cases} \tag{5}$$

$$c(\underline{i}, 0) = h(i_b + [i_w - 1]^+)\tau(\underline{i}, 0) + l\mathbf{1}(i_w = 0), \tag{6}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function and $[x]^+ = \max(x, 0)$.

Similarly, when stage 2 is idle, the states where $r_w < i_w \leq R_w$ and $i_b < R_b$ will have only one action available, that is, to continue to be idle (action $a = 0$). For example, for the demand arrival states $\underline{i} = (i_b, i_w, 0, d)$ with $r_w + 1 < i_w \leq R_w$, and $i_b < R_b$, we have

$$\tau(\underline{i}, 0) = \frac{1}{\lambda + \mu_1}, \tag{7}$$

$$P_{\underline{i}0\underline{j}} = \begin{cases} \lambda\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b, [i_w - 1]^+, 0, d), \\ \mu_1\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b, [i_w - 1]^+, 0, a_b), \end{cases} \tag{8}$$

$$c(\underline{i}, 0) = h(i_b + [i_w - 1]^+)\tau(\underline{i}, 0). \tag{9}$$

For the states where $i_w \leq r_w$, if there are units in the buffer, there are two actions available: $a = 0$ corresponding to the *wait* action, and $a = 1$ corresponding to the *start working* action. For example, for the arrival to the buffer states $\underline{i} = (i_b, i_w, 0, a_b)$ with $0 \leq i_b < R_b - 1$, and $i_w \leq r_w$, if we choose not to start, then we have

$$\tau(\underline{i}, 0) = \frac{1}{\lambda + \mu_1}, \tag{10}$$

$$P_{\underline{i}0\underline{j}} = \begin{cases} \lambda\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b + 1, i_w, 0, d), \\ \mu_1\tau(\underline{i}, 0) & \text{for } \underline{j} = (i_b + 1, i_w, 0, a_b), \end{cases} \tag{11}$$

$$c(\underline{i}, 0) = h(i_b + 1 + i_w)\tau(\underline{i}, 0)), \tag{12}$$

while, if we choose to start working, we have

$$\tau(\underline{i}, 1) = \frac{1}{\lambda + \mu_1 + \mu_2}, \tag{13}$$

$$P_{\underline{i}1\underline{j}} = \begin{cases} \lambda\tau(\underline{i}, 1) & \text{for } \underline{j} = (i_b, i_w, 1, d), \\ \mu_1\tau(\underline{i}, 1) & \text{for } \underline{j} = (i_b, i_w, 1, a_b), \\ \mu_2\tau(\underline{i}, 1) & \text{for } \underline{j} = (i_b, i_w, 1, a_w), \end{cases} \tag{14}$$

$$c(\underline{i}, 1) = h(i_b + i_w)\tau(\underline{i}, 1) + k. \tag{15}$$

Clearly, choosing action 1 whenever $i_w \leq r_w$ corresponds to the (R_w, r_w) continuous-review inventory control policy.

If the embedded Markov chain is unichain (with a set of recurrent states and a (possibly empty) set of transient states) for every deterministic policy, then the semi-Markov decision process is said to be unichain. It is clear that the process described above is unichain. It is well known that for a finite state, finite action, and unichain semi-Markov decision process, an optimal deterministic policy exists (see, e.g. Ross [21], Tijms [27] and Heyman and Sobel [15]). This policy can be located from the following fractional program:

$$\min \quad \frac{\sum_{\underline{i},a} c(\underline{i}, a) z(\underline{i}, a)}{\sum_{\underline{i},a} \tau(\underline{i}, a) z(\underline{i}, a)} \tag{16}$$

$$\text{subject to} \quad \sum_{\underline{i},a} z(\underline{i}, a) = 1, \tag{17}$$

$$\sum_{\underline{i},a} P_{\underline{i}a\underline{j}}\, z(\underline{i}, a) = \sum_{a} z(\underline{j}, a), \tag{18}$$

$$z(\underline{i}, a) \geq 0, \tag{19}$$

where $z(\underline{i}, a)$ corresponds to the fraction of transition times that the state is $\underline{i}$ and action a is chosen, i.e. the stationary probability distribution of the Markov chain. With the following substitution [7, 9],

$$y(\underline{i}, a) \triangleq \frac{z(\underline{i}, a)}{\sum_{\underline{i},a} \tau(\underline{i}, a) z(\underline{i}, a)},$$

the above fractional program corresponds to the linear program given below,

$$\min \quad \sum_{\underline{i},a} c(\underline{i}, a) y(\underline{i}, a) \tag{20}$$

$$\text{subject to} \quad \sum_{\underline{i},a} \tau(\underline{i}, a)\, y(\underline{i}, a) = 1, \tag{21}$$

$$\sum_{\underline{i},a} P_{\underline{i}a\underline{j}}\, y(\underline{i}, a) = \sum_{a} y(\underline{j}, a), \tag{22}$$

$$y(\underline{i}, a) \geq 0. \tag{23}$$

This linear program could also be obtained using the data transformation based on the uniformization technique for continuous Markov chains [24] (see also Schweitzer [23] for a general treatment). After finding a solution $\{\mathbf{y}\}$ to this program, the optimal stationary policy is obtained from,

$$P\{A = a | I = \underline{i}\} = \frac{y(\underline{i}, a)}{\sum_{a} y(\underline{i}, a)}, \tag{24}$$

Table 1
Optimal policy.

		h	l	k	p	Opt. cost	Policy
Light traffic	$\lambda = 1$	0.5	0.5	0.5	0.125	2.2503	p1
	$\mu_1 = 1$	10.0	500.0	500.0	125.0	179.0606	p2
	$\mu_2 = 2$	10.0	50.0	500.0	125.0	122.0732	p3
		10.0	0.5	5000.0	1250.0	758.3884	p4
Heavy traffic	$\lambda = 2$	0.5	500.0	50.0	10.0	514.9945	pp1
	$\mu_1 = 2$	10.0	50.0	500.0	100.0	97.9614	p2
	$\mu_2 = 1$	0.5	500.0	5000.0	1000.0	578.9747	p3
		0.5	50.0	5000.0	1000.0	117.1838	p4

for the recurrent states $\underline{i}$, i.e. $\Sigma_a y(\underline{i}, a) > 0$. For the transient states, i.e. $\Sigma_a y(\underline{i}, a) = 0$, any arbitrary action which takes the transient state into a recurrent state is optimal. Note that the optimal solution $\{y(\underline{i}, a)\}$ will be positive for exactly one action a, for every recurrent state. Thus, the optimal policy is a deterministic policy. The algorithm to obtain the optimal policy is given below.

- **Step 1**: Generate the state space according to the system dynamics.
- **Step 2**: Calculate, $\tau(\underline{i}, a)$, $P_{\underline{i}a\underline{j}}$ and $c(\underline{i}, a)$ for every $\underline{i}$ and a.
- **Step 3**: Solve the linear program (20)–(23) using a linear programming method.
- **Step 4**: Obtain the optimal deterministic policy for every recurrent state by identifying the actions such that (24) is positive. For the transient states choose an action which will take the transient state into the set of recurrent states.

Table 2
The structure of the optimal policy

(I_b, I_w)	p1	p2	p3	p4	(I_b, I_w)	p1	p2	p3	p4
(1, 3)	wait	wait	wait	wait	(1, 1)	action	wait	wait	wait
(2, 3)	wait	wait	wait	wait	(2, 1)	–	wait	wait	wait
(3, 3)	wait	wait	wait	wait	(3, 1)	–	action	wait	wait
(4, 3)	action	action	action	action	(4, 1)	–	–	action	action
(1, 2)	action	wait	wait	wait	(1, 0)	action	wait	wait	wait
(2, 2)	action	wait	wait	wait	(2, 0)	–	action	wait	wait
(3, 2)	action	wait	wait	wait	(3, 0)	–	–	action	wait
(4, 2)	action	action	action	action	(4, 0)	–	–	–	action

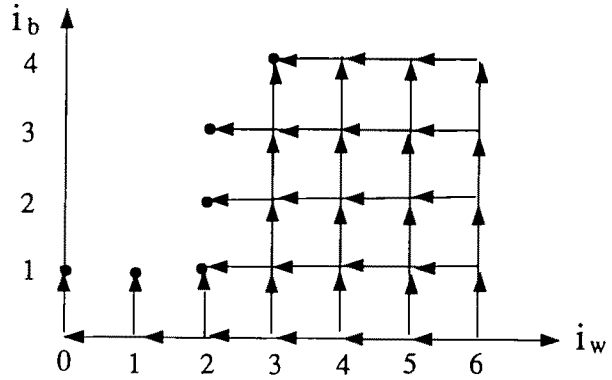

Fig. 2. Policy p1.

EXAMPLES

Let us look at some examples with $R_b = 4$ and $R_w = 6$ with $r_w = 3$. In this set of examples, we have considered both the heavy traffic case and the light traffic case. For varying values of the cost parameters, we have obtained the optimal objective function values and the optimal policies as shown in table 1.

In table 2, the p's denote the optimal policies for each example. p1 is the no look-back policy with the new reorder level $r_w = 2$. pp1 is the exact no look-back policy with the original $r_w = 3$. Specifically, the optimal policies have the following structure as shown in table 2 where (–) implies that the probability of being in state (I_b, I_w) is 0, thus, denotes a transient state.

We can use the following diagraph to present the policy structure more clearly. In these diagraphs, the up arrow (↑) represents an arrival to the buffer, the left arrow (←) represents a demand arrival and a dot (·) represents the action

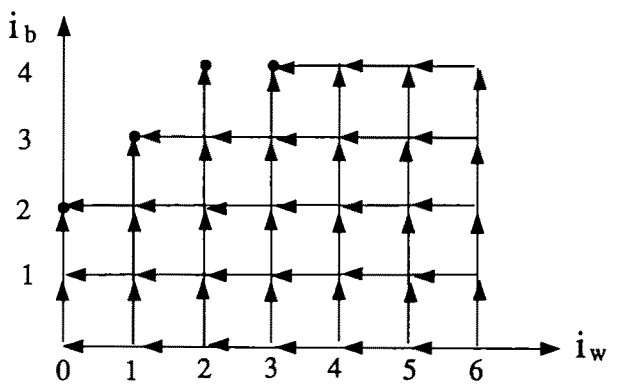

Fig. 3. Policy p2.

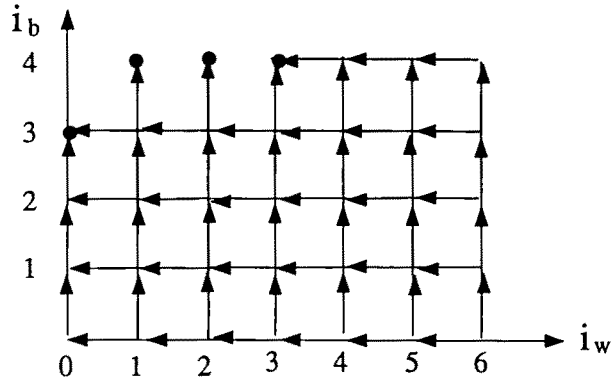

Fig. 4. Policy p3.

point. Figure 2 represents the policy structure for p1. It clearly shows that, as soon as the inventory in the warehouse drops to 2 and there are items in the buffer, stage 2 starts working. We call this kind of policy a no look-back policy. p1 policy looks like the no look-back policy with the warehouse reorder level equal to 2, except at state (4, 3). At state (4, 3), stage 2 starts working, since stage 2 is forced to work whenever the buffer is full and the inventory in the warehouse reaches the reorder level.

Figure 3 represents the policy structure for p2, the inventory in the buffer at each action state is different and the threshold value decreases with the decreasing warehouse state. When the warehouse state is low, stage 2 starts working to avoid lost sales, while for the high warehouse states, stage 2 waits until the inventory in the buffer reaches a certain value to decrease the number of set-ups and start-ups.

In policy p3 (see fig. 4), stage 2 starts working when the inventory in the warehouse is zero and the inventory in the buffer is 3. Otherwise, stage 2 will wait until the buffer is full. Policy p4 is similar to p3. But, because the set-up and start-up costs are so high, stage 2 will not start working until the buffer is full, even when there are no items in the warehouse.

In general, though, the optimal policy structure can be quite different from the policies that are presented previously. For example, policy p5 does not demonstrate a monotonic structure (see fig. 5). This policy is optimal when the same system and cost parameters as in the case where policy p1 is optimal are used but the inventory holding cost is increased to 10. When the inventory level reaches the reorder level $i_w = 3$, stage 2 starts working if the buffer level is equal to 2 or if the buffer is full, but it does not start if the buffer level is equal to 3. This policy structure corresponds to a no-threshold policy.

Because of the complexity of the optimal policy, we next consider a simple policy that is easy to implement without incurring a "much" higher cost. We refer to this suboptimal policy as the look-back policy.

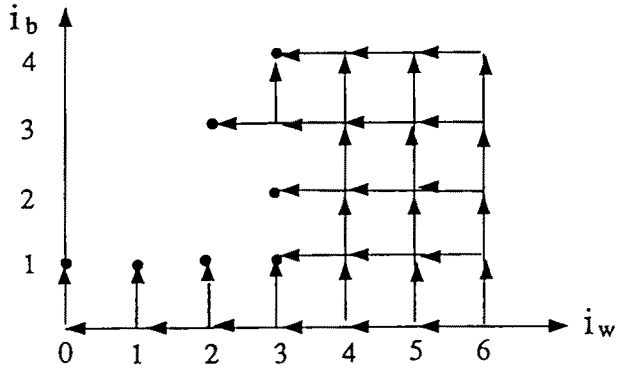

Fig. 5. Policy p5.

The look-back policy dictates that as soon as the inventory in the warehouse drops to r_w, stage 2 checks the inventory level in the buffer. If it is greater than or equal to r^*, $r^* = 1, 2, \ldots, R_b$, stage 2 starts working, otherwise, it waits until the inventory reaches r^*.

Clearly, there exist R_b choices of r^*. Under any r policy which is a policy that assigns r as the threshold level, the embedded Markov chain is unichain. Thus, given a set of cost parameters, one can calculate the optimal r^* by

$$r^* = \arg_{r=1,2,\ldots,R_b} \min V_r,$$

through the analysis of the underlying Markov chain. Table 3 demonstrates the differences between the long-run average expected cost value of the optimal policy, the suboptimal look-back policy and the no look-back policy. There is no substantial difference between the optimal and the suboptimal objective function values while the difference between the look-back and the no look-back policies could be quite significant.

Table 3
The average total cost of the optimal, the look-back and the no-look-back policies.

		h	l	k	p	Opt. cost	Subopt. cost	r^*	No look-back
Light traffic	$\lambda = 1$	0.5	0.5	0.5	0.125	2.2503	2.2519	1	2.2519
	$\mu_1 = 1$	10.0	500.0	500.0	125.0	179.0606	181.3235	3	183.4167
	$\mu_2 = 2$	10.0	50.0	500.0	125.0	122.0732	122.1016	4	128.6415
		10.0	0.5	5000.0	1250.0	758.3884	758.3884	4	845.6867
Heavy traffic	$\lambda = 2$	0.5	500.0	50.0	10.0	514.9945	514.9945	1	514.9945
	$\mu_1 = 2$	10.0	50.0	500.0	100.0	97.9614	97.9874	2	98.0088
	$\mu_2 = 1$	0.5	500.0	5000.0	1000.0	578.9747	579.0528	3	579.5016
		0.5	50.0	5000.0	1000.0	117.1838	117.1838	4	118.4046

3. Approximate analysis of the system under look-back policy

The pull system we have considered in this paper may be approximately analyzed using a decomposition scheme similar to the ones developed by Altiok and Ranjan [1], Gun and Makowski [12], Gershwin [11] and Dallery et al. [8]. Our contribution here is the incorporation of the look-back policy into the approximation. Below, we describe the approximation method and show its accuracy.

The approximation is based on replicating the behavior of the buffer and the warehouse contents in the systems that are easier to analyze. Hence, the two-stage production/inventory system can be decomposed into two subsystems as shown in fig. 6. Let us abbreviate the systems replicating the behavior of the buffer content by $\Omega(1)$ and the warehouse content by $\Omega(2)$.

3.1. DESCRIPTION OF $\Omega(1)$

$\Omega(1)$ is a two-stage system with an intermediate buffer of capacity R_b. The first stage has an exponential processing time with rate μ_1 and is always busy except when the buffer is full. It starts processing when I_b drops to $R_b - 1$. Let $P\{I_i = k\} = P_i(k)$ for $i = b, w$, be the steady-state probability that there are k units in the buffer and in the warehouse, respectively, at any point in time.

The second stage in $\Omega(1)$ has a more involved processing time. Upon process completion at stage 2 in the original system, if the number of items in the warehouse

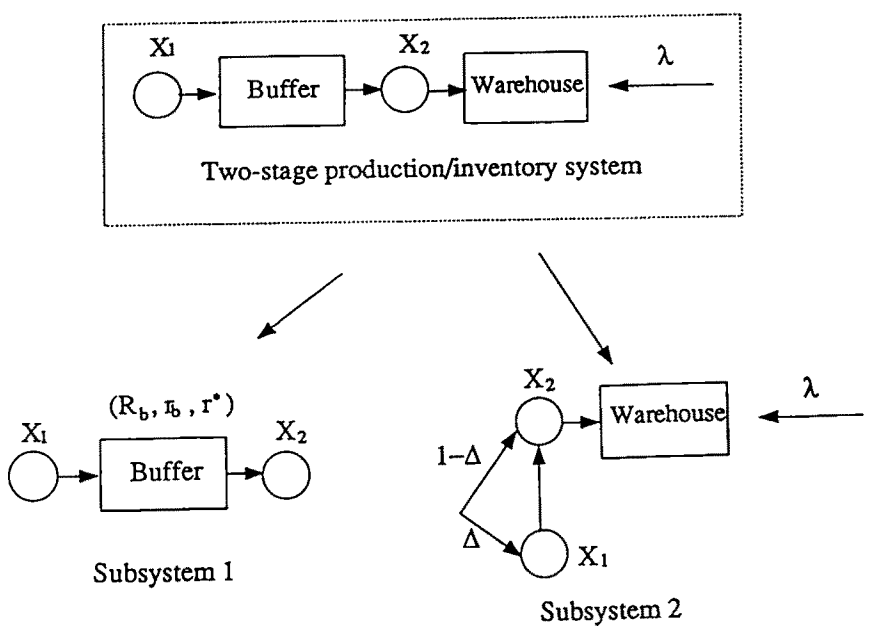

Fig. 6. Decomposition of the two-stage production system.

is R_w, stage 2 ceases its production. The probability of this event is

$$\Pi = P\{I_w = R_w - 1 \,|\, \text{a departure occurs at node 2}\}$$

Once the target level in the warehouse is reached, there have to be $R_w - r_w$ demand arrivals to occur to initiate a request for production. That is, the particular demand arrival that sees $r_w + 1$ in the warehouse and reduces it to r_w initiates the request for production. At this moment, the production starts with probability $P\{I_b \geq r^*\}$. Otherwise, the production at stage 2 is delayed until $I_b = r^*$. This delay may be interpreted as the *set-up time* for stage 2. Let S denote this set-up time for stage 2 and Z denote the time that stage 2 is idle from the moment I_w reaches R_w until stage 2 restarts its production. We call Z as the *idle time*. Then,

$$E[Z] = \frac{(R_w - r_w)}{\lambda} + P\{I_b < r^*\} E[S], \tag{25}$$

where

$$E[S] = \frac{1}{\mu_1} \sum_{k=1}^{r^*} k P_b(r^* - k). \tag{26}$$

Let $P_2 = P\{S_2 = \text{idle}\}$ be the steady-state probability that stage 2 is in the *idle* state. Then, Π can be obtained using the renewal arguments as

$$\Pi = \frac{P_2}{E[Z]\bar{o}_2}, \tag{27}$$

where $\bar{o}$ is the steady-state throughput in $\Omega(2)$.

Also, we know that stage 2 will be idle while I_w drops from R_w to r_w or the inventory in the buffer is less than r^*. Thus, P_2 can be expressed as

$$P_2 = (R_w - r_w) P_w(R_w) + \sum_{k=0}^{r^*-1} P_b(k). \tag{28}$$

The processing time at node 2 of $\Omega(1)$ is defined by

$$X_{12} = \begin{cases} X_2 & \text{with probability } 1 - \Pi, \\ X_2 + Z & \text{with probability } \Pi, \end{cases} \tag{29}$$

with an expected value of

$$E[X_{12}] = \frac{1}{\mu_2} + \frac{P_2}{\bar{o}_2}. \tag{30}$$

The distribution of Z can be modeled by a mixture of generalized Erlang distribution (MGE) with $(r^* + 1)$ phases where the first phase has a rate of μ_2, second phase has a rate of $\lambda/(R_w - r_w)$ and all the others have the rate of μ_1. Then, $\Omega(1)$ can be studied as an $M/PH/1/R_b + 1$ queue with a processing time of phase-type with $(r^* + 2)$ phases. This queueing system can be analyzed by using matrix-geometric or matrix-recursive techniques (see, e.g. Neuts [20]).

3.2. DESCRIPTION OF $\Omega(2)$

$\Omega(2)$ is a single-stage production/inventory system replicating the behavior of the warehouse contents. We analyze this system from the empty cell point of view. In the original system, once the production starts at stage 2 (after a possible set-up time), it continues until $I_w = R_w$. During this period, a departure at stage 2 may leave $I_b = 0$. In this case, stage 2 waits for stage 1 to eject a unit into the buffer. The probability of this event is,

$$\Delta \triangleq P\{I_b = 0 \mid \text{a departure occurs at stage 2}\} = \frac{\mu_1 P_b(0)}{\bar{o}_1}, \tag{31}$$

where $\bar{o}_1$ is the steady-state throughput of $\Omega(1)$.

Thus, the imaginary processor producing the finished products to be stored in the warehouse has a processing time X_{21} defined by

$$X_{21} = \begin{cases} X_2 & \text{with probability } 1 - \Delta, \\ X_2 + X_1 & \text{with probability } \Delta, \end{cases} \tag{32}$$

with an expected value of

$$E[X_{21}] = \frac{1}{\mu_2} + \frac{P_b(0)}{\bar{o}_1}. \tag{33}$$

Notice also that at a request for production, in order to incorporate the look-back policy, stage 2 has to check the inventory level in the buffer. If it is less than r^*, stage 2 has to wait until it reaches r^*. We interpret this waiting time as a *start-up time* for stage 2. The start-up time S is an Erlang random variable with c^* phases. This c^* value is determined by the inventory level in the buffer at the moment of a demand arrival (Poisson arrival). For example, if the inventory level in the buffer is higher than or equal to r^*, then $c^* = 0$. If the inventory level in the buffer is less than r^*, then, $c^* = r^* - I_b$. The probability distribution of c^* can be expressed as

$$P\{c^* = r^* - j \mid j < r^*\} = \frac{P_b(j)}{P\{I_b < r^*\}}. \tag{34}$$

Table 4
The accuracy of the decomposition algorithm.

R_w	r_w	R_b	r_b	r^*	λ	μ_1	μ_2		$\bar{I}$	$\bar{K}$	$\bar{L}$	$\bar{R}$
10	6	6	5	1	1	1	2	Appr.	6.7946	3.8036×10^{-2}	0.07244	0.3699
								Exact	6.8232	5.3307×10^{-2}	0.07396	0.3227
6	3	4	3	2	1	1	2	Appr.	4.4196	6.6738×10^{-2}	0.14009	0.3339
								Exact	4.2378	8.6075×10^{-2}	0.12239	0.2852
10	6	6	5	3	2	2	1	Appr.	6.5187	4.9906×10^{-4}	1.00491	7.81×10^{-3}
								Exact	6.0147	4.1695×10^{-4}	1.00467	7.66×10^{-3}
10	6	6	5	4	2	1	2	Appr.	1.9531	4.8112×10^{-4}	1.00456	0.5005
								Exact	1.4560	4.5395×10^{-4}	1.00403	0.5003

A convenient way to model $\Omega(2)$ is to assume that X_{21} is an $MGE-2$ random variable. Consequently, $\Omega(2)$ becomes an $M/PH/1/R_w$ queue with a start-up time and a threshold-service policy. That is, the server waits until $(R_w - r_w)$ units accumulate. It may go through a start-up time (due to the look-back policy) and starts producing until the system is cleared. The units in this $M/PH/1/R_w$ queue are the empty cells (holes) in the original warehouse.

Let us briefly describe an iterative algorithm that relates $\Omega(1)$ to $\Omega(2)$, providing approximate values of the steady-state measures of the system.

- **Step 1**: Initialize Π and $E[Z] = (R_w - r_w)/\lambda$.
- **Step 2**: Analyze $\Omega(1)$ as an $M/PH/1/R_b + 1$ queue to obtain $P_b(\cdot)$ and $\bar{o}_1$.
- **Step 3**: Obtain Δ from (31).
- **Step 4**: Analyze $\Omega(2)$ as an $M/PH/1/R_w$ queue to obtain $P_w(\cdot)$ and $\bar{o}_2$.
- **Step 5**: If $|\bar{o}_2 - \bar{o}_1| \leq \epsilon$, stop. Otherwise, obtain Π from (27) and go to step 2.

The above iterative procedure always converges and has an acceptable error level as shown in table 4. Low level of production activity tends to reduce the rate of convergence. The convergence of the algorithm is guaranteed according to the reasoning given by Dallery and Frein [10].

3.3. PERFORMANCE MEASURES

From the analyses of $\Omega(1)$ and $\Omega(2)$, we can obtain the expected values of the performance measures in our objective function as follows:

$$\bar{I}_j = \sum_{i=0}^{i=R_j} iP_j(i), \quad \text{for } j = b, w, \tag{35}$$

$$\bar{L} = E\left[\lim_{t\to\infty}\frac{L(t)}{t}\right] = \lambda P_w(0), \tag{36}$$

$$\bar{K} = E\left[\lim_{t\to\infty}\frac{K(t)}{t}\right] = \lambda P_w(R_w), \tag{37}$$

$$\bar{R} = E\left[\lim_{t\to\infty}\frac{R(t)}{t}\right] = \bar{o}\Delta. \tag{38}$$

Notice that these measures depend on the threshold value r^*. The expected number of lost sales per unit time is obtained by using the PASTA (Poisson Arrivals See Time Averages) property of the Poisson distribution. Hence, for a given set of system parameters and cost coefficients, we can evaluate the average total cost using the approximate values of the above performance measures. Table 5 shows the exact and the approximate values of the performance measures for varying values of the system parameters. The accuracy of the approximation is quite reasonable.

In table 5, we compare the exact and approximate values of r^* and the associated average total cost. The results appear to be highly satisfactory with a relative error range of (0.0023, 0.1677) in the total cost. Considering the practicality of the sub-optimal policy with respect to the complexity of the optimal policy, one may prefer to live with the above error level.

4. Justification of the look-back policy

In this section, we will briefly justify the use of suboptimal policy by making reference to the parameters of the approximation. Let us choose two values for r^* namely r_1^* and r_2^* such that

$$r_1^* < r_2^* \leq R_b.$$

Table 5
The comparison of the exact and the approximate values of r^*.

	h	l	k	p	Exact r^*	V (exact)	Appr. r^*	V (appr.)
$\lambda = 1$	0.5	0.5	0.5	0.125	1	2.2519	4	2.2021
$\mu_1 = 1$	10.0	500.0	500.0	125.0	3	181.3235	1	188.8067
$\mu_2 = 2$	10.0	50.0	500.0	125.0	4	122.1016	4	120.9519
	10.0	0.5	5000.0	1250.0	4	758.3834	4	760.1205
$\lambda = 2$	0.5	500.0	50.0	10.0	1	514.9945	2	509.5881
$\mu_1 = 2$	10.0	50.0	500.0	100.0	2	97.9874	4	100.8751
$\mu_2 = 1$	0.5	500.0	5000.0	1000.0	3	579.0528	4	554.0160
	0.5	50.0	5000.0	1000.0	4	117.1838	4	97.5359

We can write down the expected total cost expression as follows:

$$V_{r_i^*} = h\bar{I}_b(r_i^*) + h\bar{I}_w(r_i^*) + l\lambda P_w(0, r_i^*) + k\lambda P_w(R_w, r_i^*) + p\bar{o}(r_i^*)\Delta(r_i^*). \tag{39}$$

Hence, an analysis of the cost expression that shows

$$V_{r_1^*} > V_{r_2^*}$$

may justify the look-back policy.

As r^* strictly increases, clearly $E[S]$ increases which in turn increases $E[Z]$ as apparent from (25)–(26). Hence, from (30) we can write

$$E[X_{12}, r_1^*] < E[X_{12}, r_2^*],$$

which results in

$$\bar{I}_b(r_1^*) < \bar{I}_b(r_2^*).$$

This can be shown not only by treating $\Omega(1)$ as an $M/M/1/N$ queueing system, but also becomes intuitively correct.

A simple argument can be used for $\bar{I}_w$. Since $\bar{o}$ drops down as r^* increases, the finished-product input rate to the warehouse decreases whereas the demand rate remains the same and consequently the average contents of the warehouse decreases, that is,

$$\bar{I}_w(r_1^*) > \bar{I}_w(r_2^*).$$

The number of set-ups per unit time $\bar{K}$ decreases as r^* increases,

$$\bar{K}(r_1^*) > \bar{K}(r_2^*).$$

As r^* increases, clearly $P_w(0)$ increases (the warehouse will be emptier) and consequently,

$$\bar{o} = [1 - P_w(0)]\lambda,$$

decreases. Meanwhile, as r^* increases, Δ (the probability that stage 2 is starving) decreases. Thus, the number of start-ups per unit time $\bar{R}$ shows the following behavior:

$$\bar{R}(r_1^*) > \bar{R}(r_2^*).$$

Finally, as r^* increases, $P_w(0)$ increases so that the number of lost sales per unit time $\bar{L}(t)$ shows,

$$\bar{L}(r_1^*) < \bar{L}(r_2^*).$$

We can combine the above arguments in the average total cost expression:

$$V_{r_1^*} - V_{r_2^*} = h[\bar{I}_b(r_1^*) - \bar{I}_b(r_2^*)] + h[\bar{I}_w(r_1^*) - \bar{I}_W(r_2^*)] + l\lambda[P_w(0, r_1^*) - P_w(0, r_2^*)] + k[\bar{K}(r_1^*) - \bar{K}(r_2^*)] + p[\bar{o}(r_1^*)\Delta(r_1^*) - \bar{o}(r_2^*)\Delta(r_2^*)]. \quad (40)$$

It is clear that, under any circumstances, r^* can be chosen so that (40) becomes positive. The magnitude of (40) closely provides a threshold for r^* to justify the look-back policy. Obtaining explicit expressions for the differences in (40) are outside the scope of this paper.

References

[1] T. Altiok and R. Ranjan, Analysis of production lines with general service times and finite buffers: A two-node decomposition approach, Eng. Costs Prod. Econ. 17 (1989) 155–165.

[2] T. Altiok and R. Ranjan, Multi-stage, pull-type production/inventory systems, Technical report, Dept. of IE, Rutgers Univ. (1991).

[3] G. Bitran and L. Chang, A mathematical programming approach to a deterministic kanban system, Manag. Sci. 33 (1987) 427–441.

[4] C. Bell, Characterization and computation of optimal policies for operating an $M/G/1$ queueing system with removable server, Oper. Res. 19 (1971) 208–218.

[5] J. Bard and B. Golany, Determining the number of kanbans in a multi-product, multi-stage production system, Int. J. Prod. Res. 29 (1991) 881–895.

[6] J. Buzacott, Queueing models of kanban and MRP controlled production systems, Eng. Costs Prod. Econ. 17 (1989) 3–20.

[7] A. Charnes and W.W. Cooper, Programming with linear fractional functionals, Naval Res. Log. Quarterly 9 (1962) 181–186.

[8] Y. Dallery, R. David and X.L. Xie, An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers, IIE Trans. 20 (1988) 280–283.

[9] C. Derman, On sequential decisions and Markov chains, Manag. Sci. 9 (1962) 16–24.

[10] Y. Dallery and Y. Frein, On decomposition methods for tandem queueing networks with blocking, Oper. Res. 41 (1993) 386–399.

[11] S.B. Gershwin, An efficient decomposition algorithm for unreliable tandem queueing systems with finite buffer, in: *Proc. 1st. Int. Workshop on Queueing Networks with Blocking*, eds. H. Perros and T. Altiok (North-Holland, 1989).

[12] L. Gun and A. Makowski, An approximation method for general tandem queueing systems subject to blocking, in: *Proc. 1st. Int. Workshop on Queueing Networks with Blocking*, eds. H. Perros and T. Altiok (North-Holland, 1989).

[13] I.S. Gopal and T.E. Stern, Optimal cell blocking policies in an integrated services environment, *Conf. on Information Sciences and Systems* (1983) pp. 383–388.

[14] D.P. Heyman, Optimal operating policies for $M/G/1$ queueing systems, Oper. Res. 16 (1968) 362–382.

[15] D.P. Heyman and M.J. Sobel, *Stochastic Models in Operations Research, Vol. II: Stochastic Optimization* (McGraw–Hill, New York, 1982).

[16] W. Hopp and M. Spearman, Throughput of a constant WIP manufacturing line subject to failures, Int. J. Prod. Res. 29 (1991) 635–655.

[17] U.S. Karmarkar, Kanban systems, Working Paper Series QM8612, The Graduate School of Management, University of Rochester, Rochester, NY (1986).
[18] O. Kimura and H. Terada, Design and analysis of a pull system. A method of multi-stage production control, Int. J. Prod. Res. 19 (1981) 241–253.
[19] D. Mitra and I. Mitrani, Analysis of a kanban discipline for cell coordination in production lines. I, Manag. Sci. 36 (1990) 1548–1566.
[20] M.F. Neuts, *Matrix-Geometric Solution in Stochastic Models* (Johns Hopkins University Press, Baltimore, 1981).
[21] S. Ross, *Applied Probability Models with Optimization Applications* (Holden-Day, San Francisco, 1971).
[22] K.W. Ross and D.H.K. Tsang, Optimal circuit access policies in an ISDN environment: A Markov decision approach, IEEE Trans. Commun. COM-37 (1989) 934–939.
[23] P.J. Schweitzer, Iterative solution of the functional equations of undiscounted Markov renewal programming, J. Math. Anal. Appl. 34 (1971) 495–501.
[24] R. Serfozo, An equivalence between continuous and discrete Markov decision processes, Oper. Res. 27 (1979) 616–620.
[25] M. Sobel, Optimal average cost policy for a queue with start-up and shut-down costs, Oper. Res. 17 (1969) 145–162.
[26] K. So and S. Pinault, Allocating buffer storages in a pull system, Int. J. Prod. Res. 26 (1988) 1959–1980.
[27] H.C. Tijms, *Stochastic Modeling and Analysis: A Computational Approach* (Wiley, Chichester, 1986).
[28] P. Zipkin, A kanban-like production control system: Analysis of simple models, Technical Report 89-1, Business School, Columbia Univ. (1989).