IIE Transactions (2000) **32**, 1027–1041

Control of manufacturing networks which contain a batch processing machine

JOHN J. NEALE[1] and IZAK DUENYAS[2]

[1]*Department of Industrial and Operations Engineering, and*
[2]*The Business School, University of Michigan, Ann Arbor, MI 48109, USA*
E-mail: duenyas@umich.edu

Received September 1998 and accepted August 1999

We consider the control of a batch processing machine which is part of a larger manufacturing network of machines. Systems consisting of a batch processing machine and one or more unit-capacity machines in tandem are considered. The objective is to minimize the average time that jobs spend in the entire system. We present algorithms to determine the optimal policies for certain finite horizon, deterministic problems. We then discuss the structure of the optimal policies for infinite horizon, stochastic problems, and investigate the benefit of utilizing information about upstream and downstream unit-capacity machines in the control of the batch machine. We develop a simple heuristic scheduling policy to control the batch machine which takes into account the state of other machines in the network. Computational results demonstrate the effectiveness of our heuristic over a wide range of problem instances.

1. Introduction

A batch processing machine is a single machine that can process up to a certain number of jobs (the capacity of the machine) at the same time. This type of machine is found in many manufacturing and service environments. The most commonly cited examples are diffusion tubes and burn-in ovens in semiconductor manufacturing, but other examples include plating baths, heat-treating furnaces in the metalworking industries, kilns for drying lumber, material handling and transportation vehicles, oven and metallization steps in flat panel display manufacturing, and bus, subway, and elevator operations (Deb and Serfozo, 1973; Glassey and Weng, 1991; Powell and Humblet, 1986). The control of a single batch processing machine has received significant attention in the recent production literature. However, in most practical applications a batch machine is part of a larger network of machines responsible for manufacturing a product. While the analysis of a single batch machine in isolation is useful for providing methods which serve as subprocedures in a global control methodology, it is necessary to study the entire network to understand the interactions between machines.

In this paper we consider the control of a batch processing machine that is part of a larger network of machines. We consider networks that consist of a batch processing machine and one or more unit-capacity machines in a series arrangement. A unit-capacity machine is one which processes a single job at a time. Ahmadi *et al.* (1992) and Gurnani *et al.* (1992) describe many examples of manufacturing systems that contain both batch and unit-capacity machines. One example is semiconductor wafer fabrication, in which a photolithography expose operation (unit-capacity) is followed by the operation of diffusion in a furnace (batch).

We consider both finite horizon, deterministic problems and infinite horizon, stochastic problems. We assume that all jobs belong to the same job family (i.e., have the same processing requirements). The processing time of a batch is independent of the number of jobs in the batch. Once processing on a machine is initiated, it cannot be interrupted. We further assume that the jobs arrive to the first machine in the network dynamically over time. The buffers in front of each machine are assumed to be unlimited. If the batch processing machine is available and the number of jobs ready to be processed on it is less than its capacity, a non-trivial control decision must be made to either process a partial batch immediately or wait for additional jobs to arrive. The tradeoff involved is utilization of the batch capacity versus delay of jobs currently in the queue. The objective considered in this work is to minimize the average time that jobs spend in the entire system, or equivalently to minimize the average number of jobs in the system. This objective is desirable because long production cycle times can have an adverse effect on product yield, meeting promised delivery dates, responsiveness to changes in market demand, and inventory holding costs. In highly competitive industries

like the semiconductor industry, short production cycle times represent a significant competitive advantage.

To be able to refer to the problems under study in this paper in a concise manner, we will use the notation of Ahmadi *et al.* (1992) to describe the networks. Let β denote a batch processing machine, δ a unit-capacity machine (which they refer to as a *discrete* processor), and $\rightarrow$ denote the system configuration. For example, $\beta \rightarrow \delta$ represents a two-machine system with a batch processing machine feeding a unit-capacity machine, and $\delta \rightarrow \delta \rightarrow \beta$ represents a three-machine system with a unit-capacity machine feeding another unit-capacity machine feeding a batch processing machine.

In the next section we present a review of previous related work. In Section 3 we present algorithms to determine the optimal policies for certain finite horizon, deterministic problems. We discuss the structure of the optimal policies for infinite horizon, stochastic problems in Section 4, and investigate the benefit of information about upstream and downstream unit-capacity machines in the control of the batch machine. We also develop a simple heuristic scheduling policy to control the batch machine that takes into account the state of other machines in the network. Finally, Section 5 contains some concluding remarks and identifies future research directions.

2. Related work

The control of a manufacturing network that contains a batch processing machine does not seem to have been examined extensively in the production literature to date. To the best of our knowledge only two papers have considered this topic. Ahmadi *et al.* (1992) examine the static problems of minimizing the makespan and the sum of job completion times in two-machine flowshops containing batch and unit-capacity machines. They analyze problem complexity, present polynomial time procedures for some problems, and propose a heuristic for the NP-complete problem of minimizing $\sum C_j$ for a $\beta \rightarrow \delta$ system. Unlike our work in Section 3, they do not consider non-zero job release dates. They do, however, allow job-dependent processing times on the unit-capacity machine. Gurnani *et al.* (1992) consider a two-stage system consisting of multiple unit-capacity machines in parallel feeding a single batch processing machine. Their objective is to minimize the sum of a delay cost, a fixed service cost, and a capacity utilization cost at the batch machine. They assume a control limit policy for loading the batch machine and use a renewal approximation to compute an approximately optimal control limit. This control limit policy does not take into account the current state of the upstream unit-capacity machines.

While the problem of controlling a network that contains a batch machine has not received significant attention, many authors have considered batch processing machines in isolation. A server capable of processing several customers simultaneously is called a bulk server in queueing terminology, and there is extensive literature in queueing theory on bulk processing. Most of this literature, however, focuses on performance evaluation rather than control. Chaudhary and Templeton (1983) devote an entire book to bulk arrival and bulk service queues. Neuts (1967) studies the characteristics of a bulk service queue with Poisson arrivals of a single job family. He proposes a control limit policy for controlling this queue. Deb and Serfozo (1973) and Barnett and Kleitman (1978) prove that a control limit policy minimizes the sum of the holding costs and the service costs for this system. Medhi (1975) derives the waiting time distribution for this system under a control limit policy and exponential service times. Powell and Humblet (1986) consider this system in the context of less-than-truckload trucking networks and develop efficient computational procedures to describe the characteristics of the queue. Makis (1985) considers the case where the waiting time of any job cannot exceed a constant T.

The problem of scheduling a single batch processing machine has received attention recently in the deterministic scheduling literature. This work is largely motivated by the operations of diffusion and burn-in in semiconductor manufacturing (Uzsoy *et al.*, 1992, 1994) provide a comprehensive review of production planning and scheduling models in the semiconductor industry). These papers consider three cases. In the first case, all jobs belong to the same job family. Ikura and Gimple (1986), Lee *et al.* (1992), and Li and Lee (1997) consider different performance measures for this case in the presence of release dates and dues dates. The second case considered is multiple, incompatible job families. In this case jobs have different processing requirements on the batch machine and only jobs with the same processing requirements can be served together in a batch. This case models the operation of diffusion. Uzsoy (1995) and Mehta and Uzsoy (1998) consider the problems of minimizing the makespan, the maximum lateness, and the total tardiness on batch machines with incompatible families. The third case considered is multiple, compatible job families. In this case jobs with different processing requirements can be mixed together in the same batch, with the processing time of that batch being determined by the job in the batch with the longest processing time. This case models the operation of burn-in. Lee *et al.* (1992) and Lee and Uzsoy (1999) present optimal algorithms and heuristics for a number of scheduling criteria for the case of compatible families. Chandru *et al.* (1993a, 1993b) and Hochbaum and Landy (1997) address the problem of minimizing the total completion time. Uzsoy and Yang (1997) extend this to the total weighted completion time. Uzsoy (1994) considers jobs with non-identical capacity requirements. Brucker *et al.* (1998) consider regular performance measures for bounded and unbounded models

and present several results on the complexity of batch machine scheduling problems.

The highly stochastic environment of semiconductor manufacturing has led to recent work in the semiconductor manufacturing literature to address batch processing under stochastic assumptions. Avramidis *et al.* (1998) develop computational procedures to determine optimal threshold policies for a batch machine subject to uncertain job arrivals and processing times. Glassey and Weng (1991), Fowler *et al.* (1992), and Weng and Leachman (1993) develop heuristics that use information about future arrivals to minimize waiting time in queue for the single and multiple incompatible family cases. Duenyas and Neale (1997) partially characterize the optimal policies and develop heuristics for a stochastic batch machine with incompatible job families. Neale and Duenyas (1997) do the same for the compatible job families case.

In a recent paper, Robinson *et al.* (1995) address the issue of the use of upstream and downstream information in scheduling semiconductor batch operations. Their approach to the problem is different than ours. In particular, they develop heuristics that use information on the times when jobs will arrive from the upstream machines to the batch machine and the time when the downstream machine will be starved. However, they assume that arrival times to the batch machine from the upstream machine are known (although they allow for some prediction errors) and also that the time when the downstream machine is starved is predicted with no error. In contrast, we assume that we only have information on the number of jobs at each machine. This permits us to compare the effect of information from upstream and downstream machines on the optimal objective function. Our results on the optimal objective value function also confirm Robinson *et al.*'s observation (which they obtained using their heuristic) that information on upstream machines result in much greater improvements in time spent in system than information on downstream machines.

3. Finite horizon, deterministic problems

In this section, we examine the problem of minimizing the sum of completion times (or equivalently the average completion time) over a finite time horizon for networks containing a batch processing machine. We assume that there are N jobs to be processed through the network and that each job j arrives to the first machine in the network at time r_j. In addition, we arrange the r_j's in ascending order so that $r_j \leq r_{j+1}$ for $j = 1, \ldots, N-1$. All jobs belong to the same family and require a deterministic processing time p_B on the batch machine and a deterministic processing time p_{U_i} on the ith unit-capacity machine in the network (when there is only one unit-capacity machine in the network, we will drop the subscript i). The buffers in front of each machine are assumed to be unlimited. We let K denote the capacity of the batch machine and $(x)^+$ represent $\max\{x, 0\}$. When there is more than one unit-capacity machine in the network, we will refer to the ith unit-capacity machine as δ_i.

3.1. *A $\delta \rightarrow \beta$ system*

It is easy to show with a simple interchange argument that it is optimal to process each job on the unit-capacity machine at the first time when the machine is available after that job's arrival time. Inserted idle time on δ will never produce a benefit. If more than one job is waiting when the machine becomes available, we assume the job to be processed is selected by the First-In-First-Out (FIFO) rule. Note that the FIFO rule is not required for optimality. Any waiting job could be selected. The FIFO assumption, however, reduces the complexity of the problem while still producing an optimal schedule. Let a_j denote the time at which job j completes processing at δ and is thus available for processing at β. Then $a_1 = r_1 + p_U$, and the remaining a_j's can be determined by the recursive equation:

$$a_j = \max\{r_j, a_{j-1}\} + p_U \quad \text{for } j = 2, \ldots, N. \tag{1}$$

Given the a_j's it remains to determine an optimal schedule at the batch machine so that the sum of the completion times is minimized. When the number of jobs available for processing at β is less than the machine's capacity, a non-trivial decision must be made to either process the partial batch immediately or wait for additional arrivals. We note that the $O(N^3)$ dynamic programming algorithm developed by Ahmadi *et al.* (1992) for the "batch dispatching problem" could be used to determine an optimal schedule at β for this problem. However, we present an alternative dynamic programming algorithm of similar time complexity which is slightly less complex to describe and implement. Our dynamic programming algorithm is based on the following two observations:

1. Decisions are only required when a job arrives and the batch machine is available or when the batch machine completes service and there are jobs waiting.
2. If the decision to serve a batch is made, the number of jobs served in that batch is the minimum of the number of jobs waiting and the capacity of the machine. If all waiting jobs cannot be served in the batch, then the jobs to be served can be selected arbitrarily.

Algorithm DP1

Let t represent the current time (decision epoch) and $n(t)$ the number of jobs that have been processed on δ but are waiting to be processed on β at time t. The pair $(t, n(t))$ constitutes the state of our dynamic program. By observation 1, it follows that the only times at which a decision

is necessary are either when a job has arrived to the batch machine or when the batch machine has just completed a service. Thus, we need only consider values of t of the form $a_j + kp_B$ for $j = 1, \ldots, N$ and $k = 0, \ldots, N-1$. These N^2 times represent all possible arrival times and service completion times. Clearly $N \geq n(t) \geq 0$, so the total number of states is $O(N^3)$.

For each state, the decision is whether to serve a batch of the available jobs or idle until the next arrival to the batch machine. Let $f(t, n(t))$ denote the minimum sum of completion times from time t until the end of the horizon (i.e., until all jobs have been completed) given that $n(t)$ jobs are waiting at the batch machine. Then

$$f(t,n(t)) = \min \begin{cases} f(\min\{a_j : a_j > t\},\ n(t)+1), \\ \min\{n(t), K\}(t+p_B) \\ \quad + f\big(t+p_B, (n(t)-K)^+ \\ \quad + \big|\{j : t < a_j \leq t+p_B\}\big|\big), \end{cases}$$

where $|\{j : t < a_j \leq t + p_B\}|$ represents the number of jobs that arrive to the batch machine during the batch service time. The first term in the minimization represents the decision to idle until the next arrival, while the second term represents the decision to serve a batch of the waiting jobs. Since there are $O(N^3)$ states and a comparison of two values is considered for each state, the complexity of the algorithm is $O(N^3)$. We feel that this formulation is slightly more intuitive than the dynamic programming formulation of Ahmadi *et al.* (1992). Rather than simply deciding whether to idle or to serve, their recurrence relation decides the number of continuous batches to be served without any idle time. Our formulation also avoids the side calculations required by Ahmadi *et al.*'s formulation to determine the number of jobs that would be served in each batch.

The boundary conditions

$$\begin{aligned} \min\{a_j : a_j > t\} &= \infty \quad \text{for } t \geq a_N, \\ f(\infty, \bullet) &= \infty, \\ f(\bullet, n(t)) &= \infty \quad \text{for } n(t) > N, \end{aligned}$$

$$f(t,n(t)) = \begin{cases} 0 & \text{if } n(t) = 0 \quad \text{for } \infty > t \neq a_j + kp_B, \\ \infty & \text{if } n(t) \geq 1 \quad j = 1, \ldots, N,\ k = 0, \ldots, N-1, \end{cases}$$

are required to start our backward DP algorithm. The minimum total completion time for the $\delta \to \beta$ system is given by $f(a_1, 1)$.

3.2. *A $\beta \to \delta$ system*

Again, when the number of jobs available for processing at the batch machine is less than the machine's capacity, a non-trivial decision must be made to either process the partial batch immediately or wait for additional arrivals. However, for this system a job's completion time on the batch machine is not the job's completion time for the entire system. The state of the downstream unit-capacity machine must be considered when controlling the batch machine. Ahmadi *et al.* (1992) show that the problem of minimizing the sum of completion times for a $\beta \to \delta$ system without release times ($r_j = 0$ for all j) but with job-dependent processing times on δ is NP-complete in the strong sense. We develop a pseudo-polynomial time algorithm for a $\beta \to \delta$ system operating under our assumptions by modifying Algorithm DP1.

It is easy to show that inserted idle time at the unit-capacity machine is never optimal. As a result, the unit-capacity machine is kept busy as long as there are jobs available to be processed. Since all jobs have the same processing time at δ, the order in which the available jobs are processed on δ does not affect the sum of completion times. The following dynamic programming algorithm will minimize the sum of completion times for a $\beta \to \delta$ system. Without loss of generality, we assume that the release times r_j and the processing times p_B and p_U are integer values.

Algorithm DP2

Again, let t represent the current time (decision epoch) and $n(t)$ represent the number of jobs available to be processed on β at time t. For a $\beta \to \delta$ system we must also keep track of the amount of work waiting to be performed at δ. Let $w(t)$ be the total remaining processing time of all jobs waiting to be processed at δ at time t. The three values $(t, n(t), w(t))$ constitute the state of our dynamic program. Once again decisions are only necessary at β at either a job arrival or service completion time, so we need only consider the N^2 values of t of the form $r_j + kp_B$ for $j = 1, \ldots, N$ and $k = 0, \ldots, N-1$. Clearly $N \geq n(t) \geq 0$ and $Np_U \geq w(t) \geq 0$, so the total number of states is $O(N^4 p_U)$.

For each state, the decision is again whether to serve a batch of the available jobs or idle until the next arrival to the batch machine. Let $f(t, n(t), w(t))$ denote the minimum sum of completion times from time t until the end of the horizon for all jobs not yet served at β given that $n(t)$ jobs are waiting at β and $w(t)$ time units of processing are waiting at δ. Then

$$f(t,n(t),w(t)) = \min \begin{cases} f(r(t),\ n(t) + |\{j : r_j = r(t)\}|, \\ \quad (w(t) - r(t) + t)^+), \\ S(t,n(t),w(t)) + \ f(t+p_B,\ (n(t)-K)^+ + a(t), \\ \quad (w(t) - p_B)^+ + B(n(t))p_U), \end{cases}$$

where

$$\begin{aligned} r(t) &= \min\{r_j : r_j > t\}, \\ B(n(t)) &= \min\{n(t), K\}, \\ S(t,n(t),w(t)) &= \sum_{i=1}^{B(n(t))} (t + \max\{p_B, w(t)\} + ip_U), \text{ and} \\ a(t) &= |\{j : t < r_j \leq t + p_B\}|. \end{aligned}$$

Notice that $r(t)$ represents the time of the next arrival to β, $B(n(t))$ represents the number of jobs that would be served at β, $S(t, n(t), w(t))$ represents the sum of completion times for the $B(n(t))$ jobs served at β, and $a(t)$ represents the number of jobs that arrive to β during β's service time. The first term in the minimization again represents the decision to idle until the next arrival, while the second term represents the decision to serve a batch of the waiting jobs. Since there are $O(N^4 p_U)$ states and a comparison of two values is considered for each state, the complexity of the algorithm is $O(N^4 p_U)$.

The boundary conditions

$$\begin{aligned} \min\{r_j : r_j > t\} &= \infty \quad \text{for} \quad t \geq r_N, \\ f(\infty, \bullet, \bullet) &= \infty, \\ f(\bullet, n(t), \bullet) &= \infty \quad \text{for} \quad n(t) > N, \\ f(\bullet, \bullet, w(t)) &= \infty \quad \text{for} \quad w(t) > Np_U, \end{aligned}$$

$$f(t, n(t), \bullet) = \begin{cases} 0 & \text{if } n(t) = 0 \quad \text{for } \infty > t \neq r_j + kp_B, \\ & \qquad\qquad\quad j = 1, \ldots, N, \\ \infty & \text{if } n(t) \geq 1 \quad k = 0, \ldots, N-1, \end{cases}$$

are required to start our backward DP algorithm. The minimum total completion time for the $\beta \to \delta$ system is given by $f(r_1, 1)$.

3.3. *Systems with multiple unit-capacity machines*

Consider a $\delta \to \beta \to \delta$ system. As for the $\delta \to \beta$ system, it is easy to show that it is optimal to process each job on δ_1 at the first time when the machine is available after that job's arrival to the system. Since all jobs have identical processing requirements on each machine, it does not matter in which order the jobs are processed if there is more than one job waiting at δ_1 (we will assume a FIFO order). Thus, Equation (1) can again be used to determine the times at which each job is available for processing at the batch machine. Given the a_j's, the remaining task is to determine the schedule for β and δ_2 that minimizes the sum of the completion times for the entire system. Notice that this remaining problem is equivalent to the problem of minimizing the total completion time for a $\beta \to \delta$ system and Algorithm DP2 can be used (with the r_j's replaced by the a_j's) to solve this remaining problem. Consequently, the problem of minimizing the sum of completion times for a $\delta \to \beta \to \delta$ network can be solved in $O(N^4 p_{U2})$ operations.

For networks with multiple upstream unit-capacity machines feeding a single downstream batch machine, it is easy to show that the optimal policy at each of the upstream unit-capacity machines is to process a job whenever one is available. As a result, Equation (1) can be used for each unit-capacity machine δ_i to determine the release time of each job to the next unit-capacity machine δ_{i+1}. This process will eventually produce the arrival time for each job to the batch machine, and as we saw in Section 3.1 the problem of scheduling jobs on the batch machine can be solved in $O(N^3)$ operations. Thus, a fixed number of additional upstream unit-capacity machines does not add complexity to the problem of minimizing the sum of completion times.

This is not the case, however, for networks with multiple downstream unit-capacity machines. As we saw in Section 3.2, the amount of processing time at a downstream unit-capacity machine must be taken into account when scheduling jobs on the batch machine. The work at each additional downstream unit-capacity machine must be included as part of the state for a dynamic programming algorithm, and thus each additional downstream machine δ_i increases the complexity of the problem by Np_{U_i}.

4. Infinite horizon, stochastic problems

In this section, we examine the problem of minimizing the long-run average number of jobs in the system (or equivalently the long-run average time that jobs spend in the system) over an infinite time horizon for networks containing a batch processing machine. We assume that jobs arrive to the first machine in the network according to a Poisson process with rate λ. All jobs belong to the same job family and require an exponentially distributed processing time with mean $1/\mu_B$ on the batch machine and an exponentially distributed processing time with mean $1/\mu_{U_i}$ on the ith unit-capacity machine in the network. All processing and interarrival times are assumed to be independent and the buffers in front of each machine are assumed to be unlimited. Again we let K denote the capacity of the batch processing machine and $(x)^+$ represent $\max\{x, 0\}$. We define the traffic intensity at the batch machine (TI_B) to be $\lambda/K\mu_B$. Note that this is the traffic intensity or utilization of the batch machine when jobs are served in batches of size K. Because we may serve partial batches, the "effective" traffic intensity will likely be higher. Similarly, the traffic intensity at the ith unit-capacity machine (TI_{U_i}) is defined to be λ/μ_{U_i}. As a necessary condition for stability, we assume that the traffic intensities at all machines are less than one.

The following notation will be used throughout this section to describe the current state of the system:

n_{U_i} = the total number of jobs at the ith unit-capacity machine (includes all jobs waiting for service plus any job currently being served);
n_B = the number of jobs waiting in queue to be processed at the batch machine;
n_S = the number of jobs currently being processed at the batch machine ($n_S = 0$ if the batch machine is currently idle).

4.1. *A $\delta \to \beta$ system*

Clearly it is never optimal to idle at the unit-capacity machine when jobs are available for processing. Addi-

tionally, since all jobs have the same processing requirements on each of the machines, the order in which jobs are served does not affect the objective. Consequently, no control decisions are necessary at the unit-capacity machine. δ is simply kept busy whenever jobs are available and jobs are served according to the FIFO rule. The control of the batch processing machine, however, is not so simple. When the number of jobs available for processing at β is less than the machine's capacity, a non-trivial control decision must be made to either process the partial batch immediately or wait for additional arrivals. We formulate the problem of controlling the batch machine to minimize the long-run average number of jobs in the entire system as a Markov Decision Process (MDP).

The process is reviewed at those points in time when a job arrives to the network or a machine completes a service. As a result of the memoryless property of the exponential distribution, the time between consecutive review points is the minimum of exponential random variables which is itself an exponential random variable. As in Lippmann (1975), we define the uniform rate $\Lambda = \lambda + \mu_U + \mu_B$ so that the time between review points is exponentially distributed with constant parameter Λ and the probability of the next review point being an arrival is λ/Λ, the probability of the next review point being a δ service completion is μ_U/Λ, and the probability of the next review point being a β service completion is μ_B/Λ. The optimal long-run average number of jobs per period in the system, g, and the relative values of the various starting states, $V(n_U, n_B, n_S)$, must satisfy the following dynamic programming equations:

$$
\begin{aligned}
&V(n_U, n_B, 0) + g \\
&\quad = \min \begin{cases}
1/\Lambda[n_U + n_B + \lambda V(n_U + 1, (n_B - K)^+, \\
\quad \min\{n_B, K\}) + \mu_U V((n_U - 1)^+, (n_B - K)^+ \\
\quad + \min\{n_U, 1\}, \min\{n_B, K\}) \\
\quad + \mu_B V(n_U, (n_B - K)^+, 0)], \\
1/\Lambda[n_U + n_B + \lambda V(n_U + 1, n_B, 0) \\
\quad + \mu_U V((n_U - 1)^+, n_B + \min\{n_U, 1\}, 0) \\
\quad + \mu_B V(n_U, n_B, 0)],
\end{cases}
\end{aligned} \tag{2}
$$

for $n_U \geq 0$ and $n_B \geq 0$, and

$$
\begin{aligned}
&V(n_U, n_B, n_S) + g \\
&\quad = \frac{1}{\Lambda}[n_U + n_B + n_S + \lambda V(n_U + 1, n_B, n_S) \\
&\qquad + \mu_U V((n_U - 1)^+,\ n_B + \min\{n_U, 1\}, n_S) \\
&\qquad + \mu_B V(n_U, n_B, 0)],
\end{aligned} \tag{3}
$$

for $n_U \geq 0$, $n_B \geq 0$, and $K \geq n_S \geq 1$.

Note that equation (2) applies when the batch machine is currently empty and Equation (3) applies when the batch machine is currently processing a batch. When the batch machine is empty, two actions are possible. The first term in the minimization in Equation (2) represents the decision to serve at the batch machine. If this action is selected, β will serve a batch of size $\min\{n_B, K\}$ and all jobs will remain in the system until the next event which changes the state of the system (an expected time of $1/\Lambda$). At the next event the system will be in one of three different states, and the relative value of each of these states is weighted by the probability of a transition to that state. Similarly, the second term in the minimization in Equation (2) represents the decision to idle at the batch machine until the next event. Note that if this action is selected, our formulation allows for a fictitious service completion at the batch machine so that the time until the next event remains exponentially distributed with parameter Λ. This fictitious service completion, however, does not change the state of the system.

A value iteration algorithm can be used to solve the above Markov decision process for specific values of the problem parameters λ, K, μ_U, and μ_B. We used such an algorithm to solve a number of sample problems and can draw some conclusions about the structure of the optimal policies for a $\delta \rightarrow \beta$ system from these numerically solved examples. We initially conjectured that the optimal policy at the batch machine would have the following form: for each n_U, a control limit $l(n_U)$ exists such that it is optimal to serve if $n_B \geq l(n_U)$ and to idle if $n_B < l(n_U)$; furthermore, the control limits $l(n_U)$ are increasing in n_U. While many of the numerous examples we solved had this intuitively appealing structure, we found counter-examples to the conjecture of monotonic control limit values.

For example, the optimal policy for controlling the batch machine when $\lambda = 1.00$, $K = 7$, $\mu_U = 2.50$, and $\mu_B = 0.48$ consists of non-monotonic control limits as shown in Fig. 1. Note that as n_U increases from one to two the optimal control limit *decreases* from seven to five. While this result is at first surprising, we believe that an intuitive explanation exists. When $n_U = 1$, the queue in front of the unit-capacity machine is empty. The job j that is currently in service at δ will be available for processing at β after its service is completed (an expected time of 0.4 for this example). Before the next job $(j + 1)$ becomes available for processing at β, two events must happen: an arrival to δ and a service completion at δ. The expected time for this to occur is 1.4. When $n_U \geq 2$, on the other hand, job j still becomes available for processing at β after an expected time of 0.4, but job $j + 1$ now becomes available at β after an expected time of only 0.8. Thus the interarrival time at β between jobs j and $j + 1$ is significantly larger for $n_U = 1$ than it is for $n_U = 2$. We believe that this difference is responsible for the decrease in control limits. There is more incentive to delay service until the arrival of job j when $n_U = 1$ since job j will have to wait a longer time for the following batch to form if it

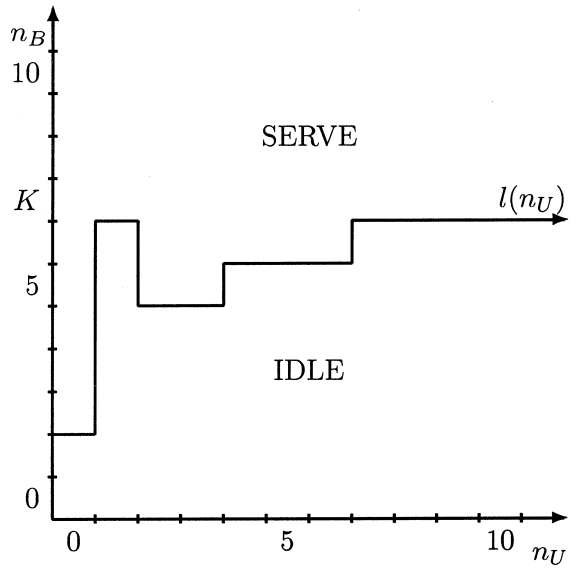

Fig. 1. Optimal policy for a $\delta \rightarrow \beta$ system with $\lambda = 1.00$, $K = 7$, $\mu_U = 2.50$, and $\mu_B = 0.48$.

is not included in the next batch served at β. This results in a greater control limit for $n_U = 1$ than for $n_U = 2$.

Since most of the work to date on the control of a batch processing machine has considered the problem of a batch machine in isolation, it is of considerable interest to know how much can be gained by taking into account information about other machines in the network when controlling a batch machine that is in reality part of a larger manufacturing network. For a $\delta \rightarrow \beta$ system, we use the above Markov decision process formulation to determine the optimal long-run average number of jobs in the system. We then compare this value to the long-run average number of jobs that results if the optimal policy for a batch machine in isolation is used to control β. The difference between these two values represents the benefit of utilizing the current state of δ when controlling β in a $\delta \rightarrow \beta$ network.

To determine the optimal policy for β in isolation, we note that since the arrivals to δ are Poisson with rate λ and the processing time at δ is exponential, the arrival process at β is Poisson with rate λ (Wolff, 1989). We can then use the result of Deb and Serfozo (1973) that the optimal policy for a batch machine with Poisson arrivals is a control limit policy. In order to compare the performance of this myopic control limit policy with the performance of the globally optimal control policy for a $\delta \rightarrow \beta$ system, we tested both policies on a number of sample problems.

To generate a sample problem, we must specify four parameters: the arrival rate; the batch capacity; and the service rate at each of the two machines. We generated sample problems by selecting a set of values for each of these four parameters that covers the range of values of interest. Rather than specifying the service rates directly, we found it more intuitive to select different levels for the traffic intensity at each machine. We then solved for the service rates using $\mathrm{TI}_B = \lambda/K\mu_B$ and $\mathrm{TI}_U = \lambda/\mu_U$. For TI_U we selected the levels of 0.2, 0.4, 0.6, and 0.8. These levels evenly cover the possible space from low to high traffic intensity. For TI_B we selected the levels of 0.3, 0.4, 0.5, and 0.6. We selected more moderate values for TI_B since extreme values are not of much interest (for large values the optimal policy is to always serve a batch of size K, and for small values there is very little waiting time regardless of the policy employed). For the batch capacity, we selected the levels of four and seven. These values were selected to study the impact of differing batch capacities. We were not able to consider batch sizes larger than seven since the size of the state space in the value iteration algorithm is directly proportional to the batch capacity. Capacities larger than seven result in a state space too large for efficient numerical solution. For all sample problems we set the arrival rate equal to one. Only one level of λ was tested because changing λ while keeping K, TI_U, and TI_B the same does not affect the optimal policy. This is because the service rates are automatically adjusted up or down with the arrival rate when the traffic intensities are kept the same. Consequently the system behaves exactly the same except that jobs move through it more quickly or more slowly. The 32 different combinations of these levels represent a wide variety of possible scenarios over which to test the performance of the different control policies. The data for the 32 different sample problems are displayed in Table 1.

For each sample problem, we used the Markov decision process formulation of Equations (2) and (3) to calculate the optimal long-run average number of jobs in the $\delta \rightarrow \beta$ system. To avoid the difficulties of an infinite state space, an upper limit was placed on the number of jobs allowed in each queue. Consequently, the value iteration algorithm's solution is a lower bound on the actual optimal long-run average number of jobs in the system. We note that the upper limit on each queue size was very large and as a result the lower bounds are very tight. For each sample problem we also calculated the long-run average number of jobs that results when the optimal control limit policy for the batch machine in isolation is used to control β in the $\delta \rightarrow \beta$ network. We used a Markov decision process formulation to determine the optimal control limit value for the batch machine in isolation, and used a value iteration algorithm to calculate the long-run average number of jobs in the Markov chain which results when β is controlled according to this control limit policy. The same upper limit was placed on each machine's queue size, so the resulting long-run average number of jobs is again a lower bound on the true value.

Table 1 summarizes the results that we obtained for a $\delta \rightarrow \beta$ system. It contains the lower bound on the truly optimal average number of jobs in the system (LB(OPT)),

Table 1. Sample problems for a $\delta \rightarrow \beta$ system

Example	λ	K	TI_B	TI_U	$LB(OPT)$	CL	$LB(CL)$	$LB(CL)/LB(OPT)$
1	1.0	4	0.3	0.2	2.2767	1	2.4430	1.0730
2	1.0	4	0.3	0.4	2.6713	1	2.8597	1.0705
3	1.0	4	0.3	0.6	3.5674	1	3.6930	1.0352
4	1.0	4	0.3	0.8	6.1193	1	6.1930	1.0120
5	1.0	4	0.4	0.2	3.2040	2	3.3522	1.0463
6	1.0	4	0.4	0.4	3.5873	2	3.7688	1.0506
7	1.0	4	0.4	0.6	4.4603	2	4.6022	1.0318
8	1.0	4	0.4	0.8	7.0470	2	7.1022	1.0078
9	1.0	4	0.5	0.2	4.3340	3	4.4808	1.0339
10	1.0	4	0.5	0.4	4.6525	3	4.8975	1.0527
11	1.0	4	0.5	0.6	5.5152	3	5.7308	1.0391
12	1.0	4	0.5	0.8	8.1224	3	8.2308	1.0133
13	1.0	4	0.6	0.2	5.7813	4	5.8996	1.0205
14	1.0	4	0.6	0.4	6.1326	4	6.3162	1.0299
15	1.0	4	0.6	0.6	6.9362	4	7.1496	1.0308
16	1.0	4	0.6	0.8	9.4989	4	9.6496	1.0159
17	1.0	7	0.3	0.2	3.9837	2	4.1644	1.0454
18	1.0	7	0.3	0.4	4.3607	2	4.5810	1.0505
19	1.0	7	0.3	0.6	5.2779	2	5.4144	1.0259
20	1.0	7	0.3	0.8	7.8431	2	7.9144	1.0091
21	1.0	7	0.4	0.2	5.5731	3	5.7659	1.0346
22	1.0	7	0.4	0.4	5.9024	3	6.1826	1.0475
23	1.0	7	0.4	0.6	6.7890	3	7.0159	1.0334
24	1.0	7	0.4	0.8	9.4118	3	9.5159	1.0111
25	1.0	7	0.5	0.2	7.4516	4	7.6576	1.0276
26	1.0	7	0.5	0.4	7.7424	4	8.0743	1.0429
27	1.0	7	0.5	0.6	8.5710	4	8.9076	1.0393
28	1.0	7	0.5	0.8	11.2348	4	11.4076	1.0154
29	1.0	7	0.6	0.2	9.8628	6	10.0218	1.0161
30	1.0	7	0.6	0.4	10.1424	6	10.4384	1.0292
31	1.0	7	0.6	0.6	10.9282	6	11.2718	1.0314
32	1.0	7	0.6	0.8	13.5264	6	13.7718	1.0181

the optimal control limit for the batch machine in isolation (CL), and the lower bound on the average number of jobs that results when this myopic control limit policy is used for the $\delta \rightarrow \beta$ system (LB(CL)). On average, the number of jobs in the system was reduced by 3.25% by utilizing the current state of δ when controlling β. In some examples, the reduction was larger than 7%. These numbers clearly indicate that taking into account the current state of an upstream unit-capacity machine in the control of a batch processing machine can make a significant difference, especially in industries with high inventory holding costs.

The results in Table 1 also indicate the situations in which it is most important to take into account information about an upstream unit-capacity machine when controlling a batch machine. The greatest differences between LB(OPT) and LB(CL) occur when the traffic intensity at the upstream unit-capacity machine is small (0.2 or 0.4). In these cases δ is rarely busy and knowledge of when there are jobs at δ is most useful in predicting the timing of arrivals to β. When the traffic intensity at δ is high (0.8), δ is almost always processing jobs and the state of δ is not as useful in predicting the timing of arrivals to β. The difference between LB(OPT) and LB(CL) is also larger for smaller values of TI_B. This is because as the traffic intensity at the batch machine increases, the optimal policy for the $\delta \rightarrow \beta$ network becomes more and more similar to the optimal policy for the batch machine in isolation: only serve a batch at β when there are at least K jobs waiting.

4.2. *A $\beta \rightarrow \delta$ system*

It is clearly never optimal to idle at the unit-capacity machine when there are jobs available, and once again the problem of controlling the batch machine to minimize the long-run average number of jobs in the system can be formulated as a Markov decision process. The formulation for a $\beta \rightarrow \delta$ system is similar to the previous formulation of Equations (2) and (3) for a $\delta \rightarrow \beta$ system. We again used a value iteration algorithm to solve the MDP for a number of sample problems and can make some

observations about the structure of the optimal policies. For each of the numerous problems that we solved, the optimal policy had the following form: for each n_U, there exists a control limit $l(n_U)$ such that it is optimal to serve if $n_B \geq l(n_U)$ and to idle if $n_B < l(n_U)$; furthermore, the control limits $l(n_U)$ are increasing in n_U. This type of policy is easy to describe and is intuitive. As the amount of work at the downstream unit-capacity machine increases, it becomes less desirable to serve a batch at β since the completed jobs will only wait in the queue at δ. As a result, the control limits increase with n_U (unlike the optimal policy for a $\delta \rightarrow \beta$ system). We have found, however, that the submodularity conditions sufficient to prove that this form of policy is optimal do not necessarily hold. Consequently, we are unable to prove that this structure holds for all cases.

To investigate the benefit of taking into account the state of a downstream unit-capacity machine in the control of the batch machine, we compared the average number of jobs that results from the globally optimal policy for a $\beta \rightarrow \delta$ system with the average number of jobs that results when the optimal policy for β in isolation is used for a $\beta \rightarrow \delta$ system. We used the input data from the 32 sample problems of Section 4.1 to test the performance of these two different control strategies for a $\beta \rightarrow \delta$ network. A value iteration algorithm was again used to calculate a lower bound on the optimal long-run average number of jobs in the system (LB(OPT)). We then compared this value to a lower bound on the long-run average number of jobs that results when the optimal policy for the batch machine in isolation is used to control β (LB(CL)). The difference between these two values represents the gain that can be achieved by utilizing information about the downstream unit-capacity machine in the control of the batch machine.

Table 2 presents the results that we obtained. On average, the number of jobs in the $\beta \rightarrow \delta$ system was reduced by 0.38% by taking into account the current state of δ when controlling β. The largest reduction achieved was 1.21%. Note that the average and maximum percent differences between LB(OPT) and LB(CL) are significantly smaller for the $\beta \rightarrow \delta$ network than they were for

Table 2. Sample problems for a $\beta \rightarrow \delta$ system

Example	λ	K	TI_B	TI_U	$LB(OPT)$	CL	$LB(CL)$	$LB(CL)/LB(OPT)$
1	1.0	4	0.3	0.2	2.5954	1	2.6027	1.0028
2	1.0	4	0.3	0.4	3.2029	1	3.2283	1.0079
3	1.0	4	0.3	0.6	4.2616	1	4.3128	1.0120
4	1.0	4	0.3	0.8	7.0231	1	7.1083	1.0121
5	1.0	4	0.4	0.2	3.5608	2	3.5610	1.0001
6	1.0	4	0.4	0.4	4.2504	2	4.2528	1.0006
7	1.0	4	0.4	0.6	5.4331	2	5.4411	1.0015
8	1.0	4	0.4	0.8	8.3858	2	8.4045	1.0022
9	1.0	4	0.5	0.2	4.7512	3	4.7513	1.0000
10	1.0	4	0.5	0.4	5.5084	3	5.5173	1.0016
11	1.0	4	0.5	0.6	6.7998	3	6.8187	1.0028
12	1.0	4	0.5	0.8	9.9601	3	9.9827	1.0023
13	1.0	4	0.6	0.2	6.2262	4	6.2328	1.0011
14	1.0	4	0.6	0.4	7.0372	4	7.0732	1.0051
15	1.0	4	0.6	0.6	8.4459	4	8.4915	1.0054
16	1.0	4	0.6	0.8	11.8620	4	11.8949	1.0028
17	1.0	7	0.3	0.2	4.4947	2	4.4980	1.0007
18	1.0	7	0.3	0.4	5.3352	2	5.3539	1.0035
19	1.0	7	0.3	0.6	6.7114	2	6.7585	1.0070
20	1.0	7	0.3	0.8	9.9379	2	10.0342	1.0097
21	1.0	7	0.4	0.2	6.1921	3	6.1943	1.0004
22	1.0	7	0.4	0.4	7.1791	3	7.1931	1.0020
23	1.0	7	0.4	0.6	8.7899	3	8.8302	1.0046
24	1.0	7	0.4	0.8	12.4268	3	12.5222	1.0077
25	1.0	7	0.5	0.2	8.1676	4	8.1717	1.0005
26	1.0	7	0.5	0.4	9.2819	4	9.3017	1.0021
27	1.0	7	0.5	0.6	11.1133	4	11.1688	1.0050
28	1.0	7	0.5	0.8	15.1885	4	15.3272	1.0091
29	1.0	7	0.6	0.2	10.6433	6	10.6448	1.0001
30	1.0	7	0.6	0.4	11.8741	6	11.8995	1.0021
31	1.0	7	0.6	0.6	13.9143	6	13.9627	1.0035
32	1.0	7	0.6	0.8	18.4842	6	18.5669	1.0045

the $\delta \rightarrow \beta$ network. Thus, the benefit of utilizing information about the state of an upstream unit-capacity machine appears to be an order of magnitude larger than that of utilizing information about the state of a downstream unit-capacity machine. For a $\beta \rightarrow \delta$ system, simply operating the batch machine as if it were alone results in an average number of jobs that is within 1% of optimal for almost all examples. Our results are consistent with those of Robinson *et al.* (1995).

The results in Table 2 also indicate the situations in which it is most important to take into account information about a downstream unit-capacity machine when controlling a batch machine. The greatest differences between LB(OPT) and LB(CL) occur when the traffic intensity at the downstream unit-capacity machine is large and the traffic intensity at the batch machine is small. In these cases, the unit-capacity machine is frequently busy and it is most important to consider the amount of work waiting at δ when deciding whether to process a batch at β. If there is a large queue at δ, then it is best to idle at β and serve a fuller batch in the future since a batch served at the present time would only join the queue at δ. Note that it is most important to utilize the state of a downstream unit-capacity machine when its traffic intensity is high, while from Section 4.1 it is most important to utilize the state of an upstream unit-capacity machine when its traffic intensity is low.

4.3. *Systems with multiple unit-capacity machines*

For networks that contain more than one unit-capacity machine, it is of interest to know how much additional gain can be achieved by taking into account the current state of additional unit-capacity machines in the network. To examine this, we consider a $\delta \rightarrow \delta \rightarrow \beta$ network. For a $\delta \rightarrow \delta \rightarrow \beta$ system , we again formulated the problem of minimizing the long-run average number of jobs in the entire system as a Markov decision process. A value iteration algorithm was used to calculate a lower bound on the optimal long-run average number of jobs in the system (LB(OPT)). This value was compared to a lower bound on the long-run average number of jobs that results when the optimal policy for the $\delta \rightarrow \beta$ system is used to control the $\delta \rightarrow \delta \rightarrow \beta$ system (LB($\delta \rightarrow \beta$)). The difference between these two values represents the benefit of taking into account the states of both the upstream unit-capacity machines (δ_1 and δ_2) rather than just the immediately upstream unit-capacity machine (δ_2). We also compared LB($\delta \rightarrow \beta$) to a lower bound on the long-run average number of jobs that results when the optimal control limit policy for the batch machine in isolation is used to control the $\delta \rightarrow \delta \rightarrow \beta$ system (LB(CL)). The difference between these two values represents the benefit of taking into account the state of the immediately upstream unit-capacity machine (δ_2) rather than controlling the batch machine as if it were in isolation.

We generated 27 sample problems over which to calculate these values. The levels of 0.2, 0.5, and 0.8 were used for the traffic intensities at the unit-capacity machines, while the levels of 0.3, 0.45, and 0.6 were used for the traffic intensity at the batch machine. The capacity of the batch machine was set equal to four and the arrival rate was set equal to one. The 27 different combinations of these parameter levels and the results that we obtained for each sample problem are presented in Table 3.

On average, LB($\delta \rightarrow \beta$) was 0.75% greater than LB(OPT), with the greatest percent difference being 3.43%. LB(CL) averaged 2.60% greater than LB($\delta \rightarrow \beta$), with a maximum percent difference of 6.58%. Note that the average and maximum percent differences between LB(OPT) and LB($\delta \rightarrow \beta$) are significantly smaller than the average and maximum percent differences between LB($\delta \rightarrow \beta$) and LB(CL). Thus, the greatest gains result from considering the immediately upstream unit-capacity machine, with much smaller gains achieved by looking one more unit-capacity machine upstream. Consequently, it seems that there are diminishing returns from taking into account additional upstream unit-capacity machines in the control of the batch machine. The largest differences between LB(OPT) and LB($\delta \rightarrow \beta$) occur when the traffic intensities at the unit-capacity machines are small. It is most important to consider the current state of the additional upstream unit-capacity machine (δ_1) when the traffic intensity at δ_2 is small (0.2). In this case δ_2 is rarely busy and the state of δ_1 is most useful for predicting the timing of arrivals to δ_2 which can in turn be used to predict the timing of arrivals to β.

We also investigated the benefits of taking into account the number of jobs at more than one unit-capacity machine downstream from the batch processing machine. However, since the benefits from looking at even a single downstream machine were fairly modest, we found that considering the number of jobs in two downstream machines had very minor effects in all of our examples. Therefore, based on our numerical examples, we can conclude that the greatest benefit can be obtained by using information on the number of jobs at a single upstream machine.

4.4. *A heuristic policy*

As discussed in the previous subsections, it is most critical to take into account the current state of an immediately upstream unit-capacity machine when controlling a batch processing machine that is part of a larger manufacturing network. However, the optimal policies for a $\delta \rightarrow \beta$ network can be quite complicated (for example, see Fig. 1). They are also computationally expensive. The number of iterations required by the value iteration algorithm to find the optimal policy is problem dependent, so we cannot provide a general rule for the computation time required. The number of iterations, however, typically increases

Table 3. Sample problems for a $\delta \rightarrow \delta \rightarrow \beta$ system

Example	λ	K	TI_B	TI_{U2}	TI_{U1}	$LB(OPT)$	$LB(\delta \rightarrow \beta)$	$LB(\delta \rightarrow \beta)/LB(OPT)$	$LB(CL)$	$LB(CL)/LB(\delta \rightarrow \beta)$
1	1.0	4	0.3	0.2	0.2	2.4430	2.5267	1.0343	2.6930	1.0658
2	1.0	4	0.3	0.2	0.5	3.2086	3.2767	1.0212	3.4430	1.0508
3	1.0	4	0.3	0.2	0.8	6.2152	6.2209	1.0009	6.3871	1.0267
4	1.0	4	0.3	0.5	0.2	3.2729	3.2840	1.0034	3.4430	1.0484
5	1.0	4	0.3	0.5	0.5	4.0325	4.0340	1.0004	4.1930	1.0394
6	1.0	4	0.3	0.5	0.8	6.9860	6.9870	1.0001	7.1458	1.0227
7	1.0	4	0.3	0.8	0.2	6.3232	6.3232	1.0000	6.3969	1.0117
8	1.0	4	0.3	0.8	0.5	7.0732	7.0732	1.0000	7.1469	1.0104
9	1.0	4	0.3	0.8	0.8	10.0131	10.0131	1.0000	10.0868	1.0074
10	1.0	4	0.45	0.2	0.2	3.8761	3.9765	1.0259	4.1497	1.0436
11	1.0	4	0.45	0.2	0.5	4.6226	4.7265	1.0225	4.8997	1.0366
12	1.0	4	0.45	0.2	0.8	7.6303	7.6691	1.0051	7.8420	1.0225
13	1.0	4	0.45	0.5	0.2	4.6412	4.6830	1.0090	4.8997	1.0463
14	1.0	4	0.45	0.5	0.5	5.3824	5.4330	1.0094	5.6497	1.0399
15	1.0	4	0.45	0.5	0.8	8.3558	8.3849	1.0035	8.6013	1.0258
16	1.0	4	0.45	0.8	0.2	7.7498	7.7565	1.0009	7.8523	1.0124
17	1.0	4	0.45	0.8	0.5	8.5035	8.5065	1.0004	8.6023	1.0113
18	1.0	4	0.45	0.8	0.8	11.4446	11.4454	1.0001	11.5412	1.0084
19	1.0	4	0.6	0.2	0.2	5.9092	5.9983	1.0151	6.1194	1.0202
20	1.0	4	0.6	0.2	0.5	6.6007	6.7483	1.0224	6.8694	1.0179
21	1.0	4	0.6	0.2	0.8	9.6381	9.6982	1.0062	9.8196	1.0125
22	1.0	4	0.6	0.5	0.2	6.6016	6.6636	1.0094	6.8694	1.0309
23	1.0	4	0.6	0.5	0.5	7.3506	7.4136	1.0086	7.6194	1.0278
24	1.0	4	0.6	0.5	0.8	10.3143	10.3451	1.0030	10.5519	1.0200
25	1.0	4	0.6	0.8	0.2	9.6621	9.6684	1.0007	9.8197	1.0156
26	1.0	4	0.6	0.8	0.5	10.4105	10.4184	1.0008	10.5697	1.0145
27	1.0	4	0.6	0.8	0.8	13.3488	13.3551	1.0005	13.5066	1.0113

when the number of states becomes larger. Consequently the computation time is sensitive to the size of the batch capacity, the number of machines, and the upper limits placed on the queue sizes. We solved the relatively small problems in Table 1 using the Pascal programming language on a Sun SPARC II workstation. Each problem typically required a number of minutes of CPU time. Since the batch capacity in applications such as semiconductor burn-in operations can number in the thousands (Hochbaum and Landy, 1997), this approach is not practical for many real-world applications. Consequently, in this subsection we propose a simple heuristic policy to control the batch machine which utilizes the current state of the immediately upstream unit-capacity machine.

Our heuristic is motivated by the structure of the optimal policies for a $\delta \rightarrow \beta$ network. As discussed in Section 4.1, the optimal policies for this network have the following form: for each n_U, there exists a control limit $l(n_U)$ such that it is optimal to idle at the batch machine if $n_B < l(n_U)$ and to serve if $n_B \geq l(n_U)$. The control limits $l(n_U)$ are not necessarily increasing in n_U. However, we can make the following observation about the control limit values. For each of the sample problems considered in Section 4.1, the greatest jump in the control limit values occurs when n_U increases from zero to one (i.e., $|l(1) - l(0)| \geq |l(j+1) - l(j)|$ for all $j \geq 0$). Moreover, in over 70% of the sample problems, the *only* jump in the control limit values occurs when n_U increases from zero to one (i.e., $l(j+1) = l(j)$ for all $j \geq 1$). Thus, for $n_U > 1$ the control limits rarely change, and when they do the change is relatively small. This motivates the development of a heuristic policy which consists of two control limit values: one for $n_U = 0$ and the other for $n_U \geq 1$.

Given the current state of the $\delta \rightarrow \beta$ system, when the batch machine is available a heuristic policy must decide whether to serve a batch of size $\min\{n_B, K\}$ or idle until the next event which changes the state of the system. We note that when $n_B = 0$ serving is not an option, and when $n_B \geq K$ idling will never produce a benefit. Consequently, a decision is only necessary when $K > n_B > 0$. Our heuristic uses the following logic to make this control decision. We approximate the benefit of idling until the next arrival to the batch machine, and compare this value to the cost of idling until the next arrival. If the cost of idling exceeds the benefit, the decision is to serve the n_B jobs at β immediately; otherwise, the decision is to idle at β until the next event which changes the state of the system, at which time the decision process begins all over again. This logic is similar to the Next Arrival Control Heuristic (NACH) of Fowler *et al.* (1992) for a batch processing machine in isolation.

Let t represent the expected time until the next arrival to the batch machine. If we idle at β until the next arrival, each of the n_B jobs currently in queue will be delayed an expected time of t. The cost of idling is thus $n_B t$. The advantage of idling until the next arrival is that this arrival can then be included in the same batch as the n_B jobs currently in queue. Had we not idled, this next arrival would likely have had to wait for the n_B jobs to complete service before it could have been served. The expected time that this next arrival would have had to wait is equal to $1/\mu_B - t$, the difference between the expected service time for the batch and the expected time until the next arrival. The benefit of idling is thus the $1/\mu_B - t$ time units of waiting avoided for the next arrival.

Some simple algebra reveals that the cost of idling exceeds the benefit when $n_B > 1/(t\mu_B) - 1$. Consequently, our heuristic is a control limit policy which chooses to idle when $n_B < \min\{\lceil 1/(t\mu_B) - 1 \rceil, K\}$ and to serve when $n_B \geq \min\{\lceil 1/(t\mu_B) - 1 \rceil, K\}$, where $\lceil x \rceil$ represents the smallest positive integer strictly greater than x. Note that the control limit value depends on the time until the next arrival, t, which in turn depends on the current state of the upstream unit-capacity machine. If $n_U \geq 1$ then $t = 1/\mu_U$, the expected time until the next service completion at the unit-capacity machine. If $n_U = 0$ then $t = 1/\lambda + 1/\mu_U$, since both an arrival and a service completion must occur before the next arrival to β. Thus, after inserting these values for t, our heuristic policy consists of two control limits:

$$l_1 = \min\left\{ \left\lceil \frac{\lambda\mu_U}{\mu_B(\mu_U + \lambda)} - 1 \right\rceil, K \right\} \quad \text{for } n_U = 0,$$

and

$$l_2 = \min\left\{ \left\lceil \frac{\mu_U}{\mu_B} - 1 \right\rceil, K \right\} \quad \text{for } n_U \geq 1.$$

The full heuristic policy can be stated as follows:

if the batch machine is available then
  if $n_U = 0$ then
    if $n_B < l_1$ then idle until the next event which changes the state of the system
    if $n_B \geq l_1$ then serve a batch of size $\min\{n_B, K\}$
  if $n_U \geq 1$ then
    if $n_B < l_2$ then idle until the next event which changes the state of the system
    if $n_B \geq l_2$ then serve a batch of size $\min\{n_B, K\}$.

We will refer to this heuristic as the Two Control Limit Heuristic (TCLH).

To test the performance of TCLH for a $\delta \rightarrow \beta$ network, we used a value iteration algorithm to calculate a lower bound on the long-run average number of jobs that results when TCLH is used to control β. This lower bound (LB(TCLH)) was calculated for the 32 sample problems from Section 4.1. The results are presented in Table 4 along with the lower bounds that were calculated in Section 4.1 for the optimal policy for the network (LB(OPT)) and the optimal policy for the batch machine in isolation (LB(CL)). On average, LB(TCLH) was only 1.59% greater than LB(OPT). The maximum difference between LB(TCLH) and LB(OPT) was 4.66%. Note that this is an improvement over LB(CL) in both average performance and worst case performance. LB(CL) was 3.25% greater than LB(OPT) on average, with a maximum difference of 7.30%. Thus, TCLH results in a long-run average number of jobs within 1.5% of optimal on average and consistently outperforms the optimal policy for the batch machine in isolation. In addition, TCLH is easier to calculate than the optimal policy for the batch machine in isolation which requires the solution of a Markov decision process. TCLH simply requires the closed-form calculation of two control limits and is thus essentially instantaneous. TCLH's simplicity and good performance make it a good candidate for implementation in manufacturing networks which contain a batch processing machine.

Table 4. Heuristic policy performance for a $\delta \rightarrow \beta$ system with exponential interarrival and service times

Example	$LB(TCLH)$	$LB(OPT)$	$LB(CL)$	$\frac{LB(TCLH)}{LB(OPT)}$	$\frac{LB(CL)}{LB(OPT)}$
1	2.2767	2.2767	2.4430	1.0000	1.0730
2	2.6756	2.6713	2.8597	1.0016	1.0705
3	3.5684	3.5674	3.6930	1.0003	1.0352
4	6.1930	6.1193	6.1930	1.0120	1.0120
5	3.2811	3.2040	3.3522	1.0241	1.0463
6	3.5873	3.5873	3.7688	1.0000	1.0506
7	4.5407	4.4603	4.6022	1.0180	1.0318
8	7.0583	7.0470	7.1022	1.0016	1.0078
9	4.5358	4.3340	4.4808	1.0466	1.0339
10	4.7581	4.6525	4.8975	1.0227	1.0527
11	5.6123	5.5152	5.7308	1.0176	1.0391
12	8.2646	8.1224	8.2308	1.0175	1.0133
13	5.9093	5.7813	5.8996	1.0221	1.0205
14	6.3604	6.1326	6.3162	1.0371	1.0299
15	7.0322	6.9362	7.1496	1.0138	1.0308
16	9.6563	9.4989	9.6496	1.0166	1.0159
17	4.1124	3.9837	4.1644	1.0323	1.0454
18	4.4221	4.3607	4.5810	1.0141	1.0505
19	5.2979	5.2779	5.4144	1.0038	1.0259
20	7.8930	7.8431	7.9144	1.0064	1.0091
21	5.6458	5.5731	5.7659	1.0130	1.0346
22	5.9076	5.9024	6.1826	1.0009	1.0475
23	6.9078	6.7890	7.0159	1.0175	1.0334
24	9.4799	9.4118	9.5159	1.0072	1.0111
25	7.7158	7.4516	7.6576	1.0355	1.0276
26	7.8689	7.7424	8.0743	1.0163	1.0429
27	8.7159	8.5710	8.9076	1.0169	1.0393
28	11.3774	11.2348	11.4076	1.0127	1.0154
29	10.1819	9.8628	10.0218	1.0324	1.0161
30	10.3006	10.1424	10.4384	1.0156	1.0292
31	11.0940	10.9282	11.2718	1.0152	1.0314
32	13.7697	13.5264	13.7718	1.0180	1.0181

We note that TCLH can easily be modified to allow for non-exponential interarrival or service time distributions. In order to test the performance of our heuristic for interarrival and service time distributions with less variability than the exponential distribution (coefficient of variation = 1), in the remainder of this section we modify TCLH for a $\delta \rightarrow \beta$ system with uniform interarrival and service time distributions with range = mean (coefficient of variation = 0.289). Specifically, we assume that the time between arrivals of jobs to δ is uniformly distributed with mean and range equal to $1/\lambda$, and that jobs require a uniformly distributed processing time on δ with mean and range $1/\mu_U$ and a uniformly distributed processing time on β with mean and range $1/\mu_B$.

TCLH can be modified for uniform interarrival and service time distributions as follows. Let t again represent the expected time until the next arrival to β. Because TCLH uses only the mean of β's service time distribution, the cost of idling and the benefit of idling remain $n_B t$ and $1/\mu_B - t$ as before. Thus, the policy again consists of control limits which depend on the expected time until the next arrival to β. The control limit value as a function of t is $\min\{\lceil 1/(t\mu_B) - 1 \rceil, K\}$. However, the calculation of t is different for uniformly distributed interarrival and service times than it was for exponentially distributed interarrival and service times. Because the uniform distribution does not have the memoryless property of the exponential distribution, when $n_U \geq 1$ we must keep track of the elapsed time since the start of the current service on δ in order to determine the expected time until the next arrival to β. Similarly, when $n_U = 0$ we must keep track of the elapsed time since the last arrival to δ in order to determine the expected time until the next arrival to β. This information can be used to calculate t as follows. When $n_U \geq 1$, the expected time until the next arrival to β given that t_S time units have elapsed since the start of the current service at δ is given by the equation

$$t = \begin{cases} \dfrac{1}{\mu_U} - t_S & \text{if } t_S \leq \dfrac{1}{2\mu_U}, \\ \left(\dfrac{3}{2\mu_U} - t_S\right)\Big/2 & \text{if } t_S > \dfrac{1}{2\mu_U}. \end{cases} \tag{4}$$

Similarly, when $n_U = 0$ the expected time until the next arrival to β given that t_A time units have elapsed since the last arrival to δ is given by the equation

$$t = \begin{cases} \dfrac{1}{\lambda} - t_A + \dfrac{1}{\mu_U} & \text{if } t_A \leq \dfrac{1}{2\lambda}, \\ \left(\dfrac{3}{2\lambda} - t_A\right)\Big/2 + \dfrac{1}{\mu_U} & \text{if } t_A > \dfrac{1}{2\lambda}. \end{cases} \tag{5}$$

When the batch machine is available and $K > n_B > 0$, the appropriate value of t can be found from Equation (4) or Equation (5) and substituted into the above expression for the control limit value. If the number of jobs at the batch machine is less than the resulting control limit value, the decision is to idle until the next event which changes the state of the system. Otherwise, the decision is to serve the n_B jobs immediately. Note that we will continue to refer to this heuristic as the Two Control Limit Heuristic (TCLH) even though the heuristic policy for uniform interarrival and service time distributions no longer consists of just two control limit values.

To test the performance of TCLH on a $\delta \rightarrow \beta$ system with uniform interarrival and service time distributions, we used the data from the 32 sample problems of Section 4.1. Because a Markov decision process formulation is not valid for uniform interarrival and service time distributions, the globally optimal policy for the $\delta \rightarrow \beta$ system could not be determined. Instead, we used simulation to compare the performance of our heuristics with the best single control limit policy for the $\delta \rightarrow \beta$ network (BestCL). BestCL was found by simulating the performance of all possible control limit values, $CL = 1, 2, \ldots, K$, and selecting the one which resulted in the minimum average number of jobs in the system. Note that BestCL is the best single control limit policy for the entire network, not just the best control limit policy for the batch machine in isolation. Thus, by definition BestCL performs at least as well for a $\delta \rightarrow \beta$ system as the optimal control limit policy for the batch machine in isolation. All policies were tested using a GPSS/H simulation program. The simulations were run for 1 000 000 time units with the average number of jobs calculated every 20 000 time units (with a 4000 time unit warmup period). We report the average number of jobs in the system for each policy as well as 95% confidence intervals. Table 5 summarizes the results.

On average, the number of jobs in the systems controlled by TCLH was 3.55% less than the number of jobs in the systems controlled by the best single control limit policy for the $\delta \rightarrow \beta$ network. In some examples the average number of jobs resulting from TCLH was as much as 8.91% less than the average number of jobs resulting from BestCL. Note that the average difference between TCLH and BestCL for uniform interarrival and service times (3.55%) is larger than than the average difference between LB(TCLH) and LB(CL) for exponential interarrival and service times (1.65%). Thus the performance of our heuristic (when compared to the optimal control limit policy for the batch machine in isolation) appears to be even stronger for uniform interarrival and service time distributions than it was for exponential interarrival and service time distributions.

5. Conclusion

In this paper, we have explored the control of manufacturing networks consisting of a batch processing machine and one or more unit-capacity machines in tandem. This work was motivated by the many examples of manufac-

Table 5. Heuristic policy performance for a $\delta \rightarrow \beta$ system with uniform interarrival and service times

Example	*TCLH*	*BestCL*	*BestCL/TCLH*
1	1.7667 ± 0.0023	1.9163 ± 0.0016	1.0847
2	1.9474 ± 0.0024	2.1209 ± 0.0014	1.0891
3	2.1675 ± 0.0024	2.3457 ± 0.0015	1.0822
4	2.5322 ± 0.0036	2.6918 ± 0.0033	1.0630
5	2.3758 ± 0.0029	2.4354 ± 0.0027	1.0251
6	2.5675 ± 0.0026	2.6397 ± 0.0028	1.0281
7	2.7854 ± 0.0031	2.8657 ± 0.0030	1.0288
8	3.1468 ± 0.0039	3.2160 ± 0.0041	1.0220
9	2.9975 ± 0.0030	3.0940 ± 0.0043	1.0322
10	3.1805 ± 0.0029	3.2929 ± 0.0035	1.0353
11	3.4092 ± 0.0040	3.5191 ± 0.0044	1.0322
12	3.7537 ± 0.0053	3.8668 ± 0.0061	1.0301
13	3.6127 ± 0.0048	3.7515 ± 0.0041	1.0384
14	3.7960 ± 0.0055	3.9567 ± 0.0040	1.0423
15	4.0129 ± 0.0047	4.1818 ± 0.0049	1.0421
16	4.3985 ± 0.0059	4.5313 ± 0.0056	1.0302
17	3.1516 ± 0.0034	3.2623 ± 0.0049	1.0351
18	3.3355 ± 0.0038	3.4660 ± 0.0047	1.0391
19	3.5586 ± 0.0043	3.6853 ± 0.0046	1.0356
20	3.9112 ± 0.0054	4.0346 ± 0.0050	1.0316
21	4.2400 ± 0.0046	4.3459 ± 0.0057	1.0250
22	4.4239 ± 0.0054	4.5454 ± 0.0060	1.0275
23	4.6379 ± 0.0054	4.7719 ± 0.0063	1.0289
24	4.9966 ± 0.0068	5.1231 ± 0.0075	1.0253
25	5.3236 ± 0.0077	5.4608 ± 0.0071	1.0258
26	5.5137 ± 0.0078	5.6598 ± 0.0077	1.0265
27	5.7186 ± 0.0074	5.8929 ± 0.0082	1.0305
28	6.0990 ± 0.0079	6.2376 ± 0.0086	1.0227
29	6.4357 ± 0.0096	6.5463 ± 0.0108	1.0172
30	6.6086 ± 0.0095	6.7432 ± 0.0099	1.0204
31	6.8165 ± 0.0108	6.9707 ± 0.0089	1.0226
32	7.2048 ± 0.0090	7.3268 ± 0.0088	1.0169

turing systems which contain both batch and unit-capacity machines. We presented polynomial time dynamic programming algorithms which minimize the sum of the completion times for $\delta \rightarrow \beta$ and $\beta \rightarrow \delta$ systems with deterministic release times and processing times. We showed that these algorithms can be extended to solve systems with any number of upstream unit-capacity machines in polynomial time, but the time required to solve systems with downstream unit-capacity machines is exponential in the number of downstream machines. We then discussed the structure of the optimal policies for stochastic networks containing a batch processing machine. The benefit of taking into account the current state of an upstream unit-capacity machine was found to be an order of magnitude larger than the benefit of taking into account the current state of a downstream unit-capacity machine. Utilizing the current states of additional upstream or downstream unit-capacity machines produced diminishing returns. Consequently, it appears to be most critical to consider the current state of a single upstream unit-capacity machine when controlling a batch processing machine which is part of a larger manufacturing network. We developed a simple heuristic control policy, TCLH, which utilizes the current state of an upstream unit-capacity machine. TCLH was found to perform nearly as well as the computationally expensive and complicated optimal policies, and appears to work well for different interarrival and service time distributions. Our heuristic's simplicity and good performance make it a good candidate for implementation in manufacturing networks which contain a batch processing machine, such as those found in semiconductor manufacturing.

Further research should focus on the control of manufacturing networks containing a batch processing machine which serves multiple job families (either incompatible or compatible). Jobs could require different processing times on the batch machine, on the unit-capacity machines, or on all machines. Networks which contain more than one batch processing machine should also be considered, as should networks in which the machines are not all in a series configuration. Additional investigation is also suggested for stochastic networks containing batch machines in which the interarrival and service times are not exponentially or uniformly distributed.

Acknowledgements

This research is partially supported by Grants No: DMI-9308290, DMI-9424596, and DMI-9625061 from the National Science Foundation and a grant from the Center for Display Technology and Manufacturing at the University of Michigan. We are grateful to Area Editor, Professor Reha Uzsoy and an anonymous referee whose comments improved the paper.

References

Ahmadi, J.H., Ahmadi, R.H., Dasu, S. and Tang, C.S. (1992) Batching and scheduling jobs on batch and discrete processors. *Operations Research*, **40**, 750–763.

Avramidis, A.N., Healy, K.J. and Uzsoy, R. (1998) Control of a batch processing machine: a computational approach. *International Journal of Production Research*, **36**, 3167–3181.

Barnett, A. and Kleitman, D.J. (1978) Some optimization problems with bulk-service queues. *Studies in Applied Mathematics*, **58**, 277–290.

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M., Potts, C.N., Tautenhahn, T. and van de Velde, S.L. (1998) Scheduling a batching machine. *Journal of Scheduling*, **1**, 31–54.

Chandru, V., Lee, C.Y. and Uzsoy, R. (1993a) Minimizing total completion time on batch processing machines. *International Journal of Production Research*, **31**, 2092–2121.

Chandru, V., Lee, C.Y. and Uzsoy, R. (1993b) Minimizing total completion time on a batch processing machine with job families. *Operations Research Letters*, **13**, 61–65.

Chaudhary, M.L. and Templeton, J.G.C. (1983) *A First Course in Bulk Queues*, Wiley, New York, NY.

Deb, R.K. and Serfozo, R.F. (1973) Optimal control of batch service queues. *Advances in Applied Probability*, **5**, 340–361.

Duenyas, I. and Neale, J.J. (1997) Stochastic scheduling of a batch processing machine with incompatible job families. *Annals of Operations Research*, **70**, 191–220.

Fowler, J.W., Phillips, D.T. and Hogg, G.L. (1992) Real-time control of multiproduct bulk-service semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, **5**, 158–163.

Glassey, C.R. and Weng, W.W. (1991) Dynamic batching heuristic for simultaneous processing. *IEEE Transactions on Semiconductor Manufacturing*, **4**, 77–82.

Gurnani, H., Anupindi, R. and Akella, R. (1992) Control of batch processing systems in semiconductor wafer fabrication facilities. *IEEE Transactions on Semiconductor Manufacturing*, **5**, 319–328.

Hochbaum, D.S. and Landy, D. (1997) Scheduling semiconductor burn-in operations to minimize total flowtime. *Operations Research*, **45**, 874–885.

Ikura, Y. and Gimple, M. (1986) Efficient scheduling algorithms for a single batch processing machine. *Operations Research Letters*, **5**, 61–65.

Lee, C.Y., Uzsoy, R. and Martin-Vega, L.A. (1992) Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, **40**, 764–775.

Lee, C.Y. and Uzsoy, R. (1999) Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, **37**, 219–236.

Li, C.L. and Lee, C.Y. (1997) Scheduling with agreeable release times and due dates on a batch processing machine. *European Journal of Operational Research*, **96**, 564–569.

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

Makis, V. (1985) Optimal control of a batch service queueing system with bounded waiting time. *Kybernetika*, **21**, 262–271.

Medhi, J. (1975) Waiting time distribution in a Poisson queue with a general bulk service rule. *Management Science*, **21**, 777–782.

Mehta, S.V. and Uzsoy, R. (1998) Minimizing total tardiness on a batch processing machine with incompatible job families. *IIE Transactions on Scheduling and Logistics*, **30**, 165–178.

Neale, J.J. and Duenyas, I. (1997) Control of a batch processing machine serving compatible job families. Technical Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, 48109.

Neuts, M.F. (1967) A general class of bulk queues with Poisson input. *Annals of Mathematical Statistics*, **38**, 759–770.

Pinedo, M. (1995) *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Englewood Cliffs, NJ.

Powell, W. and Humblet, P. (1986) The bulk service queue with a general control strategy: theoretical analysis and a new computational procedure. *Operations Research* **34**, 267–275.

Robinson, J.K., Fowler, J.W. and Bard, J.F. (1995) The use of upstream and downstream information in scheduling semiconductor operations. *International Journal of Production Research*, **33**, 1849–1869.

Uzsoy, R. (1994) Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research*, **32**, 1615–1635.

Uzsoy, R. (1995) Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*, **33**, 2686–2708.

Uzsoy, R., Lee, C.Y. and Martin-Vega, L.A. (1992) A review of production planning and scheduling models in the semiconductor industry part I: system characteristics, performance evaluation and production planning. *IIE Transactions on Scheduling and Logistics*, **24**, 47–61.

Uzsoy, R., Lee, C.Y. and Martin-Vega, L.A. (1994) A review of production planning and scheduling models in the semiconductor industry part II: shop floor control. *IIE Transactions on Scheduling and Logistics*, **26**, 44–55.

Uzsoy, R. and Yang, Y. (1997) Minimizing total weighted completion time on a single batch processing machine. *Production and Operations Management*, **6**, 57–73.

Weng, W.W. and Leachman, R.C. (1993) An improved methodology for real-time production decisions at batch-process work stations. *IEEE Transactions on Semiconductor Manufacturing*, **6**, 219–225.

Wolff, R.W. (1989) *Stochastic Modeling and the Theory of Queues*, Prentice Hall, Englewood Cliffs, NJ.

Biographies

John Neale obtained his Ph.D. from the University of Michigan Industrial and Operations Management Department in 1998. He is currently a supply chain management analyst for HP.

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*.

Contributed by the Scheduling Department.