Estimating job waiting times in production systems with cross-trained setup crews

UFUK KULA,1 IZAK DUENYAS2 and SEYED M. R. IRAVANI3
1Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, M1, USA
E-mail: ukula@engin.umich.edu
2University of Michigan Business School, Ann Arbor, USA
E-mail: duenyas@umich.edu
3Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, USA
E-mail: s-iravani@nwu.edu

Received January 2003 and accepted February 2004

We model a production system with multiple machines, each serving a variety of jobs. The machines require setups to switch from one job to another. Setup operations are performed by a limited number of setup crews who are cross-trained to perform all setup operations. We develop an approximation model that takes into account the effect of delays caused by unavailability of the setup crews, and obtain average job waiting times in the system. A numerical study demonstrates that our approximation performs very well. Our study also provides insights into the importance of explicitly modeling the effects of cross-trained setup crews on performance measures, and whether and when the cross-training of setup crews improves system performance.

1. Introduction

In recent years, many firms have emphasized workforce agility in the form of cross-trained workers who can perform several different tasks, and can therefore serve flexibly where needed. In fact, one of the key factors in the success of the Toyota Production System is the use of cross-trained teams (Hopp and Spearman, 2000). When workers are cross-trained, they are able to work on several work stations in the system. A common example of cross-trained workers in many plants are the setup crews who are responsible for setting up machines whenever they switch from one type of job to another job type. In many plants, these machines include presses that must be set up by special crews trained in quick die exchanges.

There are few models in the literature which can be used to provide measures of performance improvement resulting from increased levels of cross-training of workers in stochastic production environments. In particular, we know of no published study that models how waiting times and inventory levels in production systems are affected by having a limited number of partially or fully cross-trained setup crews.

This came to our attention recently in our work with a major office furniture company. In one of their factories, the presses were set up by several crews, each trained and responsible for only a limited subset of the presses in the factory. Due to the very high variability in the order patterns from the customers and the wide variety of products, the firm produces to order. In this environment, how long the customers were made to wait for their furniture was a key performance measure. However, due to the long setup times, the firm could not afford to make small batches but needed to batch multiple orders of products from the same family, which led to longer lead times. Also, it would often be the case that when one of the presses needed to be set up, there would be a delay while waiting for a setup crew as they were busy setting up another press at that time. We were asked the following questions: (i) how many setup crews are required in order to achieve a given level of waiting time at a set of machines? and (ii) would cross-training the setup crews so that each crew could set up any press improve performance?

Clearly, in order to address the first question, we need a model that estimates performance as a function of the number of setup crews and their level of cross-training. In this paper, we develop an approximate model that can be used as a tool for addressing this question. The firm independently came to the conclusion that the performance of their system would improve if the setup crews were fully cross-trained so that all crews could attend to all presses in the factory. However, when this was implemented, some managers responsible for production for individual product families complained that Work In Process (WIP) levels
at some presses had become worse as a result and that their jobs were being delayed longer (although there were some improvements in others). This experience motivated us to develop a model that studies the effect of cross-training of setup crews on expected waiting times and WIP in manufacturing systems.

In this paper, we study a production system with multiple machines, each producing a variety of different items. Machines require setups when they switch between products. Unlike the existing literature on stochastic scheduling of products with non-zero setup times which assumes unlimited capacity for setup operations, we consider the situation where setup capacity is limited and provided by fully cross-trained setup crews. We develop an approximate model to obtain the average waiting times for jobs of different types in the system which would enable the firm to evaluate its performance in terms of weighted average waiting times for the different types of orders.

2. Literature

When the setup capacity is unlimited (i.e., there is a dedicated setup crew for each machine), the analysis of the problem with multiple machines reduces to the analysis of independent single-machine problems. The single-machine problem has been extensively studied in the context of polling systems. For a comprehensive review of polling systems, see Borst and Boxma (1997) and Takagi (2000). One implicit assumption in all polling models in the literature is that there is a setup crew dedicated to the machine at all switchover instants to perform the required setups. However, in practice it is often the case that setups cannot be performed in a timely manner. One common reason for setup completion delays is that the setup crew is busy performing setups or some other tasks on other machines. If such setup delays are not properly accounted for, it will be misleading to use the existing polling models in production settings where the unavailability of setup crews is crucial.

One way to reduce setup crew unavailability is to create an agile workforce by cross-training existing setup personnel so that they can perform setups on most or all of the machines. Although workforce agility has lately become an important strategy to buffer against variability in production systems, most of the papers on workforce agility study the impact of cross-training workers who are responsible for processing the jobs rather than the crews responsible for the setups. Gel et al. (2002) formulate work-sharing among cross-trained workers in serial production lines as a Markov decision problem and provide insights into the structure of the optimal policy by focusing on production system characteristics such as ability to preempt the shared task, granularity of the shared task, and variability of task times. Oztolaza et al. (1990) and McClain et al. (1992, 2000) present evidence that cross-training is an effective way of balancing production lines. Bartholdi and Eisenstein (1996) and Bartholdi et al. (1999) consider serial production lines with fully cross-trained workers, and show that under a “bucket brigade” policy, assignment of workers in the order of the slowest to fastest maximizes the throughput of the line. Other examples of relevant studies are Iravani et al. (1997), Ahn et al. (1999) and Hopp et al. (2002) among others. For a review of studies on production systems with agile workforce, see Hopp and Van Oyen (2004).

To the best of our knowledge, none of the papers on workforce agility has specifically considered agility of setup crews, which is our focus in this paper. We are interested in how increased agility, such as setup crews that are trained to set up multiple types of machines, influences system performance. In the paper, we address the question of whether full flexibility of setup crews (i.e., each crew can set up all machines in the system) would always improve performance at each machine. Pooling servers in this way should increase the availability of servers when setup requests are issued. That is, from the point of view of the machine, pooling servers decreases the effective wait that the machine experiences for the setup (since the effective wait includes the actual setup time as well as the wait for the crew). Previous polling literature has shown that decreasing setup times may result in increased expected waiting times (Srinivasan et al., 1995; Cooper et al., 1998, 1999). In our study, we also show that increasing setup crew flexibility can result in longer average wait times for some jobs. (It is straightforward to show that under an optimal setup crew sequencing policy that allows crews to preempt setups, it is not possible for increasing setup crew flexibility to lead to a worse performance for all jobs.)

The remainder of the paper is organized as follows. In Section 3, we formulate the model as a continuous-time Markov chain, and we show that even for a simple case with two machines and one setup crew and exponential job processing times, the problem size is enormous. Thus, in Section 4 we develop an approximation method that reduces the size of the problem by decomposing the system with shared setup crews into smaller systems with independent setup crews. Our approximation method calculates the waiting times in a system for all job types. In Section 5, we present a computational study in which we evaluate the accuracy of our approximation method and provide some managerial insights on design and control issues in systems with cross-trained setup crews.

3. Model formulation

We consider $M$ machines (polling systems). Each machine produces (serves) a variety of jobs in a produce-to-order fashion. Setup (switching from one job type to another) requires the services of a setup crew. The facility has a total of $K < M$ fully cross-trained setup crews who can set up
Cross-training setup crews for production systems

any machine. We model each machine $m$ as a polling system serving $k_m$ job types in an exhaustive manner. Upon serving all jobs of a type, the machine operator asks for the service(s) of the setup crew who then set up the machine for the next job type. For each machine (polling system) we assume a dormant operator (server) (Eisenberg, 1971, 1994; Srinivasan and Gupta, 1996), so that if the machine has no jobs of any type to serve when the current job type is exhausted, no setup request is issued. Of course, when a setup request is issued, a crew may not be available to perform the setup. If no crew is immediately available to carry out the setup operation, the machine remains set up for the current job type. If further jobs of the type for which the machine is already set up arrive, the machine starts another exhaustive cycle and the previous setup request is canceled until once again the machine finishes serving the current job type. Conceptually, the periods in which the setup crew is busy setting up other machines can be seen as vacations (non-serving intervals) taken by the setup crew. In the context of traditional polling models vacations correspond to a switchover period, or a period between two consecutive visits to a queue by a server. The general approach to modeling polling systems as vacation models involves relating the joint probability generating function of a number of jobs for each job class at consecutive server departure epochs and solving the resulting system of equations, (Takagi, 1986). Although there is a literature on queueing models with vacations, (see Doshi (1990) for a review), these do not model our situation where the server (machine) and setup crew both take (inter-related) vacations.

We assume that the cross-trained crews process setup requests according to a priority scheme that prioritizes setup requests based on priorities given to machines (e.g., machine 1 has a higher priority than machine 2 when both have setup requests). In practice, different machines may process jobs of different levels of importance and urgency, and thus our modeling of a priority-based scheme for serving setup requests covers a broad range of situations. We note at this point that our focus on make-to-order queues is not restrictive since if the queues were make-to-stock and a base-stock policy were used at each queue, this would be exactly equivalent to the system we analyze.

We will denote queue $i$ of polling system $m$ as $Q_{im}$, and let $\lambda_{mi}$ and $\mu_{mi}$ be the arrival and service rates at queue $Q_{im}$, respectively, where $m = 1, 2, \ldots, M$, and $i = 1, 2, \ldots, k_m$. We assume that a setup crew can set up queue $Q_{im}$ with rate $\bar{\mu}_{im}$. Finally, we assume that machine 1 has the highest setup priority and machine $M$ has the lowest.

In order to show the complexity of the analysis of systems with a cross-trained setup crew, we consider the simplest possible system of interest, namely, a system with two machines ($M = 2$), two queues at each machine ($k_1 = k_3 = 2$), and one fully cross-trained setup crew ($K = 1$), where job interarrival times, service times, and setup times follow the exponential distribution. We will show that even for this simple system, the analysis is very cumbersome. For this reason, we later develop an approximation method that can handle large systems with more general processing and setup time distributions.

We first simplify the notation for our two-machine system by reindexing the queues so that queues 1 and 2 ($Q_1$ and $Q_2$) refer to the two queues in machine 1, while queues 3 and 4 ($Q_3$ and $Q_4$) refer to the two queues in machine 2 (see Fig. 1(a) for a diagram of this system). We can then define

Fig. 1. (a) Two polling systems with one fully cross-trained setup crew; and (b) the two decomposed and independent polling systems, each with a dedicated setup crew.
the status of the setup crew, where:

\[ \Pi = (n_1, n_2, n_3, n_4, i, j, v), \]  

where \( n_l \) represents the number of jobs at queue \( Q_l, l = 1, 2, 3 \) and 4. The variables \( i = 1 \) and 2 and \( j = 3 \) and 4 represent the job types currently being processed at machine 1 and machine 2, respectively, and \( v, v = 0, 1 \) and 2 denotes the status of the setup crew, where:

\[ v = \begin{cases} 
0 & \text{if the setup crew is idle,} \\
1 & \text{if the setup crew is performing a setup at machine } m, m = 1, 2. 
\end{cases} \]

The transition rates in the continuous-time Markov chain correspond to: (i) arrivals of new jobs; (ii) process completion of jobs; and (iii) setup completion at a machine. These transitions are detailed as follows: Define vector \( \bar{n} = (n_1, n_2, n_3, n_4) \), and write the state as \( \Pi = (\bar{n}, i, j, v) \). If \( \Pi' \) is a row vector of seven zeros with its \( u \)th element being one \((u = 1, 2, 3 \) and 4), then the transition rates \( q_{\Pi,\Pi'} \) are:

**Transitions due to arrivals:**

\[
q_{\Pi,\Pi'} = \begin{cases} 
\lambda_u & \text{if C1 or C2 holds, } u = 1, 2, 3 \text{ and 4}, \\
\lambda_1 & \text{if C3 or C4 holds}, \\
\lambda_2 & \text{if C5 or C6 holds}, \\
\lambda_3 & \text{if C7 or C8 holds}, \\
\lambda_4 & \text{if C9 or C10 holds}. 
\end{cases}
\]

**Transitions due to job processing completion:**

\[
q_{\Pi,\Pi'} = \begin{cases} 
\mu_u & \text{if D1 holds, } u = 1, 2, 3 \text{ and 4}, \\
\mu_1 & \text{if D2, D3, or D11 holds}, \\
\mu_2 & \text{if D4, D5, or D11 holds}, \\
\mu_3 & \text{if D6, D7, or D10 holds}, \\
\mu_4 & \text{if D8, D9, or D10 holds}. 
\end{cases}
\]

**Transitions due to setup completion:**

\[
q_{\Pi,\Pi'} = \begin{cases} 
\bar{\mu}_i & \text{if S1, or S3 holds}, \\
\bar{\mu}_j & \text{if S2, or S4 holds}. 
\end{cases}
\]

**Diagonal elements of the transition rate matrix:**

\[
q_{\Pi,\Pi} = \begin{cases} 
-\sum_{\Pi' \neq \Pi} q_{\Pi,\Pi'} & \text{for all } \Pi = (\bar{n}, i, j, v), \\
0 & \text{otherwise}. 
\end{cases}
\]

Conditions C1–C8, D1–D11 and S1–S4 are described in the Appendix.

Let \( \Pi \) be the vector of steady-state probabilities of being in one of the states described above, and let \( Q \) be the transition rate matrix of the Markov chain. Performance analysis of the system requires first obtaining values of \( \Pi \) by solving the system of equations \( \Pi Q = 0 \). This clearly does not have a closed-form solution, and we would have to solve it numerically. However, to obtain a numerical solution, we need to truncate the state space by assuming that each queue \( l = 1, 2, 3 \) and 4 has a maximum capacity of \( C_l \); the result is that the number of equations we need to solve is of the order of \( O(12C_1C_2C_3C_4) \), a very large number in practice even for this simplest system.

For medium-sized systems (i.e., systems with more than two machines, each having more than two queues), developing these equations is very difficult, not to mention that solving them requires a significant amount of computing power. Furthermore, extending the analysis to systems with any number of non-exponential machines, queues, or setup crews is even more troublesome. This leads us to the obvious need for a simple approximation that generates reasonably accurate performance measures of the system, while being easy to compute.

### 4. Approximating the average waiting time of a job

In this section, we develop an algorithm in order to approximate the waiting time that jobs experience in polling systems with shared cross-trained setup crews. In our approximation, we assume general distributions for job processing times and Erlang-P distributions for the setup times. Thus, we assume that a setup of queue \( Q_i \) consists of \( P \) phases, each being exponential with rate \( P\bar{\mu}_i \). We use our two-machine polling system with one setup crew to describe our approximation method, although our approach is easily extended to larger systems. In Section 5, we test our approximation in a system with four machines, three job types for each machine, and two setup crews. Our approximation focuses on estimating the likelihood of the unavailability of the setup crew upon request, and the corresponding average delay that machines experience in waiting for setup crews.

We first develop an estimate for the average delay that a machine experiences after its operator requests a setup. In the actual system, if the machine requests a setup but receives another job that is of the same type for which it is already set up, it will process that job and issue another setup request when it once again runs out of jobs of that type. However, in our approximation, we will combine the delay the machine experiences waiting for the setup and the duration of the setup; in addition, we will assume that no new jobs can be initiated during the time that the machine is waiting for setup crews.

In our two-machine system, at any point in time the status of \( Q_i, i = 1, 2, 3 \) and 4 of each polling system \( m, m = 1 \) and 2 can be one of the following, depending on the availability of the setup crew:

- **Machine is processing a job at queue** \( Q_i \).
- **A setup request has been issued at queue** \( Q_i \) and the setup crew is unavailable.
- **The setup crew is performing a setup for job type** \(-i\). In addition, since we assume the setup times are of Erlang type-P, we also denote the current phase of the setup that the crew is at as \( p \).
Cross-training setup crews for production systems

Let

- \((S_i, R_j, p)\) represent the states of the system when the setup crew is performing phase \(p\) of a type-\(i\) job setup, \(i = 1\) and 2, \(p = 1, \ldots, P\), and there is a type-\(j\), \(j = 3\) and 4 job setup request in the system. This includes states \((S_1, R_3, p), (S_2, R_3, p), (S_1, R_4, p)\) and \((S_2, R_4, p)\) for \(p = 1, \ldots, P\).
- \((R_i, S_j, p)\) represent the states of the system when the setup crew is performing phase \(p\) of a type-\(j\) job setup, \(j = 1\) and 2, \(p = 1, \ldots, P\), and there is a type-\(i\), \(i = 1\) and 2 job setup request. This includes states \((R_1, S_3, p), (R_2, S_3, p), (R_1, S_4, p)\) and \((R_2, S_4, p)\) for \(p = 1, \ldots, P\).
- \((S_i, R_j, p)\) represent the states of the system when the setup crew is processing phase \(p\) of a type-\(j\) job setup, \(i = 1\) and 2, \(p = 1, \ldots, P\), and machine 2 is processing a type-\(j\) job, \(j = 3\) and 4. This includes states \((S_1, P_3, p), (S_1, P_4, p), (S_2, P_3, p)\) and \((S_2, P_4, p)\) for \(p = 1, \ldots, P\).
- \((P_i, S_j, p)\) represent the states of the system when machine 1 is processing a type-\(i\) job, \(i = 1\) and 2, and machine 2 is processing a type-\(j\) job, \(j = 3\) and 4. This includes states \((P_1, P_3), (P_2, P_3), (P_1, P_4)\) and \((P_2, P_4)\).

Notice that states \((R_i, R_j, p)\), for \(i = 1\) and 2, \(j = 3\) and 4, and \(p = 1, \ldots, P\) are not feasible since both servers cannot be waiting for a setup. Similarly, states \((R_i, P_j)\) are not feasible. This is because state \((R_i, P_j)\) is equivalent to state \((S_i, P_j, 1)\), since in both states the system is performing phase 1 of a type-\(i\) job setup.

In order to find the rates at which the system enters and exits the above states, our approximation method assumes that each polling system has a dedicated setup crew; we also define the expected length of a busy period, \(1/r_i\), at \(Q_i\) as the expected time between the time instant at which the setup crew completes a setup at \(Q_i\) and the time at which the machine operator requests a setup for the next queue in the polling cycle. Note, for example, that \(r_1\) is the rate at which the system moves from states \((P_1, P_j)\) to \((S_2, P_j, 1)\), and \((P_1, S_j, p)\) to \((R_2, S_j, p)\) for \(j = 3\) and 4 and \(p = 1, \ldots, P\).

We approximate the distribution of the length of the busy period at \(Q_i\) with an exponential distribution with rate \(r_i\). This allows us to develop a continuous-time Markov chain in order to approximate the steady-state probabilities of the system being in states \((S_i, R_j, p), (R_i, S_j, p), (S_i, P_j, p), (P_i, S_j, p)\) and \((P_i, P_j)\) for \(i = 1\) and 2, \(j = 3\) and 4 and \(p = 1, \ldots, P\). The balance equations for the Markov chain for Erlang-P setup time distributions are as listed in Table 1.

Note that our continuous-time Markov chain assumes that the rates \(r_i\) for each queue \(Q_i\) are known. However, this is not the case for polling systems which share a setup crew. We later describe how our approximation uses an iterative approach to find accurate estimates of these rates.

Assuming that the \(r_i\)'s are known, we obtain the expected delay at each queue caused by the unavailability of the setup crew when a setup request has been issued for a type-\(i\) job, \(i = 1\) and 2. For example, for a type-1 job, the states in which there is a delay are \((R_1, S_3, p)\) and \((R_1, S_4, p)\), \(p = 1, \ldots, P\). Note that a setup request for a type-1 job could follow state \((P_2, P_2)\) when machine 1 finishes processing the jobs at queue 2. In this case, there would be no delay for the initiation of the setup since the setup crew is idle. On the other hand, once the system enters state \((R_1, S_3, p), p = 1, \ldots, P\), for example, there is a delay and the system leaves this state with rate \(p\tilde{\mu}_3\). Thus, the expected delay is the average time that the Markov chain spends in these states, given that a setup request results in the Markov chain entering these states.

Recall that our model requires that when the system is empty, the server resumes its service at the queue in which the first arrival occurs. Eisenberg (1994) studies this case where he calls “Stop-immediately/Jump to arrival”. We use Eisenberg’s results to obtain an approximation for the length of the busy period, and for the waiting time that jobs experience. Note that Eisenberg’s results are for a polling system with one dedicated setup crew. In Eisenberg (1994) it is shown that:

\[
1/r_i = \rho_i/\lambda \gamma_i, \tag{6}
\]

where \(\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_N\) for a polling system with \(N\) queues, \(\rho_i\) is the traffic intensity at \(Q_i\), and \(\gamma_i\) is the cycle rate at \(Q_i\) (i.e., the reciprocal of the average time between two consecutive visits to queue \(Q_i\)). \(\gamma_i\) can be found by solving Equation (3.15) in Eisenberg (1994).

Our approximation method decomposes the system of \(M\) polling systems and \(K\) setup crews into \(M\) separate polling systems, each with a dedicated setup crew. The basic idea stems from the fact that the unavailability of the shared setup crew results in a machine waiting time for setup operations. If this waiting time can be accurately estimated and added to the machine setup times, then each machine can be considered to have a dedicated setup crew and revised setup times. These revised setup times in the corresponding system with a dedicated setup crew are larger than the original setup times in the system with a shared setup crew.

As an example, consider our two-machine system with one setup crew. Our approximation method decomposes this system into two independent one-machine problems, each with a dedicated setup crew (see Fig. 1(b)). Our approximation method uses an iterative process to approximate the average job waiting times as follows. It first uses Eisenberg’s results to find an initial estimate for the \(r_i\)'s by assuming that each polling system has its dedicated setup crew. These initial estimates are then used in the balance equations listed in Table 1 in order to obtain initial estimates for the steady-state probabilities of the states of the system and the expected delays for the setup crew. The average setup time in each queue of the decomposed polling
systems are then revised by adding the expected delay for the
setup crew to the original average setup times.

We continue to assume that the revised setup times have
an Erlang-P distribution, since this keeps the coefficient of
variation of the revised setup time distribution the same as
before, and since this assumption has led to good approxi-
mutations in our numerical studies. Then, using the revised
average setup times in the decomposed system, and keep-
ing the same distribution for the setup times as before, our
approximation method obtains the new estimate for the $r_i$’s
(i.e., we keep $P$ the same, but change the mean in the Erlang
distribution). This in turn leads to new steady-state prob-
abilities for the system of equations listed in Table 1, and
therefore new estimates for the average delays for setup at
each queue. This iterative process continues until the ex-
pected delays converge. Although we do not have an ana-
lytical proof that convergence is guaranteed, we can report
that every example in the nearly 200 examples we have tested
contained convergence. After convergence of the $r_i$ values,
we then calculate the latest revised values for the average
setup times at the various queues used in the decomposed
systems, and the average job waiting times. These values are
the estimates for the waiting times in the system.

Clearly, the approach described above can be extended
to more than two machines and one setup crew. For exam-
ple, in a case with $M$ machines and $K$ setup crews, where
the setups have an Erlang-P distribution, one would have
to identify the state of each machine and setup crew. Since
each machine is either: (i) being set up; (ii) producing; or
(iii) waiting for a setup crew, it can be in one of three
states. The setup crews either would be setting up one of
the $M$ machines and would be in phase $p$ of this setup or
they would be idle. Thus, an upper bound on the number of
states in our balance equations is $3^M \times (P + 1)^K$. Note
that our assumption is that there is a priority ordering of
machines such that lower-indexed machines have a higher

---

Table 1. The balance equations for the Markov chain

<table>
<thead>
<tr>
<th>State</th>
<th>Balance equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(S_1, R_1, 1)$</td>
<td>$p \mu_1 \pi(S_1, R_1, 1) = r_1 \pi(S_1, R_1, p_1)$</td>
</tr>
<tr>
<td>$(S_1, R_1, p)$</td>
<td>$p \mu_1 \pi(S_1, R_1, p) = r_1 \pi(S_1, R_1, p_1) + p \mu_1 \pi(S_1, R_1, p-1)$, $p = 2, \ldots, P$</td>
</tr>
<tr>
<td>$(S_1, R_1, 1)$</td>
<td>$r_1 \pi(S_1, R_1, 1)$</td>
</tr>
<tr>
<td>$(S_1, R_1, p)$</td>
<td>$r_1 \pi(S_1, R_1, p) = r_1 \pi(S_1, R_1, p_1) + p \mu_1 \pi(S_1, R_1, p-1)$, $p = 2, \ldots, P$</td>
</tr>
<tr>
<td>$(S_1, R_1, 1)$</td>
<td>$(p \mu_1 + r_1) \pi(S_1, R_1, 1) = r_2 \pi(P_1, R_1) + p \mu_1 \pi(S_1, R_1, 1)$</td>
</tr>
<tr>
<td>$(S_1, R_1, p)$</td>
<td>$(p \mu_1 + r_1) \pi(S_1, R_1, p) = r_2 \pi(P_1, R_1) + p \mu_1 \pi(S_1, R_1, p-1)$, $p = 2, \ldots, P$</td>
</tr>
<tr>
<td>$(S_1, R_1, 1)$</td>
<td>$(p \mu_2 + r_1) \pi(S_1, R_1, 1) = r_2 \pi(P_1, R_1) + p \mu_1 \pi(S_1, R_1, 1)$</td>
</tr>
<tr>
<td>$(S_1, R_1, p)$</td>
<td>$(p \mu_2 + r_1) \pi(S_1, R_1, p) = r_2 \pi(P_1, R_1) + p \mu_1 \pi(S_1, R_1, p-1)$, $p = 2, \ldots, P$</td>
</tr>
</tbody>
</table>

---

Kula et al.
setup priority. If this is not the case and machines are of equal priority so that the crews carry out those setups on a First-Come First Served (FCFS) basis, we would then also need to keep track of the order of the setup requests. This will significantly increase the size of the state space. In this case an upper bound on the number of states would be \(3^M \times (P + 1)^K \times \sum_{i=1}^{M-1} [M!/(M-i)!]\). Similar to other queueing systems with multi-class customers and FCFS service discipline, the number of states can become very large for large systems.

5. Computational study and managerial insights

In this section, we report the results of a computational study to test the effectiveness of our approximation. We also provide some managerial insights based on our study.

5.1. Performance evaluation of the heuristic

We conducted an extensive computational study to test the effectiveness of our approximation. Our study consisted of four groups. First, we conducted a study on a system with two polling systems, each with two queues served by a single cross-trained setup crew. In our experimental design, we chose utilization levels of 0.6, 0.75 and 0.9 for the utilization of each polling system. In particular, a utilization of 0.6 was generated by selecting vectors (0.6, 0.3) for \(\lambda\) and (2, 1) for \(\mu\). Similarly, a utilization of 0.75 was generated by considering vectors (0.35, 0.80) for \(\lambda\) and (1, 2) for \(\mu\). Finally, utilization of 0.9 was generated by assigning vectors (1, 0.4) for \(\lambda\) and (2, 1) for \(\mu\). For average setup times in queues \(Q_1, Q_2, Q_3\), and \(Q_4\) we chose vectors (8, 8, 8, 8), (8, 8, 2, 2) and (2, 2, 2, 2). The total number of combinations created for this group (i.e., group 1) was thus 27, all of which are listed in Table 2.

Similarly, in group 2, we considered two polling systems, each with three queues, and once again generated 27 examples using the data given in Table 2. (Note that each group corresponds to nine problem instances, three combinations of arrival and service rates, and three combinations of setup times.) In group 3, we increased the number of queues in each polling system to five queues. Finally in group 4, we had four polling systems, each with three queues, but unlike the other examples, we had two setup crews.

For all groups in Table 2, we considered exponential, Erlang-2 and Erlang-4 distributions for setups and processing in order to examine the effects of variability on the performance of our heuristic. We observed that different processing distributions had no effect on the accuracy of our heuristic. Due to space constraints, Table 2 presents only the results where both service and processing distributions were Erlang-2. In the examples in Table 2, machines with lower indices had a higher setup priority when there were multiple machines requesting a setup.

In order to test the accuracy of our approximation, we performed a simulation study to obtain the average waiting time of each job class. For each case, the simulation was run until all job classes had at least a million job arrivals. The warm-up period was between 100,000 and 200,000 jobs depending on traffic intensity. As can be observed from Table 2, the differences between the approximation and the simulation results are fairly small, with the average percentage difference under 2.5% in all cases. The maximum difference between the simulated value and the approximate value observed among all the cases was 4.25%. In most cases, the difference between the heuristic and the simulation was between 0.50 and 2.25%.

Our numerical results show that the approximation works very well in estimating the average waiting time of jobs in the systems. The reason why our approximation works can be explained as follows. It is well known that the expected waiting time in cyclic exhaustive polling

<table>
<thead>
<tr>
<th>Group</th>
<th>((\lambda, \mu))</th>
<th>Setup time combinations</th>
<th>Avg. percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.60, 0.30)</td>
<td>(2, 1)</td>
<td>(8, 8, 8, 8), (8, 8, 2, 2), (2, 2, 2, 2)</td>
</tr>
<tr>
<td></td>
<td>(0.35, 0.80)</td>
<td>(1, 2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1.00, 0.40)</td>
<td>(2, 1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(0.2, 0.4, 0.6)</td>
<td>(1, 2, 3)</td>
<td>(8, 8, 8, 8, 8), (8, 8, 2, 2, 2), (2, 2, 2, 2, 2)</td>
</tr>
<tr>
<td></td>
<td>(0.25, 0.50, 1.00)</td>
<td>(1, 2, 4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.30, 0.60, 0.90)</td>
<td>(1, 2, 3)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(0.12, 0.24, 0.48, 0.96, 0.96)</td>
<td>(1, 2, 4, 8)</td>
<td>(2, 2, 2, 2, 2, 2, 2, 2)</td>
</tr>
<tr>
<td></td>
<td>(0.15, 0.30, 0.60, 1.20, 1.20)</td>
<td>(1, 2, 4, 8)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0.18, 0.36, 0.42, 1.44, 1.44)</td>
<td>(1, 2, 4, 8)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(0.60, 0.40, 0.30)</td>
<td>(2, 4, 3)</td>
<td>(1, 1, 1, 0.5, 0.5, 0.5, 1, 1, 0.5, 0.5, 0.5, 0.5)</td>
</tr>
<tr>
<td></td>
<td>(0.20, 0.40, 0.60)</td>
<td>(1, 2, 3)</td>
<td>(3, 3, 3, 3, 3, 3, 1.5, 1.5, 1.5, 1.5, 1.5)</td>
</tr>
<tr>
<td></td>
<td>(0.50, 1.00, 0.60)</td>
<td>(2, 4, 2)</td>
<td>(8, 8, 6, 6, 6, 6, 8, 8, 8, 6, 6)</td>
</tr>
</tbody>
</table>
systems with deterministic switchover times depends only on the sum of switchover times needed to complete a tour of all queues. The expected waiting time at a queue is independent of how much of the total switchover time comes from each pair of queues. The expected waiting time at a queue is independent of how much of the switchover time comes from each pair of queues. However, for non-deterministic switchover times, expected waiting times depend on the individual switchover distributions via their variances (Srinivasan et al., 1995; Cooper et al., 1996; Borst and Boxma, 1997). Van der Mei (2000) shows that in heavy traffic, the waiting time distributions in systems with non-deterministic switchover times depend on the individual switchover distributions only through the sum of expected switchover times. This implies that as the traffic intensity in a standard polling system increases, the effect of individual switchover distributions on waiting time distributions vanishes. It seems that our approximation does a reasonable job of approximating the mean delay imposed by the wait for the setup crew; also, the previous results demonstrate that this is the most important moment of the wait to estimate, especially as the system approaches heavy traffic.

5.2. Managerial insights

From a manager’s perspective, it is extremely valuable to explore the effect of the level of cross-training of the setup crew on waiting times that jobs experience. More specifically, it is important to see if and when increasing the level of cross-training of the crew improves the waiting times of all jobs. As an example, consider a system with six machines (polling systems), each serving three types of jobs and attended by two setup crews. Consider a situation where we have two setup crews, and the crews process setup requests in a FCFS manner. In scenario (a), setup crew 1 is trained only to attend to machines 1, 2 and 3, whereas setup crew 2 is trained only to serve machines 4, 5 and 6. In scenario (b), both setup crews are fully cross-trained so that they can serve a setup request from any machine. The question is whether increasing the level of cross-training of setup crews necessarily benefits the waiting times at all machines.

To address this question, we first considered fully symmetrical systems (i.e., systems with equal arrival, service, and setup rates for all machines) with parameters as shown in Table 3. All six examples in Table 3 consider six identical polling systems, each serving three types of jobs. For each example, Table 3 provides the average waiting times $w_i$ for queue $i = 1, 2$ and 3 in only one of the six identical polling systems. The examples have machine utilizations varying from 0.3 to 0.9, and the average setup times equal to either five or two. We again studied cases where the job processing times and setup times are exponential, Erlang-2 and Erlang-4 distributed, and found that the insights were independent of distribution. Therefore, in Tables 4 and 5, we present the results only for the Erlang-4 setup and processing distributions.

As Table 3 clearly indicates, the system with a fully cross-trained crew performs better in all examples. The decrease

<table>
<thead>
<tr>
<th>Example</th>
<th>Queue</th>
<th>$\rho$</th>
<th>$s_i$</th>
<th>Partially cross-trained setup crew, $w_i$ (a)</th>
<th>Fully cross-trained setup crew, $w_i$ (b)</th>
<th>Percentage improvement, $\frac{(a) - (b)}{(a)} \times 100$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.20</td>
<td>5</td>
<td>12.29</td>
<td>11.45</td>
<td>6.83</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.20</td>
<td>2</td>
<td>2.89</td>
<td>2.46</td>
<td>14.88</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.05</td>
<td>5</td>
<td>15.92</td>
<td>15.46</td>
<td>2.89</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.10</td>
<td>5</td>
<td>4.59</td>
<td>4.46</td>
<td>2.48</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.10</td>
<td>2</td>
<td>24.25</td>
<td>24.23</td>
<td>5.36</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.40</td>
<td>5</td>
<td>21.71</td>
<td>21.35</td>
<td>6.36</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.20</td>
<td>2</td>
<td>19.01</td>
<td>18.63</td>
<td>6.73</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.30</td>
<td>2</td>
<td>7.59</td>
<td>7.01</td>
<td>7.64</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.40</td>
<td>2</td>
<td>54.40</td>
<td>51.48</td>
<td>5.37</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.20</td>
<td>5</td>
<td>72.65</td>
<td>68.71</td>
<td>5.42</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.30</td>
<td>5</td>
<td>63.57</td>
<td>60.15</td>
<td>5.38</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.40</td>
<td>2</td>
<td>24.82</td>
<td>23.56</td>
<td>5.08</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.20</td>
<td>2</td>
<td>33.22</td>
<td>31.48</td>
<td>5.24</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.30</td>
<td>2</td>
<td>29.08</td>
<td>27.60</td>
<td>5.09</td>
</tr>
</tbody>
</table>
### Table 4. Comparison of systems using partially cross-trained crews and fully cross-trained crews for various utilization rates, Erlang-4 processing and Erlang-4 setup times in asymmetrical systems

<table>
<thead>
<tr>
<th>Example</th>
<th>Machines</th>
<th>Queue</th>
<th>$\rho$</th>
<th>$s_i$</th>
<th>Partially cross-trained setup crew, $w_i$ (a)</th>
<th>Fully cross-trained setup crew, $w_i$ (b)</th>
<th>Percentage improvement $\left[\frac{(a) - (b)}{(a)}\right] \times 100$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 3</td>
<td>1</td>
<td>0.20</td>
<td>5</td>
<td>12.29</td>
<td>7.75</td>
<td>36.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.05</td>
<td>5</td>
<td>15.92</td>
<td>10.85</td>
<td>31.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.05</td>
<td>5</td>
<td>15.73</td>
<td>10.55</td>
<td>32.93</td>
</tr>
<tr>
<td></td>
<td>4, 5, 6</td>
<td>1</td>
<td>0.40</td>
<td>5</td>
<td>54.50</td>
<td>63.08</td>
<td>-15.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.20</td>
<td>5</td>
<td>72.65</td>
<td>84.60</td>
<td>-16.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.30</td>
<td>5</td>
<td>63.57</td>
<td>73.94</td>
<td>-16.31</td>
</tr>
<tr>
<td>2</td>
<td>1, 2, 3</td>
<td>1</td>
<td>0.20</td>
<td>5</td>
<td>12.29</td>
<td>8.63</td>
<td>29.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.05</td>
<td>5</td>
<td>15.92</td>
<td>11.98</td>
<td>24.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.05</td>
<td>5</td>
<td>15.73</td>
<td>11.69</td>
<td>25.68</td>
</tr>
<tr>
<td></td>
<td>4, 5, 6</td>
<td>1</td>
<td>0.20</td>
<td>2</td>
<td>2.89</td>
<td>4.22</td>
<td>-46.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.05</td>
<td>2</td>
<td>4.59</td>
<td>6.77</td>
<td>-47.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.05</td>
<td>2</td>
<td>4.46</td>
<td>6.70</td>
<td>-50.22</td>
</tr>
<tr>
<td>3</td>
<td>1, 2, 3</td>
<td>1</td>
<td>0.20</td>
<td>5</td>
<td>12.29</td>
<td>7.50</td>
<td>38.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.05</td>
<td>5</td>
<td>15.92</td>
<td>11.98</td>
<td>33.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.05</td>
<td>5</td>
<td>15.73</td>
<td>10.20</td>
<td>35.16</td>
</tr>
<tr>
<td></td>
<td>4, 5, 6</td>
<td>1</td>
<td>0.40</td>
<td>2</td>
<td>24.82</td>
<td>33.34</td>
<td>-34.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.20</td>
<td>2</td>
<td>33.22</td>
<td>44.58</td>
<td>-34.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0.30</td>
<td>2</td>
<td>29.08</td>
<td>38.67</td>
<td>-32.98</td>
</tr>
</tbody>
</table>

In the average waiting times varied from approximately 3% to about 15%. The improvement in performance can be explained by the pooling effect achieved by having fully cross-trained workers. As examples in Table 3 show, pooling the setup crew in symmetrical systems results in shorter setup times for all job types. The effects of fully cross-training the setup crew in symmetrical systems are most significant when machine utilization is low and setup times are short. This can be clearly seen in example 2 in Table 3. On the other hand, examples 5 and 6 demonstrate that as machine

### Table 5. Comparison of average waiting times for Erlang-4 processing and setup times at four machines each with two job types, for fully cross-trained setup crew compared with a setup crew assigned to each machine

<table>
<thead>
<tr>
<th>Example</th>
<th>Queue</th>
<th>$\rho_i$</th>
<th>Average setup times</th>
<th>Waiting times, system with a shared crew (c)</th>
<th>Waiting times standard polling model (d)</th>
<th>Percentage difference $\left[\frac{(c) - (d)}{(d)}\right] \times 100$ (%)</th>
<th>Setup crew utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.50</td>
<td>2</td>
<td>16.56</td>
<td>15.34</td>
<td>7.95</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.40</td>
<td>2</td>
<td>20.01</td>
<td>18.51</td>
<td>8.10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.50</td>
<td>10</td>
<td>61.47</td>
<td>56.27</td>
<td>9.24</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.40</td>
<td>10</td>
<td>73.97</td>
<td>67.33</td>
<td>9.37</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.20</td>
<td>2</td>
<td>2.53</td>
<td>1.89</td>
<td>33.86</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.1</td>
<td>2</td>
<td>3.47</td>
<td>2.59</td>
<td>33.98</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.20</td>
<td>10</td>
<td>18.67</td>
<td>12.77</td>
<td>46.20</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.1</td>
<td>10</td>
<td>21.12</td>
<td>14.36</td>
<td>47.10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.50</td>
<td>10</td>
<td>66.72</td>
<td>56.27</td>
<td>18.57</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.40</td>
<td>10</td>
<td>80.45</td>
<td>67.63</td>
<td>18.96</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.50</td>
<td>2</td>
<td>27.00</td>
<td>15.34</td>
<td>76.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.40</td>
<td>2</td>
<td>32.76</td>
<td>18.51</td>
<td>76.98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.30</td>
<td>10</td>
<td>20.61</td>
<td>19.63</td>
<td>4.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.30</td>
<td>10</td>
<td>20.61</td>
<td>19.63</td>
<td>4.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.30</td>
<td>2</td>
<td>8.12</td>
<td>4.57</td>
<td>77.68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.30</td>
<td>2</td>
<td>8.12</td>
<td>4.57</td>
<td>77.68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
utilization increases, the pooling effect resulting from having fully cross-trained setup crews decreases due to the decrease in the number of setups performed in the system.

To test whether the same pooling effect applies in asymmetrical systems, we created the three examples shown in Table 4. The interesting result in these cases is that when the crews are fully cross-trained and they continue to serve setup requests in a FCFS manner, the waiting times at some queues could increase while those at others decrease. For instance, in example 1 of Table 4, fully cross-training the setup crew decreases waiting times at machines 1, 2 and 3, but increases waiting times at machines 4, 5 and 6. It is important to explore why increasing the level of cross-training makes the performance of machines 4, 5 and 6 worse.

Notice that this behavior can be explained by the differences in utilization between machines 1–3 and machines 4–6. Machines 4–6 each have a utilization level of 0.9, while machines 1–3 have a utilization level of 0.3. Interestingly, in a polling system where machines require setup when they exhaust jobs of one type, low-utilization stations require setups more often. Thus, when both setup crews are available to serve all machines, machines 1–3 start making disproportionate use of the setup crew which was previously available only for machines 4–6. This results in a better performance for jobs at machines 1–3 and a worse performance for jobs at machines 4–6. This pattern is the same in examples 2 and 3 of Table 4.

In example 2 of Table 4, all machines have the same utilization but machines 1–3 have longer setup times than machines 4–6. Therefore, full cross-training once again results in a higher use of the setup crews at machines 1–3, leading to a better performance for those machines and a worse performance for the others. Finally, example 3 shows a situation where machines 1–3 have both lower utilizations and longer setup times; the magnitude of the effect described above is thus significantly increased.

Our practical experience indicates that in many factories, supervisors responsible for particular operations may show some resistance to full cross-training of workers. This resistance is driven by the fear that performance at their particular operation may suffer. Our examples in Table 4 have illustrated situations where this could happen. We conjecture that the pooling effect caused by full cross-training benefits all stations in symmetric systems. On the other hand, in asymmetric systems, this pooling effect may result in some stations overusing the now pooled setup crews while other stations may in fact experience longer delays. In the examples we presented, the crews served jobs in a FCFS manner both before and after cross-training. However, this resulted in machines with low utilizations or longer setup times dominating the usage of the crews after cross-training; as a result, jobs at other machines experienced longer delays. These patterns demonstrate that after cross-training crews, managers should re-evaluate the priority scheme used for setup requests, and should make adjustments if the performance of certain job types will become significantly worse.

It is also important for managers to identify situations where using existing polling models, which do not model the effect of sharing setup crews, can result in misleading conclusions. Since many systems with setup crews attending multiple machines exist in practice, it is worthwhile to analyze when the job waiting times are significantly affected by the use of the shared setup crews. Conversely, it is important to understand when the delay induced by waiting for a setup crew is likely to be small and can be safely ignored.

Table 5 contains the results of an example with four machines, each processing two different types of jobs. All processing and setup time distributions are Erlang-4. This example is typical of the many we have examined to gain insights into the above issue. Table 5 represents the average job waiting times for the case in which two fully cross-trained setup crews attend different machines, and for the case where each machine is served by a dedicated setup crew. Thus, the difference between the two cases would give us one indication of the importance of modeling the delays induced by waits for setup crews.

In examples 1, 2, 3 and 4 in Table 5, all four machines and the types of jobs, their arrival and service rates, etc., are identical, and therefore we present the results for only one machine. Examples 1 and 2 have a machine utilization of 0.9 whereas examples 3 and 4 have a machine utilization of 0.3. In example 5, machines 1 and 2 have utilization levels of 0.9, and machines 3 and 4 have utilization levels of 0.6, setup times are 10, 2, 10 and 2 at machines 1–4, respectively. The last column of Table 5 shows the overall utilization of the fully cross-trained setup crew.

The examples in Table 5 show that a failure to take into account the wait for shared setup crews can result in significant underestimation of job waiting times. Even in these simple examples, the underestimation is as high as 77%. The underestimation as a percentage becomes significant when the crew utilization is high (as in examples 3–5). Example 5 also demonstrates that in an asymmetrical system, the effect of ignoring the delay for the setup crew varies a lot from machine to machine. This fact demonstrates that the common practice of inflating waiting times equally (by the same absolute or percentage level) to account for the effect of setup crew delays could result in significantly misleading estimates, and thus emphasizes the need for the analysis we have provided in this paper.

Acknowledgements

We are grateful to Candace Yano and an anonymous referee for their insightful comments that greatly improved the paper. The work of the second and third authors has been supported in part by the National Science Foundation, grant DMI-0000125.
References


Appendix

- **C1:** Condition C1 represents the arrival of a type-1 job, \( u = 1, 2, 3 \) and 4 when the crew is performing a setup. This corresponds to the transition from all \( n = (i, j, v) \), to \( n' = n + 1^u \), \( u = 1, 2, 3 \) and 4, and \( v \neq 0 \).

- **C2:** Condition C2 represents the arrival of a type-2 job, \( u = 1, 2, 3 \) and 4 when both of the machines have at least one job each and the setup crew is idle. This corresponds to the transition from all \( n = (i, j, 0) \), to \( n' = n + 1^u \), \( u = 1, 2, 3 \) and 4 except \((n_1 = 0, n_2 = 0)\) and \((n_3 = 0, n_4 = 0)\).

- **C3:** Condition C3 represents the arrival of a type-1 job to machine 1 when machine 1 is idle and the last setup performed at machine 1 was type-1 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (0, 0, n_3, n_4, 1, j, 0) \) to \( n' = (1, 0, n_3, n_4, 1, j, 0) \), \( j = 3 \) and 4.

- **C4:** Condition C4 represents the arrival of a type-1 job to machine 1 when machine 1 is idle and the last setup performed at machine 1 was a type-2 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (0, 0, n_3, n_4, 2, j, 0) \) to \( n' = (1, 0, n_3, n_4, 2, j, 1) \), \( j = 3 \) and 4.

- **C5:** Condition C5 represents the arrival of a type-2 job to machine 1 when machine 1 is idle and the last setup performed at machine 1 was a type-1 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (0, 0, n_3, n_4, 1, j, 0) \) to \( n' = (0, 1, n_3, n_4, 2, j, 1) \), \( j = 3 \) and 4.

- **C6:** Condition C6 represents the arrival of a type-2 job to machine 1 when machine 1 is idle and the last setup performed at machine 1 was a type-2 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (0, 0, n_3, n_4, 2, j, 0) \) to \( n' = (0, 1, n_3, n_4, 2, j, 0) \), \( j = 3 \) and 4.

- **C7:** Condition C7 represents the arrival of a type-3 job to machine 2 when machine 2 is idle and the last setup performed at machine 2 was a type-3 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 0, 0, i, 3, 0) \) to \( n' = (n_1, n_2, 1, 0, i, 3, 0) \), \( i = 1 \) and 2.

- **C8:** Condition C8 represents the arrival of a type-3 job to machine 2 when machine 2 is idle and the last setup performed at machine 2 was a type-4 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 0, 0, i, 4, 0) \) to \( n' = (n_1, n_2, 1, 0, i, 3, 2) \), \( i = 1 \) and 2.

- **C9:** Condition C9 represents the arrival of a type-4 job to machine 2 when machine 2 is idle and the last setup performed at machine 2 was a type-3 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 0, 0, i, 3, 0) \) to \( n' = (n_1, n_2, 0, 1, i, 4, 2) \), \( i = 1 \) and 2.

- **C10:** Condition C10 represents the arrival of a type-4 job to machine 2 when machine 2 is idle and the last setup...
performed at machine 2 was a type-4 job setup and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 0, 0, i, 4, 0) \) to \( n' = (n_1, n_2, 0, 1, i, 4, 0) \), \( i = 1 \) and 2.

- **D1:** Condition D1 represents the completion of a type-4 job, \( n = 1, 2, 3, 4 \) when there is more than one type-4 job in the system and the setup crew is idle. This corresponds to the transition from all \( n = (\mathbf{t}, i, j, 0) \) to \( n' = n - 1^i \), where \( n_u > 1 \) for \( u = 1, 2, 3 \) and 4 and \( v = 0 \).

- **D2:** Condition D2 represents the completion of a type-1 job when there is one type-1 job, at least one type-2 job at machine 1 and the setup crew is idle. This corresponds to the transition from all \( n = (1, n_2, n_3, n_4, 1, j, 0) \) to \( n' = (0, n_2, n_3, n_4, 2, j, 1) \), \( n_2 > 0 \), \( j = 3 \) and 4.

- **D3:** Condition D3 represents the completion of a type-1 job when there is one type-2 job, no type-2 job at machine 1 and the setup crew is idle. This corresponds to the transition from all \( n = (1, 0, n_1, n_3, 1, j, 0) \) to \( n' = (0, 0, n_1, n_3, 1, j, 0) \), \( j = 3 \) and 4.

- **D4:** Condition D4 represents the completion of a type-2 job when there is one type-2 job, at least one type-1 job at machine 1 and the setup crew is idle. This corresponds to the transition from all \( n = (1, 0, n_1, n_3, n_4, 1, j, 0) \) to \( n' = (0, 0, n_1, n_3, n_4, 1, j, 0) \), \( j = 3 \) and 4.

- **D5:** Condition D5 represents the completion of a type-2 job when there is one type-2 job, no type-1 job at machine 1 and the setup crew is idle. This corresponds to the transition from all \( n = (0, 1, n_2, n_4, 2, j, 0) \) to \( n' = (0, 0, n_2, n_4, 2, j, 0) \), \( j = 3 \) and 4.

- **D6:** Condition D6 represents the completion of a type-3 job when there is one type-3 job, at least one type-4 job at machine 2 and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 1, n_3, 3, 0) \) to \( n' = (n_1, n_2, 0, n_3, i, 4, 2) \), \( n_4 > 0 \), \( i = 1 \) and 2.

- **D7:** Condition D7 represents the completion of a type-3 job when there is one type-3 job, no type-4 job at machine 2 and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 1, 0, i, 3, 0) \) to \( n' = (n_1, n_2, 0, 0, i, 3, 0) \), \( i = 1 \) and 2.

- **D8:** Condition D8 represents the completion of a type-4 job when there is one type-4 job at least one type-3 job at machine 2 and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 3, 1, i, 4, 0) \) to \( n' = (n_1, n_2, 3, 0, i, 3, 2) \), \( n_3 > 0 \), \( i = 1 \) and 2.

- **D9:** Condition D9 represents the completion of a type-4 job when there is one type-4 job, no type-3 job at machine 2 and the setup crew is idle. This corresponds to the transition from all \( n = (n_1, n_2, 0, 1, i, 4, 0) \) to \( n' = (n_1, n_2, 0, 0, i, 4, 0) \), \( n_3 > 0 \), \( i = 1 \) and 2.

- **D10:** Condition D10 represents the completion of a type-j job, \( j = 3 \) and 4 and the setup crew is setting up machine 1. This corresponds to the transition from all \( n = (\mathbf{u}, i, j, 1) \) to \( n' = \mathbf{n} - 1^i \), when \( n_j > 0 \).

- **D11:** Condition D11 represents the completion of a type-i job, \( i = 1 \) and 2 and the setup crew is setting up machine 2. This corresponds to the transition from all \( n = (\mathbf{u}, i, j, 1) \) to \( n' = \mathbf{n} - 1^i \), when \( n_i > 0 \).

- **S1:** Condition S1 represents a type-i, \( i = 1 \) and 2 job setup completion at machine 1 and machine 2 is processing a type-j job, \( j = 3 \) and 4. This corresponds to the transition from all \( n = (\mathbf{u}, i, j, 1) \) to \( n' = (\mathbf{u}, i, j, 0) \), when \( n_i > 0 \).

- **S2:** Condition S2 represents type-j, \( j = 3 \) and 4 job setup completion at machine 2 and machine 1 is processing a type-i job, \( i = 1 \) and 2. This corresponds to the transition from all \( n = (\mathbf{u}, i, j, 2) \) to \( n' = (\mathbf{u}, i, j, 0) \), when \( n_i > 0 \).

- **S3:** Condition S3 represents a type-i, \( i = 1 \) and 2 job setup completion at machine 1 when machine 2 is idle, the last setup performed at machine 2 was a type-j job setup, \( j = 3 \) and 4 and there is a type-j, \( j' = 3 \) and 4 job setup request at machine 2. This corresponds to the transition from all \( n = (\mathbf{u}, i, j, 1) \) to \( n' = (\mathbf{u}, i, j, 2) \), \( j' = 3 \) and 4, and \( j \neq j' \).

- **S4:** Condition S4 represents a type-j, \( j = 3 \) and 4 job setup completion at machine 2 when machine 1 is idle, the last setup performed at machine 1 was a type-i job setup, \( i = 1 \) and 2 and there is a type-i, \( i' = 1 \) and 2 job setup request at machine 1. This corresponds to the transition from all \( n = (\mathbf{u}, i, j, 2) \) to \( n' = (\mathbf{u}, i', j, 1) \), \( i = 1 \) and 2, \( i' = 1 \) and 2, and \( i \neq i' \).

### Biographies

Ufuk Kula completed his PhD in Industrial and Operations Engineering at the University of Michigan in December 2003. His interests are in production and inventory control.

Izak Duenyas is Associate Dean for Faculty Development and Research and John Psarouthakis Professor of Manufacturing Management at the University of Michigan Business School, and also holds a joint appointment as a professor of industrial and operations engineering. His research interests are in supply chain management, production control and investments in new technology.

Seyed Iravani is Pentair-Nugent Assistant Professor in Manufacturing in the Department of Industrial Engineering and Management Sciences at Northwestern University. His research interests are in the stochastic modeling of manufacturing systems and the analysis and control of queueing systems. He is on the Editorial Board of IEEE Transactions, and he is also an Associate Editor for Management Science and Naval Research Logistics.