Benefits of Skill Chaining in Serial Production Lines with Cross-Trained Workers

Wallace J. Hopp
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208-3119, hopp@northwestern.edu

Eylem Tekin
Department of Operations Research, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina 27599-3180, eylem@unc.edu

Mark P. Van Oyen
Department of Information Systems and Operations Management, Loyola University Chicago, 25 East Pearson Street, Chicago, Illinois 60611-2196, mvanoye@orion.luc.edu

To gain insight into the potential logistical benefits of worker cross-training and agile workforce policies, we study simple models of serial production systems with flexible servers operating under a constant work-in-process (CONWIP) release policy. Two important and interrelated issues are: (a) how to decide which skill(s) are strategically most desirable for workers to gain, and (b) how to coordinate these workers to respond dynamically to congestion. We address these by considering two cross-training strategies: a straightforward capacity-balancing approach, which we call cherry picking (CP), and an innovative overlapping zone strategy that we call skill chaining. Our comparison shows that skill-chaining strategies have the potential to be robust and efficient methods for implementing workforce agility in serial production lines.

Key words: cross-training; workforce; agile production; CONWIP

History: Accepted by William S. Lovejoy, received August 29, 2001. This paper was with the authors 14 months for 2 revisions.

1. Introduction
In recent decades, the shift from mass production of highly standardized products to small-batch production of customized or niche products has forced most manufacturing organizations to use some form of flexible capacity. One attractive form of flexible capacity is the use of cross-trained workers who can shift their capacity to where it is needed. Cross-trained workers can improve efficiency in the form of higher throughput, lower inventory, shorter cycle times, and/or improved service without significant additional investment in equipment and labor. In this paper, we focus on the impact of workforce agility in serial production lines on operational efficiency in the sense of achieving a low work-in-process (WIP) to throughput ratio. From Little’s law, it follows that such efficiency also implies a low cycle time regime, as advocated by time-based competition principles.

Most real-life production lines possess both short-term variability in their operations and imbalanced average workloads across stations. In such environments, cross-trained workers can improve efficiency in two primary ways. First, lines that are imbalanced with respect to average work content will cause some workers to idle occasionally. Cross-training enables workers to divide their effort between stations, thereby improving their utilization and the resulting throughput of the line. We refer to this benefit as capacity balancing. Second, even if average workloads are balanced between stations, variability in process times (due to changes in product mix, supply disruptions, equipment failures, worker behavior, and the like) will cause workers to idle from time to time. Such variability is often accommodated through WIP buffers between operations, however, designers and managers of production lines are under constant pressure to reduce WIP (see Conway et al. 1988, McClain et al. 1992). Cross-training can improve productivity without increasing WIP by allowing a worker starved at one station to switch to another station. We refer to this benefit as variability buffering. From a queueing perspective, variability buffering is achieved via cross-training by partial pooling of servers (see Mandelbaum and Reiman 1998 for a treatment of full pooling). An effective agile workforce design uses a “reasonable” amount of cross-training to achieve higher operational efficiency both by balancing capacity and by buffering variability.

Implementing workforce agility involves three levels of decisions. The first is whether or not to use
flexible workers at all (as opposed to using some other form of flexibility or a traditional inflexible system). The second is which workers to cross-train for which skills (i.e., what skill pattern to introduce). The third is, given the skill sets of workers, how to assign the cross-trained workers to different tasks over time (i.e., what worker coordination policy to use). In general, the third decision must be resolved in order to address the second decision, which must be resolved in order to address the first decision. In this research, we concentrate primarily on the choice of skill pattern by contrasting the performance of two cross-training strategies, each operating under a well-suited worker coordination policy. Our goal is to provide general, strategic insights into the selection of an effective skill pattern. We also illustrate the practical utility of our results through a model of a real-world system with 10 workers and 18 total skills that can achieve a 17.8% increase in throughput with the addition of only two skills and no change in WIP level.

1.1. Research Issues
We begin with a qualitative overview of the key issues involved in this study. Consider a serial line with several stations, each with a single specialist worker assigned to it, and an imbalanced average work content across stations. Without cross-training, the maximum attainable capacity for the line is the capacity of the bottleneck. However, this capacity could be increased by cross-training workers to shift excess capacity from low-utilization stations to the high-utilization stations. Hence, the first question we address is which skills to provide to workers to maximize capacity. For instance, consider the four-station line in Figure 1, where the third station is the bottleneck. If Station 3 is sufficiently undercapacitated, a commonsense approach is to borrow capacity from underutilized workers at Stations 1, 2, and 4, as indicated with dotted lines representing cross-training of workers. We have observed this approach in a number of real systems (e.g., an IBM circuit board plant). Goldratt (1992) also emphasizes this strategy in his Theory of Constraints philosophy, which is widely used in practice and taught in many business schools. We term the strategy illustrated in Figure 1 cherry picking (CP), because capacity is “picked” from other stations to augment the bottleneck.

Although CP is intuitively appealing when variability is considered, there may be a competing rationale for selecting skills according to a different pattern. Specifically, to provide better flexibility, it may be desirable to have the ability to shift capacity from any station to any other station. One way to do this is via a skill-chaining pattern like that illustrated in Figure 2(a). In contrast to the CP pattern of Figure 1, where all workers can directly help the bottleneck station, in Figure 2(a) only Worker 2 is trained to help directly at the bottleneck. However, Workers 1 and 4 can indirectly help the bottleneck station by absorbing some or all of the work content at Stations 1, 2, and 4. The notion of chaining was introduced by Jordan and Graves (1995) in the context of process flexibility for a single-stage (i.e., parallel tasks) manufacturing system, extended to multistage (i.e., serial tasks) manufacturing systems by Graves and Tomlin (2003), and used by Benjaafar et al. (1998) to consider equipment flexibility. Brusco and Johns (1998) considered minimizing workforce staffing costs in the maintenance operations of a paper mill and indicated that chaining of worker skills is particularly useful among other cross-training structures.

In this paper, we contrast the CP and skill-chaining strategies. First, we show how to balance capacity in a given line under each skill pattern. Second, we investigate how these two skill patterns perform in systems with variability. While chaining would appear
to offer superior variability buffering, it does so at the cost of requiring more skills than a direct capacity-balancing approach because it cross-trains workers at highly utilized station(s) even though this is not needed for balancing capacity. So, we also contrast CP with partial chaining strategies having the same number of skills. For example, we compare the strategy shown in Figure 1 with the partial chaining strategy in Figure 2(b). In the latter, Worker 3 (the bottleneck worker) is not cross-trained because an analysis of capacity reveals that with unlimited WIP all capacity can be 100% utilized without cross-training the bottleneck worker. The authors have observed a partial skill chaining pattern at the Ruud Lighting, Inc. located in Kenosha, Wisconsin (see §6 for a discussion). In this context, we consider the questions of (a) whether cross-training of overutilized workers can be attractive, and (b) in what environments it is worth having the additional skills of the skill-chaining pattern.

Addressing these issues requires specifying a coordination policy that assigns workers to tasks over time. To make fair comparisons of skill patterns, it is important to use an optimal (or near-optimal) coordination policy for each skill pattern. In this paper, for systems with exponential processing times, we compute the optimal policies and performance of various skill patterns by using Markov decision process (MDP) models. For systems with nonexponential processing times, we use simulation to investigate the performance of the two skill patterns under a range of heuristic coordination policies, some original to this paper and others drawn from the literature and industrial practices. A comparison of these heuristics with respect to the optimal policy is also provided for exponential processing times and lines with three and four stations.

The results of our study show that when capacity is fairly imbalanced, but variability is low, the direct capacity-balancing approach (CP) can be used provided careful consideration is given to the worker coordination policy. When capacity imbalance is modest, but variability is significant, the strategies based on skill chaining with a worker coordination policy that controls queue lengths is more robust. When both factors are significant, hybrid strategies that combine elements of both CP and chaining are effective. Finally, we observe that adding a skill (e.g., by cross-training bottleneck workers) to “complete the chain” can result in surprisingly large performance benefits. On the whole, our research suggests that skill chaining may be an attractive component of practical workforce agility strategies.

1.2 Previous Research
Treleven (1989) provided a survey of the workforce agility literature up to 1989. This covered early studies on systems where machines and workers are simultaneously limiting resources (e.g., there are fewer workers than the machines, so workers are cross-trained to make each resource operable when it is needed). A more recent survey of the literature was given by Hopp and Van Oyen (2002).

A broad class of literature considers scheduling of cross-trained workers to tasks on the basis of clock time, for instance, using shifts or a predetermined sequence of tasks. In general, these studies present a constrained optimization problem to produce a workforce schedule for a given demand forecast. Examples include models for airlines and railways, health care, call-center operations, and production systems (see, e.g., Berman et al. 1997, Campbell 1999, Eaves and Rothblum 1988, Lee and Vairaktarakis 1997).

Another branch of the literature on serial production lines with cross-trained workers has been typified by TSS (Toyota Sewn Products System) lines, where there are more stations than workers. In TSS lines, workers are cross-trained to do all tasks within a zone of consecutive stations, possibly the entire line. Each worker moves down the line carrying an item and working on it at each station in the assigned zone until she either reaches the end of her zone or is preempted by another worker coming back upstream. For such lines with deterministic process times, researchers have shown that when workers are sequenced from slowest to fastest, a stable partition of zones will emerge and the line will balance itself (Bartholdi and Eisenstein 1996, Bartholdi et al. 1999). Bischack (1996), Zavadlav et al. (1996), and McClain et al. (2000) considered TSS-like systems with random processing times. In general, these studies used simulation models to investigate the appropriate size of worker zones and the worker coordination policy. They showed that provision for buffers is necessary to cope with variability, and assigning workers to overlapping zones increases efficiency when coupled with half-full buffer worker coordination policies (i.e., if the buffer is less than half full, work upstream of that buffer; otherwise work downstream).

Other than TSS lines, Downey and Leonard (1992) investigated assembly lines where workers are cross-trained for every task and showed that after each job completion, assigning a worker to the station with the maximum workload (i.e., queue length times the mean process time) works well. Van Oyen et al. (2001) investigated systems with fully cross-trained workers who can collaborate on a single job, and showed that the policy that maximizes throughput is the “expedite policy” (i.e., workers are assigned to the same job from start to finish). Gel et al. (2001) focused on hierarchical zone cross-training, in which skills of some workers are subsets of the skills of other workers. For two-machine tandem lines, they established
a “fixed before shared” principle, which says that broadly skilled workers should give strict priority to the tasks for which only they are trained.

Although research on TSS lines and systems where workers are fully cross-trained offer valuable insights, it does not directly address the decision of how to invest in skills. Moreover, for many production lines, the motivation for worker cross-training is not a result of having fewer workers than machines. Even when each machine is staffed, cross-training can enable a line to cope with capacity imbalances and variability in process times so that it can achieve high throughput with low WIP. Unfortunately, there is very little in the technical literature to guide a manager on the effective pattern and the necessary amount of cross-training, or the role that cross-training plays in reducing WIP. In this paper, we take a step towards providing these insights.

The remainder of this paper is as follows. In §2, we introduce two skill-pattern strategies and provide methodologies to balance a line under each skill pattern. In §3, we model each skill pattern as an MDP model (when process times are exponentially distributed), introduce heuristic worker coordination policies (when process times are general), and investigate the performance of these heuristics under each skill pattern. Section 4 compares the two skill patterns in a variety of environments. We consider hybrid skill-pattern strategies in §5, illustrate our insights with a brief case study in §6, and conclude in §7.

2. Skill-Pattern Strategies

We consider asynchronous flow lines having \( N \) stations with one worker assigned to each station (as her base station). Service times at a station are i.i.d. and independent of other stations. A CONstant Work-In-Process (CONWIP) release policy (see Hopp and Spearman 2000, Spearman et al. 1990) is employed, because it is a simple, broadly applicable release policy, and it allows us to quantify the trade-off between WIP and cross-training as methods for buffering variability. (Without flexible workers the system corresponds to a closed queueing network with a fixed number of jobs in the network. Cross-trained workers add an element of control to the queueing network, where the instantaneous job completion rate of a station is governed by the number of workers assigned to it.) Workers are assumed to be always available and able to switch between stations without cost or changeover time. As such, our models are most applicable to short or U-shaped production lines or systems where tasks are facilitated by information systems (e.g., phone calls, computer files) and are routed to workers. Workers cannot work on the same job simultaneously, but we assume there is sufficient equipment available for more than one worker to work at a station with one job per busy worker. Tasks cannot be preempted once begun (which is usually the case in our experience because preemption often takes time and can affect quality).

Workers can vary in speed and are benchmarked relative to a “standard worker” by defining the speed factor of each worker relative to this worker. We denote the speed factors by \( v_w \), \( w = 1, \ldots, N \), and further assume that a worker’s speed factor applies uniformly across all tasks. (We relax this assumption in §3.) Letting \( T_n \) denote the mean processing time at station \( n \) for a standard worker, this implies that worker \( w \) will have a mean processing time of \( T_n/v_w \) at station \( n \). We express the capacity of the line (i.e., the maximum rate it can achieve given full cross-training and infinite WIP) as \( \lambda = \sum_{n=1}^{N} v_w \sum_{n=1}^{N} T_n \) jobs per unit time. We define a line to be balanced if it has sufficient cross-training to achieve throughput of \( \lambda \) as WIP approaches infinity. Note that in a balanced line, utilization of all workers will approach 100% as WIP approaches infinity. In §§2.1 and 2.2, we rigorously define CP and D-skill-chaining (DSC) skill-pattern strategies, respectively, and show how each can achieve a balanced line.

### 2.1. Strategy 1: Cherry Picking

To determine how many additional skills are needed to balance a line, we start by considering the problem of how to achieve a specified throughput target of \( \lambda \leq \bar{\lambda} \). This requires that each station has capacity to process at least at rate \( \bar{\lambda} \). Assuming that initially each station is attended by one worker, for each station and worker pair we can compute the quantity \( |v_w (\lambda T_n/v_w - 1)| = |\lambda T_n - v_w| \), representing the surplus or shortage of work units at station \( n \). We define set \( \mathcal{J} \) to include stations with \( h_n \pm \lambda T_n - v_w > 0 \), \( n \in N, N \pm \{1, \ldots, N\} \) (i.e., the set of stations that need help from other workers to produce at rate \( \lambda \)). Let \( \mathcal{J} = N \setminus \mathcal{J} \) denote the workers with \( s_w = - (\lambda T_n - v_w) > 0 \), \( w \in N \) (i.e., the set of workers who have excess capacity). This suggests that the simplest cross-training strategy one could follow to obtain throughput \( \lambda \) is to pick a worker who has excess capacity and assign that worker to one or more station(s) that require help. If we do this in a manner such that the line is balanced (i.e., has capacity \( \bar{\lambda} \)) with the minimum number of additional skills, we term this strategy cherry picking (CP). In other words, CP finds the minimal number of assignments of workers in \( \mathcal{J} \) to stations in \( \mathcal{J} \) so that all stations have capacity of at least \( \lambda \); if \( \lambda = \bar{\lambda} \), then the line is balanced. We define the following decision variables for \( n \in \mathcal{J} \) and \( w \in \mathcal{J} \):

\[ x_{nw} = \text{number of standard worker time units of effort that worker } w \text{ allocates to station } n \text{ per unit time,} \]

\[ f_{nw} = \begin{cases} 1 & \text{if worker } w \text{ is cross-trained for station } n, \\ 0 & \text{otherwise.} \end{cases} \]
By definition, $\sum_{w=1}^{N} s_w \geq \sum_{n=1}^{N} h_n$ as long as $\lambda \leq \hat{\lambda}$. The mathematical formulation for this problem can be stated as follows:

$$\min \sum \sum f_{nw}$$

subject to

$$\sum_{w \in J} x_{nw} \leq s_w, \quad w \in J,$$

$$\sum_{w \in J} x_{nw} = h_n, \quad n \in J,$$

$$x_{nw} \leq s_w f_{nw}, \quad n \in J, \quad w \in J,$$

$$x_{nw} \geq 0, \quad f_{nw} \in [0, 1], \quad n \in J, \quad w \in J.$$

This formulation is equivalent to the transportation problem with fixed edge costs, where in this case fixed edge costs are all equal to 1 and flow costs are equal to zero. This problem is known to be NP-hard (Garey and Johnson 1979). Although computationally problematic for large problems, it is not difficult to find the optimal CP strategy for moderately sized lines. In addition, we can bound the number of skills that will be needed as follows. (All proofs are given in the online appendix, available at mansci.pubs.informs.org/companion.html.)

**Theorem 1.** For a CP strategy, the maximum number of additional skills required to balance the line is $N - 1$, where $N$ denotes the number of stations.

To illustrate this result, consider a line with four stations with identical workers and the mean processing times of 1, 1, 1, 2. Cross-training allows this line to attain a throughput of $\hat{\lambda} = 0.8$. CP cross-trains Workers 1, 2, and 3 ($N - 1 = 3$ additional skills) to enable each of them to allocate 20% of their effort to Station 4, which balances the line. Although CP fully addresses the issue of capacity balancing, it does not necessarily address the variability buffering issue. Hence, it may be desirable to provide more than the minimum amount of cross-training needed to balance the line.

### 2.2. Strategy 2: D-Skill Chaining

To provide enhanced variability buffering, it may be desirable to create a skill pattern that allows every worker to directly or indirectly (via paths through other workers) redirect effort from her base station to any other station. To achieve this, the network of skill assignments should be “strongly connected,” which requires a minimum of $N$ additional skills (Nemhauser and Wolsey 1988). We propose such a pattern, called 2-skill chaining (2SC), which is a special case of a more general pattern we term D-skill chaining (DSC).

We define regular DSC to be the case where each worker is cross-trained for her base station and for the next $D - 1$ stations (a total of $D$ skills per worker). As an example of regular 2SC, consider the line shown in Figure 2 with four stations denoted by 1, 2, 3, 4, where the workers are trained for the following skill sets: $(1, 2)$, $(2, 3)$, $(3, 4)$, $(4, 1)$. Notice that every station has exactly two workers capable of covering it and that stations are chained together by overlapping work assignments.

Note, however, that DSC does not necessarily feature cross-training for adjacent stations and can be defined as follows: Each worker covers exactly $D$ unique stations, each station is served by $D$ unique workers, and the graph with arcs (skills) connecting workers and stations is connected. There are many ways to construct such a $D$-skill chain. These can be enumerated (possibly with some double counting) by indexing the stations in all possible sequences and applying regular DSC to the indices. To balance capacity, we identify the minimal value of $D$ such that DSC enables the line to attain a throughput of $\hat{\lambda}$ as WIP tends to infinity. To keep our discussion simple, we consider only regular DSC, which is done to the “right” in the sense that worker $n$ is trained for her base station, $n$, and also $n \oplus 1, n \oplus 2, \ldots, n \oplus (D - 1)$. However, our approach is straightforward to extend to other DSC structures. Here, we are using $\oplus$ to denote mod $N$ arithmetic defined as follows: For $s, i \in \mathcal{N}$, $s \oplus i = s + i$ if $s + i \leq N$; otherwise, it equals $s + i - N$. Similarly, we let $\ominus$ denote mod $N$ subtraction defined as $s \ominus i = s - i$ if $s - i \geq 1$; otherwise, it equals $s - i + N$. In the sequel, we will simply use DSC to refer to regular DSC unless otherwise specified.

With $d$ degrees of cross-training, we denote by $\alpha^d(w, n)$ the number of standard worker time units of effort that worker $w$ allocates to station $n$ per unit time, whether actually servicing jobs or merely idling. In general, $\alpha^d(w, n)$ is a function of the task/worker assignment policy. Clearly, when $d = 1$, $\alpha^d(w, n) = v_w$ for all $w = n \in \mathcal{N}$, and is zero otherwise because there is no cross-training. We formally define the minimal degree of skill chaining necessary to achieve capacity of $\hat{\lambda}$ ($\lambda \leq \hat{\lambda}$) as $D = \min\{d \in \mathcal{N}: \Theta(d) \geq \hat{\lambda}\}$. In turn, $\Theta(d)$ is specified by

$$\max \Theta(d) = \min_{n=1, 2, \ldots, N} \sum_{w=1}^{N} \frac{\alpha^d(w, n)}{T_n}$$

subject to

$$\sum_{w \oplus (d-1)} \alpha^d(w, n) = v_w, \quad w \in \mathcal{N}$$

$$\alpha^d(w, n) \geq 0,$$

$$w \in \mathcal{N}, \quad n \in \{w, w \oplus 1, \ldots, w \oplus (d - 1)\},$$

$$\alpha^d(w, n) = 0,$$

$$w \in \mathcal{N}, \quad n \in \{w \oplus d, \ldots, w \oplus (N - 1)\}.$$
As before, we say the line is balanced if it has capacity of $\bar{\lambda}$. We can now state the following.

**Theorem 2.** To achieve a capacity of $\lambda \leq \bar{\lambda}$ in a non-identical worker line using DSC, it is necessary and sufficient to use $D = K + 1$, where $K = \max_{i,j} k_{ij}$ and

$$k_{ij} \triangleq \min_{k=0,1,\ldots,N-1} \left\{ k : \sum_{n=i}^{\infty} (\lambda T_n - \hat{v}_n) - \sum_{n=0}^{\infty} \hat{v}_n \leq 0 \right\}$$

for $i, j = 1, \ldots, N$. (3)

In Theorem 2, $K$ is determined by the pair, say $(t, l)$, that achieves the maximum in Equation (3). We call the sequence of stations that starts with $t$ and has length $l$ a bottleneck cell. To determine the bottleneck cell, Equation (3) considers any segment of stations in the line with length $j, j = 1, \ldots, N$, beginning with station $i$, $i \in \mathcal{N}$, and determines the number of workers, denoted by $k$, it would require from the preceding stations to achieve the desired capacity. Note that when workers are standard (i.e., $\hat{v}_n = 1, w \in \mathcal{N}$), Equation (3) reduces to

$$k_{ij}^S \triangleq \left\lceil \sum_{n=i}^{\infty} (\lambda T_n - 1) \right\rceil.$$ (4)

Hence, if we define $\bar{L} = \max_{i,j} k_{ij}^S$, then for lines with identical workers, it is necessary and sufficient to use $D = \bar{L} + 1$-level skill chaining to achieve a capacity of $\lambda$. This rounds up the excess work content for each segment of the line to the nearest integer to find the minimal number of workers required to help downstream stations.

An important consequence of Theorem 2 is that it leads to the solution of a more general problem. Consider a production line where there are $N$ teams of workers trained in a DSC pattern, each team having a base station and $(D - 1)$ cross-trained tasks. More specifically, team $i$, $i \in \mathcal{N}$, has station $i$ as a base station, and is trained for tasks $i$, $i \oplus 1$, $\ldots$, $i \oplus (D - 1)$. Let $S_i$ be the size of team $i$, so the workforce has size $W = \sum_i S_i$. If we index workers by team $(i)$ and identity $(j)$ such that $(ij)$ is the $j$th person in team $i$, and denote the speed of that worker by $\hat{v}_{ij}$, then $v_i = \sum_{j=1}^{S_i} \hat{v}_{ij}$ gives the overall speed-based capacity of team $i$. Hence, we can state the following.

**Corollary 1.** Given that each station is staffed with a team of workers, each having different speeds, we can find the minimum value of $D$ needed to balance the line under the DSC strategy by aggregating each team into a single worker with speed $v_i$ at station $i$, for $i = 1, \ldots, N$, and applying Theorem 2.

Note that the basic assumption of arbitrarily high WIP enables this aggregation by keeping all workers busy all the time, subject to the workload and cross-training conditions. In the sequel, we assume a single worker per station.

DSC is well suited to variability buffering because its symmetrical cross-training allows worker capacity to be shifted from any station to any other. One drawback of DSC is that it must be possible to cross-train in pairs. In practice, some stations may require a specialist that will not or cannot service another station. Similarly, in a straight and long production line, it may not be possible to have a worker serving both Station 1 and station $N$. Conversely, physically compact lines such as U-shaped lines are especially attractive for 2SC. Walk-times encourage the selection of adjacent workstations for worker skill sets, keeping their effect negligible. Another drawback of chaining is the number of additional skills required. For example, consider a six-station serial line with identical workers and $T_n = n$ for $n = 1, 2, \ldots, 6$, which enables a throughput of 6/21 when balanced. CP assigns Worker 1 to Station 6, Worker 2 to Station 5, and Worker 3 to Station 4. Therefore, the minimum number of skills that balances this line is $6 + 3 = 9$. In contrast, DSC requires $D = 3$, which leads to a total of 18 skills in theory, although only 11 skills are actually used to balance the line (i.e., when Worker 1 is cross-trained for Stations 2 and 3, Worker 2 for Station 4, Worker 3 for Station 5, and Worker 4 for Station 6, the line is balanced). The remainder of the skills only serve a variability-buffering function.

### 3. Worker Coordination Policies

In this section, we address the question of how to dynamically assign workers to tasks over time, given their skill sets. This is necessary to examine how CP and DSC behave under conditions of variability.

#### 3.1. Optimal Coordination Policies

We define $\mu_{wn} = \hat{v}_w/T_n$ to be the rate at which worker $w$ processes jobs at station $n$, and assume that process times are distributed exponentially with mean $1/\mu_{wn}, w, n \in \mathcal{N}$. Given worker skill sets, the problem is to maximize throughput by finding the optimal assignment of workers to tasks. To show the complicated structure of this optimization problem, we formulate it as a MDP for the simplest case: 2SC. The decision epochs are the job completion times. The state of the system at any decision epoch is represented by a vector $(x, s) = (x_1, x_2, \ldots, x_N, s_1, s_2, \ldots, s_N)$, where $x_n$ is the number of jobs waiting for service in station $n$’s queue, and $s_w$ is the status of worker $w$. For each worker $w \in \mathcal{N}, s_w \in \{0, 1, 2\}$, where 0 refers to the idling state, and 1(2) refers to the worker processing at her base (secondary) station. Because we do not allow preemption, it is possible that one or more workers idle at some states under
the optimal policy. Note that \( \sum_{n=1}^{N} x_n + \sum_{n=1}^{N} I_{(s_n \neq 0)} = \text{WIP} \), where \( I_{(s_n \neq 0)} \) is the indicator function of set \( A \). For each worker \( w \), the actions are (1) I: Idle, (2) W1: Start processing at the base station, and (3) W2: Start processing at the secondary station. Further details of the MDP formulation are given in §3 of the online appendix.

As an example, consider a simple case of a four-station line with identical workers and identical average processing times of stations. Applying the value iteration algorithm, we can solve for the optimal coordination policy. Table 1 depicts the optimal decisions for allocating Worker 1 for some selected states at a WIP level of eight. This shows that the optimal worker coordination policy can be an idling policy, even in a simple system where there is no difference in worker speeds or station processing times. Optimal decisions also change based on the current locations of the remaining workers in the line, even when station queue lengths remain the same. Furthermore, the structure of the optimal coordination policy depends on other system parameters such as the WIP level, mean processing times, and worker speeds, which makes it very difficult to implement.

The skill assignments under CP also depend on system parameters in a complex way, which makes the optimal policy complicated to express and implement. However, real-life applications generally require an easy-to-implement worker policy. Therefore, to facilitate comparison between different skill-pattern strategies, in the next section we introduce a range of easy-to-implement policies motivated by the literature and industrial practice.

### 3.2. Heuristic Worker Coordination Policies

We consider nine heuristic worker coordination policies that we categorize as static, queue-length-based, and workload-based policies. Static policies (Fixed-Before-Shared, Zoned Craft, and PRIority) do not use any information regarding the state of the system. Queue-length-based policies (MaxQueue, MaxQueue-Gap, Uniform Buffer) use real-time information on the number of jobs in queue. Workload-based policies (MaxLoad, MaxGap, Time Buffer) use available information to compute the total average processing times in queues. We first define some additional notation to describe our policies:

- \( T_{ij} \): raw processing time of the line \( (T_0 = \sum_{n=1}^{N} T_n) \)
- \( S_w \): set of stations for which worker \( w \) is skilled (i.e., the skill set of \( w \)).
- \( Q_n(t) \): observed queue length at workstation \( n \) at time \( t \).
- \( W_n(t) \): average workload at station \( n \) at time \( t \), which is computed by \( W_n(t) = Q_n(t) \times T_n \).
- \( G_n(t) \): the difference (gap) between the workloads of stations \( n \) and \( n+1 \), i.e., \( G_n(t) = W_n(t) - W_{n+1}(t) \).
- \( G_n^2(t) \): the difference (gap) between the queue lengths of stations \( n \) and \( n+1 \), i.e., \( G_n^2(t) = Q_n(t) - Q_{n+1}(t) \).

The full set of coordination policies we implemented are given below. As will be seen from the following descriptions, the PRI, ML, MQG, MG, UB, and TB heuristics were developed as part of this research project, while the others were taken from the literature.

- **Fixed-Before-Shared Policy** (FBS). Assigns worker \( w \) to her base station first if there are any available jobs. This policy was shown to be effective for systems with hierarchical cross-training (Gel et al. 2001).
- **Zoned Craft Policy** (ZC). This policy is only applicable to designs where all workers are cross-trained for consecutive stations, which includes DSC but generally not CP. Beginning from her base station, worker \( w \) carries a job and processes it successively until the end of her zone (i.e., until station \( w + D - 1 \)), unless she bumps into an idle worker downstream. Then she returns to her base station and gets the next available job; or, if there is no available work, she successively checks the downstream stations for available jobs within her zone. This is a form of *craft production* as analyzed in Van Oyen et al. (2001) and is in the spirit of *bucket brigade* policies discussed in Bartholdi and Eisenstein (1996) and *leave* policies developed in McClain et al. (2000).
- **PRIority Policy** (PRI). Assigns worker \( w \) to the nonempty station with the highest priority. Priorities for stations in \( S_w \) are determined based on the ranking of the fractions computed as \( x_{nu} \) for CP and \( \alpha^2 \) for DSC (i.e., the station with the largest long-run effort allocation fraction for that worker is given the highest priority).

| Table 1: Examples of Optimal Coordination Policies for a Four-Station Line with Identical Workers and Identical Mean Process Times |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 4 | 1 | 1 | 1 | W2 |
| 0 | 1 | 0 | 4 | 1 | 1 | 2 | W2 |
| 0 | 1 | 0 | 4 | 1 | 2 | 1 | W2 |
| 0 | 1 | 0 | 4 | 1 | 2 | 2 | W2 |
| 0 | 1 | 0 | 4 | 2 | 1 | 1 | W2 |
| 0 | 1 | 0 | 4 | 2 | 2 | 1 | W2 |
| 0 | 1 | 0 | 4 | 2 | 2 | 1 | W2 |
| 0 | 1 | 0 | 4 | 2 | 2 | 1 | W2 |


MaxQueue Policy (MQ). Assigns worker \( w \) to station \( n^{*} \) with the longest queue; i.e.,
\[
n^{*} = \arg \max_{n \in S_n, Q_n(t) > 0} Q_n(t).
\]

Askin and Iyer (1993) implemented this policy to minimize throughput times in cellular manufacturing systems. Nelson (1967) found this policy effective in minimizing the flow time in a job shop system when compared to random, first-come-first-served and assignment to shortest operation time coordination policies.

Maxload Policy (ML). Assigns worker \( w \) to station \( n^{*} \) with the largest workload; i.e.,
\[
n^{*} = \arg \max_{n \in S_n, Q_n(t) > 0} W_n(t).
\]

Downey and Leonard (1992) studied a different version of this policy for systems with finite buffers. In their study, when a worker becomes idle, she goes to the station with the maximum production index defined by \( \max[\min(Q_n, B - Q_{n+1})S_n] \) for all stations that are not currently staffed. Here, \( Q_n \) is the queue length of station \( n \), \( B \) is the size of the buffer, and \( S_n \) is the mean processing time at station \( n \). If in the above expression \( B \) tends to infinity (infinite buffer), this policy reduces to the maxload policy.

MaxQueueGap Policy (MQG). Assigns worker \( w \) to station \( n^{*} \) with the maximum gap between its queue length and that of the next station; i.e.,
\[
\text{MaxQueueGap Policy (MQG)}
\]
\[
n^{*} = \arg \max_{n \in S_n, Q_n(t) > 0} G_n(t).
\]

We developed this policy and found it to be nearly optimal for symmetric three-station lines.

MaxGap Policy (MG). Assigns worker \( w \) to station \( n^{*} \) with the maximum gap between its workload and that of the next station; i.e.,
\[
n^{*} = \arg \max_{n \in S_n, Q_n(t) > 0} G_n(t).
\]

Buffer Policies. These policies are specifically applicable to 2SC where each worker is cross-trained for two consecutive stations. The idea is that each buffer is assigned a certain threshold value. After a job completion, worker \( w \) processes a job from station \( w + 1 \) if the queue length of station \( w + 1 \) exceeds the threshold value. Otherwise, she processes a job from station \( w \).

We consider two different buffer threshold values in this study: the Uniform Buffer (UB) and the Time Buffer (TB) policies. The UB sets the threshold values equal to WIP/\( N \) for all buffers. The idea behind this approach is to balance the queue lengths at each station. Because the UB only balances queue lengths, it is not sensitive to different workloads. To account for this, the TB considers the relative magnitudes of mean processing times at consecutive stations and their ratio to the total mean processing time of the line. The TB policy uses threshold values of
\[
B_n = \frac{\text{WIP}}{N} \times \frac{T_n}{T_{n+1}} \times \frac{T_0}{N} \times \frac{(T_n + T_{n+1})}{2}.
\]

The first term in the above expression refers to the point at which the workload is balanced over all stations and the other terms are for scaling relative workloads at consecutive stations. In particular, \( T_n/T_{n+1} \) scales \( B_n \) for relative workload at stations \( n \) and \( n + 1 \) (a smaller mean processing time at the upstream station decreases the threshold value). The last term scales \( B_n \) to correct imbalance that places a large or small amount of work at stations \( n \) and \( n + 1 \). In other words, if there is a large work content at stations \( n \) or \( n + 1 \), then we want to lower the threshold. These two buffer policies were suggested by the half-full buffer policies implemented and shown to be effective in the literature (e.g., Ostolaza et al. 1990, Zavadlav et al. 1996, McClain et al. 2000) for open systems with finite buffers.

3.3. Performance of Heuristic Policies

In this section, we examine the performance of the heuristic worker coordination policies: (1) find out how well the heuristics perform with respect to the optimal policy by studying three- and four-station lines (the longest we could solve optimally), and (2) investigate whether a heuristic or class of heuristics performs uniformly well under 2SC and/or CP.

3.3.1. Method of Investigation. We used simulation to compute the performance of heuristics and to consider different variability levels and longer lines. For our simulation analysis, systems were simulated using a computer program written in C. Each simulation run began with an empty system, and ended after 120,000 jobs exited the line, including a warm-up period of 10,000 jobs. Each run was replicated 10 times. For variance reduction purposes, the same sequence of random numbers was used for each simulation, and each station had its own random number stream. At a confidence level of 99%, all standard errors were within 0.3%.

We studied three, four, and six-station tandem lines, with each worker having a speed of 1 at their primary task. For each line length, we set the sum of the mean processing times (raw processing time of the line) equal to the number of stations (\( N \)) so that the capacity of all systems is 1. To highlight the effects of capacity imbalance, we considered different designs with one or more bottleneck stations. For three-station lines, Station 1 was the bottleneck. For four-station...
Skill Chaining in Serial Production Lines

For each design, we chose the mean processing times at the bottleneck stations as 25%, 50%, and 75% greater than the average processing time, with equal mean processing times at all other stations. We focused on environments that can be balanced under chaining with $D = 2$. (In §5, we discuss situations where $D = 2$ is not enough to achieve capacity balancing and suggest a hybrid approach.) We generated processing times from a gamma distribution. We considered coefficient of variability (cv) values of 0.2, 1, and 3 to correspond to low variability (e.g., automated equipment), significant variability (e.g., labor-dominated or service sector tasks), and high variability (e.g., unreliable machines or diverse product mix). For each capacity imbalance level, we used the three cv values, and also set cv values of the bottleneck stations to high (cv = 3) and others to low (cv = 0.2), and vice versa. When comparing the heuristics with the optimal policy, we could only consider cv = 1 for which gamma distribution reduces to exponential distribution. The complete set of experiments is given in §4 of the online appendix.

To consider cases where workers do not perform at their secondary station as efficiently as at their primary station, we studied three efficiency levels for workers: no loss in efficiency (i.e., standard worker performance regardless of the task type), and 10% and 20% increase in the mean processing times at the secondary stations. As a result, for example, for a six-station line we use three capacity imbalance levels, three bottleneck designs, five variability scenarios, three worker efficiency levels, and nine heuristic worker allocation policies (a total of 1,215 environments to compute throughput at a given WIP level).

In a CONWIP line, one can measure performance in two ways. First, we could set a WIP level and observe the resulting throughput. Second, we could find the necessary amount of WIP to achieve a desired throughput. The cross-training strategy that achieves the target throughput with the lowest WIP is best. In the sequel, we use both approaches to demonstrate our results.

### 3.3.2. Comparison of the Heuristics

We compared the performance of the heuristics with that of the optimal policy for three and four-station lines and exponential processing times. Tables 2 and 3 present the average and maximum percentage errors from the optimal for four-station lines. In Tables 2 and 3, L, M, and H refer to low, medium, and high capacity imbalance, respectively. Each level of imbalance labeled L, M, or H reports the average and maximum deviation of heuristics from the optimal for a given policy and a total of six problem instances, i.e., 3 WIP levels as $2N$, $3N$, $4N$, and two designs. The last column of the tables presents the percent errors relative to the best heuristic for each problem instance, and the last row (indicated by “#”) gives the number of times each heuristic performed the best.

These tables show a wide range of performance among heuristics. However, in general, the best policy for each environment performs well. However, the heuristics get worse with the decrease in efficiency at the secondary stations. As the efficiency at the secondary stations decreases, the optimal policy tends to idle the cross-trained worker when there are no jobs to process at the primary station, instead of assigning her to the secondary station where she will be very slow. Thus, nonidling policies become increasingly suboptimal with the decrease in efficiency.

In Table 2, average and maximum errors show that Fixed-Before-Shared, MaxQueue, and MaxQueueGap policies are most effective under CP. Although the PRI policy has the best performance for many of the cases.
(40 out of 54), it may perform poorly for high-capacity imbalance cases. In such environments, the priority policy usually assigns workers to the bottleneck station(s), which causes jobs to accumulate at the nonbottleneck stations. Because there is only one worker available for the nonbottleneck station(s) under CP, a bottleneck worker may starve as too much WIP accumulates downstream of the bottleneck. Similarly, workload-based policies such as MaxLoad and MaxGap do not perform as well as their queue-length-based counterparts. From Table 3, we see that for 2SC, queue-length-based policies, especially UB, perform very well. In contrast, static policies (FBS, ZC, and PRI) and workload-based policies (ML, MG, and TB) have relatively high errors.

An immediate conclusion of our results is that under both skill-pattern strategies, queue-length-based policies almost always perform better than workload-based policies. This result is not limited to four-station lines and exponential processing times, but is true if we search for the best heuristic for longer lines with different processing-time distributions (see §5 of the online appendix for comparison of the heuristics in six-station lines). The reason workload-based policies can perform poorly, especially in highly imbalanced cases, is that if a worker is cross-trained for a short and a long task, workload-based policies tend to assign the worker to the station with long processing times even if there are not many jobs there. Thus, a worker may spend insufficient time at stations with short process times. Using workload-based policies with CP can produce even more suboptimal performance because when a worker is assigned to one short and one long task, this worker is the only person who can do the short task. Assigning this worker to the longer task frequently creates more workload imbalance in the system. This leads us to:

Conclusion 1. For both CP and 2SC, queue-length-based policies outperform workload-based policies.

A second observation from our tests is that the performance of CP depends strongly on the worker coordination policy and the characteristics of the environment. We observe dramatic decreases in performance

| Table 3: Average and Maximum Deviations of the Heuristic Policies from the Optimal for Four Stations and 2SC Strategy |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 2SC             | FBS             | ZC              | PRI             | MO              | ML              | MOG             | MG              | UB              | TB              | Best             |
| No efficiency loss |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| L               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 5.47            | 2.83            | 5.47            | 1.01            | 1.00            | 0.40            | 0.63            | 0.84            | 0.81            | 0.38             |
| Max             | 7.34            | 3.33            | 7.34            | 2.08            | 2.17            | 0.93            | 2.05            | 1.97            | 2.17            | 0.88             |
| M               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 5.78            | 3.92            | 5.78            | 1.35            | 2.28            | 0.71            | 1.80            | 1.08            | 2.74            | 0.65             |
| Max             | 7.17            | 5.24            | 7.17            | 2.27            | 4.88            | 1.33            | 4.02            | 2.20            | 7.83            | 1.33             |
| H               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 5.99            | 5.26            | 9.44            | 1.91            | 5.46            | 1.37            | 5.28            | 1.56            | 6.58            | 1.10             |
| Max             | 7.61            | 9.39            | 13.42           | 2.51            | 10.19           | 2.02            | 10.55           | 2.55            | 13.42           | 1.70             |
| Efficiency loss = 10% |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| L               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 4.37            | 4.50            | 4.37            | 2.70            | 2.65            | 2.00            | 2.34            | 0.60            | 1.10            | 0.56             |
| Max             | 5.51            | 5.42            | 5.51            | 3.28            | 3.16            | 2.83            | 3.03            | 0.99            | 1.63            | 0.98             |
| M               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 4.72            | 5.13            | 4.72            | 2.40            | 3.32            | 1.76            | 2.97            | 0.75            | 3.29            | 0.75             |
| Max             | 5.52            | 7.52            | 5.52            | 2.69            | 5.13            | 2.34            | 4.56            | 1.30            | 8.28            | 1.30             |
| H               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 5.14            | 5.93            | 9.34            | 2.34            | 6.19            | 1.82            | 6.08            | 1.20            | 7.17            | 1.14             |
| Max             | 6.21            | 10.92           | 15.00           | 2.63            | 11.24           | 2.46            | 11.91           | 1.78            | 15.00           | 1.76             |
| Efficiency loss = 20% |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| L               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 3.75            | 6.58            | 3.75            | 4.78            | 4.69            | 3.98            | 4.43            | 0.82            | 1.83            | 0.82             |
| Max             | 4.44            | 8.53            | 4.44            | 6.52            | 6.32            | 5.73            | 6.12            | 1.39            | 2.83            | 1.39             |
| M               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 4.14            | 6.71            | 4.14            | 3.82            | 4.73            | 3.15            | 4.49            | 0.83            | 4.23            | 0.83             |
| Max             | 4.72            | 10.21           | 4.72            | 5.15            | 5.98            | 4.60            | 5.42            | 1.18            | 9.04            | 1.18             |
| H               |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 4.67            | 6.84            | 9.56            | 3.04            | 7.17            | 2.53            | 7.10            | 1.15            | 8.02            | 1.15             |
| Max             | 5.40            | 12.74           | 16.71           | 3.49            | 12.41           | 3.50            | 13.30           | 1.71            | 16.71           | 1.71             |
| Overall         |                 |                 |                 |                 |                 |                 |                 |                 |                 |                  |
| Ave.            | 4.89            | 5.30            | 6.28            | 2.59            | 4.17            | 1.97            | 3.90            | 0.98            | 3.98            | 0.82             |
| Max             | 7.61            | 12.74           | 16.71           | 6.52            | 12.41           | 5.73            | 13.30           | 2.55            | 16.71           | 1.76             |
| #               | 0               | 0               | 0               | 0               | 3               | 9               | 4               | 23              | 5               |                  |
if the right policy is not used. On the other hand, although the performance of 2SC also depends on the environment and the worker coordination policy, we do not observe nearly the variation we do in CP. This leads us to:

**Conclusion 2.** 2SC is a more robust skill-pattern strategy than CP with respect to worker coordination policies.

Finally, we note that 2SC consistently produces good results when implemented with the UB policy for almost all types of environments. The UB policy assigns the worker to the upstream (downstream) station if the buffer in between the two stations is less than (more than) WIP/N. So, the UB policy drives the buffer levels toward WIP/N, which allows a job to be available for each worker most of the time, provided that the line has at least the critical WIP level. No policy is as consistently effective for CP as the UB policy is for 2SC.

### 4. Analysis of Chaining

In the previous section, we investigated how the heuristics performed under the two skill-pattern strategies. Assuming that each skill-pattern strategy is operated under the best heuristic worker coordination policy, we now examine their performance for different production lines. Because by definition both CP and 2SC balance the line, the value of each skill pattern rests on how effective it is at variability buffering. We begin with a comparison of CP and 2SC. Figures 3 and 4 show the trade-off between WIP level and throughput (TH) for CP and 2SC when there is no efficiency loss and 50% efficiency loss, respectively. For readability, we use the following shorthand notation for each point in our experimental design. L, M, and H refer to low, medium, and high, respectively; and the first three letters of the label refer to capacity imbalance, cv of the bottleneck station(s), and cv of the nonbottleneck stations, respectively. So, for example, MLH refers to a system with medium capacity imbalance (bottleneck mean work content 50% greater than the average), low cv at the bottleneck station (cv = 0.2), and high cv at other stations (cv = 3).

In both cases, as variability increases, 2SC increasingly outperforms CP. However, when there is sufficient efficiency loss on the secondary tasks, at low variability and high capacity imbalance CP can perform better than 2SC, but only at high WIP levels. Figure 4 shows that 2SC outperforms CP in case HLL for WIP up to two times the critical WIP level, but at higher WIP levels CP performs better. We can conclude that if there is high variability in the system and/or we want to run the system lean (i.e., with low WIP), 2SC seems to be a better strategy than CP. However, it also requires additional cross-training.

This raises the question of whether the additional cross-training required in chaining is worthwhile. We address this question in two steps by: (1) comparing systems with equal numbers of skills, with one system resembling the chained structure much more closely than the other, and (2) investigating whether there is a loss in the system’s performance if we omit the skills given to the bottleneck workers in the 2SC strategy.

#### 4.1. Comparison to Cherry Picking

To highlight the dynamics underlying the performance of chaining, let us first consider a four-station line with a single bottleneck at Station 1. The CP solution is to cross-train all nonbottleneck Workers 2, 3, and 4 to help at the bottleneck, resulting in three additional skills in the system. Now suppose that instead we were to use three additional skills but in a different manner, so as to “partially chain” the stations. There are two ways to chain the stations with links...
chaining is usually superior to CP. The improvement which are presented in Table 4, show that partial ence between partial chaining and CP. The results, put optimally, and computed the percentage differ-

For the line in Figure 5, we compared CP and upstream partial chaining at different capacity imbal-

<table>
<thead>
<tr>
<th>Stations</th>
<th>WIP</th>
<th>Eff. Loss = 0</th>
<th>Eff. Loss = 10%</th>
<th>Eff. Loss = 20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>8</td>
<td>10.30</td>
<td>9.32</td>
<td>8.35</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>7.67</td>
<td>6.94</td>
<td>6.25</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>5.67</td>
<td>5.01</td>
<td>4.40</td>
</tr>
<tr>
<td>M</td>
<td>8</td>
<td>9.98</td>
<td>9.00</td>
<td>8.09</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>6.45</td>
<td>5.47</td>
<td>4.63</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>3.82</td>
<td>2.75</td>
<td>1.79</td>
</tr>
<tr>
<td>H</td>
<td>8</td>
<td>9.26</td>
<td>8.33</td>
<td>7.49</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>4.92</td>
<td>3.69</td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2.23</td>
<td>0.74</td>
<td>-0.55</td>
</tr>
</tbody>
</table>

from partial chaining is most pronounced in systems with low capacity imbalance, low WIP levels, and high worker efficiency at secondary stations. In such systems, chaining makes spare capacity more flexible, because excess from almost any nonbottleneck station can be shifted to any other station, including the bottleneck. When worker efficiency on secondary tasks decreases, so does the capacity a worker can allocate to the secondary tasks. When there is a substantial decrease in efficiency on the secondary tasks, shifting capacity indirectly to help the bottleneck station may not be appropriate because the gain in flexibility is offset by the loss in efficiency, which erodes the capacity of each worker.

We also compared CP with partial chaining for six-station lines by using simulation to investigate the effect of different variability levels (see §6 of the online appendix for numerical results). For both skill patterns, the best configuration (upstream or downstream) and heuristic were chosen to determine the throughput at different WIP levels. We observed that the benefit of partial chaining increases as the system becomes more variable. These observations lead us to:

**Conclusion 3.** Indirect cross-training via partial chaining uses the same number of skills as CP (within feasibility limits) and usually outperforms the direct capacity-balancing approach of CP, particularly when variability is high and the WIP level is low.

This result provides the powerful insight that even when the same number of skills are used, it is better to chain them in an indirect path to the bottleneck than to use them to train workers to help directly at the bottleneck.

### 4.2. Value of Completing the Chain

In the partially chained systems above, we omitted cross-training the bottleneck worker(s) to help at other stations. One might speculate that this would have little impact on performance because the partially chained line is already balanced. To find out if this is true, we first illustrate the effect of cross-training bottleneck workers to complete the chain in Figures 6 and 7, which depict results for lines with mean process times given by the vector \( T \) above the graph. We start with a line with all specialists and add skills successively to form a complete chain of skills. Figure 6 illustrates this process for a slightly imbalanced six-station line (each station with a \( cv = 1 \)), with the first station
being the bottleneck. We add skills towards upstream in a sequence that achieves the maximum line capacity at each step. We first cross-train Worker 2 for Station 1, then Worker 3 for Station 2, and proceed in this manner until we cross-train Worker 6 for Station 5, which balances the line (i.e., attains the maximum capacity of one) with five additional skills. Then, we complete the chain by cross-training bottleneck Worker 1 for Station 6. Figure 7 presents a similar analysis for a line with two bottlenecks, where the bottleneck stations together possess 50% of the raw processing time, \(cv = 0.2\) at bottleneck stations and \(cv = 3\) at nonbottleneck stations. Similarly, starting with specialists, we first cross-train Workers 4 and 1 for bottleneck stations.

Table 5 presents the impact of completing the chain for balanced lines (i.e., the mean work contents of all stations are equal) with varying line lengths. Assuming that all workers except Worker 1 are cross-trained based on upstream partial chaining, for each line we complete the chain by cross-training Worker 1 for the last station. For three and four stations, the throughputs for the upstream partial chain and 2SC are computed optimally, using the MDP. For six stations, we picked the best heuristic for each environment. From Table 5, one can observe that the benefit of completing the chain is largest at low WIP levels and in short lines. Intuitively, this benefit decreases as the efficiency on the secondary tasks increases.

Overall, our experimental analyses show that completing the chain has a pronounced effect for almost all systems except for those with a really severe bottleneck or almost no variability. (See §7 of the online appendix for the throughput increase obtained by completing the chain for six-station lines at different variability and WIP levels.) For high variability levels (i.e., \(cv = 3\)), the benefit of completing the chain can be more than 20%. Moreover, completing the chain by cross-training a bottleneck worker can result in substantial improvements in lines with a low-variability bottleneck and high-variability nonbottlenecks. The

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Value of Completing the Chain for (cv = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIP</td>
<td>Three station</td>
</tr>
<tr>
<td>No efficiency loss</td>
<td></td>
</tr>
<tr>
<td>2N</td>
<td>16.83</td>
</tr>
<tr>
<td>3N</td>
<td>10.89</td>
</tr>
<tr>
<td>4N</td>
<td>7.77</td>
</tr>
<tr>
<td>Efficiency loss = 10%</td>
<td></td>
</tr>
<tr>
<td>2N</td>
<td>13.40</td>
</tr>
<tr>
<td>3N</td>
<td>8.89</td>
</tr>
<tr>
<td>4N</td>
<td>6.51</td>
</tr>
<tr>
<td>Efficiency loss = 20%</td>
<td></td>
</tr>
<tr>
<td>2N</td>
<td>10.52</td>
</tr>
<tr>
<td>3N</td>
<td>7.21</td>
</tr>
<tr>
<td>4N</td>
<td>5.42</td>
</tr>
</tbody>
</table>
reason is that a low-variability station has more reliable capacity to give to the next station, and because all stations are indirectly connected by the chained workers, this effect benefits the entire line. Figure 8 summarizes the influence of imbalance, bottleneck variability, and nonbottleneck variability on the benefit of completing the chain. This leads us to:

**Conclusion 4.** Completing the chain by cross-training the bottleneck worker(s) can produce substantial increases in throughput, particularly when the line is short and nearly balanced, variability of nonbottleneck stations is high, and variability of the bottleneck is low.

In serial lines, completing the chain allows capacity from any station to be shifted (indirectly) to any other station. So, even shifting capacity from, or through, the bottleneck can be very valuable.

## 5. Hybrid Approaches

We have shown 2SC to have good capacity balancing and excellent variability-buffering capability. However, the CP approach can provide a lower-cost way to balance capacity. In this section, we illustrate the practical value of combining these approaches in cases of extreme line imbalance. For systems where 2SC cannot balance the line and variability is significant, we could use $D \geq 3$. Because this would require many additional skills, we propose using a hybrid approach: 2SC plus CP. That is, all stations are first chained based on 2SC. Then, we cross-train the workers who still have extra capacity with the skills for stations that still need help. In Figures 9 and 10, we show TH versus WIP under different cross-training strategies for two highly imbalanced systems. For the line in Figure 9, $D = 4$ balances the line with a total of 18 additional skills and performs almost as well as full cross-training (FXT). However, 2SC + CP only requires eight additional skills, achieves high throughput with a reasonable amount of WIP, and is much better than CP or 2SC. Figure 10 presents a very unrealistic situation where 91.66% of the total work content is at one station (Station 1). To balance this line by using DSC, FXT ($D = 6$) is necessary, resulting in 30 additional skills. On the other hand, 2SC + CP requires only 10 additional skills. As Figure 10 shows, 2SC + CP is much more effective than CP or 2SC, which have five and six additional skills, respectively.

Based on our investigations, we suggest the following guidelines for setting a cross-training strategy: (1) When both imbalance and variability are low, then no cross-training is necessary for good performance (although there may be many reasons to cross-train that are not based on logistical considerations). (2) When capacity is imbalanced, but variability is low, then CP can be used effectively with careful consideration of the worker coordination policy. When capacity imbalance is modest, but variability is significant, then 2SC with a queue-length-based policy should be effective. (3) Finally, when both factors are significant, hybrid strategies that combine both 2SC and CP may be the best approach.
6. Case Study: Ruud Lighting

Although our research suggests that skill chaining may be an effective cross-training strategy for manufacturing and service systems, industrial examples are not yet common. One example of a company that is already using partial chaining is Ruud Lighting, Inc. of Kenosha, Wisconsin. Over the years, Ruud has made use of paced assembly lines, transfer lines, and various asynchronous (unpaced) lines. Having a very high level of product variety, their continuous improvement efforts led them to search for a very efficient and flexible system.

Ruud produces industrial/security lighting products on two parallel lines where each is configured as in Figure 11(a). The two lines are identical, nearly balanced, and consist of five workstations along a common bench. Each line handles three product families with a total of more than 1,000 different models, of which 100 models constitute the majority of demand. Production takes place in an area about 65 feet by 22 feet for a pair of parallel assembly lines spaced about 14 feet apart. The space is compact and cross-training (partial downstream chaining) is implemented to minimize worker movement (walking).

We begin by simulating the existing Ruud system with two separate lines with downstream partial chaining at a CONWIP level of 12 and \( \text{cv} = 1 \). We assume that both lines are balanced and we employ the MQ policy as an approximation of the operating policy used by the workers. The resulting system throughput is \( 1.61 \pm 0.00156 \) jobs per minute. If instead we partially chain the workers against the material flow as shown in Figure 11(b), throughput is \( 1.63 \pm 0.0022 \), which shows that the direction of partial chaining is not very important in this system.

The research of this paper suggests that completing the chain would be helpful. To do this on the individual lines, Worker 1 would need to perform Task 5 and Worker 10 would do Task 6, which is not done at present due to the walk times that would be required. However, if these straight lines were converted to U-shaped cells, as shown in Figure 11(c), walk times would not be a problem. Simulation shows that this would cause throughput to increase to \( 1.92 \pm 0.0021 \), a 17.8% improvement over the original situation. Unfortunately, Ruud has not adopted the U-shaped layout because of material-handling difficulties.

Another approach for completing the chain is to link the two lines by cross-training Worker 6 for Station 1 in Line 1, and Worker 5 for Station 10 in Line 2, as shown in Figure 11(d). Again, the distance is so short between the two lines that the zero switchover time is a reasonable assumption. When we add one skill to each line in this manner, the resulting throughput is \( 1.90 \pm 0.0017 \). Hence, without any change in layout, it is possible to increase throughput about as much as could be done using two U-shaped lines.

Although Ruud found this latter approach appealing, what they have chosen to do instead is to link the two lines by allowing additional workers to float between them. There is ample bench space and enough tooling to allow the floater to contribute. If two additional floater workers help the two ends of the lines (where Workers 5 and 6 are still specialists), our simulations predict an increase in throughput to \( 2.12 \pm 0.0025 \), 30% improvement, which is well above the 20% increase in labor capacity represented by the addition of two workers.

The Ruud case shows that chaining can be a practical strategy for implementing workforce agility in real-world settings. It also illustrates that the basic principles we have identified in this paper (e.g., the value of completing the chain) can carry over to more complex systems than the ones we have used in our analyses.

7. Conclusions

In this paper, we investigated a number of worker cross-training architectures for increasing efficiency in production lines. Our results provide the following managerial insights into the choice of skill-pattern strategies and effective worker coordination policies.

1. It can be ineffectively cross-train to balance capacity directly (i.e., via CP). Strategies that rely on 2SC to shift capacity indirectly between stations are usually much more effective, particularly in systems with high variability or low WIP.

2. 2SC performs robustly well across a variety of simple queue-length-based policies. This suggests that the coordination policy can be selected to fit worker preferences or other practical system considerations without significant performance loss, which makes 2SC an attractive skill pattern for practical applications. On the other hand, selection of a worker coordination policy under CP is much more sensitive to the characteristics of the environment, which makes it more difficult to implement.
3. The flexibility created by 2SC is so substantial that the throughput for a given WIP level does not exhibit diminishing returns to the number of skills added. Instead, 2SC frequently yields the greatest marginal benefit from the addition of the final skill required to complete the chain, even though this skill trains the bottleneck worker to help out at a nonbottleneck station.

We believe that skill-chaining strategies are valuable not only in serial lines, but also in a wide range of systems, including those with multiple products and those with a parallel or a network flow structure, such as cells with automated equipment. Further research is needed to determine the potential benefits of chaining in such environments.

An electronic companion is available at mansci.pubs.informs.org/companion.html.

Acknowledgments
The authors acknowledge Juta Pichitlamken and Amit Sachdeva at Northwestern University for their work on preliminary simulation studies related to this research. We thank Christopher Ruud and Scot Siebers of Ruud Lighting for involving us in their operational design. Finally, we thank Bill Lovejoy, two anonymous associate editors, and two anonymous referees for many helpful comments and for pushing us to make our analysis more complete and rigorous. This work was partially supported by the National Science Foundation under Grant Nos. DMI-9732868 and DMI-0098821.

References