RC22628 (W0211-050) November 6, 2002
Mathematics

IBM Research Report

Principles for Partial Resource Flexibility Decisions in Flow Shop Environments

Richard L. Daniels*,Joseph B. Mazzola, Dailun Shi*****

*Terry College of Business
University of Georgia
Athens, GA 30602-6262
U.S.A.

**McDonough School of Business
Georgetown University
Washington, DC 20057
U.S.A

***IBM T. J. Watson Research Center
1101 Kitchawan Road
Route 134/P O Box 218
Yorktown heights, NY 10598
U.S.A.

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Principles for Partial Resource Flexibility Decisions in Flow Shop Environments

Richard L. Daniels*
Joseph B. Mazzola**
Dailun Shi***

*Terry College of Business
University of Georgia
Athens, GA 30602-6262
U.S.A.

**McDonough School of Business
Georgetown University
Washington, DC 20057
U.S.A

***IBM T. J. Watson Research Center
1101 Kitchawan Road
Route 134/P O Box 218
Yorktown heights, NY 10598
U.S.A.

October 2002

Abstract

Resource flexibility refers to the ability to dynamically reallocate resource units from one stage of a production process to another in response to shifting system bottlenecks. Recent research has demonstrated that substantial improvements in operational performance can be realized in both serial and parallel production environments through the effective utilization of resource flexibility. In these contexts the resource was assumed to exhibit complete flexibility, i.e., each resource unit can be assigned to any stage of the system. This research explores the extent to which the operational benefits associated with resource flexibility can be achieved in a flow shop environment using a partially flexible resource. Focusing on labor flexibility, we propose corresponding metrics for partial flexibility and formulate a model for flow shop scheduling with partial resource flexibility (FSPRF). We also present a branch-and-bound algorithm and a heuristic for FSPRF. And on the basis of a set of computational experiments, we suggest the importance of the distribution of flexibility on system performance, and characterize important attributes for those flexibility distributions that yield superior results. The conclusions drawn from this research provide significant insight into the management of flow shops with a work force that is cross-trained to achieve partial flexibility. Moreover, we adapt the principles developed by Jordan and Graves (1995) for partially flexible manufacturing plants to the flow shop scheduling environment, and we link these principles in a novel way to recent research on self-buffering flow lines.

1. Introduction

With intensified competition, shortened product life cycles, and time-sensitive customer demand, flexibility has become more and more crucial to the success of a firm. In many business environments, the flexibility of resources has been recognized as an important competitive weapon. To achieve high returns from flexibility investments, the relationship between the flexibility embodied in the resources and the associated operational performance need to be clearly understood. Furthermore, acquiring adequate flexibility doesn't guarantee a competitive edge. The actual return from any investment in flexibility depends heavily on how the flexibility is managed.

Overall, resource flexibility decisions must be made at three different levels: strategic, tactical and operational. At the strategic level, a firm determines an appropriate level of investment in resource flexibility and the types of flexibility (e.g., technology, facility or workforce) in which to invest. Decisions at this level consider the dynamic business environment in which the firm operates, and the objective is to choose a role for flexibility that reflects the firm's strategic long-term need. At the tactical level, the optimal composition/distribution of a given investment in flexibility among types and units of resource is determined. The objective at this level is to determine the set of stages to which each unit of resource can be assigned. The operational level then considers the problem of scheduling jobs and assigning individual resource units over time so as to optimize system performance, to efficiently utilize the existing flexible resources in day-to-day operations, and to ultimately materialize the potential benefits from the flexibility investment. This research addresses issues primarily at the tactical and operational levels, i.e., determining the right amount and mix of resource flexibility to include in the system, and determining how to utilize this resource in the

daily production schedule. Based on a set of computational experiments, important insights are obtained into all three levels of decision making in resource flexibility.

While there are many types of flexible resources, labor, given the inherent flexibility of the human worker, offers perhaps the most common example of a widely available flexible resource. In addition, flexible machines, including flexible manufacturing systems, offer another means for achieving system flexibility (see, for example, Sethi and Sethi 1990). We focus exclusively on labor flexibility in flow shops. The ubiquity of the worker-staffed assembly line offers strong motivation for understanding the role of labor flexibility in this setting. Our research contributes to the increased understanding of the benefits of partially flexible labor in flow shops; in so doing, it also offers a basis for unifying recent work on partial manufacturing flexibility (e.g., Jordan and Graves 1995) and self-balancing flow lines (see, e.g., Bartholdi and Eisenstein 1996 and Zadalav, McClain, and Thomas 1996).

We study labor flexibility from a scheduling perspective, because production scheduling is one of the most important and challenging manufacturing functions that must be performed repetitively. Nevertheless, a significant gap exists between scheduling theory and the situation faced by schedulers in practice. For example, classic scheduling models (see e.g., Baker 1994, Morton and Pentico 1993) assume that processing times are fixed and known prior to scheduling. This assumption implies that all resource inputs required for an operation are available in the right quantities and mix whenever they are needed. In practice, schedulers must determine not only the timing of an operation, but also the appropriate resources (machines, workers and capital) for the associated operation. The processing time of an operation is therefore often a function of the amount and mix of resources dedicated to the operation.

When job processing times depend on the amount of resource allocated to a task, flexible resources can be used to break processing bottlenecks, thereby enhancing system effectiveness and efficiency. Recent research has established that when resources exhibit complete flexibility,

i.e., when each unit of resource can be allocated to any stage of the production process, substantial improvements in operational performance can be realized in both serial and parallel production environments (e.g., Daniels and Mazzola 1993, 1994 and Daniels, Hoopes, and Mazzola 1996, 1997).

This research extends the work on resource flexibility from complete flexibility to partial flexibility. This extension was motivated by our observation of flow lines used to fabricate and assemble telephone switching equipment, where tasks are performed sequentially at a series of work stations, and the time required to perform each task is a function of the number of workers assigned to the associated work station. In production line (flow shop) environments such as these, partial (labor) resource flexibility is a particularly important issue that is addressed by training each worker to perform a subset of the tasks occurring within the line. We explore modeling issues associated with scheduling partially flexible labor in flow shops when processing times depend on the amount of labor allocated to tasks; moreover, we investigate the impact of partial resource flexibility on system performance. We focus on obtaining insight into the relationship between the operational performance of the system and system characteristics such as the amount (or degree) of resource flexibility present within the system and also the manner in which flexibility is allocated among the work force. In so doing we propose measures for capturing the amount and mix of resource flexibility contained within the production system.

The study of scheduling with resource flexibility draws from the scheduling literature as well as from the literature on manufacturing flexibility. In addition to the articles mentioned previously, flexible resource scheduling is related to project scheduling (see, e.g., Talbot 1982, Lawrence and Morton 1993, or the survey by Ozdamar and Ulusoy 1995), scheduling with controllable processing times (see, e.g., Vickson 1980, Van Wassenhove and Baker (1982), Daniels and Sarin 1989, Trick 1994, or Alidaee and Kochenberger 1996), and flexible manufacturing (Fine and Freund 1990 or Sethi and Sethi 1990). For a discussion of scheduling

(including flow shop scheduling) see, e.g., Baker (1994), and for recent trends in the area of scheduling, see Lee, Lei, and Pinedo (1997).

In the next section we define the concept of skill matrix, which establishes workers' cross-training skills. Section 2 also presents measures for partial flexibility, addressing both the amount and the allocation of flexibility among the workforce. Based on these flexibility measures, attributes for skill matrices that yield good operational performance are discussed in this section as well. The flow shop scheduling with partial resource flexibility is formulated in Section 3, and problem complexity is also established. Section 4 presents a branch-and-bound algorithm to solve FSPRF, and Section 5 discusses heuristic algorithms for approximate solutions to FSPRF. The computational experiments summarized in Section 6 provide important insights into flow shop scheduling with partial workforce flexibility. Section 7 ends the paper with a summary and suggestions for future research.

2. Skill Matrices, Flexibility Metrics and Effective Skill Matrices

Let $W = \{1,2,\cdots,w\}$ denote the set of workers, each trained to staff a subset of work stations M={1,2,…, m}. The workers can be divided into distinct groups such that each worker in a group W_g is trained to operate on the same subset of stations, i.e., workers in W_g are identically skilled. Let G index the set of these worker groups. Thus, the sets $\{W_g, g \in G\}$ form a partition of W; that is, $W_g \subseteq W$, $W_g \bigcap W_{g'} = f$ if $g \neq g'$, and $\bigcup_{g \in G} W_g = W$. Let $w_g = |W_g|$ be the number of workers for $g \in G$. The stations that can be operated by any particular group of workers can be identified, as can the different groups of workers that are capable of staffing a specific station. In particular, for $g \in G$, define:

M_g = { $j \in M$: station j can be staffed by any worker in group g},

and define $m_g = |M_g|$ be the number of such stations. Similarly, for $j \in M$, let

G_j = { $g \in G$: workers in group W_g can be allocated to station j},

and let $g_j = |G_j|$ be the number of such groups.

The worker-station skill matrix S is a useful tool for establishing which workers can operate which station s. Specifically, the matrix S=(s_{hj}), where $h \in W$, $j \in M$, is defined as:

$$s_{hj} = \begin{cases} 1, & \text{if worker h can staff station j;} \\ 0, & \text{Otherwise.} \end{cases}$$

For example, the $m \times m$ unit matrix S = I_{mxm} represents the skills consistent with the classic flow shop, where there is no labor flexibility since each worker is permanently assigned to a fixed work station. At the other extreme, S = E_{mxm}, the $m \times m$ matrix containing all ones, captures the complete flexibility case studied by Daniels and Mazzola (1993, 1994).

The following notation for skill matrices is adopted for convenience. Given a skill matrix S=(s_{hj}), the support of the matrix S, denoted supp(S), is defined as $\{(h, j) \in W \times M \mid s_{hj} = 1\}$. For each row of the matrix S corresponding to a worker $h \in W$, let M_h^S = { $j \in M$ | s_{hj}=1} be the stations which worker h has skills to operate, and $m_h^S = |M_h^S|$ $= \sum_{j \in M} s_{hj}$ be the number of such stations. Similarly, for each column of S corresponding to station $j \in M$, define W_j^S ={ $h \in W$ |s_{hj}=1} and $w_j^S = |W_j^S|$ $= \sum_{h \in W} s_{hj}$.

Let $\bar{k}_j$ be number of possible modes in which jobs can be processed on station j. Since it would not be possible to assign more than w_j^S number of workers to station j, we assume without loss of generality that the maximum processing mode $\bar{k}_j \le w_j^S$ for any $j \in M$. In some instances, however, it is feasible and potentially advantageous to allow $\bar{k}_j < w_j^S$, since some of the workers trained to staff station j may also be trained to staff other stations, and during the

course of the dynamic allocation of workers the increased flexibility afforded by $w_j^S > \bar{k}_j$ may prove to be beneficial.

Example 1. Consider a problem with 4 stations and 4 workers. Suppose that

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}.$$

In this example worker 1 can staff stations 1, 2 and 4; worker 2 can staff stations 2 and 4, and so forth. There are 3 groups of workers: $W_1=\{1\}$, $W_2=\{2\}$, and $W_3=\{3,4\}$; hence, $w_1= w_2=1$ and $w_3=2$. Also, $M_1=\{1, 2, 4\}$, $M_2=\{2, 4\}$, $M_3=\{1, 3, 4\}$. Similarly, $G_1=\{1, 3\}$, $G_2=\{1, 2\}$, $G_3=\{3\}$, and $G_4=\{1, 2, 3\}$.

This example illustrates that the maximum processing mode on a station, and hence of any jobs on the station, depends on the skill matrix S. Since $s_{h1}=1$ for 3 workers, the fastest processing speed of any job on station 1 is achieved by assigning all 3 workers to it. On the other hand, station 4 is capable of processing jobs up to 4 modes. Regardless of the skill matrix, however, the number of possible processing modes of any job is bounded by a number $\bar{k}$, which can represent, e.g., the total number of workers or the maximum number of possible processing modes of all jobs.

2.1 Metrics for Partial Resource Flexibility

One of the objectives of this research is to understand better the ways in which system performance changes as the degree of resource flexibility increases. This requires the development of appropriate metrics for resource flexibility that capture the level (amount) of resource flexibility as well as the mix of the flexible resource with respect to stations. Define flexibility metric:

$$f^S = (\sum_{h \in W} \sum_{j \in M} s_{hj}) / wm .$$

Clearly, for any skill matrix S, f^S assumes a value on the interval [0,1], with greater values of f^S corresponding to more flexible systems. Given that entries in a skill matrix can be either 0 or 1, the possible values of f^S are discrete. A real number $f \in [0,1]$ is an attainable value if there exists a skill matrix S satisfying $f^S = f$ that yields a feasible flow shop scheduling problem.

The metric f^S measures the level of flexibility. Within each value of f^S, we can also measure the mix of flexibility by quantifying the degree to which flexibility is allocated across stations and workers. For each station $j \in M$, we define $b^S = \max_{j,j' \in M} \{\left| w_j^S - w_{j'}^S \right|\}$. If we consider the station that has the largest number of workers who are trained to staff it, and the station that has the smallest such number of workers, this metric represents the difference between these two values. The number b^S seeks to measure how evenly resource flexibility is distributed throughout the system, with smaller values of b^S corresponding to more evenly distributed flexibility. We refer to b^S as the station-balance metric or simply s-balance metric, since it concentrates primarily on the stations.

An alternative mix metric is established by defining the station-worker-balance metric or simply sw-balance metric $\hat{b}^S = \max_{j,j' \in M} \{\left| mw_j^S - mw_{j'}^S \right|\}$, where $mw_j^S = \sum_{h \in W} (s_{hj} / m_h^S)$ (recall that $m_h^S = \sum_{j \in M} s_{hj}$ is the number of stations that worker h has skills on). This measure takes into account that a worker trained on more than one station would need to effectively split his time across stations; hence, the adjustment factor m_h^S is incorporated into the metric.

For the matrix S appearing in Example 1, f^S =11/16. For each column 1 through 4 in S, the values of w_j^S are 3, 2, 2, and 4, respectively. Thus, the s-balance measure is $b^S = 4 - 2 = 2$.

The sw-balance measure for this skill matrix is $\hat{b}^S = 3/2 - 2/3 = 5/6$. Moreover, f^S=1 and $b^S = \hat{b}^S = 0$ when S=E_{mxm}; while f^S=1/m and $b^S = \hat{b}^S = 0$ for S=I_{mxm}.

2.2 Attributes of Effective Skill Matrices

Matrix S is an effective skill matrix if S results in superior system performance relative to skill matrices having the same value of f^S. In a flow shop with w workers and m stations, a skill matrix S has $w \times m$ entries (each entry being either 0 or 1). Thus, there are total of 2^{wm} possible skill matrices. For w=m=3, there are 512 skill matrices; this number increases to 65,536 when w=m=4 and reaches more than 33 million for w=m=5. This explosive growth in the number of skill matrices motivates the characterization of effective skill matrices, i.e., a set of attributes that allow the decision maker to confine attention to a manageable subset of skill matrices to identify the best distribution of flexibility. We define a set of such attributes in this section.

Without loss of generality, for each $j \in M$, we require that $W_j^S \neq \varnothing$ (or equivalently $w_j^S \geq 1$) for any skill matrix S; otherwise, the corresponding flow shop scheduling problem will be infeasible (this property also implies that at least one worker can staff every station). Similarly, we require that every worker $h \in W$ can staff at least one station (i.e., $M_h^S \neq \varnothing$); otherwise, any such workers can be eliminated from consideration. Moreover, we are concerned with the overall flexibility of a workforce without explicitly considering which workers possess which skills. Hence, if two skill matrices S and S' differ only in the order of their rows (i.e., if there exists a permutation of the rows of S such that the corresponding rearrangement of rows gives rise to S'), then the two matrices S and S' define precisely the same set of worker skills. The skill matrices S and S' are then said to be equivalent. Throughout the remainder of our discussion, we do not distinguish between equivalent skill matrices.

Both the s-balance metric b^S and the sw-balance metric $\hat{b}^S$ measure how evenly skills are distributed across stations. Intuitively, high values of b^S (or $\hat{b}^S$) should be associated with poor operational performance, since jobs are more likely to be delayed when at least one station lacks the additional labor needed to break system bottlenecks. Therefore, we define a skill matrix S to be s-balanced whenever $b^S \leq 1$; and it is said to be sw-balanced if $\hat{b}^S \leq 1$.

The computational experiments described in Section 6 indicate that for any attainable value of flexibility metric f, the skill matrix yielding the best operational performance is both s-balanced and sw-balanced, suggesting that these are two important attributes of effective skill matrices. Unfortunately, a large number of feasible skill matrices are still both s-balanced and sw-balanced. For example, 33 of the 57 feasible skill matrices are both s-balanced and sw-balanced when m=w=3; and this number increases to 578 out of 2306 feasible skill matrices when m=w=4, and over 28,000 out of over 270,000 when m=w=5. Thus, s-balance and sw-balance do not provide a sufficiently precise means for distinguishing between effective and ineffective distributions of flexibility among a workforce. This observation motivates consideration of a third group of skill matrices, which we refer to as chains. Chains form a proper subset of matrices that are both s-balanced and sw-balanced. Importantly, the number of chains is quite manageable and, as will be discussed in Section 6, chains consistently yield excellent system performance.

A chain is most easily conceptualized in flow shops in which the number of workers is equal to the number of stations, i.e., w=m. For any skill matrix S, the corresponding worker-station graph $G(S) = (V^S, E^S)$ is the bipartite graph with one set of vertices in V^S corresponding to the set W of workers and the other set of vertices corresponding to the set M of stations; thus there are 2m vertices in the graph. An arc $(h, j) \in E^S$ if and only if $s_{hj} = 1$ in S.

Jordan and Graves (1995) studied process flexibility in the context of the assignment of products to plants, and observed that limited flexibility (i.e., each plant builds only a few, but not all products) results in large benefits when the plant-product assignments follow a chain in the corresponding plant-product bipartite graph. The basic idea behind a chain is to form a continuous path in the bipartite graph and then maintain the continuity of the path while also creating cycles of maximal length (before creating subcycles). We now extend the concept of chains to flow shop scheduling with partial resource flexibility.

Let S be an $m \times m$ skill matrix. For $1 \le k \le m$, the skill matrix S is said to be a k-chain if M_h^S ={h, h+1, …, h+k-1} for all $h \in W$, where it is understood that if $j \in M_h^S$ and j>m, then j is taken to be the unique station $j' \in \{1,2,\cdots,m\}$ satisfying $j \equiv j' \,(\text{mod}\, m)$. Any skill matrix S that is equivalent to a k-chain is also said to be a k-chain. The k-chain (of dimension $m \times m$) is denoted by $C^{m,k}$. Observe that |supp($C^{m,k}$)| = km. An arbitrary skill matrix S is then said to be a chain if there exists some $k \in \{1,2,\cdots,m-1\}$ satisfying $\text{supp}(C^{m,k}) \subseteq \text{supp}(S) \subset \text{supp}(C^{m,k+1})$. As before, any skill matrix S that is equivalent to a chain is also said to be a chain. Thus, S defines a chain if either it is equivalent to some k-chain, or it can be obtained by taking a k-chain and adding some extra 1's in the positions of $\text{supp}(C^{m,k+1}) \setminus \text{supp}(C^{m,k})$. Note that $C^{m,k}$ is sometimes referred to in the literature as a circulant (see, e.g., Balas 1975).

If we define the $m \times m$ matrix $\Delta^{m,k} = (d_{r,j}^{m,k})$ by

$$d_{h,j}^{m,k} = \begin{cases} 1, & \text{if } j \equiv h + k \,(\text{mod}\, m), \\ 0, & \text{otherwise}, \end{cases}$$

for $h \in W$, $j \in M$, it then follows that $C^{m,k+1} = C^{m,k} + \Delta^{m,k}$, for k=1,…, m-1. This is illustrated in the following example.

Example 2. For m=w=5, we have that

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$C^{5,2} \quad + \quad \Delta^{5,2} \quad = \quad C^{5,3}$$

Figure 1. The Bipartite Graphs for 2-chain $C^{5,2}$ and 3-chain $C^{5,3}$

The bipartite graphs corresponding to the 2-chain $C^{5,2}$ and the 3-chain $C^{5,3}$ are shown in Figure 1. Observe that in the 2-chain for m=w=5, worker 1 is trained to staff stations 1 and 2; worker 2 is trained on stations 2 and 3; and so forth, with worker 5 trained on stations 5 and 1. Also, as can be seen in the graph for $C^{5,2}$, the 2-chain forms a continuous path and the corresponding cycle is of maximal length in that the chain covers all of the workers and stations without forming subcycles. These properties as they correspond to the 3-chain are also observed in the graph for $C^{5,3}$. The worker skill patterns defined by chains lend themselves readily to the flow shop scheduling environment in that they involve training workers to operate a set of stations that occur consecutively in the process. This is an important observation in that the resulting worker skill patterns can be seen to relate directly to other recent research findings on worker flexibility in flow shops, as we discuss in Section 7.

The following result will prove helpful.

Proposition 1. If a $m \times m$ skill matrix S defines a chain, then its s-balance and sw-balance measures satisfy

$$b^S \leq 1;$$

$$\hat{b}^S \leq \frac{2}{k^* + 1},$$

where $k^* \in \{1,2,\cdots,m-1\}$ is the maximum number satisfying $\text{supp}(C^{m,k^*}) \subseteq \text{supp}(S)$.

All proofs are contained in the Appendix. Observe that the right hand side of the inequality for $\hat{b}^S$ in Proposition 1 is not greater than 1; hence chains are both s-balanced and sw-balanced. The relationship among s-balanced skill matrices, sw-balanced skill matrices and chains is further illustrated by the following example.

Example 3. Consider the skill matrices

$$S_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},\ S_2 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},\ S_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix},\ S_4 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

By definition, $b^{S_1} = 2$, $\hat{b}^{S_1} = 3/2$, $b^{S_2} = 1$, $\hat{b}^{S_2} = 3/2$, $b^{S_3} = 2$, $\hat{b}^{S_3} = 1$ and $b^{S_4} = \hat{b}^{S_4} = 1$. Hence, S_1 is neither s-balanced nor sw-balanced, S_2 is s-balanced but not sw-balanced, S_3 is sw-balanced but not s-balanced, while S_4 is both s-balanced and sw-balanced but not a chain.

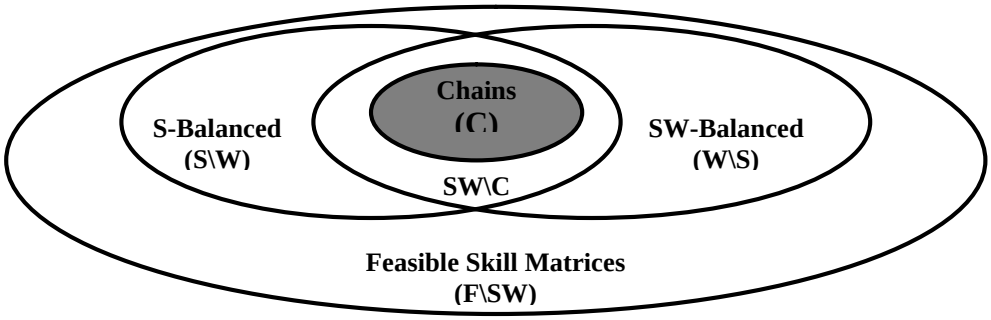

Figure 2. The Relationships among Different Kinds of Skill Matrices

Because of these relationships, the set of feasible skill matrices can be partitioned into five non-overlapping subsets as depicted in Figure 2. The first subset is the set of matrices that

are feasible but neither s-balanced nor sw-balanced. The second subset includes s-balanced matrices that are not sw-balanced, while the third subset contains sw-balanced matrices that are not s-balanced. The fourth is the subset of matrices that are both s-balanced and sw-balanced but not chains, and the last subset is the set of chains. As in the Figure 2, these five subsets of matrices are denoted, respectively, as F\SW, S\W, W\S, SW\C and C.

Given the values of m and w, let S(m,w) be the number of skill matrices (recall that equivalent skill matrices are treated as identical); and let C(m,m) be the number of chains when m=w. The following result gives the number of skill matrices and the number of chains.

Theorem 1. In a flow shop with m stations and w workers:

$$\text{S(m,w)} = \binom{w + 2^m - 2}{2^m - 2}$$

$$\text{C(m,m)} = (m-1)2^m - (m-2)$$

Table 1 presents the number of different kinds of skill matrices for various values of m and w. Information is also provided concerning the size of the other non-overlapping subsets depicted in Figure 2. These numbers and Theorem 1 highlight the explosive growth in the number of possible ways to distribute resource flexibility as problem size increases, and proved useful in guiding the design of computational experiments discussed in Section 6.

3. Problem Formulation

We are given a set of jobs (indexed by) N = {1, 2,…, n} which are to be processed sequentially on a set of stations, M = {1, 2, … , m}, which constitute a flow shop. The jobs are to be processed in the same order on each of the stations, and we refer to the processing of job $i \in N$ on station $j \in M$ as (job) operation (i, j). Each operation can be performed in any one of a set $K_{ij} = \{1, \cdots, \bar{k}_{ij}\}$ of possible processing modes. For each $k \in K_{ij}$, operation (i,j) requires $\hat{r}_{ijk}$

units of renewable resource (e.g., labor) and has a processing time of $\hat{p}_{ijk}$ time units. We assume without loss of generality that the sets K_{ij} are ordered so that $\hat{p}_{ijk_1} \le \hat{p}_{ijk_2}$ whenever $k_1 \ge k_2$. Let C_{ij} denote the completion time of operation (i,j). Consistent with the measure of system performance for flow shop problems typically encountered in the literature, the objective is to minimize the total makespan C_{max} of all the jobs. At any point in time there is a fixed (and known) amount of the resource available, and the resource may be specific to a particular subset of stations. We assume that each operation must be processed without preemption and that once a specific set of workers has been assigned to an operation, this assignment remains fixed for the duration of it processing.

To formulate the flow shop scheduling problem with partial resource flexibility with respect to the skill matrix S, which we refer to as FSPRF(S), define the decision variables:

$$Y_{ii'} = \begin{cases} 1, & \textit{if job } i \textit{ precedes job } i' \\ 0, & \textit{otherwise} \end{cases}$$

$$X_{ijkt} = \begin{cases} 1, & \textit{if job } i \textit{ completes} \text{ processing } \textit{in station } j \textit{ in} \text{ mode } k \textit{ at time } t \\ 0, & \textit{otherwise} \end{cases}$$

$$a_{hjt} = \begin{cases} 1, & \textit{if} \text{ worker } h \textit{ is assigned to station } j \textit{ during } [t-1, t) \\ 0, & \textit{otherwise} \end{cases}$$

The number of workers, stations, and jobs, are specified, along with the data on processing times and resource requirements (all of which are assumed to be integer-valued). Let $\overline{C}(S)$ be the optimal makespan of FSPRF with respect to a feasible skill matrix S = (s_{hj}). FSPRF(S) can then be formulated as the following 0-1 mixed integer programming problem (Let $\hat{T} = \{1, 2, ..., \hat{t}\}$, where $\hat{t}$ is a known upper bound on the optimal makespan, and let $\hat{M}$ be an arbitrarily large positive integer).

$\overline{C}(S)$ = *Minimize* C_{max}(S)

s.t. $C_{\max}(S) \geq C_{im}, \quad i \in N$ (1)

$$p_{ij} = \sum_{k \in K_{ij}} \hat{p}_{ijk} \sum_{t \in \hat{T}} X_{ijkt}, \quad i \in N, j \in M \tag{2}$$

$$C_{ij} - C_{i'j} + \hat{M}(1 - Y_{i'i}) \geq p_{ij}, \quad i \in N, i' \in N \setminus \{i\}, j \in M \tag{3}$$

$$Y_{ii'} + Y_{i'i} = 1, \quad i \in N, i' \in N \setminus \{i\} \tag{4}$$

$$C_{ij} \geq C_{i,j-1} + p_{ij}, \quad i \in N, \ j \in M \tag{5}$$

$$C_{ij} = \sum_{t \in \hat{T}} t \sum_{k \in K_{ij}} X_{ijkt}, \quad i \in N, j \in M \tag{6}$$

$$\sum_{t \in \hat{T}} \sum_{k \in K_{ij}} X_{ijkt} = 1, \quad i \in N, j \in M \tag{7}$$

$$a_{hjt} \leq s_{hj}, \quad h \in W, j \in M, t \in \hat{T} \tag{8}$$

$$\sum_{j \in M} a_{hjt} \leq 1, \quad h \in W, t \in \hat{T} \tag{9}$$

$$\sum_{k \in K_{ij}} \sum_{t \in \hat{T}} X_{ijkt} [a_{hjt} s_{hj} - \hat{r}_{ijk}] = 0, \quad i \in N, j \in M \tag{10}$$

$$\sum_{k \in K_{ij}} \sum_{t=\hat{p}_{ijk}}^{t} X_{ijkt} [\sum_{l=t-p_{ij}+1}^{t} a_{hjl} - \hat{p}_{ijk}] = 0, \quad i \in N, j \in M, h \in W, t \in \hat{T} \tag{11}$$

$X_{ijkt}, Y_{ii'}, a_{hjt} \in \{0,1\}, \ i \in N, i' \in N \setminus \{i\}, \ j \in M, k \in K_{ij}, h \in W, t \in \hat{T}$ (12)

In this formulation the objective function minimizes schedule makespan as defined by constraints (1). Constraints (2) determine the unique processing time of each operation. Constraints (3-5) determine the job sequence and also enforce precedence requirements (define $C_{i,0} = 0, i \in N$). Constraints (6) define the completion time of each operation, and constraints (7) ensure that exactly one processing mode and one completion time is selected for each operation.

Constraints (8) ensure that workers are assigned only to work stations for which they have the requisite skills, and Constraints (9) specify that at each point in time, a worker can be assigned to at most one station. Constrains (10) require that if operation (i,j) is completed at time t in mode k, then exactly $\hat{r}_{ijk}$ appropriately-skilled workers are assigned to it during the time interval [t-1,t). Constrains (11) then require that if any worker $h \in W$ is assigned to an operation during the final time interval of its processing, then that worker must also have been assigned to the operation for the previous $\hat{p}_{ijk} - 1$ time intervals. (We note that this formulation allows for solutions that could possibly contain an extraneous "assignment" of an idle worker to a job operation in a period before the job operation is even begun (provided, of course, that the assignment does not result in a suboptimal solution); rather than imposing additional constraints to prevent such assignments, it is sufficient to simply ignore them when interpreting the final solution.)

When w=m, FSPRF contains as special cases both the classic flow shop scheduling problem (i.e., $S=I_m$) and the (complete) flexible resource flow shop scheduling problem introduced in Daniels and Mazzola (1994) (i.e., $S = E_m$). Since the flexible-resource flow shop scheduling problem is NP-hard in the strong sense for m = 2 (Daniels and Mazzola 1994), the same computational complexity holds for FSPRF. The foregoing remarks concerning the complexity of FSPRF assume that the skill matrix S is not fixed as an input parameter. For a specified (feasible) skill matrix S, the problem FSPRF(S) is also easily seen to generalize the classical flow shop scheduling problem; hence, it too is NP-hard in the strong sense (see, e.g., Garey and Johnson 1976 and 1979).

Given a pair of skill matrices S_1 and S_2, we say that S_1 is a *restricted case* of S_2, denoted as $S_1 \prec S_2$, if $\text{supp}(S_1) \subset \text{supp}(S_2)$. Whenever $S_1 \prec S_2$, any feasible solution to FSPRF(S_2) also defines a feasible solution to FSPRF(S_1), hence we immediately have the following.

Proposition 2. If $S_1 \prec S_2$, then $\overline{C}(S_2) \leq \overline{C}(S_1)$.

Let $\overline{C}(\hat{f}) = \min_{f^S = \hat{f}} \overline{C}(S)$ denote the minimum makespan of all skill matrices of a specified value $\hat{f}$ of the flexibility metric. It follows as an immediate corollary to this proposition that $\overline{C}(\hat{f}_1) \leq \overline{C}(\hat{f}_2), \quad if \ \hat{f}_2 \leq \hat{f}_1$.

4. A Branch-and-bound Algorithm for FSPRF

In this section we define a branch-and-bound algorithm for obtaining optimal solutions to FSPRF. Because of the inherently complex nature of the problem, the utility of the algorithm is not primarily as a tool for solving instances of FSPRF arising in practice. Rather, its principal objective is to provide essential insight into scheduling with partial resource flexibility, allowing, e.g., examination of the direct benefits that accrue as the amount of resource flexibility increases. It also provides a basis for characterizing the structure of effective skill matrices and for identifying properties of optimal schedules. In Section 6 we discuss important insights obtained by applying the algorithm to a set of test problems and examining the resulting optimal solutions. From information obtained in these experiments, we define a heuristic for FSPRF which facilitates further exploration of these principles and furthers our understanding of flow shop scheduling with partial labor flexibility.

The FSPRF branch-and-bound algorithm solves FSPRF(S), for each skill matrix S, by utilizing a two-level branching procedure. On the first level, the process branches through all possible sequences of the n jobs. Given a skill matrix S and a specific sequencing of jobs, the second level systematically constructs all possible feasible schedules of operations on each station in conjunction with associated assignments of workers to operations. The construction of these feasible schedules involves the generation of nodes in the search tree corresponding to partial schedules. Specifically, for a given job sequence, the branching procedure begins at time t = 0, and after scheduling the initial operations, it generates a new set of partial schedules each

time a currently scheduled operation completes processing. For our purposes we adopt the convention that if an operation has a processing time of p and begins processing at time t, then the station and workers assigned to it are occupied during the (half-closed) interval [t,t+p).

4.1. Branching

To describe the branching procedure in detail, we establish some terms and notation. Let Θ denote the set of all operations (i,j), and assume a job sequence $p = \{ i_1, i_2, \ldots, i_n\}$ is given. At each point in time t occurring either at the beginning of the scheduling horizon or when a currently scheduled operation completes processing, a partial schedule is constructed, evaluated and (possibly) stored as a node in the search tree for further completion. At any point t, let Ω_t represent the set of operations that have been scheduled (in the partial schedule) to begin strictly prior to time t. Each operation $(i, j) \in \Omega_t$ has an associated (scheduled) operating mode, processing time, and resource amount, denoted by k_{ij}, $p_{ij} = \hat{p}_{ijk_{ij}}$, and $r_{ij} = \hat{r}_{ijk_{ij}}$, respectively; in addition, the specific set of workers assigned to each operation in Ω_t is known. Throughout the remainder of our discussion we assume for each operation $(i, j) \in \Theta$ that $r_{ijk} = k$, that the total amount of resource R is equal to the number of workers w, and that workers have partial flexibility as specified by the skill matrix $S = (s_{hj})$.

Let W_t^F represent the set of workers who are free (i.e., unassigned to a station) at time t; this set will include those workers who were free at time t-1, together with those workers who complete the processing of an operation at time t. Let E_t be the set of operations that are eligible to begin at time t, taking into account the specified sequence of jobs, operation precedence constraints, and the availability of a worker to perform the operation (i.e., $(i_q, j) \in E_t$ if for $q \geq 2$, (i_{q-1},j) has been completed; for $1 \leq q \leq n$ and $j \geq 2$, its predecessor operation $(i_q,j-1)$ has been completed; and there exists a worker $h \in W_t^F$ with $s_{hj} = 1$).

Recall that the set W of workers is partitioned into groups $W_g \subseteq W$, $g \in G$. Define $W_g^t = W_g \bigcap W_t^F$, and $w_g^t = \left|W_g^t\right|$; thus, the set W_g^t is the set of workers from group $g \in G$ who are free at time t, and w_g^t is the number of such workers. Define M_t^F to be the (sub)set of stations M that are free (or become free) at time t and for which there exists an operation $(i, j) \in E_t$. Recalling the earlier definitions of $M_g, g \in G$ and $G_j, j \in M$, for each $g \in G$, we define the set $M_g^t = M_t^F \bigcap M_g$, which is the set of eligible stations at time t to which any worker in group g can be assigned, and for each $j \in M$, the set $G_j^t = \{g \in G_j : W_g^t \neq \varnothing\}$, which is the set of groups at time t that are capable of providing at least one worker to be assigned to station j.

At a branching point t, the branching procedure begins with the partial schedule contained in an active node of the search tree (with the partial schedule corresponding to $\Omega_0 = \varnothing$ occurring at t = 0) and then generates new nodes of the search tree corresponding to all possible subsets of jobs in E_t that satisfy the operation processing precedence constraints and that can be run simultaneously on the stations that are idle at time t; in addition, for each of these operations, all corresponding processing modes that are possible using the number of workers in each group that are available at time t must also be taken into account. While it is essential to consider all possible combinations of the applicable groups of workers and to keep track of the number of workers in each of the groups assigned to an operation, because workers within each group are considered to be identically skilled, it is not necessary to enumerate all possible individual assignments of workers to operations.

The set of branches generated from a node of the search tree must also include branches in which some or perhaps all of the labor resource from each of the relevant groups is strategically withheld. Because workers must complete their tasks before moving on to new

operations, it is sometimes optimal at a branching point t to deliberately withhold labor that could otherwise be assigned to an operation in E_t.

We now describe an approach for generating all of the possible branches at time t. First, identify M_t^F. From this (already knowing W_t^F), identify G_j^t, for each $j \in M_t^F$. All possible subsets of M_t^F (including the empty set) must then be considered as candidates for stations that become active at time t, and for each of these subsets, all possible (and feasible) combinations of processing modes for the operations on these stations must also be considered; moreover, for each processing mode on each station, it is also necessary to consider all combinations of workers from each of the groups that are able to supply labor for this operation. In regard to this last point, we again note that it is necessary only to consider the number of workers from each group assigned to an operation and not all possible specific assignments of individual workers to operations. Once the number of workers to be assigned from a group is specified, the particular workers can be selected arbitrarily.

4.2. Dominance Relations and Bounds

The branching procedure described in the previous section generates a set of new nodes, corresponding to time t, to be included in the search tree. A large number of potential branches can arise as the many possible combinations of operations and worker assignments are taken into account. We now discuss dominance relations and bounds, which are used to fathom nodes (i.e., eliminated from further consideration in the search tree without interfering the ability of the algorithm to identify an optimal solution) to expedite the searching for the optimal solution.

Suppose we are considering a node that is generated at time t using the branching procedure. Recall that this node corresponds to a partial schedule that is constructed for a fixed sequence of jobs p, and also recall the definitions of Ω_t, E_t, and W_t^F from the earlier discussion. Define the (sub)set of workers $W_t^B = W \setminus W_t^F$, which is the set of workers who are

busy processing operations at time t. Each worker $h \in W_t^B$ is assigned to a unique operation $(i_h, j_h) \in \Omega_t$; let $\hat{m}_t(h)$ be the station, j_h, to which worker h is assigned. Paralleling the definitions of W_t^F and W_t^B, we define M_t^B to be the (sub)set of stations $j \in M$ that are busy at time t, where $M_t^B = M \setminus M_t^F = \bigcup_{h \in W_t^B} \{\hat{m}_t(h)\}$.

At time t each node generated by the branching procedure involves the selection of subsets $\tilde{M}_t \subseteq M_t^F$, $\tilde{E}_t \subseteq E_t$, and $\tilde{W}_t \subseteq W_t^F$, where $\tilde{M}_t$ is the set of stations in M_t^F on which a new operation is begun at time t, $\tilde{E}_t$ is the set of operations in E_t that are started at time t, and $\tilde{W}_t$ is the set of workers in W_t^F who are assigned to work on operations in $\tilde{E}_t$. We can then define $\hat{W}_t^B = W_t^B \bigcup \tilde{W}_t$ and $\hat{M}_t^B = M_t^B \bigcup \tilde{M}_t$ to be the sets of workers and stations, respectively, that are busy immediately after time t (when the new operations in $\tilde{E}_t$ have been scheduled). For each operation $(i, j) \in \tilde{E}_t$, let $\tilde{k}_{ij}$ be its newly scheduled processing mode.

Again, assume that a skill matrix S is specified. For a particular node, suppose there exists a worker $h \in W_t^F \setminus \tilde{W}_t$ who can be assigned to an operation in $(i, j) \in \tilde{E}_t$ and thus allow its processing mode to be increased or that there exists an operation $(i', j') \in E_t \setminus \tilde{E}_t$ with $j' \in M_t^F \setminus \tilde{M}_t$ and $s_{hj'=1}$ so that worker h can be assigned to begin this operation on station j'. As discussed earlier it is necessary sometimes to strategically withhold resources; i.e., to reserve (labor) resources so that they are available for use at subsequent branching points. We therefore seek conditions under which the deliberate withholding of workers is not necessary, and thus the node defined by the corresponding partial solution can be fathomed.

The two theorems that follow provide different sets of conditions for fathoming (a node corresponding to) a partial solution. First, for any worker $h \in W$, define

$$t_h^{'} = \min\{C_{ij} : (i,j) \in \Omega_t \bigcup \widetilde{E}_t, C_{ij} > t, \text{and there exists j'} \in \mathrm{M_t^F} \textit{ with } s_{hj} = s_{hj'} = 1\}$$

Observe that $t_h^{'}$ is the earliest time after time t when worker h could possibly be assigned to a new operation. For an operation $(i,j) \in \Omega_t \bigcup \widetilde{E}_t$, by allowing for $j' \in M_t^F$ to either equal j or not, this definition takes into account stations on which no operation is assigned at time t (i.e., that are deliberately left idle) but for which a new worker will become available when one of the current operations completes processing, and at that time, this new worker could possibly work with worker h on a new operation on that station.

Theorem 2. For a partial schedule s, at time t if there exist: (i) an operation $(i',j') \in E_t$ with $k_{i'j'} < \bar{k}_{i'j'}$ (where $k_{u'j'} = 0$ if $(i',j') \in E_t \setminus \widetilde{E}_t$), (ii) a subset of workers $H \subseteq W_t^F \setminus \widetilde{W}_t$ with $s_{hj'} = 1$, for all $h \in H$, and (iii) a processing mode k' satisfying $k' = k_{i'j'} + |H| \leq \bar{k}_{i'j'}$ and $t + \hat{p}_{i'j'k'} < \min\{\min_{h \in H}\{t_h^{'}\}, t_{i'j'}^{*}\}$, where

$$t_{i'j'}^{*} = \begin{cases} \min\{C_{ij} : (i,j) \in \Omega_t \bigcup \widetilde{E}_t,\ j \in \hat{M}_t^B, \textit{ and there exists} \\ \qquad \hat{h} \in \hat{W}_t^B \textit{ with } s_{\hat{h}j'} = 1 \textit{ and } \hat{m}_t(\hat{h}) = j'\} + \hat{p}_{i'j'\bar{k}_{i'j'}}, \textit{ if } (i',j') \in E_t \setminus \widetilde{E}_t, \\ +\infty, \qquad\qquad \textit{Otherwise,} \end{cases}$$

then the node corresponding to s can be fathomed.

Observe that the definition of $t_{i'j'}^{*}$ simply means that if operation (i',j') is eligible to start at time t but is not included in $\widetilde{E}_t$, then $t_{i'j'}^{*}$ assumes a value that is equal to the earliest completion time of any operation in $\Omega_t \bigcup \widetilde{E}_t$ to which there is a worker h currently assigned who could also work on operation (i',j'). We assume that min taken over the empty set is equal to $+\infty$ and that the sum of $+\infty$ and any real number is equal to $+\infty$.

Theorem 2 extends a similar dominance result from Daniels and Mazzola (1994) to accommodate partial resource flexibility. In identifying sets $H \subseteq \hat{W}_t^F \setminus \widetilde{W}_t$ and associated processing modes k' satisfying the conditions of the Theorem, it is important to note that when

searching for a value of k' satisfying the requisite conditions for a particular operation (i', j') for which there are appropriately trained workers in $\hat{W}_t^F$, it could easily be the case that the next larger processing mode $k_{i'j'}+1$ might not satisfy the conditions, but that there exists a larger processing mode that indeed satisfies the conditions (this is because condition (iii) involves the processing time in a particular mode). In addition, as suggested in the proof of Theorem 2, it is sufficient to look at maximal values of |H| and k' satisfying the conditions, which facilitates the fathoming of a greater number of nodes. In this regard, it is also important to observe that maximal sets H may be limited to workers in $\hat{W}_t^F$ from the same group, i.e., since $\min_{h\in H}\{t_h^{'}\}$ is taken into account.

We now provide another set of conditions for fathoming nodes, generalizing a second result from Daniels and Mazzola (1994).

Theorem 3. For a partial schedule s, at time t, following the decision to schedule a set of operations $\tilde{E}_t \subseteq E_t$ (where $\tilde{E}_t$ is possibly empty), if there exists a subset of workers $H \subseteq \hat{W}_t^F$ and an operation $(i', j') \in \Omega_t$ satisfying (i) $s_{hj'} = 1$, for all $h \in H$, (ii) $k_{i'j'} + |H| \leq \bar{k}_{i'j'}$, and (iii) $t_{i'j'}^s + \hat{p}_{i'j'k'} \leq \min_{h\in H}\{t_h^{'}\}$, where $t_{i'j'}^s$ is the scheduled starting time of operation (i', j'), then s can be fathomed.

We now turn our attention to the calculation of bounds that can be used to fathom a partial schedule s. We assume that an upper bound $\overline{C}_{\max}$ on the optimal solution is known.

Assuming that sequence p of jobs is given, a lower bound on the makespan of any completion s^+ of s can be obtained by appending to the partial schedule the schedule of all remaining operations formed by processing each operation in its fastest possible processing mode (i.e., by relaxing the resource constraints for these operations) and scheduling each

operation to occur as early as the precedence constraints permit. This bound, $C^{+}_{\max}$, can then be used to fathom the node corresponding to s whenever $C^{+}_{\max} \geq \overline{C}_{\max}$.

Another bound based on the amount of remaining resource can also be utilized to fathom nodes. The application of such a bound, which is based on the saturated-processor bound of Garey and Johnson (1975), was used in the algorithm defined in Daniels and Mazzola (1994) for the complete resource flexibility case. Denote the set of operations that are being processed at time t by Θ^{R}_{t}, and observe that $\Theta^{R}_{t} = \Omega_{t} \bigcup \tilde{\tilde{E}}_{t}$. Again, let $\overline{C}_{\max}$ be the current upper bound on the optimal makespan. For each operation $(i, j) \in \Theta$, compute $k_{ij} = \min_{k \in K_{ij}} \{\hat{r}_{ijk} \bullet \hat{p}_{ijk}\}$, and recall that w^{S}_{j} denotes the number of workers who are trained (according to the skill matrix S) to operate station j, for each $j \in M$. If

$$\sum_{(i,j) \in \Theta \setminus \Theta^{R}_{t}} k_{ij} \geq |W|(\overline{C}_{\max} - t) - \sum_{(i,j) \in \Theta^{R}_{t}} \hat{r}_{ijk_{ij}}(C_{ij} - t)$$

or if for any $j' \in M$,

$$\sum_{(i,j) \in \Theta \setminus \Theta^{R}_{t} \text{ with } j = j'} k_{ij} \geq w^{S}_{j'}(\overline{C}_{\max} - t) - \sum_{(i,j) \in \Theta^{R}_{t} \text{ with } j = j'} \hat{r}_{ijk_{ij}}(C_{ij} - t)$$

then insufficient labor is available for the remaining operations to yield a schedule with makespan less than $\overline{C}_{\max}$, and the node can be fathomed. Note in the second set of inequalities that the two summations are taken over zero or one operation, since there can be at most one operation on station j' at time t (we assume that summation over the empty set yields a value of zero).

An additional lower bound can be obtained from the current partial solution at a given node as follows. For each station $j \in M$, let $(\hat{i}_{j}, j)$ be the last operation scheduled on that station in the partial schedule, and let $\overline{C}_{j}$ be its completion time. Let $\hat{C}_{\min} = \min_{j \in M} \{\overline{C}_{j}\}$, and

$\hat{C}_{\max} = \max_{j \in M}\{\overline{C}_j\}$. Similarly, for the incumbent solution, for each $j \in M$ let C_j^* be the completion time of operation $(\hat{i}_j, j)$, and define $C_{\min}^* = \min_{j \in M}\{C_j^*\}$, and $C_{\max}^* = \max_{j \in M}\{C_j^*\}$. If $\hat{C}_{\min} \geq C_{\max}^*$, the node can be fathomed. Also, if $\hat{C}_{\max} < C_{\min}^*$, a better incumbent solution can be constructed by replacing the corresponding partial schedule in the incumbent solution with the partial schedule s, while starting all the remaining operations $C_{\min}^* - \hat{C}_{\max}$ earlier than in the incumbent solution.

To solve an instance of FSPRF to optimality for each possible value of the skill metric, f^S, it is necessary to solve FSPRF(S) for all feasible skill matrices S. In the computational experiments described in Section 6, we solve instances of FSPRF to optimality for n = 5 jobs, m = 3 stations, and w = 3 workers utilizing this branch-and-bound algorithm. The optimal solutions from these problems offered valuable insight for designing the heuristic defined in Section 5, as well as for understanding characteristics of effective skill matrices. To improve the performance of the branch-and-bound algorithm, the second two levels of the heuristic (which is described in the next section) were applied for each skill matrix S to provide a good initial starting solution. In addition, all skill matrices S' for which $S' \prec S$ were identified, thus enabling the application of Proposition 1 to identify an even better starting solution.

In another set of the computational experiments, FSPRF was solved for each possible value $\hat{f}^S$ while considering only a restricted set of skill matrices. For these problems, the FSPRF problems were solved in order of increasing value of the flexibility metric $\hat{f}^S$. The heuristic was applied once for each value of $\hat{f}^S$, and the corresponding solution, along with the optimal solution from the previous value of $\hat{f}^S$, was used to provide a starting solution for the branch-and-bound algorithm. In addition, if the branch-and-bound algorithm determined that a particular skill matrix S could not improve upon an incumbent solution for that value of the

flexibility metric, then the algorithm proceeded immediately to the next skill matrix without solving problem FSPRF(S) to optimality.

5. A Heuristic for FSPRF

In this section we define a heuristic for obtaining approximate solutions to FSPRF. The heuristic operates on three levels, which address, respectively, the selection of a skill matrix, the sequencing of jobs, and the scheduling of operations. In the remainder of our discussion we focus on flow shops in which the number of workers is equal to the number of stations; i.e., w=m.

On the first (or highest) level the heuristic subprocedure HSKILL is employed to determine an effective skill matrix S that conforms with a specified value of the flexibility metric $\hat{f}$. Recall that M_h^S denote the set of stations on which worker $h \in W$ is trained as the worker's skill set. The HSKILL subprocedure determines each worker's skill set (i.e., makes the worker-skill assignments) by forming a chain in the worker-station graph, G(S), corresponding to skill matrix S. Within the context of chains, additional worker-skill assignments that might be required to achieve the desired value $\hat{f}$ of the flexibility metric are made based on the marginal benefit associated with each possible assignment.

For a given a value of m and a specified value $\hat{f}$, the HSKILL heuristic identifies the largest value of k for which $km = \left|\text{supp}(C^{m,k})\right| \le \hat{f}m^2$, or equivalently, $k \le \hat{f}m$; call this value $\hat{k}$. The skill matrix S is initially set equal to $C^{m,\hat{k}}$. If $\hat{k} = \hat{f}m$, HSKILL terminates, and the FSPRF heuristic then proceeds to the next level. Alternatively, if $\hat{k} < \hat{f}m$, then an additional $\hat{m} = \hat{f}m^2 - \hat{k}m$ number of 1's must be added to positions of $\text{supp}(\Delta^{m,\hat{k}}) = \text{supp}(C^{m,\hat{k}+1}) \setminus \text{supp}(C^{m,\hat{k}})$ to complete the skill matrix S. The selection of these $\hat{m}$ elements is

performed on the basis of ratios that estimate the benefit of increasing the skill set corresponding to each of the workers.

To compute these ratios we first focus attention on the total processing time required on each station if all jobs are processed in the same operating mode. The relative magnitudes of these total times measure the potential of each station to become a system bottleneck. Recall that $\bar{k}_{ij}$ is the number of possible processing modes for operation (i,j), and $\bar{k}_j = \max_{i \in N} \bar{k}_{ij}$ is the maximum number of processing modes on station j. Without loss of generality, we assume that all operations have the same maximum number $\bar{k}$ of possible processing modes. For each station $j \in M$ and processing mode $k \in \overline{K} = \{1, \cdots, \bar{k}\}$, define $\hat{P}_{jk} = \sum_{i \in N} \hat{p}_{ijk}$ to be the minimum total time required processing all jobs on station j in mode k.

Next, consider the set M of stations as ordered by the columns of skill matrix S. For any pair $j_1, j_2 \in M$, define the subset $M^{j_1, j_2} = \{j_1 + 1, j_1 + 2, \cdots, j_2\}$. Again, we apply the convention that if $j_1 + l > m$ for any l, then it is replaced by the unique $l' \in M$ satisfying $l' \equiv j_1 + l \pmod{m}$. For example, if m=5, $M^{1,5}$={2,3,4,5} and $M^{3,2}$={4,5,1,2}. For $j_1, j_2 \in M$ and a processing mode $k \in \overline{K}$, we identify the set $\hat{P}_k^{j_1, j_2} = \{\hat{p}_{lk} \mid l \in M^{j_1, j_2}\}$. For any $1 \le q \le \left| M^{j_1, j_2} \right|$, we then define the subset $M_{k,\{q\}}^{j_1, j_2}$ to be the q elements of M^{j_1, j_2} corresponding to the q smallest elements of $\hat{P}_k^{j_1, j_2}$ (ties, if any, are broken arbitrarily). Thus, for example, if m=5, k=2, j_1=1, j_2=5, with $\{\hat{p}_{j2} \mid j \in M\} = \{80,100,120,90,100\}$, then $M^{1,5}$={2,3,4,5} (as noted above) and $\hat{P}_2^{1,5}$={100,120,90,100}; for q=2, $M_{2,\{2\}}^{1,5}$={2,4} or $M_{2,\{2\}}^{1,5}$={4,5} if another alternative is chosen for ties; and $M_{2,\{3\}}^{1,5}$={2,4,5} for q=3.

Finally, to complete the skill matrix S, the HSKILL heuristic fills $\hat{m}$ additional 1's in the positions of $\text{supp}(\Delta^{m,\hat{k}}) = \text{supp}(C^{m,\hat{k}+1}) \setminus \text{supp}(C^{m,\hat{k}})$. For each $(j_1, j_2) \in \text{supp}(\Delta^{m,\hat{k}})$, we compute the ratio:

$$r^{\hat{k}}_{j_1,j_2} = \min_{k'=1,2,\cdots,\min\{\hat{k},\bar{k}-1\}} \left\{ \frac{\hat{P}_{j_1 1} + \frac{1}{k'+1}(\sum_{l \in M^{j_1,j_2}_{1,\{k'\}}} \hat{P}_{l1})}{\frac{k'}{k'+1}(\hat{P}_{j_2 1}) + \frac{1}{k'+1}(\hat{P}_{j_2,k'+1}} \right\}.$$

The $\hat{m}$ positions corresponding to the $\hat{m}$ smallest ratios $r^{\hat{k}}_{j_1,j_2}$ (with ties broken arbitrarily) are included in skill matrix S, thus completing the HSKILL subprocedure. The application of HSKILL heuristic, along with the rationales and computations of these ratios is illustrated in the following example.

Example 4. Using the data from the example problem in Daniels and Mazzola (1994) with n=4 jobs, m=4 stations, and w=4 workers, we assume here that $\bar{k} = 3$ is the maximum processing mode of any job. The job processing times as given in Table 2.

Table 2. Job Processing Times of a 4-station and 4-job Example Problem

Job i	$\hat{p}_{i11}$	$\hat{p}_{i12}$	$\hat{p}_{i13}$	$\hat{p}_{i21}$	$\hat{p}_{i22}$	$\hat{p}_{i23}$	$\hat{p}_{i31}$	$\hat{p}_{i32}$	$\hat{p}_{i33}$	$\hat{p}_{i41}$	$\hat{p}_{i42}$	$\hat{p}_{i43}$
1	14	13	12	25	15	10	28	16	12	10	5	4
2	38	19	13	30	24	20	10	8	7	16	12	10
3	20	14	11	5	4	3	16	13	11	22	17	15
4	26	18	14	20	10	7	24	17	16	8	7	6

From the data in Table 2, we compute the aggregate (total) processing times $\hat{p}_{jk}$ for each station and processing mode as indicated in the following table.

Station j	$\hat{P}_{j1}$	$\hat{P}_{j2}$	$\hat{P}_{j3}$
1	98	64	50
2	80	53	40
3	78	54	46
4	56	41	35

Now, suppose that HSKILL is to determine a skill matrix for $\hat{f} = 7/16$. In this case, m=4, $\hat{k} = 1$ and $\hat{m} = \hat{f}m^2 - \hat{k}m = 3$ elements from supp($\Delta^{4,1}$) must selected using the ratios $r_{12}^1 = (98 + 80/2)/(80/2 + 53/2) = 2.075$, $r_{23}^1 = (80 + 78/2)/(78/2 + 54/2) = 1.803$, $r_{34}^1 = (78 + 56/2)/(56/2 + 42/2 = 1.381$, and $r_{41}^1 = (56 + 98/2)/(98/2 + 64/2) = 1.296$. Hence selecting the elements corresponding to the 3 smallest ratios, HSKILL sets skill matrix S equal to $C^{4,1}$, and in addition, set s_{23}= s_{34}= s_{41}=1. Since the ratio r_{hj}^1 seeks to estimate the benefit of increasing the skill set of worker h by include station j in the 1-chain $C^{4,1}$, it is not surprising that r_{41}^1 yields the best ratio. Observe that station 1 with only one worker is much more likely to be a bottleneck than is station 4 (since $\hat{P}_{11} = 98$ is the highest and $\hat{P}_{41} = 56$ is the lowest among 4 stations). Additionally, if all operations on station 1 could be processed in mode 2 using a second worker, then the total (minimum) processing time would drop by almost 35%. Thus, if worker 4 were cross-trained to staff station 1 in additional to station 4, there is high potential for an improvement in system performance. Note that worker 4 would still need to allocate his or her total time between the two stations; hence, the ratio incorporates estimates of the average processing time increase on station 4 (reflecting time spend on station 1) and the average (faster) processing time on station 1 (resulting from some of the operations being performance in a faster processing mode).

We conclude this example by considering a case in which $\hat{f} = 7/8$. In this case, $\hat{k} = 3$ and HSKILL must select 2 additional elements from supp($\Delta^{4,3}$) to include in S, which is initially set equal to $C^{4,3}$. Computing the ratios

$$r_{14}^3 = \min\left\{\{\frac{98 + (56)/2}{(56)/2 + (41)/2}\}, \{\frac{98 + (78 + 56)/3}{(56)(2/3) + (35)/3}\}\right\} = \min\{2.598, 2.912\} = 2.598$$

$$r_{21}^{3} = \min\left\{\{\frac{80+(98)/2}{(98)/2+(64)/2}\},\{\frac{80+(78+56)/3}{(98)(2/3)+(50)/3}\}\right\} = \min\{1.593, 1.520\} = 1.520$$

$$r_{32}^{3} = \min\left\{\{\frac{78+(80)/2}{(80)/2+(53)/2}\},\{\frac{78+(56+80)/3}{(80)(2/3)+(40)/3}\}\right\} = \min\{1.744, 1.850\} = 1.744$$

$$r_{43}^{3} = \min\left\{\{\frac{56+(78)/2}{(78)/2+(54)/2}\},\{\frac{56+(80+78)/3}{(78)(2/3)+(46)/3}\}\right\} = \min\{1.439, 1.614\} = 1.439$$

HSKILL then also sets $s_{43}=s_{21}=1$.

The HSKILL heuristic is now summarized.

Heuristic HSKILL

Input: Problem size and problem data, including the number of workers w, which is equal to the number of stations m, and a specified value $\hat{f}$ of the flexibility metric.

Output: A skill matrix S with FM(S)= $\hat{f}$ that defines a chain.

0. **Initialize**. Determine the maximum $\hat{k} \in M$ satisfying $\hat{k} \le \hat{f}m$. Initially, set $S = C^{m,\hat{k}}$. If $\hat{k} = \hat{f}m$, then stop. Otherwise, let $\hat{m} = \hat{f}m^2 - \hat{k}m$, and go to step 1.
1. **Select remaining worker skill assignments**. For each ordered pair $(h, j) \in \text{supp}(\Delta^{m,\hat{k}})$, compute the ratio $r_{hj}^{\hat{k}}$. For each of the $\hat{m}$ smallest ratios (breaking ties arbitrarily), set $s_{hj}=1$ for the corresponding pairs (h,j), and then stop.

The second level of the FSPRF heuristic consists of the HSEQ heuristic subprocedure, which determines the sequence in which the n jobs should be processed on each station. The sequence determined at this level is used in the subsequent level by the subprocedure HSCHED to generate a feasible schedule to FSPRF. Note that a feasible schedule consists of a schedule of operation start times on each station, together with an assignment of specific workers to each operation (thus also specifying the processing mode and labor resource requirement of each operation). By utilizing a job sequencing heuristic in conjunction with the operation scheduling subprocedure (HSCHED), and alternating between the two, HSEQ iteratively considers a number of different job sequences.

Specifically, an initial set of operation processing times is calculated using the skill matrix S determined by the HSKILL heuristic and assuming that each operation is processed in

its fastest mode for this skill matrix. Given this fixed set of processing times, by relaxing the resource constraint, a classical flow shop scheduling problem is obtained. HSEQ then employs the flow shop scheduling heuristic of Nawaz, Enscore, and Ham (NEH) (1983) to determine a sequence. This sequence is then input to the next level heuristic HSCHED routine, which returns a feasible schedule of operation start times and also a corresponding set of operation processing times. The schedule that is returned by HSCHED is then compared with the best schedule, HBEST, with corresponding makespan value $C_{\max}^{HBEST}$, and these are updated if an improved schedule is realized. The new set of processing times is then used as input to the NEH heuristic and a new sequence is determined. The procedure is repeated in this manner until the same sequence is generated in two consecutive iterations, or until the number of iterations reaches the limit, MAXIT. We now summarize the HSEQ subprocedure.

Heuristic HSEQ

Input: Problem data, together with a skill matrix S and a specified value for the parameter MAXIT.

Output: A heuristic solution for FSPRF consisting of the schedule HBEST with corresponding makespan value $C_{\max}^{HBEST}$.

0. **Initialize**. Set $C_{\max}^{HBEST} = +\infty$. Calculate the initial vector of processing times **p**=$\{p_{ij} \mid (i, j) \in \Theta\}$ by setting the initial set of resource requirements **r**=$\{r_{ij} = \sum_{r \in W} s_{rj} \mid (i, j) \in \Theta\}$ and then determining the corresponding vector **P** of processing times by assuming operation (i,j) is processed by r_{ij} units of workers.
1. **Determine sequence**. Using the vector **p** of processing times, call the NEH flow shop heuristic, returning a sequence p of n jobs. If p is the same sequence obtained in the previous iteration or if the number of applications of this step exceeds MAXIT, then stop. Otherwise, go to step 2.
2. **Determine feasible schedule**
   - **a.** Using the sequence p, call HSCHED, returning a feasible schedule, SCHED, with makespan $C_{\max}^{SCHED}$. If $C_{\max}^{SCHED} < C_{\max}^{HBEST}$, update the best schedule found, HBEST and its makespan.
   - **b.** Using the processing times corresponding to SCHED, update vectors **p** and **r**. Return to step 1.

As mentioned above, the HSCHED subprocedure represents the third level of the FSPRF heuristic. At this level we are given skill matrix S, a specified job sequence $p = (i_1, \cdots, i_n)$ in which jobs will be processed on each station, and the problem data, including processing times $\hat{p}_{ijk}$ and resource requirements $\hat{r}_{ijk}$, for each operation (i,j), where $k \in \overline{K}$. Since both S and the sequence p are specified, we denote the subproblem occurring at this level by FSPRF(S,p).

The HSCHED heuristic constructs a feasible schedule to FSPRF(S,p) similarly to the construction used in the branch-and-bound algorithm, except that instead of creating multiple branches, HSCHED constructs a solution in a single pass by randomly selecting eligible operations and corresponding operation processing modes, and then allowing processing modes to be increased and additional operations to be scheduled through application of Theorems 1 and 2 (to avoid unnecessary withholding of the labor resource). Because of its randomized construction and its ability to generate feasible schedules rapidly, the procedure is repeated for a specified number, NUMREP, of iterations, and then returns the best solution found, SCHED, for problem FSPRF(S, p) .

The HSCHED heuristic systematically schedules job operations and assigns processing modes and sets of workers to these operations by beginning at time t=0, and then proceeding in order of increasing t as jobs complete processing. Recall that Ω_t is defined as the set of job operations in the partial solution scheduled to start before time t, and W_t^F is the set of workers who are free at time t. For each $j \in M$, let $W_{jt}^F \subseteq W_t^F$ be the set of workers available at time t who can operate station j, and $\hat{w}_{jt}$ be the number of such workers. E_t again denotes the set of operations that are eligible to begin at time t and $\tilde{E}_t$ represents the set of operations that are actually scheduled to begin at time t. . We now summarize the HSCHED subprocedure.

Heuristic HSCHED

Input: Problem data, a skill matrix S, a sequence p , and a parameter, NUMREP, indicating the number of replications of the partial-schedule construction procedure.

Output: A feasible schedule, SCHED, with makespan $C_{\max}^{SCHED}$ and a corresponding set of operation processing times and labor resource requirements.

0. **Initialize**.
   a. Set $C_{\max}^{SCHED} = +\infty$.
   b. Set t=0, $\Omega_0 = \varnothing$, and $\hat{r}_{j0} = \hat{w}_{jt}$ (which is the number of workers who are trained to operate station j).

1. **Select operations and corresponding workers at time t**. Determine E_t. If $E_t = \varnothing$, set $\tilde{E}_t = \varnothing$, and go to Step 2. Otherwise, define $k_{jt}^* = \max\{k \in K_{ij} \mid r_{ijk} = k \le \hat{w}_{jt}\}$ for each $(i, j) \in E_t$. Randomly select a subset $\tilde{E}_t \subseteq E_t$ of operations to begin at time t (along with a corresponding set processing modes) by the following steps.

a. Set $\tilde{E}_t = \varnothing$. Randomly choose an operation $(i, j) \in E_t$ and then randomly select a corresponding processing mode $k'_{jt} \in \{0,1,\cdots,k^*_{jt}\}$. Update E_t by removing operation (i,j). If $k'_{jt} > 0$, include operation $(i, j) \in \tilde{E}_t$, then randomly select $\hat{r}_{ijk'_{jt}}$ workers in W^F_{jt} to perform operation (i,j). Update W^F_{jt} by removing this subset of workers, and update $\hat{w}_{jt}$ by decreasing it by $\hat{r}_{ijk'_{jt}}$. Further update E_t by removing those operations (i', j') for which $\hat{w}_{j't}$ =0. Repeat this step until $E_t = \varnothing$.

b. Apply theorem 1 to the partial schedule generated the proceeding step. If an operation $(i, j) \in E_t$ and a corresponding set of workers in W^F_{jt} are identified for which the conditions of the Theorem are met, then assign the workers to the operation, updating $E_t, \tilde{E}_t, W^F_t, W^F_{jt}, k^*_{jt}$ and $\hat{r}_{ijk'_{ij}}$ accordingly. Repeat this until no set of workers W^F_t meeting the conditions of the Theorem is found. The, repeat this entire step applying Theorem 2 at time t to operations in Ω_t. When this has been completed, go to Step 2.

c. Apply the bounds introduced in the branch-and-bound algorithm to the current partial schedule. If it is determined that a completion of the partial schedule cannot improve on this schedule, then return to Step 0 to begin a new repetition of this subprocedure.

2. **Schedule operations at time t and proceed to next t**. If $\tilde{E}_t \neq \varnothing$, schedule each $(i, j) \in \tilde{E}_t$ to begin at time t in processing mode k'_{jt} with $\hat{r}_{ijk'_{jt}}$ workers assigned to it in Step 1a, and let its completion tie be $C_{ij} = t + \hat{p}_{ijk'_{jt}}$ Then (regardless of whether or not $\tilde{E}_t = \varnothing$) update $\Omega_t \leftarrow \Omega_t \bigcup \tilde{E}_t$. If $|\Omega_t| < mn$ (i.e., not all operations are scheduled), calculate $t' = \min\{C_{ij} \mid (i, j) \in \Omega_t \text{ and } C_{ij} > t\}$, Update $t \leftarrow t'$, and return to Step 1. Otherwise, a complete feasible schedule of job operations has been determined. Compare the makespan of this new schedule with $C^{SCHED}_{\max}$; if a better solution has been found, update SCHED and $C^{SCHED}_{\max}$. If the number of repetitions of this subprocedure during its current implementation is less than or equal to NUMREP, return to part b of Step 0 to repeat the subprocedure and generate a new solution; otherwise, stop.

Observe in Step 1a that the selection of a processing mode of 0 for the randomly selected operation corresponds to not scheduling the operation to begin at time t. This is important in that it is sometimes optimal to strategically withhold one or more workers so that they can be used in a subsequent time period; hence, to ensure that the heuristic is capable of generating an optimal solution, the procedure must accommodate the existence of idle labor

6. Computational Experiments

We performed a series of experiments to obtain a greater understanding of flow shop scheduling with partial labor resource flexibility. The principal objective of the experiments was to explore how the operational benefits derived from resource flexibility change as the level and mix of resource flexibility contained in the system vary. A second objective of the experiments

was to provide insight into effective ways of training workers by identifying characteristics of skill matrices that lead to high-quality optimal solutions. This contribution has important managerial implications for determining how resource flexibility should be distributed throughout the system (through worker cross-training) in order to optimize performance. The final objective of the experiments was to assess the performance of the FSPRF heuristic as a tool for identifying close approximations of the optimal solution.

The FSPRF branch-and-bound algorithm and heuristic were coded in C++ to conduct a series of computational experiments. The test problems were generated using an experimental design similar to that presented in Daniels and Mazzola (1994). Problem instances involve n jobs, m stations, and w=m workers. Each operation can be processed in one of three possible modes $\bar{k} = 3$, with processing mode k requiring k workers (recall that the actual number of processing modes in any particular problem instance also depends on the specified skill matrix). The normal (mode 1) processing time, $\hat{p}_{ij1}$, of each operation (i,j) is an integer randomly drawn from the uniform distribution U[10,50b_j], where the vector b = (b_1,…,b_m) specifies the bottleneck configuration of the problem. Higher processing modes for the operation are then given by

$$\hat{p}_{ijk} = [1 - a\,(1 - \frac{1}{k})]\,\hat{p}_{ij1}$$

where a is a parameter determining the marginal impact of additional labor on operation processing times. As a increases, the use of additional labor for processing the operation has a greater effect on lowering the operation's processing time. The experiments involve values of a = 0.2, 0.4, 0.6, 0.8 so that this effect can be examined in detail. The test problems also reflect m+1 different bottleneck configurations (b = (1,1,…,1), (1.5,1,…,1), (1,1.5,…,1), and (1,1,…,1.5)).

To achieve the objectives outlined above, three sets of experiments were performed. The first set of experiments involved the solution of 16 test problems involving 5 jobs, 3 stations, and

3 workers. For each of these problems, all of the 57 possible skill matrices were generated, and for each skill matrix S the resulting problem instance of FSPRF(S) was solved to optimality using the branch-and-bound algorithm. The second two levels (HSED and HSCHED) of FSPRF heuristic were applied to each problem to provide a good starting solution for the algorithm. The 16 problems were obtained by considering the four different a values in conjunction with the four different bottleneck configurations, and generating one problem instance per combination. Problem instances across a given bottleneck configuration are linked by keeping the normal processing times $\hat{p}_{ij1}$ for each operation (i,j) fixed across the four problem instances (corresponding to the four different values of a respectively); this structure allowed us to isolate the impact of a on the results.

Recall that the case of no flexibility ($S = I_{m\times m}$) corresponds to the classical flow shop (i.e., each worker permanently assigned to one and only one station); the complete flexibility case corresponds to the case of $S = E_{m\times m}$, the matrix with each entry being 1; and $\overline{C}(S)$ denote the optimal makespan of the corresponding FSPRF(S) problem. The complete flexibility benefit is defined as the reduction in the optimal makespan from no flexibility to complete flexibility, which equals $\overline{C}(I_{n\times n}) - \overline{C}(E_{n\times n})$. For any particular skill matrix S, the relative benefit achieved by S refers to the percentage of the complete flexibility benefit that can be realized by the skill matrix specified by S, which is equal to the ratio $[\overline{C}(I_{n\times n}) - \overline{C}(S)]/[\overline{C}(I_{n\times n}) - \overline{C}(E_{n\times n})]$; this quantity is then multiplied by 100 and reported as a percentage.

Figure 3 shows the distribution of the relative benefit achieved by different types of skill matrices as the flexibility metric f^S increases from 1/3 to 1 for one of the 16 test problems in the first set of experiments. We observe that a large fraction of the benefit of complete flexibility can be obtained with a relatively modest amount of partial flexibility. In this problem instance,

for example, the best skill matrix (i.e., the one with the highest relative benefit value) for f^S=0.67 captures about 95% of the complete flexibility benefit; and that even the best of f^S=0.56 provides over 80% of the benefit of complete flexibility. We also observe that the benefits of labor flexibility rise as the amount of flexibility increases, but with diminishing marginal returns (as shown by the concave property of the upper envelope of the graph).

The data in Figure 3 demonstrate considerable performance variability for different matrices with the same value of f^S; this observation conveys the importance of the proper mix of workers' skills. It is interesting to note that some skill matrices actually provided a relative benefit that is negative, i.e., for these skill matrices, improved performance would be realized if there were no flexibility in the system and each worker were simply assigned permanently to a specific station. A total of 6 such skill matrices exhibited this property for this problem instance. An example of such a skill matrix is given below.

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

For this skill set, the first worker must divide time between stations 2 and 3, while workers 2 and 3 are each dedicated to station 1. In the example, this skill matrix yields a relative benefit of –188.6%. An important managerial implication of this observation is that workers should be provided with some direction in selecting an appropriate portfolio of cross-training skills. Individual worker self-determination of skills with no consideration of the resulting mix of skills across all workers in the system can yield performance that is even worse than that occurring in a totally inflexible system.

Figure 3 also shows the difference in operational performances corresponding to skill matrices that are chains, sw-balanced, s-balanced, and others (as classified in Figure 2). We observe that matrices, which are both s-balanced and sw-balanced, yield solutions that occur

along the upper envelope of the graph. This observation provides strong evidence of the importance of skill matrices that are both s-balanced and sw-balanced (including chains). Moreover, in Figure 3 solutions corresponding to chains consistently occur near the upper envelope of the graph for each value of the flexibility metric f^S, suggesting that chains represent an important class of skill matrices. Further comparison of the performance of (either s- or sw-) balanced matrices against those that are not balanced, and chains against non-chains, reveals that skill matrices that have any of the attributes tend to result in better solutions than those that don't. These observations motivate the attributes of effective skill matrices discussed in Section 2.

Table 3 summarizes our computational experience with the 16 problems that constitute the first set of experiments. In the table each of the problem types is identified according to the bottleneck configuration (b_1, b_2, b_3) and the value of α. We next report the benefit associated with complete flexibility, which is computed by taking the difference between the optimal makespans from the normal flow shop and a completely flexible system, and then dividing the difference by the latter; the resulting amount is then multiplied by 100 and reported as a percentage. The next five columns correspond to values of the flexibility metric f^S ranging from 0.44 through 0.89 in increments of 0.11. For each value of f^S, we report the percentage of the available benefit (i.e., the benefit realized with complete flexibility) obtained by the best solution occurring among all skill matrices with the specified value of the flexibility metric f^S. Observe that no columns are included for $f^S = 0.33$ and $f^S = 1.00$, since the corresponding values in the table would (by definition) equal 0% and 100%, respectively.

We are also interested in assessing the frequency in which the best solution for each value of f^S corresponds to a skill matrix that is sw-balanced, as well as the corresponding frequency for the best-solution skill matrices that define a chain. These frequencies (reported as

percentages) are indicated in the next two columns of Table 3; note that skill matrices corresponding to $f^S = 0.33$ and $f^S = 1.00$ are not included in these values, since the matrices associated with no flexibility and complete flexibility are necessarily sw-balanced and also define chains. The last column reports the solution quality for skill matrices defining chains, i.e., for each value of f^S and each problem instance, the difference between the optimal makespan and the makespan corresponding to the best chain occurring among that set of skill matrices is reported as a percentage of the optimal makespan.

In Table 3 we observe that operational performance improves as the amount of labor flexibility in the system increases for all bottleneck configurations and for all values of a. The relationship exhibits pronounced diminishing marginal returns, with a significant improvement in performance (ranging from 20-61%) realized by adding only one skill to a single worker (i.e., for $f^S = 0.44$). This observation is consistent with Jordan and Graves (1995), who found that increasing the flexibility of production facilities even a small amount can yield sizable improvements in performance. We also note that all of the optimal solutions were obtained from skill matrices that are sw-balanced, and on the average 75% of the optimal solutions derive from chains. In addition, the quality of solutions corresponding to chains is quite high, with chains giving rise to solutions that are on the average within 0.56% of optimality.

A second, similar set of experiments was performed on problems involving n = 5 jobs with m=w=3 stations and workers, and n=4 jobs with m=w=4 stations and workers. We again use m+1 bottleneck configurations and four values of a. This set of experiments involved five replications of each problem type. To compensate for problem difficulty, we consider only those skill matrices that are sw-balanced for the m=3 problems; and for the m=4 problems we further restrict attention to skill matrices that define chains (since exhaustive evaluation of the 2306 skill matrices, or even the 1102 sw-balanced skill matrices, associated with m=w=4 effectively

renders the problem computationally intractable). The results from the first set of experiments suggest strongly that these restricted classes of skill matrices yield high-quality solutions.

The computational results are presented in Table 4 for n=5 and m=w=3 and in Table 5 for n=4 and m=w=4. Tables 4 and 5 are structured similarly to Table 3, with values for each problem type averaged over the five replications. We again see that a large portion of the available benefit attributable to labor flexibility is captured as f^S is increased modestly over its minimum value f^S = 0.33 in Table 4 and f^S = 0.25 in Table 5). In nearly all cases, over 90% of the available benefit is realized with values of f^S in the middle of the range between no flexibility and complete flexibility, again emphasizing that far less than complete flexibility is required to capture most of the associated improvements in operational performance. We also note in Table 4 that 64.75% of the optimal solutions are derived from chains, and that the best solutions generated from chains averaged within 0.7% of optimality. Thus, this set of experiments provides further evidence of the importance of chains in defining an effective class of skill matrices. This justifies the consideration of only these skill matrices when solving the n=m=w=4 problems (reported in Table 5) and provides a reasonable expectation for the resulting solutions to be close approximations to optimal solutions (that would occur if all skill matrices were considered).

Table 6 presents information on the branch-and-bound algorithm and the FSPRF heuristic for the second set of experiments. Specifically, the table reports the average CPU time (in seconds) required by the branch-and-bound algorithm to solve each instance of a problem type over all of the applicable skill matrices, as well as the average CPU time (in seconds) for the heuristic. The implementation of the FSPRF heuristic used in these experiments employed values of MAXIT = 60 and NUMREP = 100. We also report the average solution quality for the FSPRF heuristic, calculated by taking the difference between the makespans of the heuristic

solution and the optimal solution and dividing by the makespan of the optimal solution (and reported as a percentage). The results in Table 6 confirm the difficulty encountered in generating optimal solutions for even small problem instances. Even when the solution process is constrained to consider only sw-balanced skill matrices (for the n=5, m=w=3 problems) or chains (for the n=m=w=4 problems), excessively large computation times are observed. In contrast, the solution times for the FSPRF heuristic are quite reasonable. In both cases, computational times grow significantly with increasing a, suggesting that problem difficulty is most pronounced when processing times are sensitive to the number of workers assigned to an operation. The FSPRF heuristic also yields good approximations, e.g., heuristic solutions average 3.6% above the best solution for a given value of f^S. The overall quality of the heuristic solutions is therefore quite sufficient to provide important insight into the nature of the operational benefits of partial resource flexibility, which is the principal objective of these experiments.

In the third set of experiments the FSPRF heuristic was applied to problems involving n = 10 and 20 jobs and m = w = 5 stations and workers. This set of experiments involved problem types corresponding to the six bottleneck configurations and a = 0.2, 0.4, 0.6, 0.8, and five replications were generated for each problem type. For this set of experiments, MAXIT=100 and NUMREP=100 in the FSPRF heuristic. Table 7 presents the average benefit realized by the best heuristic solution for complete flexibility compared with that for the normal flow shop. The average relative benefit achieved for each value of the flexibility metric f^S is then reported (for brevity, the values of f^S are reported in increments of 0.08), as are the average CPU times for the heuristic. The results in Table 7 again show that performance improvements accumulate steadily with f^S, and that the benefits associated with labor flexibility increase most rapidly for higher values of a. As expected, the computational effort required to generate approximate

solutions grows with problem size; however, the heuristic represents a reasonable alternative for solving larger problems.

7. Concluding Observations and Remarks

This paper has addressed a flow shop scheduling problem in which processing time control is exercised through the dynamic assignment of partially-flexible labor resources. Our primary objective was to ascertain the extent to which the operational benefits attributable to labor flexibility can be captured by cross-training the work force so that each worker can be assigned to only a subset of the stages in the system. We presented metrics for partial flexibility, formulated the flow shop scheduling problem with partial resource flexibility, and characterized those skill sets that consistently yield superior operational performance. Our computational results suggest that a large portion of the available benefit associated with labor flexibility can be realized with a relatively small investment in cross training.

The experiments also demonstrated that operational performance for a given level of flexibility can vary widely depending on the mix of skills resident in the work force. A somewhat surprising result was that arbitrarily allocated partial flexibility can result in operational performance that is considerably worse than that achieved by an inflexible system. Thus, to consistently obtain high-quality solutions, scheduling, skill allocation, and resource assignment must be closely coordinated decisions.

The results further indicate that s-balanced and sw-balanced skill matrices and, in particular, skill matrices derived from chains are particularly effective in distributing skills across workers to optimize operational performance. This is an important finding in that chains constitute a direct connection with the earlier work of Jordan and Graves (1995) on partially flexible production facilities. Moreover, the work-force skill sets corresponding to chains involve training workers to operate stations occurring consecutively in the production line.

Hence, another connection is immediately established with other recent work by Bartholdi and Eisenstein (1996) and Zavadlav, McClain, and Thomas (1996) on self-buffering production lines, where the efficacy of worker flexibility across consecutive workstations has been established. Our research therefore provides a unifying framework for linking these two heretofore seemingly disparate studies. When viewed in this manner, our finding that a large percentage of the benefit of complete flexibility can be realized from a reasonably small amount of carefully selected partial flexibility connects and reinforces similar findings noted in Jordan and Graves (1995) and Zavadlav, McClain, and Thomas (1996).

This research represents an important step toward understanding the benefits of partial work force flexibility in flow shops and also toward characterizing effective ways in which workers should be cross-trained. The FSPRF problem is extremely difficult computationally. Future research can be directed toward the development of other (exact and heuristic solution approaches to the problem). Recent advances in heuristic approaches to scheduling problems have resulted in effective solution procedures for difficult scheduling problems (see, for example, Daniels and Mazzola 1993, Lee, Lei , and Pinedo 1997, and Daniels, Hoopes and Mazzola 1997). In addition, future research can also be directed toward the further characterization of effective worker-skill combinations, the consideration of other measures of performance, and the incorporation of partial resource flexibility to other production environments.

Acknowledgment

Part of this research was performed while the second author was on sabbatical leave at INSEAD.

Appendix

Proof of Proposition 2. If $S = C^{m,k^*}$, then $\hat{b}^S = 0$, $b^S = 0$, and the Proposition immediately follow. Otherwise, since $\text{supp}(C^{m,k^*}) \subset \text{supp}(S) \subset \text{supp}(C^{m,k^*+1})$, we have $b^S = 1$, and:

$$\max_{j \in M} mw_j^S \le k^* \frac{1}{k^*} + \frac{1}{k^*+1} = 1 + \frac{1}{k^*+1},$$

$$\min_{j \in M} mw_j^S \ge \frac{k^*}{k^*+1},$$

$$\hat{b}^S = \max_{j_1, j_2 \in M} \left| mw_j^S - mw_j^S \right| \le 1 + \frac{1}{k^*+1} - \frac{k^*}{k^*+1} = \frac{2}{k^*+1}. \quad \blacksquare$$

Proof of Theorem 1. Recall that the sets $\{W_g, g \in G\}$ form a partition of W, and that M_g is a subset of M of stations representing stations which any worker in group Wg can operate. Observe that there is a one-to-one mapping between the group index G and the set of non-empty subsets of the m-element set M, with $g \in G$ corresponding to $M_g \subseteq M$. Thus, the number of worker groups $|G| = 2^m - 1$ is equal to the number of non-empty subsets of an m-elements set.

Let x_g be the number of workers in group g for g=1, 2, …, 2m-1. Then the number of skill matrices is equal to the number of integer solutions to the equation:

$$x_1 + x_2 + \cdots + x_{2^m-1} = w, \text{ where } x_g \ge 0. \quad \text{(A1)}$$

Let $y_g = x_g + 1$. Equation (&) is then equivalent to:

$$y_1 + y_2 + \cdots + y_{2^m-1} = w + 2^m - 1, \text{ where } y_g \ge 1. \quad \text{(A2)}$$

The number of integer solutions to equation (A2) is equivalent to the number of ways a set of w+2m-1 elements can be divided into (2m-1) number of non-empty subsets, which is equal to the right hand side for S(m,w) in Theorem 1.

Now assume that m=w, and let k denote the number of 1's in a chain. If k is a multiplier of m, say k=k*m for some k*, there is then only one chain with k 1's in the matrix; otherwise, let k*m<k<(k*+1)m for some k* and k**=k-k*m. To generate chains of k 1's, k** extra 1's must be added to the k*-chain C_{m,k^*}, restricted to the m potential positions of $\text{supp}(C_{m,k^*+1}) \setminus \text{supp}(C_{m,k^*})$. Therefore, there are $\binom{m}{k^{**}}$ chains of k 1's. Thus, the number of chains of k 1's with $k^*m \le k \le (k^*+1)m$ is given by $2^m = \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{m}$ for any obtainable value of k*. The possible obtainable values of k are m, m+1, …, m2. These values of k can be rearranged as $m \le k \le 2m$, $2m \le k \le 3m$, …, $(m-1)m \le k \le mm = m2$. Therefore, there are total of (m-1) obtainable values of k*, namely {1,2,…,m-1}, satisfying $k^*m \le k \le (k^*+1)m$, and there are 2m number of chains for any of these (m-1) values of k*. However, chains with k=2m, 3m, …, (m-1)m are double-counted in the process, and subtracting these from the total yields the right hand side of C(m,m) in Theorem 1. $\blacksquare$

Proof of Theorem 2. Let s be such a partial schedule, and let s^+ be any completion of s. Suppose that there exists an operation (i', j') and a set of workers H that satisfy the conditions of the Theorem. Let $\tilde{s}$ be a modified partial schedule that is identical to s except for operation (i', j') and the assignment of additional workers in H. There are two cases to consider.

Case 1. If $(i', j') \in \tilde{E}_t$ in schedule s^+, then $(i', j') \in \tilde{E}_t$ in partial schedule s; hence the only difference between partial schedules s and $\tilde{s}$ is the reduced processing time of operation (i', j') in $\tilde{s}$ resulting from the assignment of workers in H to that operation. Observe that the modified partial schedule $\tilde{s}$ will also be generated by the branching procedure. In addition, the completion $\tilde{s}^+$ of $\tilde{s}$ formed by using the operations scheduled subsequently to time t with the same starting times, processing modes, and labor assignments as those occurring in s^+ will be resource feasible and the starting time of any of the subsequent operations will not be delayed beyond that occurring in s^+ (because of insufficient labor), since

$t + \hat{p}_{i'j'k'} \le \min_{h \in H}\{t_h^{'}\}$. This completion of schedule $\tilde{s}$ will also be generated by the branching procedure. Since the completion time of every operation in $\tilde{s}^{+}$ is less than or equal to the completion time of the corresponding operation in s^{+}, s can be fathomed.

Case 2. If $(i', j') \in E_t \setminus \tilde{E}_t$, then in schedule s^{+}, operation (i', j') is scheduled on station j' at some time $\hat{t}$ >t. If this operation is scheduled with a processing mode $k_{i'j'}$ that is less than or equal to the maximal processing mode $k^{'}$ satisfying the conditions in the Theorem, then the start time of operation (i', j') can be moved earlier to occur at time t and the conditions ensure that there is a subset of workers $H \subseteq W_t^F \setminus \tilde{W}_t$ who can be assigned to the operation without affecting any of other operations in $\Omega_t \bigcup \tilde{E}_t$. Call the resulting partial schedule $\tilde{s}$ and again observe that it will be generated by the branching procedure. In addition, by the definition of $t^*_{i'j'}$ and the fact that $t + \hat{p}_{i'j'k'} \le t^*_{i'j'}$, none of the workers in H will be needed for a subsequent operation before the completion of operation (i', j'). Once again, we can form the completion $\tilde{s}^{+}$ of $\tilde{s}$ by using the schedule from s^{+} for scheduling operations occurring after time t; moreover, because operation (i', j') has been shifted to an earlier time, it might be possible to shift some of subsequent operations to an earlier starting time (maintaining the same processing modes and labor assignments). Therefore, the completion times of all operations in schedule $\tilde{s}^{+}$ are no later than the corresponding completion times in s^{+}, and s can be fathomed.

On the other hand, if operation (i', j') is scheduled to begin in partial schedule s at time $\hat{t} > t$ with processing mode $k_{i'j'}$ that is strictly greater than the maximal value of $k^{'}$ satisfying the conditions of the Theorem, then it will be necessary to wait for additional workers who can operate station j' beyond those available in $\tilde{W}_t^B$. The earliest possible time that this can occur is $t^*_{i'j'} - \hat{p}_{i'j'k_{i'j'}}$. Thus, the earliest possible completion time for operation (i', j') in schedule s^{+} is $t^*_{i'j'}$. However, consider the partial schedule $\tilde{s}$ that is identical to s except that operation (i', j') starts at time t using any processing mode $k^{'}$ satisfying the conditions of the Theorem. For this partial schedule, the completion time of operation (i', j') is no later than $t^*_{i'j'}$, and using the same reasoning as above to identify a completion $\tilde{s}^{+}$ of this partial schedule, s can be fathomed. ¦

Proof of Theorem 3. Let s^{+} be any completion of s. If such a (sub)set of workers $H \subseteq \tilde{W}_t^F$ and an operation $(i', j') \in \Omega_t$ exist, then all workers $h \in H$, who would otherwise be idled until (at the earliest) time $\hat{t} = \min_{h \in H}\{t_h^{'}\}$ can be assigned to this operation. Denote the resulting partial schedule by $\tilde{s}$. With the improved processing time of this operation, its completion time will be $t^s_{i'j'} + \hat{p}_{i'j'k'}$, which is no greater than $\hat{t}$, thus we use the schedule of operations in s^{+} occurring after time t to identify a completion $\tilde{s}^{+}$ which allows s to be fathomed. ¦

References

- Alidaee, B., G.A. Kochenberger. 1996. A Framework for Machine Scheduling Problems with Controllable Processing Times. *Production and Operations Management* **5,** 391-405.
- Baker, K.R. 1994. *Elements of Sequencing and Scheduling*. Dartmouth College, Hanover, NH.
- Balas, E. 1975. Facets of the Knapsack Polytope. *Mathematical Programming* **8,** 146-164.
- Bartholdi, J.J., D.D. Eisenstein. 1996. A Production Line that Balances Itself. *Operations Research* **44,** 21-34.
- Daniels, R.L., B.J. Hoopes, J.B. Mazzola. 1996. Scheduling Parallel Manufacturing Cells with Resource Flexibility. *Management Science* **42,** 1260-1276.
- Daniels, R.L., B.J. Hoopes, J.B. Mazzola. 1997. An Analysis of Heuristics for the Parallel-machine Flexible-Resource Scheduling Problem. *Annals of Operations Research* **70,** 439-472.
- Daniels, R.L., J.B. Mazzola. 1993. A Tabu-Search Heuristic for the Flexible-Resource Flow Shop Scheduling Problem. *Annals of Operations Research* **41,** 207-230.
- Daniels, R.L., J.B. Mazzola. 1994. Flow Shop Scheduling with Resource Flexibility. *Operations Research* **42,** 504-522.
- Daniels, R.L., R.K. Sarin. 1989. Single machine Scheduling with Controllable Processing Times and Number of Tardy Jobs. *Operations Research* **37,** 981-984.
- Fine, C.H., R.M. Freund. 1990. Optimal Investment in Product-Flexible Manufacturing Capacity. *Management Science* **36**, 449-466.
- Garey, M.R., D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.
- Garey, M.R., D.S. Johnson, R. Sethi. 1976. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research* **1,** 117-129.
- Jordan, W.C., S.C. Graves. 1995. Principles of the Benefits of Manufacturing Process Flexibility. *Management Science* **41**, 577-594.
- Lawrence, S.R., T.E. Morton. 1993. Resource-Constrained Multi-Project Scheduling with Tardy Costs: Comparing Myopic, Bottleneck, and Resource Pricing Heuristics. *European Journal of Operational Research* **64**, 168-187.
- Lee, C.-Y., L. Lei, M. Pinedo. 1997. Current trends in Deterministic Scheduling. *Annals of Operations Research* **70**, 1-41.
- Nawaz, M., E.E. Enscore, I. Ham. 1983. A Heuristic Algorithm for The M-machine, N-job Flow Shop Sequencing Problem. *OMEGA* **11**, 91-95.
- Ozdamar, L., G. Ulusoy. 1995. A Survey of the Resource-Constrained Project Scheduling Problem. *IIE Transactions* **27**, 574-586.
- Sethi, A.K., S.P. Sethi. 1990. Flexibility in Manufacturing: A Survey. *International Journal of Flexible Manufacturing Systems* **2**, 289-328.
- Shi, D. 1999. Job Scheduling and Workforce Allocation in Flow Shops with Partial Resource Flexibility. *Ph.D. Dissertation*. Georgia Institute of Technology, Atlanta, GA.
- Talbot, F.B. 1982. Resource-Constrained Project Scheduling With Time-Resource Tradeoffs: The Nonpreemptive Case. *Management Science* **28**, 1197-1210.
- Trick, M.A. 1994. Scheduling Multiple Variable-Speed Machines. *Operations Research* **42**, 234-248.
- Van Wassenhove, L.N., K.R. Baker. 1982. A Bicriterion Approach to Time Cost Trade-Offs in Sequencing. *European Journal of Operational Research* **11**, 48-54.

- Vickson, R.G. 1980. Two Single-machine Sequencing Problems Involving Controllable Job Processing Times. *AIEE Transactions* **12**, 258-262.
- Zavadlav, E., J.O. McClain, L.J. Thomas. 1996. Self-buffering, Self-balancing, Self-flushing Production Lines. *Management Science* **42**, 1151-1164.

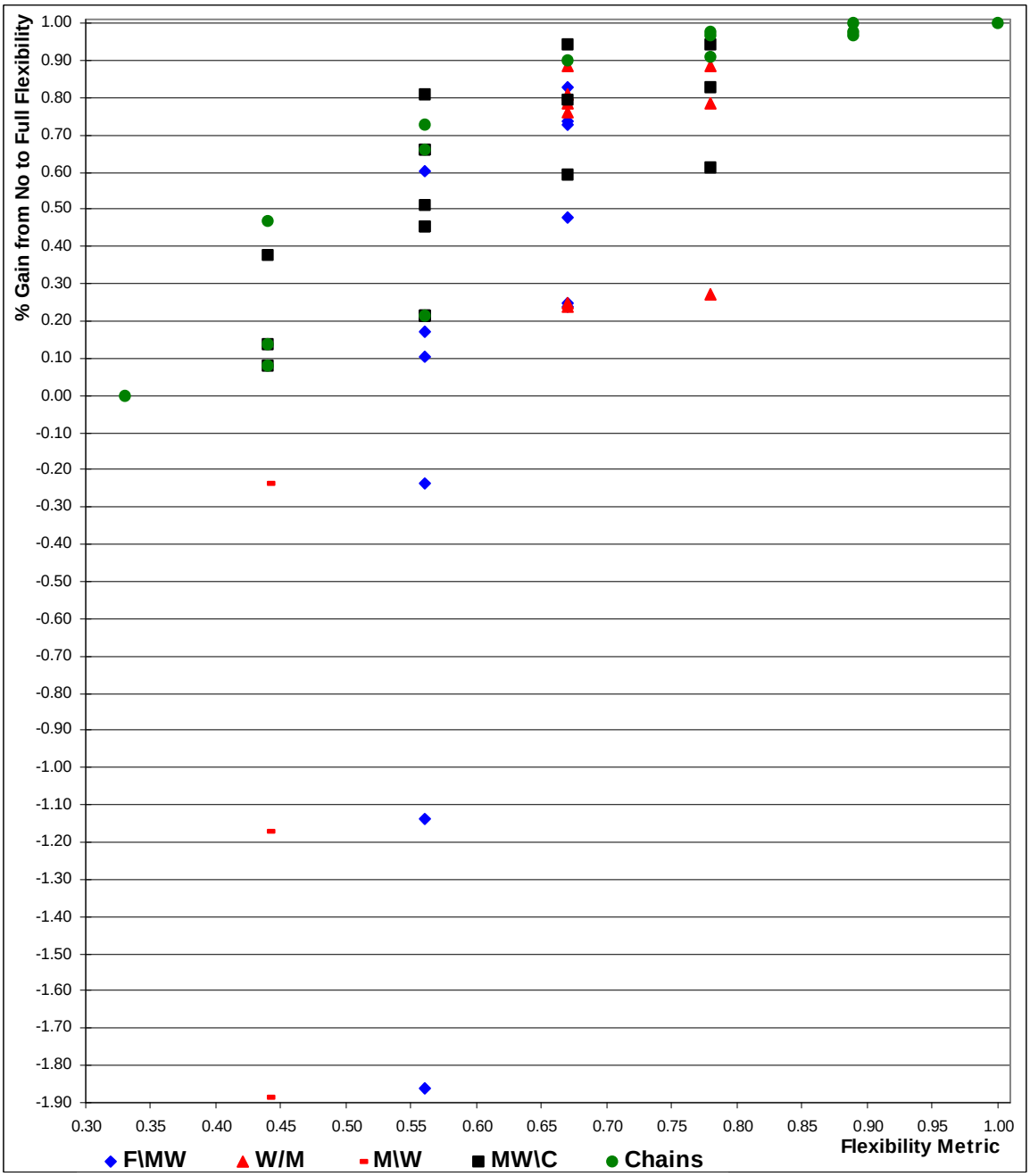

Figure 3. Distribution of Relative Benefit Achieved by Skill Matrices
(Matrices are Classified According to Figure 2)

Table 1. Number of Different Types of Skill Matrices

Stations	m=3			m=4			m=5		
Workers	w=3	w=4	w=5	w=4	w=5	w=6	w=4	w=5	w=6
S(m,w)	84	210	462	3,060	11,628	38,760	46,376	324,632	1,947,792
F(m,w)	57	168	402	2,306	9,902	35,228	33,031	270,907	1,762,957
M(m.w)	36	66	114	682	1,884	4,700	7,211	34,772	148,102
W(m,w)	42	78	132	1,102	3,076	7,638	19,241	95,282	421,152
F\MW(m,w)	12	81	246	1,100	6,382	26,156	12,810	168,990	1,304,735
M\W(m,w)	3	9	24	104	444	1,434	980	6,635	37,070
W\M(m,w)	9	21	42	524	1,636	4,372	13,010	67,145	310,120
MW\C(m,w)	18	57	90	532	1,440	3,266	6,231	28,012	111,032
C(m,w)	15	N/A	N/A	46	N/A	N/A	N/A	125	N/A

Where: S(m,w) = the number of skill matrices
F(m,w) = the number of matrices that are feasible
M(m,w) = the number of matrices that are s-balanced
W(m,w) = the number of matrices that are sw-balanced
C(m,w) = the number of matrices that are chains
F\MW(m,w) = the number of matrices that are feasible but neither s-balanced nor sw-balanced
M\W(m,w) = the number of matrices that are s-balanced but not sw-balanced
W\M(m,w) = the number of matrices that are sw-balanced but not s-balanced
MW\C(m,w) = the number of matrices that are both s-balanced and sw-balanced but not chains if chains are defined; otherwise, it is the number of matrices that are both s-balanced and sw-balanced

Table 3. Computational Experience with Initial Set of Test Problems

Bottleneck Configuration	a	Benefit of Complete Flexibility (%)	Percentage of Available Benefit Realized for Partial Flexibility Measure (ϕ^S)					Sw-balanced % Optimal	Chains % Optimal	Solution Quality for Chains (%)
			0.44	0.56	0.67	0.78	0.89			
(1,1,1)	0.2	4.7	20	70	80	90	100	100	80	0.28
	0.4	8.8	27.8	83.3	88.9	94.4	100	100	80	0.58
	0.6	13.2	42.3	84.6	88.5	92.3	96.2	100	80	0.50
	0.8	21.2	38.5	61.5	87.2	94.9	97.4	100	60	0.31
(1.5,1,1)	0.2	6.2	61.1	77.8	88.9	94.4	100	100	100	0.0
	0.4	11.2	64.5	80.6	87.1	93.5	100	100	100	0.0
	0.6	18.9	51.0	71.4	93.9	100	100	100	60	0.46
	0.8	30.0	42.3	71.8	88.7	95.8	100	100	80	0.47
(1,1.5,1)	0.2	11.1	51.6	77.4	87.1	93.5	96.8	100	100	0.0
	0.4	19.2	50.0	82.0	94.0	98.0	100	100	100	0.0
	0.6	27.5	53.7	83.6	92.5	97.0	100	100	80	0.24
	0.8	39.5	46.6	80.7	94.3	97.7	100	100	60	0.93
(1,1,1.5)	0.2	8.9	45.2	67.7	87.1	96.8	100	100	80	0.27
	0.4	16.6	48.1	68.5	88.9	94.4	96.3	100	60	0.20
	0.6	26.8	50.0	72.5	83.8	93.8	97.5	100	60	1.58
	0.8	37.3	52.4	75.7	92.2	96.1	98.1	100	20	2.55

Table 4. Computational Experience with Second Set of Experiments (n=5; m= w=3)

Bottleneck Configuration	a	Benefit of Complete Flexibility (%)	Percentage of Available Benefit Realized for Partial Flexibility Measure (ϕ^S)					Chains % Optimal	Solution Quality for Chains (%)
			0.44	0.56	0.67	0.78	0.89		
(1,1,1)	0.2	5.89	54.99	81.07	96.59	100	100	60	0.41
	0.4	11.61	48.61	79.98	91.07	97.15	99.44	52	0.70
	0.6	18.31	47.28	73.95	89.39	95.78	99.17	60	0.75
	0.8	26.85	43.18	76.34	89.48	97.06	100	72	1.16
(1.5,1,1)	0.2	8.88	53.37	83.41	91.44	96.14	98.62	84	0.14
	0.4	16.74	56.34	83.61	93.12	97.53	99.58	88	0.15
	0.6	25.19	54.43	80.41	91.99	96.95	99.74	68	0.40
	0.8	37.14	51.91	77.93	90.60	96.24	99.68	68	0.54
(1,1.5,1)	0.2	8.51	44.92	74.94	95.24	98.29	100	68	0.39
	0.4	15.97	47.18	79.57	93.01	97.59	99.29	64	0.70
	0.6	24.21	46.94	79.35	90.48	96.80	99.78	72	0.83
	0.8	34.32	44.51	79.57	91.75	97.11	99.59	64	1.20
(1,1,1.5)	0.2	9.12	48.11	80.29	92.36	98.44	100	64	0.32
	0.4	17.56	48.92	79.89	91.57	96.04	99.33	64	0.53
	0.6	27.46	49.20	77.14	92.49	98.14	99.05	40	1.01
	0.8	39.56	50.51	80.43	93.63	97.90	99.69	48	1.58

Table 5. Computational Experience with Second Set of Experiments ((n=m=w=4).

Bottleneck Configuration	a	Benefit of Complete Flexibility (%)	Percentage of Available Benefit Realized for Partial Flexibility Measure (ϕ^S)										
			0.31	0.38	0.44	0.50	0.56	0.63	0.69	0.75	0.81	0.88	0.94
(1,1,1,1)	0.2	11.55	24.42	46.45	63.27	76.70	84.44	92.60	96.37	98.49	100	100	100
	0.4	22.03	24.49	50.74	66.97	77.32	88.92	96.85	99.65	100	100	100	100
	0.6	35.62	25.22	49.56	63.53	76.34	88.06	94.01	98.54	99.12	100	100	100
	0.8	53.40	25.33	46.99	60.63	75.02	85.89	93.86	98.43	100	100	100	100
(1.5,1,1,1)	0.2	14.34	42.43	57.18	74.05	84.07	90.72	95.94	98.87	99.47	100	100	100
	0.4	27.97	45.55	61.11	71.23	81.93	90.76	94.28	97.28	98.63	100	100	100
	0.6	42.21	46.87	62.59	74.96	79.71	90.44	94.96	98.41	99.64	100	100	100
	0.8	61.62	46.22	59.28	70.48	80.38	88.40	95.70	98.34	99.86	99.86	100	100
(1,1.5,1,1)	0.2	12.14	18.97	45.55	65.84	81.68	88.41	94.05	98.08	100	100	100	100
	0.4	23.72	20.14	43.92	67.73	82.72	89.93	93.75	98.47	99.57	100	100	100
	0.6	35.76	21.73	42.69	67.49	81.54	91.67	95.52	98.81	100	100	100	100
	0.8	52.38	22.59	42.94	63.37	75.25	85.79	94.75	98.59	99.52	100	100	100
(1,1,1.5,1)	0.2	13.56	22.29	44.07	74.45	86.44	94.37	97.75	98.92	100	100	100	100
	0.4	26.63	24.83	45.83	68.62	77.61	87.50	93.4	98.48	100	100	100	100
	0.6	39.32	28.18	43.83	63.27	75.65	85,80	96.92	99.53	100	100	100	100
	0.8	58.56	24.73	43.14	60.88	74.13	86.93	95.69	98.73	99.81	100	100	100
(1,1,1,1.5)	0.2	12.01	28.74	47.71	61.60	77.49	86.71	92.33	96.06	97.78	98.52	100	100
	0.4	23.48	26.92	44.58	59.52	74.41	84.91	92.70	95.19	97.86	100	100	100
	0.6	36.08	28.25	46.30	58.46	76.26	87.34	92.79	97.05	99.32	99.71	100	100
	0.8	55.34	28.31	46.87	60.97	74.81	85.94	93.21	97.44	99.41	100	100	100

Table 6. Performance of FSPRF Branch-and-Bound Algorithm and Heuristic

Number of jobs and Stations (n,m)	Bottleneck Configuration	a	Branch-and-Bound Algorithm CPU Time (sec)	Heuristic CPU Time (sec)	Heuristic Solution Quality (Percent)
(5,3)	(1,1,1)	0.2	102.34	6.60	2.66
		0.4	446.26	19.26	3.59
		0.6	1574.32	30.04	3.63
		0.8	3707.80	76.84	4.05
(5,3)	(1.5,1,1)	0.2	64.40	12.58	2.10
		0.4	252.86	20.72	2.77
		0.6	1299.78	49.94	3.59
		0.8	3191.88	92.44	3.35
(5,3)	(1,1.5,1)	0.2	57.06	17.68	2.63
		0.4	247.84	37.66	3.38
		0.6	1190.50	74.92	3.82
		0.8	3828.62	44.46	4.59
(5,3)	(1,1,1.5)	0.2	126.76	50.40	2.70
		0.4	499.34	46.58	3.19
		0.6	1600.38	85.94	3.61
		0.8	2688.72	131.92	3.84
(4,4)	(1,1,1,1)	0.2	2369.30	22.09	3.38
		0.4	5369.30	36.10	4.45
		0.6	6309.39	68.18	4.70
		0.8	13142.17	41.80	5.25
(4,4)	(1.5,1,1,1)	0.2	3686.13	20.09	2.78
		0.4	6400.34	41.56	3.50
		0.6	6471.77	74.18	3.91
		0.8	19792.09	96.90	4.14
(4,4)	(1,1.5,1,1)	0.2	5988.95	22.03	1.89
		0.4	6751.79	24.77	2.55
		0.6	9026.93	47.50	3.95
		0.8	10597.29	79.48	4.76
(4,4)	(1,1,1.5,1)	0.2	2793.88	22.54	2.23
		0.4	3750.91	36.32	3.98
		0.6	6434.18	79.28	4.36
		0.8	11067.11	52.84	5.33
(4,4)	(1,1,1,1.5)	0.2	1786.24	24.67	2.50
		0.4	2553.24	66.11	3.32
		0.6	5693.67	89.41	3.91
		0.8	10925.00	110.59	4.55

Table 7. Computational Experience with FSPRF Heuristic on Larger Problems (m=5).

Number of Jobs (n)	Bottleneck Configuration a		(Approx.) Benefit of Complete Flexibility (Percent)	(Approx.) Percentage Complete Benefit Obtained with Partial Flexibility Measure (ϕ^S)										CPU Time (sec)
				0.24	0.32	0.40	0.48	0.56	0.64	0.72	0.80	0.88	0.96	
10	(1,1,1,1,1)	0.2	3.7	0	0	0	0	25.0	25.0	25.0	50.0	100	100	230.1
		0.4	8.2	0	0	1.4	36.2	51.0	67.1	70.8	93.3	93.3	100	427.4
		0.6	13.6	0	0	26.8	36.5	76.5	93.0	93.0	97.4	97.4	100	382.0
		0.8	22.0	0	6.3	32.3	49.6	68.3	84.1	91.1	95.5	97.9	100	631.3
10	(1.5,1,1,1,1)	0.2	4.0	25.0	25.0	50.0	50.0	59.1	84.1	84.1	84.1	100	100	210.7
		0.4	7.4	32.0	35.1	35.1	47.7	53.1	78.7	78.7	100	100	100	345.1
		0.6	15.0	12.5	20.0	20.0	58.5	66.7	75.8	81.7	92.4	100	100	463.7
		0.8	24.2	15.9	36.0	36.0	60.9	70.8	89.8	89.8	92.0	94.1	96.2	504.4
10	(1,1.5,1,1,1)	0.2	5.7	0	7.9	7.9	7.9	15.7	50.0	75.0	100	100	100	252.0
		0.4	10.1	0	11.1	30.9	30.9	36.1	92.3	92.3	92.3	100	100	287.0
		0.6	17.4	0	6.9	24.6	42.1	80.7	80.7	85.8	100	100	100	474.5
		0.8	26.8	0	10.9	42.8	50.3	73.9	83.0	85.9	91.1	97.7	98.4	794.7
10	(1,1,1.5,1,1)	0.2	4.0	0	16.0	16.0	16.0	20.0	40.0	40.0	80.0	100	100	216.9
		0.4	10.4	5.4	14.2	21.5	30.6	58.8	78.8	91.1	100	100	100	356.8
		0.6	17.8	7.3	13.0	27.2	63.3	80.9	80.9	84.8	91.7	100	100	501.3
		0.8	25.7	11.8	29.1	47.4	62.7	76.3	82.0	86.7	93.8	99.3	100	636.1
10	(1,1,1,1.5,1)	0.2	3.4	0	0	5.4	33.3	66.7	92.0	100	100	100	100	174.9
		0.4	8.7	0	7.1	7.1	16.6	50.9	92.0	92.0	100	100	100	349.6
		0.6	15.3	0	7.9	27.8	64.5	78.8	91.7	96.2	96.2	100	100	407.7
		0.8	25.1	5.6	9.1	35.0	62.0	75.1	84.6	89.4	92.8	100	100	614.7
10	(1,1,1,1,1.5)	0.2	3.4	0	0	0	25.0	50.0	75.0	100	100	100	100	209.86
		0.4	11.1	0	3.6	12.0	30.4	63.4	63.4	70.4	88.9	100	100	334.8
		0.6	17.7	4.1	4.1	44.6	66.2	80.5	86.1	92.2	98.5	100	100	457.6
		0.8	28.7	6.2	20.2	49.4	70.1	75.9	85.3	94.0	97.0	100	100	606.6
20	(1,1,1,1,1)	0.2	1.2	0	0	0	0	25.0	50.0	80.0	100	100	100	547.7
		0.4	2.7	0	0	0	0	20.0	46.1	66.1	80.0	100	100	738.5
		0.6	6.0	0	0	9.7	18.4	48.4	58.7	83.8	100	100	100	1040
		0.8	11.8	9.0	9.0	14.3	37.2	51.8	69.3	85.9	98.6	98.6	100	2040.
20	(1.5,1,1,1,1)	0.2	2.6	25.0	35.1	35.1	35.1	75.0	100	100	100	100	100	525.8
		0.4	7.2	17.3	18.7	18.7	41.1	45.4	73.4	80.0	100	100	100	988.3
		0.6	12.0	11.1	18.4	21.3	41.0	44.4	58.1	75.3	81.1	98.1	100	1561.
		0.8	18.9	32.8	32.8	40.8	48.3	67.3	84.2	88.6	93.5	95.3	97.8	1886
20	(1,1.5,1,1,1)	0.2	1.8	0	0	0	0	33.3	66.7	100	100	100	100	561.9
		0.4	5.3	0	0	0	8.1	27.0	32.7	60.4	88.8	100	100	1030.
		0.6	9.8	0	4.1	19.9	40.4	61.9	65.6	90.8	92.5	100	100	1505
		0.8	16.5	0	11.0	25.9	51.3	70.9	77.0	88.2	92.5	97.8	100	1779.
20	(1,1,1.5,1,1)	0.2	0.5	0	0	0	0	25.0	50.0	50.0	100	100	100	532.4
		0.4	5.6	0	0	0	1.9	1.9	4.5	20.8	40.8	81.9	81.9	801.6
		0.6	7.9	0	0	8.5	22.6	46.0	67.3	67.3	80.0	100	100	1434
		0.8	13.1	0	0	17.3	46.6	69.3	77.1	81.2	98.0	100	100	1782
20	(1,1,1,1.5,1)	0.2	2.6	0	0	8.3	8.3	41.7	60.0	66.7	66.7	100	100	457.9
		0.4	6.5	0	26.9	26.9	41.6	41.6	71.6	77.0	99.0	100	100	936.0
		0.6	12.5	13.0	19.5	26.2	46.2	60.5	81.2	81.2	84.8	100	100	1308
		0.8	18.9	11.1	13.8	32.2	58.3	63.7	87.2	91.8	95.6	100	100	1554
20	(1,1,1,1,1.5)	0.2	4.1	0.6	0.6	0.6	10.8	10.8	30.8	40.6	80.0	100	100	726.4
		0.4	8.7	7.3	7.3	14.1	44.3	67.6	71.0	80.7	91.5	100	100	1287
		0.6	15.4	14.5	14.5	46.1	62.8	73.6	85.5	97.0	97.0	100	100	1271
		0.8	24.8	24.5	24.6	51.6	63.4	75.7	82.8	93.2	99.2	100	100	1664.