Manufacturing & Service Operations Management

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Worker Cross-Training in Paced Assembly Lines

George Vairaktarakis, Janice Kim Winch,

Worker Cross-Training in Paced Assembly Lines

George Vairaktarakis • Janice Kim Winch
Case Western Reserve University, Department of OR and OM, 10900 Euclid Avenue, Cleveland, Ohio 44106-7235
Lubin School of Business, Department of Management and Management Sciences, One Pace Plaza, New York, New York 10038

Paced or Synchronous assembly lines are a popular class of assembly systems consisting of a series of assembly stations arranged in tandem. Every job (or order) visits all assembly stations in the same sequence and spends the same amount of time (known as the production cycle) at each station. Industries such as aircraft, fire-engine, and automobile assembly have production cycles of a few hours and are labor intensive. In spite of increased automation in such industries, human capital remains the most expensive and important contributor to a flexible production system. In this article we formulate the cross-training problem on a paced assembly line with m stations (mCT). We assume that each worker possesses a number of skills referred to as a skill vector. Our objective is to schedule a set of work orders through the assembly system so as to minimize the size of the required workforce and/or the workforce cross-training costs. We analyze the complexity of mCT and identify polynomially solvable cases. A variety of lower bounds is developed based on optimization techniques. These lower bounds are used to develop a branch and bound algorithm as well as to evaluate our heuristics. A computational experiment reports the performance of all algorithms. Using these algorithms, we examine how the formation of skill vectors affects the workforce size and draw guidelines for cross-training programs in organizations with labor intensive assembly operations.
(*Workforce Planning*; *Paced Assembly Line*; *Cross-Training*; *Integer Programming*)

1. Problem Motivation and Assumptions

The model presented in this paper is motivated by assembly operations in firetruck manufacturing. In §1.1 we describe details about the protocol of operations in this industry, and in §1.2 the relevant modeling assumptions.

1.1. Motivation

A firetruck consists of three main components: the body, the chassis and the engine. The chassis and the engine are purchased from an outside supplier, while the body and final assembly (of the three main components) take place in a paced assembly line physically located in two adjacent plants. The body related operations are performed in 8 distinct stations, and the progressively assembled body moves from station to station on a cart. A final assembly station completes the assembly line for a total of $m = 9$ stations. The size and weight of semi-finished bodies makes it impossible to maintain buffers of truck bodies between stations. This dictates synchronous movement from station to station, i.e., a common production cycle at each station. In our application the production cycle length is $c = 16$ hours. Workers are assigned to work on stations for eight hours each day, and the plant runs two shifts. At the end of the day (i.e., the 16-hour production cycle) semi-finished units are moved to the next

1523-4614/99/0102/0112$05.00

downstream station. The 16-hour production cycle allows some flexibility in deciding how many workers are needed in each of the two daily shifts. The number of workers per production cycle (i.e., 2 shifts) needed to assemble a firetruck ranges from 90 to 180. About 15% of the units assembled require a number of workers from the lower end of [90, 180], about 25% of the units require average numbers, and about 60% of the units require number of workers from the upper end of [90, 180]. Evidently, the production environment considered here is a *mixed-model assembly line* capable to process a number of different but structurally similar firetrucks.

The production manager determines the worker-hours needed for each firetruck based on experience. This number is then converted to number of workers. The management perceives that the workload estimates are mostly accurate and that any fluctuations in the actual number of hours required to complete each operation stems from fluctuations in worker performance. The discrepancies are small and handled using overtime. Still, overtime costs are immensely lower than new hires.

The majority of orders for firetrucks come from counties and municipalities that are more sensitive to taxpayer dollars than they are for delivery dates. As a result, the production planner has some flexibility on which orders should be batched together, and how they are sequenced with the objective to reduce workforce requirements. Batch sizes range from $n = 15$ to $n = 25$ units. Thus, resequencing an order or delaying it until the next batch, will delay delivery by at most about 25 days; an acceptable period for most customers. In addition, customer orders come with a delivery window (rather than an exact delivery date) of about two weeks. Batching of orders is considered to be a relatively straightforward activity of the production manager because they are based on the delivery windows and the product mix. Usually, the manager tries to batch together the same proportion of trucks having high, medium and low work content (for more details on the usual mix see §6.2).

Cross-training alternatives for our assembly line are shaped by the nature and sophistication of assembly operations. There are $r = 5$ distinct skills defined on $m = 9$ stations. Four of the skills involve the body, and each skill includes 2 stations. The first skill, say s_1, involves the first 2 stations, skill s_2 involves the next 2 stations, etc. The last skill (i.e., s_5) corresponds to final assembly and involves a single station. Moreover, each station is involved with only one skill, and workers of each skill can only work on stations involved with that skill. The specific operations involved with each skill and the corresponding stations are described below.

Skill 1: "pre-rigging"
Station 1A: pre-wiring, harnesses, pneumatic tubing
Station 1B: pneumatic tubing, external lights, overhead consoles
Skill 2: "mechanical assembly"
Station 2A: brake and other pedals, small hardware, some mechanical components
Station 2B: mechanical components, seats, interior and overhead panels, wall covering
Skill 3: "electrical assembly"
Station 3A: light hook-ups, preliminary electrical work
Station 3B: complete electrical work, dashboard
Skill 4: "cabin assembly and quality assurance"
Station 4A: final hook-ups, windshields
Station 4B: cabin, quality control

The above descriptions capture our operating environment in adequate detail. In this paper we formulate and solve the problem of scheduling production in a paced mixed-model assembly system so as to minimize total workforce size. The resulting workforce size is the sum of workers needed from each skill. Our solution ensures that, in every production cycle, the optimal number of workers per skill is never smaller than the number of workers needed to carry on the production schedule. Details on current and proposed management practices are relegated to §6.

1.2. Model Assumptions

In this subsection we discuss the assumptions made in our model and explain the motivation behind them. The first three assumptions follow directly from the operational protocol discussed in §1.1.

• Each work station requires exactly one skill and can process a single job at a time.

• Every job must perform an operation on every single work station in the same assembly sequence. The amount of work performed on each job in each station differs for different jobs.

• Jobs transfer from station to station synchronously every c time units.

The last of the above assumptions is mandated by the fact that firetrucks are large and expensive and hence it would be impractical to maintain buffers to smooth the flow of work. With respect to worker movement however, we assume the following.

• Workers are assigned to a station only after each synchronous transfer, and remain in that station for the entire production cycle of c time units.

Note that, with respect to labor hours, it would be more efficient to reschedule workers in an asynchronous fashion. Such rescheduling would avoid worker idle time. In contrast, synchronous worker movement suffers when the capacity required over a shift is not an integral number of workers. Such a loss however, is marginal considering that it is always less than the labor contribution of a single worker and the fact that the number of workers per station are 10 or more in our application. Moreover, the management can always assign one less worker to a particular station (thus totally eliminating worker idle time in that particular station), and hence force the assigned workers to utilize small amounts of overtime to finish the next job. Therefore, the *synchronous worker movement* loses very little labor capacity compared to the asynchronous mode. On the positive side, synchronous worker movement is simpler and more easily manageable while asynchronous worker movement presents the following challenges.

Our experience with manufacturers that rely heavily on labor is that managing asynchronous worker movement is an extremely difficult managerial task. Asynchronous movement allows workers to 'hide' in the crowd since it is not easy to keep track of the exact station that each worker should be at every point in time. Often, workers move from station to station not based on workforce requirements but to socialize with their fellow workers. As a result, a great amount of worker capacity is lost. A simple and easily enforceable approach to workforce management is to assign as few workers in each station as necessary to complete the subassembly of the corresponding station within the duration of a production cycle. In this case worker schedules are posted in the morning and every worker is assigned to a particular station for the entire shift in a synchronous fashion. Job enrichment is achieved by assigning workers to different stations on different days. In our experience from firetruck assembly, the labor loss in a synchronous worker movement mode is far less than the lost capacity suffered by asynchronous worker movement due to the above mentioned organizational difficulties.

Note that synchronous worker movement simplifies the workforce planning by limiting management intervention regarding communication and coordination of the workforce to the beginning of the production cycle. Also, it is easier to recognize problems on the line since for every unit of product a known set of workers are directly responsible for each station. Also, the duration of the common production cycle can be easily adjusted by adding or removing workers. For this reason, the tools that we develop in the following sections can be used not only for configuring the assembly line but also for testing the effects of design changes in the assembly process. Another assumption made in our model is the following.

• In each station, workers combine efforts to perform the required job assembly in c time units.

This assumption implies that the assembly work is equally divided among the workers assigned to each station. This assumption is made mostly for presentation convenience as it does not affect our model at all. As mentioned earlier, in our model we assume that the management uses past experience to specify the number of workers required in each work station in order to complete each job in c time units. In specifying this number the senior engineer that determines the job requirements for each station may or may not use this assumption. The overriding factor in his/her determination is his or her past experience for similar work orders. Therefore, in our model the last assumption is made for simplification purposes and as a basis for computational experimentations and it is not used explicitly in the analysis. Finally, in our base model we assume the following.

• A worker with a particular skill is capable to work in every station associated with that particular skill (and as a result the workers associated with each skill can be considered indistinguishable).

Because of this assumption, our base formulation does not capture situations where some workers are trained only for a subset of the stations associated with a skill. In other words, our formulation assumes that a cross-training program has been completed by the manufacturer and every employee has been trained to work in every station that involves his/her area of expertise. This assumption eliminates the secondary scheduling problem of assigning individual workers to stations from amongst the pool of workers trained for each skill. For our application this problem is quite secondary as the number of workers not fully trained for a skill are much fewer than the number of fully trained workers. As a result, minor rearrangement of workers easily resolves this problem. Such rearrangements are only needed when new trainees enter the pool of existing skilled workers. Considering that there are only a few such new trainees during the year, the frequency of such rearrangements is quite small. In order to capture applications where the number of new trainees is comparable to the number of fully trained workers for a skill, we show in §7 how one can extend our base model to account for this complication.

2. Literature Review

High labor specialization in assembly lines has the advantage of high production rates, low wages, quick worker training, and easy recruiting of employees. However, excessive specialization results in worker dissatisfaction presented in the form of absenteeism, tardiness, union grievances, job-related sicknesses, and sabotage. Production quality also suffers because workers are not motivated to produce high quality products. Also, since workers make only a small part of a product, no single worker is accountable for the quality of the whole product.

One of the proposals for modifying specialized jobs to provide for a broader range of job satisfaction is to cross-train workers so that they are able to work on a wide range of tasks. In this study we formalize and solve several variants of the worker cross-training problem as an optimization problem. With the help of this research, a manager can subdivide the operations involved with a product into skill vectors so that the work content of each skill is satisfactory from the worker standpoint, and the costs associated with the resulting mix of workers is economically feasible.

In our application, the paced assembly system is used to accommodate a variety of firetruck models. Hence, our assembly system is an example of a *mixed-model assembly line.* Due to market considerations recent attention to product variety has led to the use of mixed model lines where several variations of a basic stable design are allowed to be on the line at the same time. Example of such products could be different models of a car where each model may come with a variety of options. Then, depending on the model and options each car has different processing requirements. In most mixed model assembly lines the conveyance system moves at constant speed and the models are introduced in the line in equal time intervals; see Dar-El and Cother (1975), Okamura and Yamashima (1979), Thomopoulos, (1967). Safety buffers are often utilized to hedge against starvation and blockage of stations.

Some work has been done on workforce planning of assembly lines that produce a single item. Bartholdi (1992) presented a case study in which workers perform different tasks of the same item while Pinto et al (1981) have developed branch and bound and heuristic procedures for the case that workers perform the same task on different items. The importance of decentralized workforce control is captured by Bartholdi and Eisenstein (1996) where every worker of the assembly line follows a simple rule of what to do next. Ebeling and Lee (1993), developed an optimization model to assess the effect of cross-training into firm profitability.

Another approach to workforce planning, is to treat workforce as a generic resource to be allocated and scheduled appropriately. Along these lines, Daniels and Mazzola (1993) consider a flexible resource, the amount of which affects the processing time of an operation. In the context of workforce planning, this is analogous to assigning the right number of workers so as to achieve a predetermined processing time. The manufacturing environment considered in this research is a flowshop (and hence unpaced), and the authors present a tabu-search heuristic with near optimal performance. In a related paper, Daniels et al (1997) consider the flexible-resource scheduling problem on parallel identical machines.

To the best of our knowledge no research has been done on the cross-training problem in paced assembly lines with primary objective to minimize the total size of the workforce or the associated costs. The outline of the paper is as follows. In § 3 we formulate the cross-training problem. Lower bounding schemes are presented in § 4. These lower bounds are used to develop a branch and bound algorithm as well as to evaluate the heuristics developed in § 5. In § 6 we report our computational evaluation of the algorithms developed and draw conclusions on the effect of cross-training on workforce size. Also, we provide an example of how the company that motivated this research makes decisions currently. Extensions of the model are presented in § 7. We conclude in § 8.

3. Model Formulation and Basic Results

In this section we formally describe the worker cross-training problem. We first introduce the following pieces of notation.

n: number of jobs

m: number of stations in the assembly line

c: the duration of the production cycle. It is common for all the stations of the assembly line.

J_i: the ith job of the job set $J = \{J_1, J_2, \ldots, J_n\}$

ST_j: the jth station of the set $ST = \{ST_1, ST_2, \ldots, ST_m\}$ of stations

W_{ij}: the workforce requirement of job J_i on station ST_j in order to complete the jth task of J_i in c time units, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$.

We consider a serial assembly line with m stations. A target production cycle of c time units is given along with a set J of n jobs. Each job $J_i \in J$ has an associated m-tuple where the jth component, W_{ij}, is the number of workers needed to process job J_i in the jth station ST_j of the assembly line within c time units. All jobs enter the assembly line from the first station and are transferred to the next station of the line after c time units. A clarification follows on the calculation of the workforce requirements W_{ij}. Each job J_i requires p_{i1} labor hours on ST_1, p_{i2} labor hours in ST_2 … followed by p_{im} labor hours in ST_m. We assume that the assembly work for each operation can be accumulated by adding up the time each worker spends in an assembly station, and that all workers work at the same rate. Given a target production cycle of c time units, the workforce needed to complete the operation of J_i in ST_j is $W_{ij} = \lceil p_{ij}/c \rceil$. Thus, each job can be considered as a vector of workforce requirements $(W_{i1}, W_{i2}, \ldots, W_{im})$.

This operational protocol has been considered by Lee and Vairaktarakis (1997). The objective considered by the authors was the minimization of the maximum number of workers needed in the assembly system over any production cycle. Note that the number of production cycles needed to execute n jobs in the above paced assembly system is $n + m - 1$. The minimization of the total workforce size objective considered in Lee and Vairaktarakis (1997) is based on the premise that the worker tasks are of a generic nature, or that all workers are already trained to perform all assembly operations. In many industries however, assembly operations are diverse and specialized and hence each worker is trained to work on a selected subset of stations. For instance, electricians possess the skill to work on any station designed to perform electrical wiring operations, but they cannot work on any station that requires an unrelated skill outside their expertise, e.g., assembly of mechanical parts. Lee and Vairaktarakis (1997) indicate that their model is applicable to assembly environments where workers are trained to work in any of the m stations of the system. To formulate the different skills that may be possessed by the workers, we introduce the following skill vectors:

r: the number of distinct skills,

$A = \{A_l\}_{l=1}^{r}$ where $A_l = \{j_1, j_2, \ldots, j_{|A_l|}\}$ is a partition of the index set $\{1, 2, \ldots, m\}$ of stations in r parts,

$s_l = (s_{l1}, s_{l2}, \ldots, s_{lm})$; the lth skill vector, where $s_{lj} = 1$ iff $j \in A_l$; 0 otherwise.

Observe that the skill vectors $s_1, \ldots, s_r$ are binary m-tuples and $s_{lj} = 1$ indicates that all the workers assigned to work in ST_j are workers that possess the lth skill. Moreover, since A is a partition of ST, no two skills share a station ST_j. Hence, the above definition assumes that the operations involved with each skill vector are unrelated to each other. It becomes apparent that the research of Lee and Vairaktarakis (1997) corresponds to the special case where $r = 1$ and $s_1 = (1, 1, \ldots, 1)$.

In this paper, our objective is to find a permutation of jobs that minimizes the total number of skilled workers (i.e., the sum of workers needed from each skill) over the production horizon of $n + m - 1$ cycles. This cross-training problem on m stations will be referred to as *mCT*, and can be formulated as an integer program as follows:

$$x_{ij} := \begin{cases} 1 \text{ if job } i \text{ is scheduled at position } j; \\ 0 \text{ otherwise} \end{cases}$$

Z_l: the number of workers that possess the lth skill, $l = 1, 2, \ldots, r$.

$$(\textbf{mCT}) \ \text{Min} \sum_{l=1}^{r} Z_l$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad 1 \le i \le n, \tag{1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \le j \le n, \tag{2}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{k} s_{lj} W_{ij} \, x_{i,k-j+1} \le Z_l \qquad 1 \le k \le m - 1, 1 \le l \le r, \tag{3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} s_{lj} W_{ij} \, x_{i,k-j+1} \le Z_l \qquad m \le k \le n, 1 \le l \le r, \tag{4}$$

$$\sum_{i=1}^{n} \sum_{j=k-n+1}^{m} s_{lj} W_{ij} x_{i,k-j+1} \le Z_l \qquad n + 1 \le k \le n + m - 1, 1 \le l \le r, \tag{5}$$

$$x_{ij} \in \{0,1\}. \tag{6}$$

Equations (1) and (2) correspond to the assignment constraints, and equations (3)–(5) correspond to production cycles $1, 2, \ldots, n + m - 1$. In particular, equation (3) corresponds to production cycles $1, 2, \ldots m - 1$, (4) to production cycles $m, m + 1, \ldots, n$ and (5) to production cycles $n + 1, \ldots, n + m - 1$. Lee and Vairaktarakis (1997) have shown that the *mCT* problem is strongly $\mathcal{NP}$-complete even when there is a single skill ($r = 1$) and $m = 3$ stations in the assembly system. Because the *mCT* problem with $m = 3$ stations is strongly $\mathcal{NP}$-complete, the existence of a skill s_l involving three or more stations renders the corresponding *mCT* problem $\mathcal{NP}$-complete as well.

In the next subsection we identify a polynomially solvable case where each skill vector s_l consists of precisely two nonzero elements. We refer to such vectors as *2-station skills.* In particular, we consider *consecutive* 2-station skills where $A_l = \{2l - 1, 2l\}$, $l = 1, 2, \ldots, r$.

3.1. A Polynomially Solvable Case

Consider the case of *agreeable* workforce requirements defined below.

Definition 1. *An* mCT *problem with consecutive 2-station skills is said to be agreeable if for every pair of jobs i and i′,*

$$W_{i,1} + W_{i,2} \le W_{i',1} + W_{i',2}$$
$$\textit{iff } W_{i,2j-1} + W_{i,2j} \le W_{i',2j-1} + W_{i',2j}$$
$$\textit{for all } j = 2,3, \ldots, \frac{m}{2}.$$

As an example, the above definition is satisfied when the workloads in stations ST_3 and ST_4 are proportional to the workloads in ST_1 and ST_2 respectively. This would mean that a job that requires significant assembly in ST_1 (ST_2) also requires significant assembly in ST_3 (ST_4). In this case, it is easy to show that the sequence of jobs that minimizes the number Z_1^* also minimizes the number Z_l^* for all $l = 2,3, \ldots, r$. Hence, this problem becomes equivalent to minimizing the total workforce on a 2-station assembly line (2SAL) where all workers are trained to work on both stations. The 2SAL problem is the special case of *mCT* where $m = 2$ and $r = 1$ and can be solved optimally using the O-2SAL algorithm of Lee and Vairaktarakis (1997). The computational complexity of O-2SAL is $\mathcal{O}(n \log n)$ because it uses a nondecreasing order of W_{i1}'s along with a nonincreasing order of W_{i2}'s. In the rest of the paper we develop solution algorithms for the general *mCT* problem. To evaluate these algorithms we use the lower bounding schemes developed in the next section.

4. Lower Bounds

In this section we develop lower bounds to be used for evaluating our heuristics for the *mCT* problem. We start with a property that allows the identification of lower bounds on the total workforce size, by adding lower bounds on the number of workers needed for each particular skill s_l, $l = 1,2, \ldots, r$.

Indeed, for every skill s_l, $l = 1, 2, \ldots, r$, consider the subproblem

$$(\mathbf{P}_1) \qquad \text{Min } z_l$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad 1 \le i \le n,$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \le j \le n,$$

$$\sum_{i=1}^{n} \sum_{j=1}^{k} s_{lj} W_{ij} x_{i,k-j+1} \le z_l \qquad 1 \le k \le m - 1,$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} s_{lj} W_{ij} x_{i,k-j+1} \le z_l \qquad m \le k \le n,$$

$$\sum_{i=1}^{n} \sum_{j=k-n+1}^{m} s_{lj} W_{ij} x_{i,k-j+1} \le z_l$$

$$n + 1 \le k \le n + m - 1,$$

$$x_{ij} \in \{0,1\}.$$

Clearly $z_l^* \le Z_l^*$ for $l = 1, 2, \ldots, r$, where Z_l^* is the optimal number of workers in the formulation *mCT* of § 3. Therefore,

Proposition 1. $\Sigma_{l=1}^{r} z_l^* \le \Sigma_{l=1}^{*} Z_l^*$.

In light of Proposition 1, the following result provides a lower bound on the number of workers required by 2-station skill vectors that consist of adjacent stations. We refer to these vectors as *consecutive* skill vectors.

Corollary 1. *Let L be the set of consecutive 2-station skills (i.e.,* $|A_l| = 2$*), and* W_l *the optimal value of* ©-2SAL for the stations in A_l, for $s_l \in L$. $\Sigma_{s_l \in L} W_l \le \Sigma_{s_l \in L} Z_l^*$.

In the following subsections we develop two different lower bounding schemes. The first is based on surrogate relaxation (see Glover 1975) and the second on the *assembly line crew scheduling* problem (ALCS); see Hsu (1984). One advantage of the surrogate relaxation scheme is that it generates both lower and upper bounds.

4.1. Surrogate Relaxation Lower Bound

In this lower bound we relax the constraints (3)–(5) of *mCT*. Let μ be a vector of multipliers:

$$\mu = \left\{ \mu_{kl} : \mu_{kl} \ge 0, \sum_{k=1}^{n+m-1} \mu_{kl} = 1 \text{ for every } l = 1, 2, \ldots, r \right\}.$$

The multiplier μ_{kl} corresponds to the workforce size constraint for the lth skill vector during the kth production cycle. For each skill vector s_l, surrogating over all possible values of k results to replacing the constraints in the sets (3), (4), and (5) by constraints of the form

$$\sum_k \mu_{kl} \sum_i \sum_j s_{lj} W_{ij} x_{i,k-j+1} \le \sum_k \mu_{kl} Z_l = Z_l.$$

Then, surrogating over all possible skill vectors results in the following relaxation of *mCT*:

$$S(\mu) = \text{Min} \sum_l Z_l$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{l=1}^{r} \sum_{k=1}^{m-1} \mu_{kl} \sum_{i=1}^{n} \sum_{j=1}^{k} s_{lj} W_{ij} x_{i,k+1-j} \\ & + \sum_{l=1}^{r} \sum_{k=m}^{n} \mu_{kl} \sum_{i=1}^{n} \sum_{j=1}^{m} s_{lj} W_{i,m-j+1} x_{i,k+j-1} \\ & + \sum_{l=1}^{r} \sum_{k=n+1}^{m+n-1} \mu_{kl} \sum_{i=0}^{n} \sum_{j=0}^{k} s_{lj} W_{i,m-j} x_{i,n+j-k} \\ & \le \sum_{l=1}^{r} \sum_{k=1}^{n+m-1} \mu_{kl} Z_l = \sum_{l=1}^{r} Z_l, \\ & (1),(2),(6). \end{aligned}$$

Thus, to solve $S(\mu)$ optimally, we can equivalently solve:

$$\begin{aligned} S(\mu) = \text{Min} & \sum_{l=1}^{r} \sum_{k=1}^{m-1} \mu_{kl} \sum_{i=1}^{n} \sum_{j=1}^{k} s_{lj} W_{ij} x_{i,k+1-j} \\ & + \sum_{l=1}^{r} \sum_{k=m}^{n} \mu_{kl} \sum_{i=1}^{n} \sum_{j=1}^{m} s_{lj} W_{i,m-j+1} x_{i,k+j-1} \\ & + \sum_{l=1}^{r} \sum_{k=n+1}^{m+n-1} \mu_{kl} \sum_{i=1}^{n} \sum_{j=0}^{k} s_{lj} W_{i,m-j} x_{i,n+j-k} \\ & \text{s.t. } (1),(2),(6). \end{aligned}$$

This formulation corresponds to solving the assignment problem **AP** (μ):

$$\mathbf{AP}(\mu) = \text{Min} \sum_{ij} c_{ij} x_{ij} \quad \text{s.t. } (1),(2),(6). \tag{7}$$

where

$$c_{ij} := \sum_{l=1}^{r} \sum_{q=1}^{m} s_{lq} \mu_{j+q-1,l} W_{iq}.$$

If Z^* is the optimal value of mCT, then $S(\mu) \le Z^*$ for any set μ of multipliers. Hence, the solution of $S(\mu)$ can be used to obtain lower bounds for mCT. In addition, upper bounds of Z^* can be obtained as follows: For every vector μ the assignment X found by the assignment problem (7) produces a feasible sequence for mCT, say S. Starting with S we perform pairwise interchanges of jobs systematically, until no improving interchange is possible. This procedure is known in the literature as 2OPT heuristic, and has been successfully applied to difficult combinatorial problems including the Traveling Salesman problem in Lin and Kernighan (1973). The resulting sequences produce upper bounds for mCT and hence SR produces both lower and upper bounds. However, our experiments clearly indicate that the upper bounds produced by this procedure are somewhat high (average percentage relative deviation from the lower bound is in the range of 9.7%). For this reason, the following procedure is presented strictly as a lower bounding scheme.

Procedure SR
Input: W_{ij} $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$
Output: Lower bound LB on the optimal workforce level
$UB := +\infty$,
$LB := 0$.
Let

$$\mu_{kl} := \begin{cases} \dfrac{1}{n - m + 1} & \text{for } k = m, \ldots, \\ n \text{ and } l = 1, 2, \ldots, r; \\ 0 & \text{otherwise.} \end{cases}$$

Solve (AP) with $c_{ij} := \sum_{l=1}^{r} \sum_{q=1}^{m} s_{lq}\, \mu_{j+q-1,l}\, W_{iq}$, $1 \le i, j \le n$. Let S be the resulting sequence and Z_S the corresponding value
Repeat
If $(Z_S > LB)$ **then** $LB := Z_S$. Compute the per cycle workforce requirements of S, say $R_{1l}, R_{2l}, \ldots, R_{n+m-1,l}$, for $l = 1, 2, \ldots, r$. Let

$$\mu_{kl} := \begin{cases} \dfrac{R_{kl}}{\sum_{k=m}^{n} R_{kl}} & \text{for } k = m, \ldots, n; \\ 0 & \text{otherwise.} \end{cases}$$

Solve (AP) with $c_{ij} := \sum_{l=1}^{r} \sum_{q=1}^{m} s_{lq}\, \mu_{j+q-1,l}\, W_{iq}$, $1 \le i, j \le n$. Let S be the resulting sequence and Z_S the corresponding value
Until $(|LB - Z_S| < 1)$.

As seen, SR terminates when a new lower bound is found that is within 1 unit away from the incumbent lower bound. Generating multipliers according to the formula $\mu_{kl} = R_{kl}/\sum_{k=m}^{n} R_{kl}$ for $k = m, \ldots, n$, reflects the intuition that the maximum workforce requirement for a skill vector is very likely to be attained during one of the production cycles $k = m, \ldots, n$. This is because all stations are busy only during these cycles. Our computational experiments validate this intuition.

In what follows we present an alternative lower bound based on the *assembly line crew scheduling* problem (ALCS) considered in Hsu (1984).

4.2. ALCS Based Lower Bound

To motivate this lower bound, consider the following example.

Example 1. Consider the 6-job, 2-skill instance of Table 1, with $s_1 = (1, 1, 1, 1, 0, 0, 0)$ and $s_2 = (0, 0, 0, 0, 1, 1, 1)$. A generic schedule of the 6 jobs is shown in Table 2, where $J_{[i]}$ denotes the job in the ith position and the columns correspond to production cycles.

Let $A_l = \{j_1, j_2, \ldots, j_{|A_l|}\}$ be the set of stations associated with skill s_l. Notice that the cycles that contain all of the stations in A_l range from $c_{j_{|A_l|}}$ to c_{n+j_1-1}. Let

Table 1 Example of ALCS Based Lower Bound

J_i	$W_{i,1}$	$W_{i,2}$	$W_{i,3}$	$W_{i,4}$	$W_{i,5}$	$W_{i,6}$	$W_{i,7}$
J_1	7	5	5	8	7	5	5
J_2	5	6	6	6	5	6	6
J_3	7	7	7	7	7	7	7
J_4	4	7	7	4	4	7	7
J_5	6	4	5	8	6	4	5
J_6	5	5	8	6	5	5	8

Table 2 A Typical Schedule on 6 Jobs

Job/c_k	1	2	3	4	5	6	7	8	9	10	11	12
$J_{[1]}$	$W_{[1],1}$	$W_{[1],2}$	$W_{[1],3}$	$\boldsymbol{W}_{[1],4}$	$W_{[1],5}$	$W_{[1],6}$	$\boldsymbol{W}_{[1],7}$					
$J_{[2]}$		$W_{[2],1}$	$W_{[2],2}$	$\boldsymbol{W}_{[2],3}$	$\boldsymbol{W}_{[2],4}$	$W_{[2],5}$	$\boldsymbol{W}_{[2],6}$	$\boldsymbol{W}_{[2],7}$				
$J_{[3]}$			$W_{[3],1}$	$\boldsymbol{W}_{[3],2}$	$\boldsymbol{W}_{[3],3}$	$\boldsymbol{W}_{[3],4}$	$\boldsymbol{W}_{[3],5}$	$\boldsymbol{W}_{[3],6}$	$\boldsymbol{W}_{[3],7}$			
$J_{[4]}$				$\boldsymbol{W}_{[4],1}$	$\boldsymbol{W}_{[4],2}$	$\boldsymbol{W}_{[4],3}$	$W_{[4],4}$	$\boldsymbol{W}_{[4],5}$	$\boldsymbol{W}_{[4],6}$	$\boldsymbol{W}_{[4],7}$		
$J_{[5]}$					$\boldsymbol{W}_{[5],1}$	$\boldsymbol{W}_{[5],2}$	$W_{[5],3}$	$W_{[5],4}$	$\boldsymbol{W}_{[5],5}$	$\boldsymbol{W}_{[5],6}$	$W_{[5],7}$	
$J_{[6]}$						$\boldsymbol{W}_{[6],1}$	$W_{[6],2}$	$W_{[6],3}$	$W_{[6],4}$	$\boldsymbol{W}_{[6],5}$	$W_{[6],6}$	$W_{[6],7}$

$\mathcal{K}_l = \{c_{j_{|A_l|}}, c_{j_{|A_l|+1}}, \ldots, c_{n+j_1-1}\}$. In Table 2 for example, we have $\mathcal{K}_1 = \{4, 5, 6\}$ and $\mathcal{K}_2 = \{7, 8, 9, 10\}$.

For simplicity of notation, we define

$$W(k, l) = \sum_i \sum_j s_{lj} W_{ij} x_{i,k-j+1},$$

where the $x_{i,k-j+1}$'s follow the same definition as in the *mCT* formulation in § 3. $W(k, l)$ is simply the number of workers possessing the skill s_l required during the kth cycle. The minimum total workforce requirement of *mCT*, Z^*, can be expressed as $\min_S \Sigma_{l=1}^r \max_k W(k, l)$. For any fixed sequence S and skill set l, we have

$$\max_k W(k, l) \geq \max_{k \in \mathcal{K}_l} W(k, l) \geq \sum_{k \in \mathcal{K}_l} W(k, l))/|\mathcal{K}|. \qquad (8)$$

A lower bound on $\Sigma k \in \mathcal{K}_l\, W(k, l)$ in (8) can be obtained by simply choosing the $|\mathcal{K}_l|$ smallest $W_{i,j}$'s for every station j that belongs to A_l. Letting $W_{[l]}$ be the set of these $W_{i,j}$'s, we get

$$LB_l = \left\lceil \frac{\sum_{W_{i,j} \in W_{[l]}} W_{i,j}}{|\mathcal{K}_l|} \right\rceil \leq Z_l^*. \qquad (9)$$

Hence we have the lower bound,

$$LB^{(1)} = \sum_{l=1}^r = \left\lceil \frac{\sum_{W_{ij} \in W_{[l]}} W_{ij}}{|\mathcal{K}_l|} \right\rceil \leq \sum_{l=1}^r Z_l^*. \qquad (10)$$

For consecutive skill sets as in our example, we have $|\mathcal{K}_l| = n - |A_l| + 1$. The number of relevant production cycles can be calculated similarly for nonconsecutive skill vectors. In Example 1, according to (9) the value of LB_1 corresponding to A_1 is computed by finding the smallest $n - |A_1| + 1 = 3$-element sum for each of the first 4 columns of Table 1, and then taking the average of the 4 sums. Hence, $LB_1 = \lceil 60/3 \rceil = 20$. Similarly, $n - |A_2| + 1 = 4$ and $LB_2 = \lceil 63/4 \rceil = 16$. Hence, according to (10) we need at least $LB^{(1)} = 36$ workers.

The bound (10) can be significantly improved by carefully choosing the elements of the lists $W_{[l]}$. Instead of including in the set $W_{[l]}$ the $n - |A_l| + 1$ smallest elements of each station, we can identify an assignment that minimizes the contribution of the numbers in Table 2 that appear in bold. Note that the contribution to $LB^{(1)}$ of each boldfaced number $W_{[l],j}$ in cycles 4 through 6 is $W_{[i],j}/(n - |A_1| + 1)$. Similarly, the contribution of boldfaced numbers in cycles 7 through 10 is $W_{[i],j}/(n - |A_2| + 1)$. Then, the assignment problem

(AP)

$$Z_{\mathrm{AF}} = \mathrm{MIN} \sum_{i=1}^n \sum_{l=1}^r \sum_{k \in \mathcal{K}_l} \sum_{j \in A_l} \frac{W_{ij}}{n - |A_l| + 1}\, x_{i,k-j+1}$$

$$\text{s.t.} \quad (1), (2), (6), \qquad (11)$$

identifies an assignment that minimizes the contribution of the numbers in bold. We denote this lower bound by $LB^{(2)}$, i.e.,

$$LB^{(2)} = [Z_{\mathrm{AP}}] \leq \sum_{l=1}^r Z_l^*.$$

Even though our description of $LB^{(2)}$ was based on consecutive skill vectors, it is evident that AP can be adapted to handle an arbitrary partition of the set of stations to skill sets. The computational experiments presented in §6 indicate that, for the majority of cases tested, $LB^{(2)}$ is the best among the bounds developed in this section. Having generated lower bounds we turn our attention to exact and heuristic algorithms for the *mCT* problem. This is the focus of the next section.

5. Exact and Heuristic Algorithms

In this section we develop exact and heuristic algorithms for the *mCT* problem with arbitrary skill vectors. The heuristics developed are tested in §6 against the lower bounds of the previous section. To evaluate the performance of these heuristics from the optimal solution we describe next a branch and bound algorithm which is efficient for problems of small size.

5.1. Branch and Bound Algorithm for Small Problems

The root node of the search tree used in our branch and bound algorithm is at level 0 and represents the set of all possible job sequences. A node at level p represents the set of all possible sequences with the first p jobs already fixed. Branching is done by placing a new job in position $p + 1$. For each new node, lower and upper bounds are found on the workforce requirements associated with the sequences represented by that node. Below we provide more details on the lower bounding schemes used by the branch and bound algorithm.

Our computational experimentation indicated that the quality of the various lower bounds depends on the number of stations involved with a skill s_l. For instance, for consecutive 2-skill vectors, the lower bound of Corollary 1 appears to be dominant. For skill vectors s_l with $|A_l| > 2$, $LB^{(2)}$ appears dominant. However, $LB^{(1)}$ is faster to compute than $LB^{(2)}$ since it does not involve solving an assignment problem. For this reason, at each node of the branch and bound tree our algorithm uses $LB^{(1)}$ on all skill set scenarios with $|A_l| > 2$. If a node is not fathomed by $LB^{(1)}$, then $LB^{(2)}$ is obtained. If the node remains unfathomed, then an upper bound is obtained for this node by applying 2OPT on the assignment that resulted in $LB^{(2)}$ (for the case where $|A_l| = 2$, the upper bound is obtained by running 2OPT on the O-2SAL orders corresponding to 2-skills, and then selecting the best among them).

As we move deeper down the search tree, the lower bounds become tighter due to more cycles having their workforce requirements determined. For illustration, consider in Example 1 a node at level 3 which has its first three positions assigned to be the jobs J_1, J_2, J_3 respectively. The list $W_{[1]}$ for $LB_1^{(1)}$ contains W_{ij}'s from the fixed jobs and selected W_{ij}'s from the unfixed ones. Specifically, the sum in the numerator of (9) will contain the terms $W_{[1],4}$, $W_{[2],3}$, $W_{[3],2}$, $W_{[2],4}$, $W_{[3],3}$, $W_{[3],4}$ and three $W_{[j],1}$'s, two $W_{[j],2}$'s, and one $W_{[j],3}$ out of the workforce requirements W_{ij} of unfixed jobs (i.e., J_4, J_5, J_6). As for the second skill set, notice that its requirements in cycle 7 are already completed and equal $W_{[1],7} + W_{[2],6} + W_{[3],5} = 5 + 6 + 7 = 18$. Hence, $LB_2^{(1)}$ can be obtained as the maximum of 18 and the lower bound on the maximum workforce required in cycles 8, 9, and 10.

Our experiments showed that the time savings of using $LB^{(1)}$ prior to $LB^{(2)}$ (and the fact that the workforce requirements of completed cycles can be readily compared to the lower bound for each skill set) more than justify the loss in the quality of the lower bound. For this reason, the above described strategy for using $LB^{(1)}$ and $LB^{(2)}$ proved to be very effective.

The lower bound from the surrogate relaxation SR was not used in the branch and bound algorithm because it is dominated by $LB^{(2)}$. Hence, the value of procedure SR is in obtaining both a lower and upper bound by a single algorithm. The branching strategy used in the branch and bound algorithm was the "best lower bound first," and it proved superior to "depth first" search.

A minor modification was made to $LB^{(2)}$ to obtain tight lower bounds for the special case where $m = n$. The problem in such cases is that there is only one production cycle with all stations busy. Hence, cycles remain incomplete until the branch and bound search approaches the tips of the search tree. This problem is magnified as n increases. For this reason, we forced $\mathcal{K}_l$ to include several middle cycles. The best number of cycles included depends on n. In our experiment with $n = m = 15$, we found that 5 middle cycles (13 through 17) yield the best lower bounds. In general, larger values of n would require more middle cycles. Table 4 below reports the performance of the branch and bound algorithm on small problems. Next we proceed with developing a number of heuristics.

5.2. Heuristic Algorithms

In this subsection we develop and test a number of heuristic algorithms. We start by presenting a result on the worst case performance of an arbitrary heuristic for the *mCT* problem. The proof appears in the Appendix.

Let W_S and W^* be the workforce levels required for the *mCT* problem by an arbitrary sequence S and an optimal sequence S^* respectively.

THEOREM 1. $W_S/W^* \le \max_l \sum_{j=1}^{m} s_{lj}$ *and this bound is tight.*

In our experiments we tested several variations of insertion-type greedy heuristics with 2OPT applied on the resulting sequences. Such heuristics performed poorly with the average percentage deviation from the lower bound being in the range of 14%. Hence, we do not report more details for such heuristics. Our first heuristic utilizes the algorithm O-2SAL which as mentioned in §3.1 solves optimally the agreeable *mCT* problem.

Heuristic H_1
Input: W_{ij} $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$
Output: Upper bound *UB* on the optimal number of cross-trained workers
$UB := +\infty$
for $l := 1$ **to** r **do begin**
If $\Sigma_j s_{lj} > 1$, apply O-2SAL on all pairs of stations ST_j, $ST_{j'}$ with $s_{lj} = s_{lj'} = 1$
Let $S_{jj'}$ be the resulting sequence
if $(W(S_{jj'}) < UB)$ **then** $UB := W(S_{jj'})$ and save $S_{jj'}$
endfor

In H_1, we use $W(S)$ to denote the workforce requirement associated with the sequence S. The number of sequences produced in H_1 are of order $\mathcal{O}(r(\max_l |A_l|/2))$. Since each application of O-2SAL takes $\mathcal{O}(n \log n)$ time, the complexity of H_1 is $\mathcal{O}(rn \max_l |A_l|^2 \log n)$.

The following heuristic uses O-2SAL to produce sequences that are sub-optimal for two stations but may have good performance for the m-station problem. To do this systematically, we generate multipliers μ_{ij} for stations that belong to the same cross-training vector, say ST_1 and ST_2. These multipliers indicate the value that equalizes the difference $(W_{i2} - W_{j2})$ and $(W_{j+1,1} - W_{i+1,1})$ of workforce levels, i.e.,

$$\mu_{ij}(W_{i2} - W_{j2}) = (1 - \mu_{ij})(W_{j+1,1} - W_{i+1,1})$$

and are given by:

$$\mu_{ij} := \frac{W_{j+1,1} - W_{i+1,1}}{(W_{i2} - W_{j2}) + (W_{j+1,1} - W_{i+1,1})}.$$

If $0 \le \mu_{ij} \le 1$, we multiply all entries $W_{\cdot,1}$ by μ_{ij} and all entries $W_{\cdot,2}$ by $1 - \mu_{ij}$, for $r = 1, 2, \ldots, n$. Then we apply O-2SAL to this modified 2-station problem and the resulting sequence is evaluated for the m-station problem.

Heuristic H_2
Input: W_{ij} $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$
Output: Upper bound *UB* on the number of cross-trained workers
$UB \cdot= +\infty$
for every 2 stations ST_j and $ST_{j'}$ belonging to the same skill vector **do begin**
for any two production cycles k, l, $1 \le k < l \le n - 1$, **do begin**
if $W_{k,j'} \le W_{k+1,j}$ and $W_{l,j'} \ge W_{l+1,j}$ **then begin**
compute

$$\mu_{kl} = \frac{W_{l+1,j} - W_{k+1,j}}{(W_{k,j'} - W_{,j'}) + (W_{l+1,j} - W_{k+1,j})}$$

if $0 \le \mu_{kl} \le 1$ **then begin**
Apply O-2SAL with respect to $W'_{i1} = \mu_{kl} W_{i,j}$ and $W'_{i2} = (1 - \mu_{kl})W_{i,j'}$
Let $S_{kl}(j, j')$ be the resulting sequence
if $W(S_{kl}(j, j')) < UB$ then set $UB \cdot= W(S_{kl}(j, j'))$ and save $S_{kl}(j, j')$
endif
endif
endfor
endfor

This heuristic exploits the possibility that a loss due to sub-optimality of an O-2SAL sequence S for a 2-station problem may be offset by the gains realized when S is applied to the m-station problem. Our last heuristic, referred to as **Heuristic** H_3, is developed by running the branch and bound algorithm of §5.1 for 10 minutes of CPU time, or normal termination, whichever occurs earlier.

As mentioned in §3, the research of Lee and Vairaktarakis (1997), is the special case of *mCT* when $r = 1$, which disregards the cross-training issue. The lower bound used in that research is the heuristic *SR* for $r = 1$. As we mentioned earlier, the new lower bounds presented here dominated *SR* on all test problems (for all r-values considered) that we experimented with, and provides dramatic improvements in deviation and optimality gaps over *SR*. The heuristics H_1–H_3 are adaptations of similar heuristics presented in Lee and Vairaktarakis (1997) for the single skill case. No branch and bound algorithm was presented in that research. In the next section we will see that H_3 dominates H_1 and H_2, and that the latter two heuristics exhibit acceptable performance.

6. Computational Results

We test the algorithms presented in this paper on random problems with $m = 6$, 9, 12, or 15 stations and $n = 15$, 20, or 25 jobs. For each combination (m, n), the entries W_{ij} assume random values taken from the discrete uniform distribution on the interval [10,20], which is a realistic range for the application that motivated our research. For each combination, we run 10 problems on a SPARC station running Sun OS 5.4. To assess the *value of cross-training* we employed the following block design in our experiment.

For every problem generated for a particular (m, n) combination considered we tested skill vector scenarios of different lengths. We considered skill scenarios where each skill consists of *consecutive* stations. For instance, by consecutive 3-skill scenario we refer to the scenario where s_1 contains the first 3 stations, s_2 the following 3, etc. We define consecutive 4, 5 or more skill scenarios similarly. As indicated in Table 3, we consider consecutive 2- and 3-skill scenarios for $m = 6$ and $m = 9$ (in the latter case the 2-skill scenario consists of 5 distinct skills where s_5 utilizes only station 9). Also, for $m = 12$ and $m = 15$ we consider consecutive 2-, 3- and 4-skill scenarios (for $m = 15$ and the 4-skill scenario, the last skill vector utilizes the stations 13, 14, and 15 only). Finally, we report results for the case where no special skill is required to work in any of the stations, i.e., there is a single m-skill.

By our experiment not only do we want to evaluate the performance of our algorithms but also to draw conclusions on the effect of the scenario structure on the total workforce requirements. For instance, we would like to know how many additional workers are needed under the 2-skill scenario as compared with the requirements corresponding to the m-skill scenario. In the former scenario, workers are specialized but the work content is limited which may have implications on employee satisfaction. The latter scenario offers enriched jobs to the workers but the required level of cross-training (and associated costs) is very high.

For the above reasons we employ a block design where for a particular (m, n) combination, every randomly generated problem is solved for all consecutive skill scenarios considered for that (m, n) combination. For example, every randomly generated problem with $m = 9$ and $n = 15$ is solved for all values $|A_l| = 2$, 3, and 9. In this fashion we are able to draw conclusions on the effect of skill length on the size of the workforce under relatively homogeneous conditions. These results are presented later in Table 5.

In Table 3 we report the performance of H_1, H_2, and H_3 after 2OPT is applied to all the sequences produced by each heuristic. The mean and maximum (over 10 test problems) relative gap $((W_H - LB)/LB)$ 100% is reported for all heuristics where *LB* is the best lower bound obtained by H_3 (i.e., running the branch and bound algorithm for 10 minutes). Even though *SR* also computes a lower bound, it was never used in our computations because it was always worse than the lower bound provided by H_3. We summarize this experiment in Table 3.

We make the following observations. The average relative gaps of the heuristics H_1–H_3 are 7.9%, 7.1% and 5.9% respectively. Thus, the branch and bound based heuristic H_3 appears to be an excellent algorithm for solving *mCT*. Given the simplicity of H_1 and H_2 (as compared to coding H_3), they are attractive alternatives. Recall that H_1 and H_2 do not compute a lower bound. If evaluation of the proposed solution is necessary, then the SR procedure presents an alternative with decent performance (overall average deviation from the lower bound is 9.7% for the test problems of Table 3).

By looking at the averages Avg_*m* for the various values of *m*, we conclude that the mean percentage gap increases with the number of stations *m* in the assembly line. Similarly, the averages Avg_*n* indicate that the mean percentage gaps increase as *n* increases. The overall averages of the maximum percentage gaps of the heuristics indicate that H_1, H_2 and H_3 are robust since the gaps 9.2%, 8.3% and 7.0% are close to the corresponding overall mean percentage gaps.

Finally, the average performance of our heuristics

Table 3 Relative Deviations (W_H − LB/LB) · 100% of Heuristics

			H_1		H_2		H_3	
m	n	skill Size (p)	Mean	Max	Mean	Max	Mean	Max
6	15	2	4.8	6.4	2.9	5.3	1.6	4.2
		3	6.1	7.0	5.0	7.0	3.0	4.7
		6	5.3	7.1	4.7	7.1	1.9	2.4
	20	2	7.9	11.7	6.4	9.6	4.9	7.4
		3	9.0	10.3	7.6	9.0	5.3	6.7
		6	6.7	8.8	5.7	7.5	3.0	3.8
	25	2	9.0	10.5	8.2	10.5	6.1	7.5
		3	9.5	12.5	8.8	10.5	6.8	8.0
		6	7.4	8.1	6.5	8.0	4.2	5.9
9	15	2	4.4	6.1	3.5	4.8	1.6	3.9
		3	7.3	9.1	6.9	7.6	5.2	6.8
		9	4.9	5.8	4.3	5.1	3.3	3.4
	20	2	7.1	8.5	6.5	7.7	5.0	6.2
		3	10.0	11.6	8.8	10.1	7.4	8.5
		9	5.8	6.4	4.9	5.7	4.1	4.8
	25	2	8.1	9.7	6.9	8.2	6.1	7.6
		3	10.7	11.7	10.0	11.4	8.1	9.0
		9	6.7	7.3	5.6	6.5	4.5	5.7
12	15	2	6.1	7.7	4.9	6.2	3.9	5.2
		3	8.5	9.9	7.5	8.4	6.9	7.7
		4	7.3	8.0	6.6	7.4	5.5	6.3
		12	4.9	5.8	4.6	5.8	4.1	5.8
	20	2	8.8	9.9	8.0	8.9	7.4	8.3
		3	10.6	11.4	9.9	10.6	9.0	9.7
		4	8.7	9.5	8.0	8.6	7.0	7.5
		12	5.6	7.4	5.2	6.1	4.1	4.8
	25	2	10.5	12.4	9.4	11.0	9.0	10.5
		3	11.4	12.4	10.6	11.4	10.0	10.8
		4	9.2	10.2	8.7	9.5	7.6	8.4
		12	6.2	7.2	6.0	6.8	4.6	5.6
15	15	2	5.6	6.5	5.1	5.7	4.6	5.7
		3	9.4	10.4	8.6	9.1	7.9	8.6
		4	8.2	8.8	7.7	8.1	6.7	7.2
		15	3.1	3.9	2.7	3.8	2.4	3.3
	20	2	8.9	10.5	8.2	9.1	7.8	8.6
		3	11.3	12.6	10.6	11.7	10.0	10.8
		4	9.1	10.4	8.8	9.5	8.3	8.6
		15	5.9	6.7	5.6	6.7	5.3	6.7
	25	2	10.9	12.4	10.0	11.5	9.7	10.7
		3	12.2	13.2	11.5	12.8	11.0	11.8
		4	10.6	12.0	10.0	11.1	9.1	10.1
		15	6.1	6.5	5.6	6.0	4.8	5.9
		Average	7.9	9.2	7.1	8.3	5.9	7.0
		Avg.m = 6	7.3	9.2	6.2	8.3	4.1	5.6
		Avg.m = 9	7.2	8.5	6.4	7.5	5.0	6.2
		Avg.m = 12	8.2	9.3	7.5	8.4	6.6	7.6
		Avg.m = 15	8.4	9.5	7.9	8.8	7.3	8.2
		Avg.n = 15	6.1	7.3	5.4	6.5	4.2	5.4
		Avg.n = 20	8.2	9.7	7.4	8.6	6.3	7.3
		Avg.n = 25	9.2	10.4	8.4	9.7	7.3	8.4
		Avg.p = 2	7.7	9.4	6.7	8.2	5.6	7.2
		Avg.p = 3	9.7	11.0	8.8	10.0	7.6	8.6
		Avg.p = 4	8.9	9.8	8.3	9.0	7.4	8.0
		Avg.p = m	5.7	6.8	5.1	6.3	3.9	5.1

for various p-skill scenarios (i.e., the Avg.p figures) indicate very definitively that our algorithms exhibit the largest gaps for 3-skill scenarios. For greater values of p (i.e., $3 < p \leq m$), the relative gaps decrease. Recall that $LB^{(2)}$ is mostly responsible for the lower bound of skill scenarios with 3 or more skills. For $p = 2$ the average percentage gap exhibited by H_3 is 5.6%; in this case the lower bound based on O-2SAL is used.

In conclusion, the heuristics developed provide an effective set of tools for the mCT problem and offer alternative options for coding simplicity and level of optimality. In Table 4 we evaluate the effectiveness of the branch and bound algorithm of §5.1. Also, using the optimal objective values, we compute the corresponding optimality gaps for the heuristics.

The design of the computational experiment of Table 4 is identical to that for Table 3 except that we consider smaller values for the parameters n and m. We make the following observations. The average relative gaps exhibited by H_1–H_3 are 1.2%, 0.7% and 0.0% respectively, which allows us to say that all heuristics exhibit near optimal performance. The average and maximum CPU times (over the 10 randomly generated problems) required by the branch and bound algorithm are reported in the last two columns of Table 4. The overall average CPU requirement is 289.2 seconds. As expected, CPU times increase exponentially with n.

6.1. The Effect of Cross-Training on Workforce Size

Let $W(n, m, p)$ be the workforce level required by a test problem with n-jobs m-stations and the consecutive p-skill scenario. Then, the fraction $((W(n, m, p) - W(n, m, p'))/W(n, m, p))$ 100% represents the relative percentage decrease in workforce requirements between the consecutive p- and p'-skill vector scenarios. We assume that $p < p'$ and hence it is expected that $W(n, m,$

Table 4 Relative Optimality Gaps $(W_H - W^*)/W^* \cdot 100\%$ for Small Problems

			H_1		H_2		H_3		CPU sec.	
n	m	skill size	Mean	Max	Mean	Max	Mean	Max	Mean	Max
6	6	2	0.4	3.1	0.1	1.1	0.0	0.0	0.2	0.4
		3	0.2	1.2	0.0	0.0	0.0	0.0	0.4	0.6
	9	2	0.3	1.3	0.3	2.1	0.0	0.0	0.3	0.6
		3	0.6	2.3	0.5	2.9	0.0	0.0	0.8	1.0
	12	2	0.5	2.0	0.0	0.0	0.0	0.0	0.7	0.9
		3	0.1	0.6	0.0	0.0	0.0	0.0	1.1	1.4
		4	0.0	0.0	0.0	0.0	0.0	0.0	0.9	1.4
9	6	2	1.8	5.3	1.1	3.1	0.0	0.0	3.4	13.0
		3	1.6	4.3	0.6	2.1	0.0	0.0	11.7	16.5
	9	2	0.7	1.4	0.2	0.7	0.0	0.0	7.1	16.8
		3	1.1	2.2	0.5	2.1	0.0	0.0	23.0	31.5
	12	2	0.8	1.5	0.5	1.5	0.0	0.0	18.8	30.5
		3	0.7	2.2	0.4	1.6	0.0	0.0	40.5	50.0
		4	0.9	2.3	0.3	1.1	0.0	0.0	61.4	100.1
12	6	2	2.4	4.3	1.3	3.2	0.0	0.0	49.7	107.5
		3	3.1	4.4	2.4	4.4	0.0	0.0	237.1	367.6
	9	2	1.9	4.6	1.2	2.0	0.0	0.0	107.8	235.1
		3	2.1	4.2	1.3	2.8	0.0	0.0	788.8	1339.3
	12	2	1.5	2.5	1.2	2.0	0.1	0.5	829.1	2081.3
		3	1.8	2.6	1.1	2.1	0.2	1.1	1662.7	2359.1
		4	2.0	3.2	1.4	2.7	0.3	0.6	2228.7	3744.7
		Average	1.2	2.6	0.7	1.8	0.0	0.1	289.2	500.0
		Avg_m = 6	1.6	3.8	0.9	2.3	0.0	0.0	50.4	84.3
		Avg_m = 9	1.1	2.7	0.7	2.1	0.0	0.0	154.6	270.7
		Avg_m = 12	0.9	1.9	0.5	1.2	0.1	0.2	538.2	929.9
		Avg_n = 6	0.3	1.5	0.1	0.9	0.0	0.0	0.6	0.9
		Avg_n = 9	1.1	2.7	0.5	1.7	0.0	0.0	23.7	36.9
		Avg_n = 12	2.1	3.7	1.4	2.7	0.1	0.3	843.4	1462.1
		Avg_p = 2	1.1	2.9	0.7	1.7	0.0	0.1	113.0	276.2
		Avg_p = 3	1.2	2.7	0.8	2.0	0.0	0.1	307.3	463.0
		Avg_p = 4	1.0	1.8	0.6	1.3	0.1	0.2	763.7	1282.1

$p) > W(n, m, p')$; in fact this proves to be the case for all (p, p') combinations considered in our experiments. The average of these numbers over the 10 test problems for the associated (m, n) combination is reported in Table 5. The notation (2, 3) means that $p = 2$ and $p' = 3$. Similarly, $(\cdot, m)$ means that we compare the two longest skill vectors considered for (m, n) and hence $p' = m$ (for $m = 6, 9$ the second longest skill vector has 3 skills while for $m = 12, 15$ it has 4 skills). All $W(n, m, p)$ values are obtained by taking the smaller among the workforce requirements of the solutions produced by the heuristics H_1–H_3 applied to each test problem.

The last column of Table 5 records the relative percentage decrease in workforce size from the 2-skill scenario to the single m-skill scenario (i.e., $(p, p') = (2, m)$). Hence, the column headed by $((W(n, m, 2) - W(n, m, m))/W(n, m, 2))$ 100% indicates the savings in workforce size that can be obtained by cross-training workers on all assembly stations, as opposed to just 2 stations per worker. These savings average 16.4% for the problems considered. Also, it is evident that workforce savings increase with m, and decrease with n.

The x's in Table 5 indicate combinations not considered in our experiment due to the limited number of

Table 5 The Effect of Skill Length on Workforce Size

		$\frac{W(n,\ m,\ r) - W(n,\ m,\ r')}{W(n,\ m,\ r)}$ 100%						$\frac{W(n,m,2) - W(n,m,m)}{W(n,\ m,\ 2)}$ 100%	
		$(p, p') = (2, 3)$		$(p, p') = (3, 4)$		$(p, p') = (\cdot, m)$			
m	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max
6	15	5.0	6.8	x	x	6.8	8.8	11.5	12.6
	20	5.2	7.4	x	x	6.2	7.3	11.1	12.9
	25	4.1	6.1	x	x	6.0	7.3	9.8	12.2
9	15	5.8	7.6	x	x	11.7	13.5	16.8	18.9
	20	6.2	7.7	x	x	9.5	11.0	15.1	16.1
	25	6.3	8.3	x	x	8.7	9.7	14.4	15.5
12	15	4.7	6.3	3.5	4.2	12.4	13.8	19.5	20.6
	20	5.2	5.8	2.9	3.6	10.1	11.1	17.2	18.7
	25	4.9	5.8	2.9	3.1	8.9	9.5	15.9	17.0
15	15	5.8	6.6	2.5	3.3	20.0	21.4	26.5	27.8
	20	5.9	6.8	2.3	3.2	13.4	15.0	20.4	22.4
	25	5.8	6.5	2.4	2.8	11.4	12.8	18.5	19.8
	Average	5.4	6.8	2.8	3.4	10.4	11.8	16.4	17.9
	Avg_m = 6	4.8	6.8	x	x	6.3	7.8	10.8	12.6
	Avg_m = 9	6.1	7.9	x	x	10.0	11.4	15.4	16.8
	Avg_m = 12	4.9	6.0	3.1	3.6	10.5	11.5	17.5	18.8
	Avg_m = 15	5.8	6.6	2.4	3.1	14.9	16.4	21.8	23.3
	Avg_n = 15	5.3	6.8	3.0	3.8	12.7	14.4	18.6	20.0
	Avg_n = 20	5.6	6.9	2.6	3.4	9.8	11.1	16.0	17.5
	Avg_n = 25	5.3	6.7	2.7	3.0	8.8	9.8	14.7	16.1

stations in the assembly line (e.g., 4-skill scenarios for $m = 6$). Based on the combinations for $m = 12$ and $m = 15$ however, it becomes clear that workforce savings are higher for $(p, p') = (2, 3)$ than for $(p, p') = (3, 4)$. Given the definitiveness of this trend we conclude that the workforce size decreases at a decreasing rate with cross-training improvements. In fact, the ratio $(W(n, m, 2) - W(n, m, 3))/(W(n, m, 2) - W(n, m, m))$ (obtained by dividing the *Mean* column for $(p, p') = (2, 3)$ by the *Mean* column for $(p, p') = (2, m)$) ranges from 21.9% to 46.8% with an average of 34.9%. This means that in an assembly system with low levels of cross training (i.e., $p = 2$), training each worker on an additional station reduces the size of the workforce by an average of 34.9% of the total possible reduction that could be realized if we trained all workers on all stations. Summarizing our conclusions we have:

- the size of a workforce trained for 2 stations per worker is on average 16.4% larger than a fully cross-trained workforce,
- the workforce size decreases at a decreasing rate with improvements in cross-training,
- in assembly systems with low levels of cross-training, training each worker on an additional station reduces on average the workforce size by 34.9% of the reduction that could be realized by a fully cross-trained workforce.

6.2. Current and Proposed Practice

In this subsection we show how planning decisions are made currently in the company that motivated this research, and show the differences with the proposed approach. Consider the 10-job example in Table 6 with the indicated stations per skill. This example follows the descriptions of §1.1.

Table 6 Example

	skill 1		skill 2		skill 3		skill 4		skill 5	Total
J_1	10	13	9	11	14	16	11	10	15	109
J_2	8	16	8	12	17	14	15	12	17	129
J_3	13	15	11	9	16	21	19	13	18	135
J_4	18	9	17	13	11	19	11	17	9	124
J_5	21	13	19	16	9	11	20	18	21	148
J_6	13	18	16	11	22	19	17	21	16	153
J_7	9	21	18	24	16	18	9	10	18	143
J_8	17	16	19	15	19	20	22	10	22	160
J_9	18	19	20	17	21	19	11	22	17	164
J_{10}	11	21	12	21	18	14	19	16	18	150

For the above example, the plant manager would code each job by L, M, or H according to whether the total workforce requirements (in the last column of Table 6) are low, medium, or high. In this classification scheme, the jobs J_1–J_{10} are coded by *L,M,M,M,H,H, H,H,H,H*, in this order. Then, the manager intertwines the various jobs so as to decrease the adverse effect of large jobs on workforce requirements. The coded sequence *H,M,H,M,H,S,H,M,H,H*, appears as a good possibility since the tails of the sequence are occupied by *H*-jobs (notice that during the first few and the last few production cycles, part of the assembly line is not occupied by firetrucks). Also, the S-job is in the middle of the run, thus alleviating the requirement for high workforce levels during the middle of the run where all the assembly stations are busy. Similarly, the M-jobs are scheduled between H-jobs, and closer to the middle of the production run.

A starting sequence that follows the above classification is J_9, J_3, J_{10}, J_4, J_7, J_1, J_5, J_2, J_6, J_8. The number of workers needed for this sequence is 181 (39, 38, 39, 43, 22 for skills 1 through 5 respectively). From this point on the manager follows an ad hoc procedure for manual interchanges among jobs with the objective of minimizing the total workforce size. This procedure is beyond any scientific or intuitive description and requires several hours of trial and error. Also, recall that normally the number of firetrucks (jobs) in a batch may be 20 or more. This fact, and the fact that the production planner has currently no means of incorporating cross-training vectors into his/her experimentation, makes the planning decision extremely difficult.

The workforce requirements corresponding to algorithms H_1–H_3 were found to be 161, 161, and 159 respectively for this example. The optimal sequence is J_8, J_3, J_4, J_1, J_5, J_9, J_7, J_{10}, J_2, J_6 and was obtained by H_3 in 29 seconds. The individual requirements per skill are 34, 36, 36, 31, and 22 for the skill vectors 1 through 5 respectively. Evidently, the value of our algorithms is in identifying an optimal or near optimal production plan in seconds, rather than laboring for hours to obtain inferior schedules. Moreover, our algorithms incorporate the effect of skill vectors on workforce size and allow to ask "what if" questions that help determine what an effective cross-training program may look like.

7. Extensions

The objective considered in the *mCT* model is the minimization of the total workforce size. Depending on the application a potentially more relevant objective may be the minimization of cross-training costs. This is because some stations may require generic assembly skills that can be hired at low cost while others may require specialized skills that are in short supply. Let c_l denote the cost associated with training or hiring a worker for skill s_l. Then, by replacing Z_l by $c_l Z_l$ in the objective function, the *mCT* formulation can be used to minimize the total training costs.

Another application of the *mCT* model is in minimizing the cost of training *additional* workers, given the workforce on hand for each skill. Let W_l denote the number of workers on hand possessing skill s_l, $l = 1, 2, \ldots, r$. Then, by replacing the objective function of *mCT* by $\sum_l c_l(Z_l - W_l)$ we obtain a minimum-cost schedule for *additional* workforce cross-training. This variant is also useful in scheduling a new batch of n jobs following the completion of the previous production batch of n jobs. Towards this end, note that the model *mCT* is cast as a finite horizon workforce planning problem where a specific set of n jobs is to be processed only once. As a result, the last few assembly stations are idle during the first $m - 1$ production cycles of a production run of n jobs, and the first few stations are idle during the last $m - 1$ production cycles of the $(n + m - 1)$-cycle duration of the run. In practice, a manager sequentially schedules batches of jobs as soon as (s)he knows with certainty the work orders that should be included in the next production batch. Effectively, the first batch imposes a sort of initial condition on the required size Z_l of workers possessing skill s_l, $l = 1, 2, \ldots, r$ during each of the first $m - 1$ production cycles. More specifically, let S be the workforce schedule of the current production batch, and let

$\bar{W}_{kl}$: the number of workers possessing skill s_l, $1 \le l \le r$, required in S during the production cycle c_{n+k}, $1 \le k \le m - 1$.

Then,

$$A_{kl} := \max_k W_{kl} - \bar{W}_{kl},\ 1 \le l \le r,\ 1 \le k \le m - 1$$

is the number of workers possessing s_l that are available to work on the new production batch during c_k. Then, to minimize the number of additional workers needed (if any) for the next production run, it is enough to replace W_{ij} in (3) by $W_{ij} - A_{kl}$. To minimize workforce costs rather than size, the modification mentioned earlier applies.

Another objective for the cross-training problem may be one of hierarchical nature. It may be that the skill s_1 is in extremely short supply thus prompting a sequence of jobs that minimizes the number Z_1 of associated workers. Among the sequences that minimize Z_1, we may want to identify those that minimize Z_2, and so on, in a goal programming fashion. This mode of hierarchical optimization can be captured by *mCT* by appropriately selecting the cost values c_l. For instance, choosing $c_l = (2M)^{r-l}$ for $l = 1, 2, \ldots, r$, for big enough M (e.g., $M = \Sigma_{ij} W_{ij}$), forces the *mCT* formulation to hierarchically minimize the variables Z_1, $Z_2, \ldots, Z_r$.

For another extension of *mCT*, consider the assumption made in § 1.2 where it is assumed that all workers that possess a skill s_l are trained on every single station $ST_j \in A_l$. Equivalently, this assumption captures the scenario that the firm's cross-training program is completed. When the cross-training program is not completed, every worker associated with skill s_l is only trained on a subset of A_l. We refer to this scenario as *incomplete cross-training*. In §7.1 we describe how *mCT* can be used to prioritize efforts towards completing the cross-training program.

Also, in §1.2 we assumed that no two skill vectors have a station in common. In some applications this assumption may be cumbersome because: (i) existing workers may have already been trained in a skill not included in their newly determined skill vector, (ii) some skill vectors may include one or more generic skills common to all skill vectors, (iii) the cost to cross-train the workers of a particular skill vector to perform selected skills of different vectors may be negligible, and (iv) the per worker cross-training costs of a particular skill vector may be substantially greater than the corresponding costs for another skill. In the latter case the workforce requirements for the expensive skill vector can be alleviated by training workers of less demanding vectors to perform some of the easier tasks involved with the expensive skill vector. We refer to the above scenario as *overlapping skill vectors*. In §7.2 we describe how *mCT* can be used for this scenario.

7.1. Incomplete Cross-Training

Let $A_{l,1}, A_{l,2}, \ldots, A_{l,q}$ be the subsets of A_l associated with the current state of the cross-training program of the firm. Let also $M_{l,1}$ be the number of workers that have received training for all the stations in $A_{l,1}$. Similarly are defined the quantities $M_{l,2}, \ldots, M_{l,q}$. Given a sequence of jobs S, consider the transportation problem associated with the production cycle c_k on a network $N_k(S)$ with supply nodes $A_{l,1}, A_{l,2}, \ldots, A_{l,q}$, and demand nodes $ST_{j1}, ST_{j2}, \ldots, ST_{j_{|A_l|}}$, together with a dummy demand node d. Every supply node $A_{i,q'}$ is connected to all stations $ST_j \in A_{l,q'}$ for $q' = 1, 2, \ldots, q$. Also, the dummy node d is connected to all supply nodes. The supply amount for $A_{l,q'}$ is $M_{l,q'}$ for $q' = 1, 2, \ldots, q$. The demand of each node $ST_j \in A_l$ is set equal to $W_{jk}(S)$, where

$W_{jk}(S)$: the number of workers scheduled in S to work on station ST_j during the kth production cycle, $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n + m - 1$,

where S is the best among the sequences obtained by our algorithms for the *mCT* problem. Also, the demand of the node d is set equal to $\Sigma_{q'=1}^{q} M_{l,q'} - \Sigma_{j \in A_l} W_{jk}(S)$.

The solution of the above transportation problem indicates whether the current level of worker cross-training is adequate for the kth cycle, and identifies the station(s) that must be given priority to meet the workforce requirements associated with the sequence S for the production cycle c_k.

7.2. Overlapping Skill Vectors

Under this scenario the operations manager can use our algorithms after selecting a logical partition A of the m stations, and consequently form the distinct skill vectors s_l, $l = 1, \ldots, r$. Let S be the sequence of jobs obtained by our algorithms. This sequence determines the following quantities:

$W_{jk}(S)$: the number of workers that must work on station ST_j during the production cycle k according to schedule S, $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n + m - 1$.

$W_l(S)$: the number of workers required to possess the skill s_l according to S, $l = 1, 2, \ldots, r$.

Using the above values we would like to translate the workforce requirements of the mutually exclusive skill vectors s_l, to requirements for the overlapping skill vectors

$$o_i = (o_{l1}, o_{l2}, \ldots, o_{lm}),$$

$$l = 1, 2, \ldots, r', \text{ where } o_{lj} \in \{0, 1\}.$$

Hence, we seek the decision variables:

$W'_l(S)$; the number of workers possessing o_l required by sequence S, $l = 1, 2, \ldots, r'$.

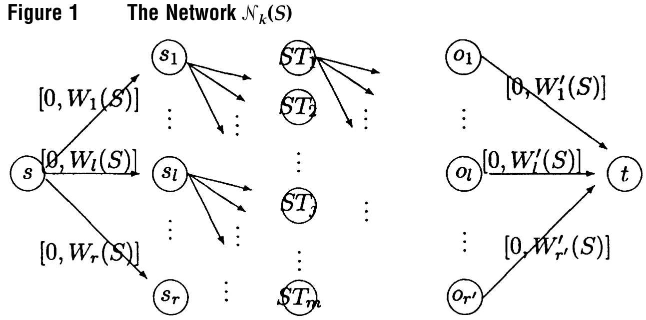
For this problem we propose using a hierarchical optimization approach by determining an order of importance for the overlapping skill vectors o_l (based on cost, duration of training, or other critical factors). As shown below, the hierarchical approach requires polynomial computational time. Indeed, for every period $k = 1, 2, \ldots, n + m - 1$, consider the network $\mathcal{N}_k(S)$:

$\mathcal{N}_k(S)$	lower bound	upper bound
(s, s_l)	0	$W_l(S)$
(s_l, ST_j) iff $s_{lj} = 1$	W_{jk}	W_{jk}
(ST_j, o_l) iff $o_{lj} = 1$	0	∞
(o_l, t)	0	$W'_l(S)$

Let us assume that the vectors $o_1, o_2, \ldots, o_{r'}$ are sorted in descending order of importance. We would like to find the minimum number $W'_1(S)$ of workers that possess skill o_1 and can accommodate the demands placed by S on the stations ST_j with $o_{1j} = 1$, over any production cycle c_k. Subsequently, we can identify the minimum number $W'_2(S)$ of workers that possess skill o_2 and can accommodate the demands placed by S on the stations ST_j with $o_{2j} = 1$ over any production cycle, and so on. The workforce sizes $W'_l(S)$ must be sufficient for every production cycle $k = 1, 2, \ldots, n + m - 1$. For this reason, we apply bisection search to find the minimum possible value of $W'_1(S) \in [0, W(S)]$ (where $W(S) = \Sigma_l W_l(S)$), say $W'_1(S)^*$, for which there exists a feasible solution for all networks $\mathcal{N}_1(S), \mathcal{N}_2(S), \ldots, \mathcal{N}_{n+m-1}(S)$. Upon finding $W'_1(S)^*$, we use bisection search on $W'_2(S) \in [0, W(S)]$, and so on, until all of $W'_1(S)^*, W'_1(S)^*, \ldots, W'_{r'}(S)^*$ are found.

For each trial value for $W'_l(S)$, we need to apply a feasible flow algorithm to all networks $\mathcal{N}_k(S)$ for $k = 1, 2, \ldots, n + m - 1$. A value $W'_l(S)$ is feasible only if there exists a feasible flow for all $\mathcal{N}_k(S)$'s. The computational complexity of the suggested procedure can be calculated as follows. A feasible flow (if one exists) in a network with lower and upper bounds may be obtained using the construction of Even (1973). He shows how to transform a network $\mathcal{N}$ with lower bounds into another network $\bar{\mathcal{N}}$ without lower bounds. From the maximum flow in $\bar{\mathcal{N}}$ one can easily determine a feasible flow for $\mathcal{N}$, when one exists. If $\mathcal{N}$ has v vertices and e edges then $\bar{\mathcal{N}}$ has $v + 2$ vertices and $e + 2v$ edges. Each of the networks $\mathcal{N}_k(S)$ described above has $\mathcal{O}(m)$ nodes, assuming that the number r' of overlapping skill vectors is no greater than m as usually happens in practice. Also, since each station is usually involved with a small number of skill vectors, we can assume that the number of arcs in $\mathcal{N}_k$ is $\mathcal{O}(m)$. The fastest known maximum flow algorithm for sparse networks without

Figure 1 **The Network $\mathcal{N}_k(S)$**



lower bounds is due to Sleator (1980) and has complexity $\mathcal{O}(|V||A|\log|V|)$ where $|V|$ and $|A|$ are the number of vertices and arcs respectively in the network. Hence, each application of the maximum flow algorithm on $\bar{\mathcal{N}}_k$ requires $\mathcal{O}(m^2 \log m)$ time.

Observe that for each trial value of $W_l'(S)$ from the range $[0, W(S)]$, $n + m - 1$ or $\mathcal{O}(n)$ applications of the maximum flow algorithm are required (assuming $n \geq m$). Therefore, the overall complexity of the procedure described above requires $\mathcal{O}(nm^2 \log m(\log W)^{r'})$ where $W = \Sigma_l W_l(S)$. Finally, note that instead of using this procedure for a single sequence S, one could apply it to a small subset of promising sequences identified by the algorithms presented in §5.

In this section we demonstrated that the formulation *mCT* is very general and can be adapted to capture cost or hierarchical objective functions, finite or infinite production horizons, cross-training programs that are in-progress or completed, and mutually exclusive or overlapping skill vectors. The purpose of this paper is not only to expose the specific manufacturing application that motivated this research, but also to compile a set of guidelines on how the level of cross-training and the partition of the assembly line into different skills affect the size of the workforce. For different costs (e.g., different salaries for employees of various skills), these guidelines will take a different form. For these reasons, we deemed *mCT* as the most appropriate model to carry our detailed analyses and experimentations.

8. Conclusion

In this paper we formulated the cross-training problem faced by a firm that has to execute a set of work orders. Worker cross-training has significant cost implications in labor intensive industries. We developed a host of lower bounds that helped the development and evaluation of exact and heuristic algorithms. We performed extensive computational experiments that showed that many of the algorithms developed exhibit near optimal performance. Using these algorithms we observed that small improvements in the level of cross-training of the workforce can significantly decrease the size of the workforce necessary to execute the orders. Also, we compared experimentally the size of a fully trained workforce required to execute a set of orders, against the size of a relatively untrained workforce. Our research serves not only as a means to quantifiably convince operations managers on the importance of worker training but also in designing efficient cross-training programs.

Extending this research, it would be useful to develop local search heuristics to compare their performance against the heuristics developed in this paper, on large problems with say $n = 100$ jobs and $m = 20$ stations.

Appendix

Proof of Theorem 1. Let S be an arbitrary sequence of jobs. Then, $W_S = \Sigma_l W_l(S)$ where $W_l(S)$ denotes the number of l-skill workers required by S. Similarly, $W^* = \Sigma_l W_l^*$ where W_l^* denotes the number of l-skill workers required by an optimal sequence S^*. Observe that

$$W_l(S) \leq \max_l W_l^* \sum_{j=1}^{m} s_{lj} \quad \text{for every } 1 \leq l \leq r.$$

This is because $W_l^* \geq s_{lj} W_{ij}$ for every combination of the indices i, j, l, and the fact that W_l is the sum of no more than $\Sigma_{j=1}^m s_{lj}$ terms. Hence,

$$W_S = \sum_{l=1}^{r} W_l(S) \leq \left(\sum_{l=1}^{r} \sum_{j=1}^{m} s_{lj}\right) \max_l W_l^*$$
$$\leq \max_l \left(\sum_{l=1}^{r} \sum_{j=1}^{m} s_{lj}\right) W_l^* = \max_l \sum_{j=1}^{m} s_{lj} W^*$$

which proves the first part of the theorem.

To see that the bound $L = \max_l \Sigma_{j=1}^m s_{lj}$ is tight, consider an arbitrary skill scenario. Let s_1 be a skill that involves precisely L stations. For simplicity, suppose that $s_1 = (1, 1, \ldots, 1, 0, \ldots, 0)$. Consider a job set of m jobs, where the first L are $J_1 = (M, 0, \ldots)$, $J_2 = (0, M, 0, \ldots)$, $\ldots$, $J_L = (0, \ldots, 0, M, 0, \ldots)$, and the remaining $m - L$ have the form $(0, \ldots, 0)$. For this instance of the *mCT* problem, the sequence starting with the jobs $J_1, J_2, \ldots, J_L$ in this order is optimal, and $W^* = M$. On the other hand, the sequence S starting with the jobs $J_L, J_{L-1}, \ldots, J_1$ in this order has $W_S = L \cdot M$ and hence $W_S/W^* = L$. This completes the proof of the theorem. □

References

Bartholdi, J. J. III. 1992. Balancing 2-sided assembly lines (a case study). Technical Report. IsyE Department. Georgia Institute of Technology.

——, D. D. Eisenstein. 1996. A production line that balances itself. *Oper. Res.* **44**(1) 21–34.

Daniels, R. L., J. B. Mazzola. 1993. A tabu-search heuristic for the flexible-resource flow shop scheduling problem. *Management Sci.* **41** 207–230.

——, B. J. Hoopes, J. B. Mazzola. 1997. An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. *Ann. Oper. Res.* **70** 439–472.

Dar-El, E. M., R. F. Cother. 1975. Assembly line sequencing for model-mix. *Internat. J. Production Research* **13**(5) 463–477.

Ebeling, A. C., C.-Y. Lee. 1994. Cross-training effectiveness and profitability in assembly line production. *Internat. J. Production Res.* **32** 2843–2859.

Even, S. 1973. *Algorithmic Combinatorics, Chapter* 10. Macmillan, New York.

Glover, F. 1975. Surrogate constraint duality in mathematical programming. *Oper. Res.* **23** 434–453.

Hsu, W.-L. 1984. Approximation algorithms for the assembly line crew scheduling problem. *Math. Oper. Res.* **9** 376–383.

Lee, C.-Y., G. Vairaktarakis. 1997. Workforce planning in mixed-model transfer lines. *Oper. Res.* **45**(4) 553–567.

Lin, S., B. W. Kernighan. 1973. An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* **21** 498–516.

Okamura, K., H. Yamashima. 1979. A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor. *Internat. J. Production Res.* **17**(3) 233–247.

Pinto, P. A., D. G. Dannenbring, B. M. Khumawala. 1981. Branch and bound and heuristic procedures for assembly line balancing with parallel stations. *Internat. J. Production Res.* **19** 565–576.

Sleator, D. D. 1980. Unpublished Ph.D. Dissertation, Stanford University.

Thomopoulos, N. T. 1967. Line balancing sequencing for mixed model assembly. *Management Sci.* **14**(2) 59–75.

Accepted by Timothy Lowe; received June 12, 1997. This paper has been with the authors for 8 months for 4 revisions. The average review cycle time was 45.4 days.