



Discrete Optimization

A multi-period machine assignment problem

Xinhui Zhang ^{a,*}, Jonathan F. Bard ^b

^a *Department of Industrial Engineering, Wright State University, 207 Russ Engineering Center, 3640 Colonel Glen Hwy, Dayton, OH 45435, United States*

^b *Graduate Program in Operations Research and Industrial Engineering, The University of Texas, ETC 5.160, C2200, Austin, TX 78712-1063, 512-471-3076, United States*

Received 22 November 2002; accepted 1 July 2004

Available online 11 September 2004

Abstract

In this paper, a multi-period assignment problem is studied that arises as part of a weekly planning problem at mail processing and distribution centers. These facilities contain a wide variety of automation equipment that is used to cancel, sort, and sequence the mail. The input to the problem is an equipment schedule that indicates the number of machines required for each job or operation during the day. This result is then post-processed by solving a multi-period assignment problem to determine the sequence of operations for each machine. Two criteria are used for this purpose. The first is to minimize the number of startups, and the second is to minimize the number of machines used per operation.

The problem is modeled as a 0–1 integer program that can be solved in polynomial time when only the first criterion is considered. To find solutions in general, a two-stage heuristic is developed that always obtains the minimum number of startups, but not necessarily the minimum number of machines per operation. In a comparative study, high quality solutions were routinely provided by the heuristic in negligible time when compared to a commercial branch-and-bound code (Xpress). For most hard instances, the branch-and-bound code was not able to even find continuous solutions within acceptable time limits.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Multi-period assignment problem; Machine scheduling; Total unimodularity; Heuristics

* Corresponding author.

E-mail addresses: xinhui.zhang@wright.edu (X. Zhang), jbard@mail.utexas.edu (J.F. Bard).

1. Introduction

The classical assignment problem (AP) has a wide range of applications that span manufacturing, personnel scheduling and transportation, to name a few. In generic terms, the problem seeks to minimize the cost of assigning a given set of jobs (tasks) to a group of machines (people). Computationally, the assignment problem is considered “easy” because it can be solved in polynomial time by several efficient algorithms. It falls into the general class of minimum cost network flow programs, all of whose A-matrices possess the total unimodularity property so the linear programming solution is guaranteed to be integral. For a detailed survey of the subject, see [Ahuja et al. \(1989\)](#).

In this paper, we study a variant of the AP that involves assigning jobs to machines over several time periods using two objectives to guide the search for solutions. The first is to minimize the number of startups or changeovers. A startup occurs whenever a new or different job begins processing on a machine. These transitions usually require extra time and effort, which translates into reduced capacity and increased labor costs. The second is to use as few machines per job as possible over the planning horizon, i.e., when two or more identical jobs are processed in different periods, we would like to assign them to the same machine whenever feasible. Although these two objectives do not conflict with each other they may not be perfectly aligned, so minimizing the number of startup, for example, does not necessarily minimize the total number of machines used.

The motivation for this problem originated from our work on equipment scheduling at United States Postal Service (USPS) processing and distribution centers (P&DCs). These facilities contain a wide variety of equipment for accepting, canceling, transporting, sorting and sequencing the mail. Although each type of equipment is designed to perform a basic task; for example, reading a barcode and sorting the mail piece to a destination, these machines must be reset as the mail type changes. The quality of the address, the degree of previous processing, and the destination are some of the characteristics that define the mail type. Each category of mail is processed as a separate job.

The input to the problem is a 7-day equipment schedule obtained by solving a large-scale mixed-integer linear program with multiple criteria. Job assignments must be made for each 1/2-hour period in a day once the equipment schedule is determined. In postal terminology, a job is called an *operation*. A full description of the equipment scheduling problem is given by [Zhang and Bard \(2005\)](#).

In a P&DC, the equipment is grouped into clusters of identical machines. As such, the scheduling model only provides the number of machines that should be running for each operation during the day, but does not specify the sequence of operations for each machine. To resolve this issue, the schedule must be post-processed by solving a multi-period assignment problem. [Aronson \(1986\)](#) considered a version of this problem designed to minimize (i) the cost of assigning a person to an activity, and (ii) the cost of transferring a person from one job to another. The latter was assumed to be sequence dependent. He presented an integer multi-commodity network flow model and developed a specialized branch and bound algorithm to find solutions. Instead of solving the linear programming relaxation, his idea was to solve a set of shortest path subproblems and branch on jobs that were assigned to more than one machine. [Gupta \(1995\)](#) investigated an asynchronous parallel algorithm to execute the same branch and bound scheme. In a similar vein, [Maxon and Bhadury \(2001\)](#) studied a multi-period assignment problem with repetitive tasks and tried to introduce a human element into the analysis. Their objective was to minimize a combination of the assignment cost and the “boredom” that results when workers are required to repeat the same task in consecutive periods. A mathematical model was proposed and a simple iterative heuristic was developed and implemented in Excel. For a discussion of traditional parallel machine scheduling with sequence-dependent setups, see, for example, [Kurz and Askin \(2001\)](#).

The multi-period assignment problem discussed in this paper differs from the above problems in two ways.

1. The assignment cost of a job to a machine is 1 because all the machines are identical.
2. The startup cost is sequence independent and similarly can be set to 1.

Condition (1), implies that the summation of the assignment costs is a constant over the planning horizon and so can be ignored, while (2) significantly reduces the number of binary variables that otherwise would have to be used to model the sequence-dependant setups in the integer programming model. Even with these simplifications, though, specific instances of the P&DC multi-period machine assignment problem can become quite large, and depending on the objective function, remain NP-hard. This has motivated us to develop two polynomial-time heuristics. For the primary objective of minimizing the number of startups, we show that the constraint region of the associated integer program is totally unimodular and that the proposed heuristic is, in fact, an exact procedure. We offer an extension for further minimizing the number of machines per operation that maintains the quality of the first solution. Because the constraint region of the combined problem is not totally unimodular, global optimality is not guaranteed, but the computations demonstrate the effectiveness of the approach.

The remainder of the paper is organized as follows. In the next section, we highlight the activities of a mail processing and distribution center to provide the context in which our multi-period assignment problem arises. In Section 3, we give a mathematical formulation of the bi-criteria problem, followed in Section 4 with the properties of the model and a description of the heuristics. Section 5 presents a computational comparison of the heuristics with a commercial branch-and-bound code (Xpress) using data provided by the Dallas P&DC. Some suggestions for future research are outlined in Section 6.

2. Description of a mail processing and distribution center

The USPS operates approximately 275 major P&DCs around the country that serve as hubs for collecting, processing, and dispatching the mail. On a typical day, a medium-size P&DC may handle over five million letters, each undergoing several operations before they are dispatched to the outside world. Only letters are considered in this paper. Flats are governed by similar processing concepts and are handled the same way, but with different equipment. The other categories are either not suited for automation or too small in volume to be of concern.

The automated processing of a letter follows three steps: (i) canceling the stamp (if one exists), (ii) reading the address and identifying the destination with a barcode, and (iii) sorting the letter to its final destination. Four types of equipment are currently deployed in P&DCs for letter processing: (1) an advanced face-canceller system (AFCS); (2) a multi-line optical character reader (MLOCR); (3) a remote barcode sorter (RBCS); and (4) a delivery barcode sorter (DBCS). Fig. 1 displays the major operations and mail flows associated with letters at the Dallas facility. Here, arcs (arrows) represent mail flows and nodes (blocks) represent the processing operations (the operation numbers and the equipment used to run these operations are also listed inside the blocks). An operation is considered major if it has more 10 hours of mail to be processed over the day. An arc is considered major if the output to the destination node consists of more than 15% of the total output at the originating node.

All mail first arrives at the P&DC's receiving area and is broken down into one of several pre-defined groups, based upon origination, physical size, quality of address, and degree of previous processing. Each group has a different composition and follows different routes (series of operations) through the facility. The major groups are (1) local collection mail gathered from the street boxes served by the P&DC, and (2) incoming mail partially processed at another P&DC and dispatched to the local P&DC for distribution.

The local collection mail can be classified as stamped, metered, pre-barcoded, and manual. The manual qualifier refers to letters that, due to peculiar physical characteristics (e.g. irregular shape), get culled as they

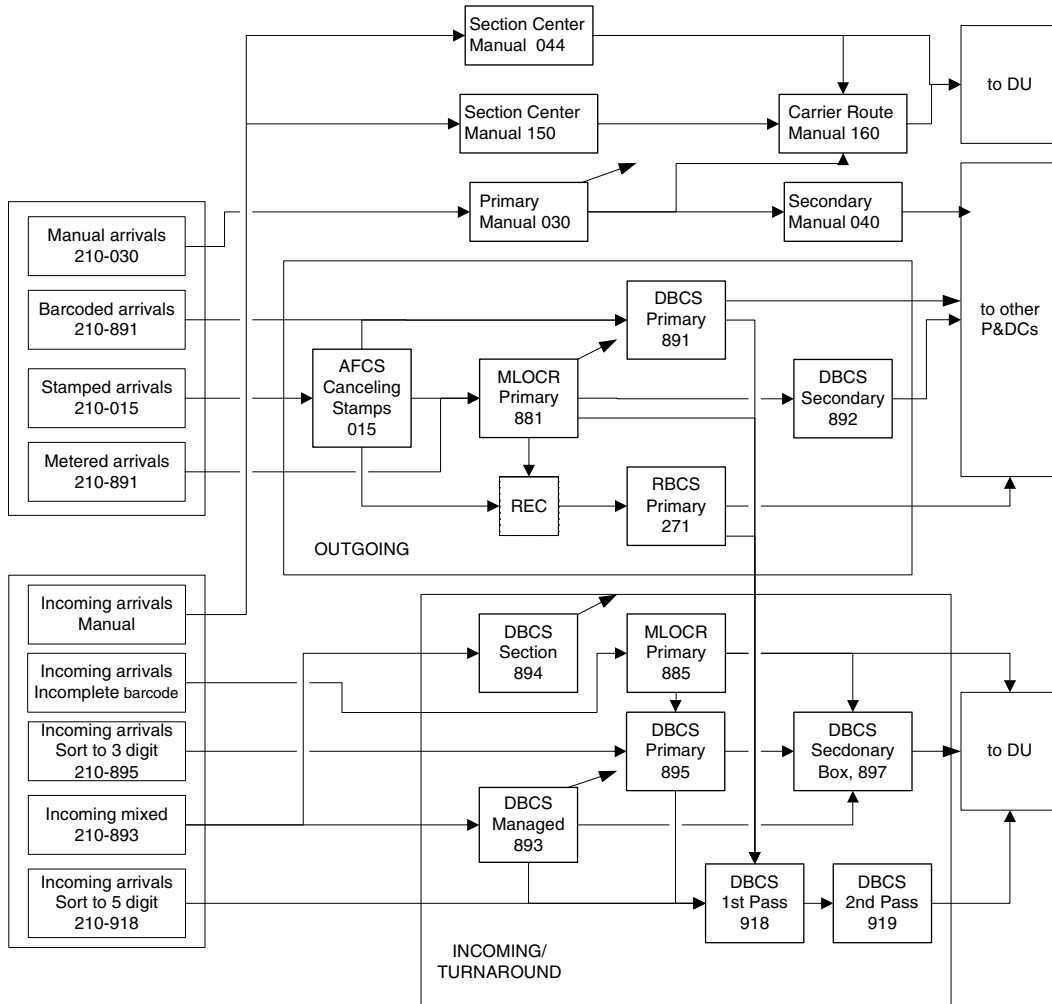


Fig. 1. Major mail flows in a P&DC.

travel through the belt feeding mail into the automation system. The stamped letters are first sent to the AFCS (advanced facer-canceller system), operation 015, where their stamps are located and canceled. Pre-barcoded letters, machine-printed envelopes and handwritten pieces are also separated at this stage for faster processing with the automation equipment.

The machine-printed envelopes as well as the metered arrivals are fed into MLOCRs (multiline optical character readers), operation 881, where the machines attempt to read the full address, and if successful, print a 3-, 5-, or 11-digit barcode on the envelope. The MLOCRs are also equipped with a limited number of bins to perform primary sorting. Output to certain other P&DCs is dispatched promptly while the remaining mail pieces require a secondary sorting, operation 892, on a DBCS (delivery barcode sorter) to channel them to more specific destinations.

The handwritten letters separated at the AFCS and machine-printed envelopes unreadable by an MLOCR are sprayed with a florescent ID on the back and sent to a RBCS (remote barcode sorter), operation 271, to obtain a barcode. While the letters are being transferred, video images of the address fields are

transmitted to a remote encoding center (the dashed REC block in Fig. 1) to be identified. The barcodes found are then retrieved and sprayed when the letters enter the RBCS to be sorted and dispatched.

The pre-barcoded arrivals with no stamps are sent directly to a DBCS, operation 891, to be sorted and dispatched. Finally, rejects that failed in the above operations, as well as the letters that are not fed into the automation system, are sent to the manual section, which consists of a primary (operation 030) and secondary (operation 040), to be separated and dispatched.

The local collection mail is also called outgoing mail because it is intended to be dispatched to other P&DCs. Outgoing mail sent from other P&DCs becomes the incoming mail for the local P&DC. There is also a certain amount of incoming mail that originates from the local service region and is called turnaround mail. Incoming and turnaround mail is mostly barcoded upon their arrival.

Mail sent to the local service area can arrive already sorted to either 3 or 5 digits. The 3-digit sorted letters are first sent to a DBCS, operation 895, to receive two additional digits. Then, together with the 5-digit sorted letters, are dispatched to firms through operation 897 or to delivery units (DUs) through delivery point sequencing operations. These operations are designed to reduce the workload of the carriers in the DUs by sorting the letters to each carrier route, operation 918, and then to walk order for each route, operations 919. The full letter automation process contains more than 20 different arrival streams, 40 operations, and more than 200 flows within the facility.

3. Mathematical formulation

The multi-period machine assignment problem under investigation decomposes by equipment type, i.e., AFCS, MLOCR, DBCS, DBCS and MANUAL, allowing us to address only one machine group at a time. The following notation is used in the development of the model.

Indices and sets

m index for machines; $m \in M$ (set of machines in a specific group)
 n index for operations (jobs); $n \in N$ (set of operations)
 t index for periods; $t \in T = \{1, 2, \dots, 48\}$

Data

$r_n(t)$ number of machines required for operation n during period t

Variables

$y_{mn}(t)$ 1 if machine m performs operation n in time period t ; 0 otherwise (it is assumed that $y_{mn}(0) = 0$ for all m and n)
 $z_{mn}(t)$ 1 if a startup of operation n occurs on machine m at the beginning of time period t ; 0 otherwise

Criterion 1 (Minimize the number of startups)

$$(P1) \quad \text{Minimize} \quad \sum_{m \in M} \sum_{n \in N} \sum_{t \in T} y_{mn}(t) [1 - y_{mn}(t-1)] \quad (\text{Objective 1})$$

$$\text{subject to} \quad \sum_{m \in M} y_{mn}(t) = r_n(t) \quad \forall n \in N, t \in T \quad (1a)$$

$$\sum_{n \in N} y_{mn}(t) \leq 1 \quad \forall m \in M, t \in T, \quad (1b)$$

$$y_{mn}(t) \in \{0, 1\} \quad \forall m \in M, n \in N, t \in T. \quad (1c)$$

The objective is to minimize the number of startups. Constraint (1a) ensures that all operations that are supposed to be running in period t are assigned to machines. Constraint (1b) specifies that a machine can be assigned to at most one operation in a single period, while constraint (1c) ensures integrality of the assignments. It is assumed that $r_n(0) = 0$ for all $n \in N$ implying that the facility is idle at $t = 0$. In the first period, exactly $r_n(1)$ machines must be turned on for operation n so there is no need to count the initial startups. This number is fixed at $\sum_{n \in N} r_n(1)$. Notice that $r_n(t)$ is the solution obtained from the equipment scheduling model and that $\sum_{n \in N} r_n(t) \leq |M|$ so the above problem is always feasible.

The objective function can be linearized with the addition of $O(|M| \cdot |N| \cdot |T|)$ variables and constraints as follows:

$$\begin{aligned}
 \text{(P1')} \quad & \text{Minimize} \quad \sum_{m \in M} \sum_{n \in N} \sum_{t \in T} z_{mn}(t) && \text{(Objective 1')} \\
 & \text{subject to} \quad (1a)–(1c) \text{ and} \\
 & \quad y_{mn}(t) - y_{mn}(t - 1) \leq z_{mn}(t) \quad \forall m \in M, n \in N, t \in T, && (1d) \\
 & \quad z_{mn}(t) \geq 0 \quad \forall m \in M, n \in N, t \in T. && (1e)
 \end{aligned}$$

Constraint (1d) and (1e) define the startup variables, $z_{mn}(t)$, which are treated as continuous. Because we are minimizing the sum of these variables, they will automatically be 0 or 1 in any optimal solution. Aronson (1986) compared this formulation with an integer multi-commodity network flow model under sequence dependant transfers and concluded that (P1') contained more constraints than the corresponding network flow model. However, for sequence independent transfers, the same argument does not hold because the number of constraints in (1d) is only around 1/2 of the corresponding number in the network flow model. For this reason, we have adopted (P1').

Criterion 2 (Minimize the number of machines used per operation)

$$\begin{aligned}
 \text{(P2)} \quad & \text{Minimize} \quad \sum_{m \in M} \sum_{n \in N} \max\{y_{mn}(t) : t \in T\} && \text{(Objective 2)} \\
 & \text{subject to} \quad (1a)–(1c).
 \end{aligned}$$

The objective is to minimize the sum of the maximum number of different machines used for each operation over the planning horizon. To linearize (P2), let β_{mn} be a binary variable indicating whether or not machine m performs operation n , and replace the ‘max’ term in the objective with β_{mn} . This leads to

$$\begin{aligned}
 \text{(P2')} \quad & \text{Minimize} \quad \sum_{m \in M} \sum_{n \in N} \beta_{mn} && \text{(Objective 2')} \\
 & \text{subject to} \quad (1a)–(1c) \text{ and} \\
 & \quad y_{mn}(t) \leq \beta_{mn} \quad \forall m \in M, n \in N, t \in T, && (1f) \\
 & \quad \beta_{mn} \in \{0, 1\} \quad \forall m \in M, n \in N, && (1g)
 \end{aligned}$$

where constraint (1f) records whether or not machine m performs operation n at time t . If $y_{mn}(t)$ is integral in a solution then β_{mn} will be integral as well so there is no need to explicitly specify the integrity requirement in constraint (1g).

Both objectives 1 and 2 must be considered when deriving solutions to the multi-period machine assignment problem. Fig. 2 illustrates the difficulty that may arise when solving each of these problems independently. In the example, an operation spans 10 periods with a requirement of two machines in periods 3, 4, 7 and 8, and 1 machine in all other periods. This is depicted on the “input” line in the figure. If we are only minimizing the number of startups, more machines than the minimum might be used, as shown in case 1. If we

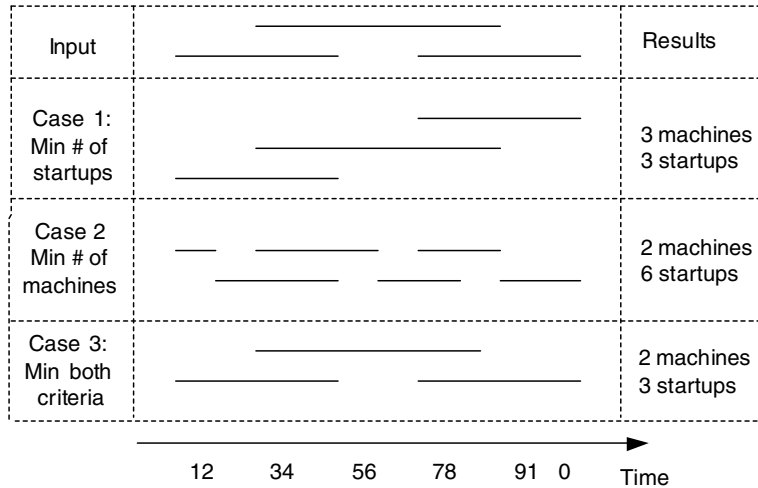


Fig. 2. Multiple solutions to problems (P1) and (P2).

are only minimizing the number of machines used, more startups than the minimum might occur, as shown in case 2. When both objectives are minimized, we achieve an assignment of 2 machines and 3 startups. This is shown in case 3.

Our real goal of first solving (P1') and then solving (P2') without allowing the number of startups to increase in the final solution, can be achieved by minimizing a weighted sum of the two objective functions.

Criterion 3 (Sequentially minimize the number of startups followed by the number of machines used per operation)

$$\begin{aligned}
 \text{(P3) Minimize} \quad & \sum_{m \in M} \sum_{n \in N} \beta_{mn} && \text{(Objective 3)} \\
 \text{subject to} \quad & \sum_{m \in M} \sum_{n \in N} \sum_{t \in T} z_{mn}(t) = z^* \\
 & \text{(1a)–(1f),}
 \end{aligned}$$

where z^* solves (P1'). Alternatively, we can replace (P3) with the following problem that combines the two objective functions.

$$\begin{aligned}
 \text{(P3') Minimize} \quad & \alpha \sum_{m \in M} \sum_{n \in N} \sum_{t \in T} z_{mn}(t) + (1 - \alpha) \sum_{m \in M} \sum_{n \in N} \beta_{mn} && \text{(Objective 3')} \\
 \text{subject to} \quad & \text{(1a)–(1f),}
 \end{aligned}$$

where the parameter $\alpha \in (0, 1)$ must be sufficiently close to 1 to assure that the minimum number of startups is obtained.

4. Model properties and algorithms

4.1. Models (P1) and (P1')

Let us first look at model (P1). For t fixed, the constraints (1a)–(1c) in (P1) are identical to those in an unbalanced minimum cost network flow problem on a bipartite graph, and hence, are totally unimodular

(TU). The periods are tied together by the objective function which is nonlinear and nonconvex even for $y_{mn}(t) \in [0, 1]$. As a result, one would not expect that the relaxed solution to (P1) or its equivalent linearization, (P1'), to be integral. The following proposition states otherwise.

Proposition 1. *If the integrality constraint (1c) in model (P1) is relaxed, at least one solution to the continuous version of the problem will be integral.*

Proof. The proof will be by induction on t . For $t = 1$, the objective function is $\sum_{m \in M} \sum_{n \in N} y_{mn}(1)$ due to the assumption that $y_{mn}(0) = 0$ for all m and n . Problem (P1) then reduces to minimizing a linear function over an integral polyhedron and so will always be integral when the simplex method is used. In actuality, the objective function is constant implying that all feasible solutions are optimal.

We now assume that the proposition is true for $t - 1$ and show that it is true for t . By the induction hypothesis, we know that there is an integral solution for periods $\tau = 1, \dots, t - 1$. Call it $y_{mn}^*(\tau)$. The current problem is

$$\begin{aligned} \text{Minimize} \quad & \sum_{m \in M} \sum_{n \in N} \sum_{\tau=2}^{t-1} y_{mn}(\tau)[1 - y_{mn}^*(\tau - 1)] + \sum_{m \in M} \sum_{n \in N} y_{mn}(t)[1 - y_{mn}^*(t - 1)] \\ \text{subject to} \quad & \sum_{m \in M} y_{mn}(\tau) = r_n(\tau) \quad \forall n \in N, \tau = 1, \dots, t, \\ & \sum_{n \in N} y_{mn}(\tau) \leq 1 \quad \forall m \in M, \tau = 1, \dots, t, \\ & y_{mn}(\tau) \in [0, 1] \quad \forall m \in M, n \in N, \tau = 1, \dots, t. \end{aligned}$$

To show that there is always an integral solution to this problem, it is sufficient to show that the principle of optimality applies; that is, the optimal solution for t forward does not change the optimal solution up to time $t - 1$. Assuming that this is true for the moment, using the induction hypothesis allows us to fix $y_{mn}^*(\tau)$ for $\tau = 1, \dots, t - 1$, and solve the linear program associated with period t , whose solution is, once again, integral.

To see why the principle of optimality holds, we propose a simple greedy procedure for solving the full problem.

1. If operation n uses machine j at time $t - 1$, then it is always optimal for operation n to continue to use machine j at time t unless operation n needs fewer machines.
2. When operation n needs fewer machines at time t , it releases arbitrarily the number of machines not needed.
3. When operation n needs more machines at time t , it selects new machines without preempting other operations now using the machines previously used by operation n .

The fact that this procedure provides an optimal assignment is based on a swapping argument. In particular, when operation n_1 changes from machine j_1 to machine j_2 at time t a startup at machine j_2 is incurred. A second startup is incurred when operation n_2 takes over machine j_1 . Therefore, it is better to assign operation n_2 to machine j_2 directly and avoid the startup associated with operation j_1 . \square

The swapping argument is similar to the one used to show that a “keep the tool needed soonest” policy minimizes the number of tool switches on a flexible machine with a finite tool magazine (e.g., see Bard, 1988). Note that the proof of Proposition 1 does not depend on the assumption that $y_{mn}(0) = 0$ for all m and n . In the next section, we provide an algorithm that implements the greedy procedure outlined in the proof.

The next question is whether or not there always exists an integral solution to (P1'), the linearized version of (P1). This is resolved below, but first we offer the following result.

Lemma 1. For $t \in T$, let

$$Y(t) = \left\{ \begin{array}{l} (y_{mn}(t), s_m(t), z_{mn}(t), w_{mn}(t)) : \sum_{m \in M} y_{mn}(t) = r_n(t), \quad \sum_{n \in N} y_{mn}(t) + s_m(t) = 1, \\ y_{mn}(t) - z_{mn}(t) + w_{mn}(t) = y_{mn}(t-1), \quad y_{mn}(t) \geq 0, \\ z_{mn}(t) \geq 0, \quad s_m(t) \geq 0, \quad w_{mn}(t) \geq 0 \quad \forall m \in M, n \in N \end{array} \right\}$$

be the feasible region of (P1') with the integrality condition relaxed and slack variables added to each of the constraints in (1b) and (1d). Then for $y_{mn}(t-1)$ fixed, $Y(t)$ is an integral polyhedron.

Proof. In matrix form, we have $Y(t) = \{(\mathbf{y}(t), \mathbf{s}(t), \mathbf{z}(t), \mathbf{w}(t)) : \mathbf{A}\mathbf{y}(t) + \mathbf{I}_{|M|}\mathbf{s}(t) = (\mathbf{r}(t), \mathbf{e}_{|M|})^T, \mathbf{I}\mathbf{y}(t) - \mathbf{I}\mathbf{z}(t) + \mathbf{I}\mathbf{w}(t) = \mathbf{y}(t-1), \mathbf{y}(t) \geq \mathbf{0}, \mathbf{z}(t) \geq \mathbf{0}, \mathbf{s}(t) \geq \mathbf{0}, \mathbf{w}(t) \geq \mathbf{0}\}$ where \mathbf{A} is an $(|M| + |N|) \times (|M| \cdot |N|)$ totally unimodular matrix, $\mathbf{I}_{|M|}$ is the $|M|$ -dimensional identity matrix, $\mathbf{e}_{|M|}$ is an $|M|$ -dimensional column vector of 1's, and \mathbf{I} is the $(|M| \cdot |N|) \times (|M| \cdot |N|)$ identity matrix. The system of constraints is

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{I}_{|M|} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{y}(t) \\ \mathbf{s}(t) \\ \mathbf{z}(t) \\ \mathbf{w}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{r}(t) \\ \mathbf{e}_{|M|} \\ \mathbf{y}(t-1) \end{pmatrix}, \tag{2}$$

where $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)^T$ such that \mathbf{A}_1 and \mathbf{A}_2 are $|M| \times (|M| \cdot |N|)$ and $|N| \times (|M| \cdot |N|)$ dimensional matrices, respectively. Now, if \mathbf{A} is totally unimodular, then so is $(\mathbf{A}, \mathbf{I})^T$, (\mathbf{A}, \mathbf{I}) and $(\mathbf{A}, -\mathbf{I})$, where \mathbf{I} has either the same number of columns or the same number of rows as \mathbf{A} , depending on its location (see Nemhauser and Wolsey, 1988; Chapter III.1). It can also be shown that if $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)^T$ is TU then so is $\begin{pmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{I} \end{pmatrix}$.

By applying these arguments in the appropriate order, we conclude that the matrix in Eq. (2) is TU, hence $Y(t)$ is an integral polyhedron. \square

Note that if $y_{mn}(t-1)$ is not fixed, the resultant matrix is not totally unimodular. Now, using the same induction proof that we used in Proposition 1, we can show the following.

Proposition 2. If the integrality constraint (1c) in model (P1') is relaxed, at least one solution to the continuous version of the problem will be integral.

The implication of this result is that the linear programming relaxation of (P1') will yield a solution to the multi-period assignment problem that minimizes the number of startups. Furthermore, the optimal value to models (P1) and (P1') is a constant and can be achieved by a simple construction algorithm rather than LP. The algorithm is presented fully below.

Algorithm 1 (Minimize_Startups)

Step 1: Set $r_n(0) = 0$ for all $n \in N$, and set $t = 1$.

Step 2: While ($t \leq 48$)

2.1 For each operation n , do the following: choose randomly $\min\{r_n(t), r_n(t-1)\}$ out of the $r_n(t-1)$ machines that were assigned operation n in the previous period $t-1$ and assign them operation n .

2.2 For each operation n not fully covered, i.e., $r_n(t) \geq r_n(t-1)$, do the following: select *randomly* $r_n(t) - r_n(t-1)$ from the unassigned machines and assign them operation n .

2.3 Update $t \leftarrow t + 1$.

Step 3: Stop.

As we can see, this algorithm follows precisely the argument used in the proof of Proposition 1. To reduce the number of startups, at time period t , an operation is assigned to those machines that were assigned the same operation in the previous period $t-1$. More specifically, when operation n needs fewer machines, it releases arbitrarily the number of machines not needed, this is done at step 2.1. When operation n needs more machines, it selects new machines without preempting other operations now using the machines previously used by operation n . This is done at step 2.2. The algorithm starts at period $t=1$ and continues until all periods have been considered. It has complexity $O(|M| \cdot |N| \cdot |T|)$ and is easy to implement. More importantly, though, it solves (P1') optimally.

Lemma 2. *Algorithm 1 provides an optimal solution to the multi-period machine assignment problem (1a)–(1d) with Objective 1' giving a schedule with the minimum number of startups.*

Proof. First, we provide a simple yet tight lower bound on the number of startups. To do this, let us sum all constraints in (1d) over $m \in M$. This leads to

$$\sum_{m \in M} z_{mn}(t) \geq \sum_{m \in M} y_{mn}(t) - \sum_{m \in M} y_{mn}(t-1) = r_n(t) - r_n(t-1),$$

where the latter equality follows from constraint (1a). Now let $Z_n(t) = \max\{0, r_n(t) - r_n(t-1)\}$ and we have $\sum_{m \in M} z_{mn}(t) \geq Z_n(t)$ implying that $\sum_{t \in T} \sum_{n \in N} Z_n(t)$ is a lower bound on the total number of startups. Second, we show that by construction, the algorithm achieves this lower bound, that is, only $Z_n(t)$ startups are incurred for operation n at time period t . In fact, this readily follows because startups only occur in step 2.2 and the number is exactly $\max\{0, r_n(t) - r_n(t-1)\}$, or $Z_n(t)$ defined above. This completes our proof. \square

4.2. Models (P2) and (P2')

We now address models (P2) and (P2'). When the integrality constraint is relaxed, (P2) reduces to minimizing a piecewise linear function over a (integral) polyhedron. Solutions to such problems generally do not occur at an extreme point of the polyhedron. As a consequence, there is no guarantee that the relaxed solution will be integral. Looking at the linearized version of the problem, (P2'), we note that the addition of constraint (1e) destroys the otherwise total unimodularity of the \mathbf{A} -matrix associated with (1a) so solving the LP is not likely to yield integral values for $y_{mn}(t)$ and β_{mn} . Nevertheless, there are some simple cases that will be stated without proof.

Lemma 3. *For model (P2):*

1. *When there is only one operations (i.e., $|N| = 1$), the problem of minimizing the number of machines per operation can be solved by a greedy algorithm that allocates as much work as possible over the time horizon to machine 1, then to machine 2, and so on.*
2. *Let R_n be the minimum number of machines required for operation n ; i.e., $R_n = \max\{r_n(t) : t \in T\}$ and let $R^* = \sum_{n \in N} R_n$. If $R^* \leq |M|$, then (P2) can be decomposed by operation and the greedy algorithm applied to each of the $|N|$ subproblems to obtain the optimal solution.*

3. When $|N| = 2$ and $R^* > |M|$, the greedy algorithm applied to operations 1 and 2 separately and the two solutions then merged with some additional logic will minimize the number of machines per operation.

When $|N| \geq 3$, it is relatively easy to construct examples that demonstrate the failure of the greedy algorithm to find the optimum. When $R^* > |M|$, we have the following conjecture.

Conjecture 1. The problem of minimizing the number of machines per operation given by model (P2) is NP-hard.

To have a means of measuring the quality of any feasible solution, we notice that the optimal objective function value has a fixed lower bound. In particular, $\sum_{m \in M} \beta_{mn}$ can be no less than $\max_{t \in T} \{r_n(t)\}$, which implies that $\sum_{n \in N} \sum_{m \in M} \beta_{mn} \geq \sum_{n \in N} \max_{t \in T} \{r_n(t)\}$. In addition, we have the following.

Proposition 3. Let Y be the feasible region of (P2') with the integrality constraint (1c) relaxed, and let $s_m(t)$ and $u_{mn}(t)$ be the slack variables added to (1b) and (1e), respectively, so that the problem is in equality form. Then there exists an optimal integer solution, call it $\{y_{mn}^*(t), \beta_{mn}^*, s_m^*(t), u_{mn}^*(t)\}$, that is an extreme point of Y .

Proof. If $\beta_{mn} = \beta_{mn}^*$ is fixed at 0 or 1 for all m and n , (P2') decomposed by time period t , and the corresponding feasible regions (1a)–(1c) form separate integral polyhedra. Therefore, each relaxed solution to subproblem t , call it $\{y_{mn}^*(t), s_m^*(t), u_{mn}^*(t)\}$, is integral and basic. If we consider the full problem with β_{mn} not fixed, the solution $\{y_{mn}^*(t), \beta_{mn}^*, s_m^*(t), u_{mn}^*(t)\}$ is also integral and basic. This follows because the number of constraints remains the same and β_{mn}^* can be taken to be nonbasic at either its upper or lower bound. This completes our proof. \square

This result implies that it may be possible to find an optimal assignment for (P2') by solving its relaxation. Our computational experience presented in Section 5 indicates that many instances solve quickly but most require some branching. The last problem that we wish to address is (P3). As a corollary to Propositions 2 and 3 we have the following.

Corollary 1. Let Y be the feasible region of (P3) with the integrality constraint (1c) relaxed, and that $s_m(t)$, $w_{mn}(t)$ and $u_{mn}(t)$ are the slack variables added to (1c)–(1e), respectively, so that the problem is in equality form. Then there exists an optimal integer solution, call it $\{y_{mn}^*(t), \beta_{mn}^*, s_m^*(t), u_{mn}^*(t), w_{mn}^*(t)\}$, that is an extreme point of Y .

4.3. Models (P3) and (P3')

In multiple objective optimization terminology, model (P3) represents a hierarchical or pre-emptive approach to a two-criteria problem, while model (P3') represents an Archimedean approach that involves a linear aggregation of the two criteria. In general, these models will not yield the same solution.

By letting α range between 0 and 1, it is possible to generate what are called the supported nondominated solutions to (P1') and (P2') (Tuytens et al., 2000). For example, consider a multi-period machine

Table 1
Sample problem with nondominated solutions

t	1	2	3	4	5
$r_1(t)$	1	2	3	1	1
$r_2(t)$	2	0	0	2	0
$r_3(t)$	1	0	0	0	1
$r_4(t)$	0	0	2	0	2
$r_5(t)$	1	3	0	2	1

Table 2
Solution for different settings of α

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Startups	18	18	18	18	18	18	18	20	20	20	
Machines per operation	–	12	12	12	12	12	12	11	11	11	11

Table 3
Nondominated solutions for different settings of α for model (P3')

Time	$\alpha = 0.6$					$\alpha = 0.7$				
	1	2	3	4	5	1	2	3	4	5
m_1	2	5	1	5	5	1	5	1	5	5
m_2	1	<i>1</i>	1	1	1	2	1	1	2	1
m_3	3	1	1	2	3	3	1	1	1	3
m_4	2	5	4	2	4	2	5	4	2	4
m_5	5	5	4	5	4	5	5	4	5	4

assignment problem with five machines, five operations, and five periods. The number of machines required for each operation is shown in Table 1 and solutions for various settings of α are reported in Table 2. Putting $\alpha = 0$ will give the optimal solution to (P1') and putting $\alpha = 1$ will give the optimal solution to (P2'). The corresponding values are 18 startups and 11 machines per operation, respectively. Finally, the full solutions for two settings of α ($\alpha = 0.6$ and $\alpha = 0.7$) are presented in Table 3. These values were selected because that is where the solution changes. Each entry in Table 3 represents the particular operation assigned to a particular machine (m_1, \dots, m_5) in time period t .

A closer examine of the solutions with the settings $\alpha = 0.6$ and $\alpha = 0.7$ reveals that while certain assignment sequences remain the same in both cases (indicated in bold), when $\alpha = 0.7$, in order to put operation 2 (in italic) on a single machine (m_2), an additional setup of operation 1 is incurred. Nevertheless, the two criteria are rather closely aligned. Intuitively, additional setups would be required if operations were assigned to more machines than necessary. For the instances that we examined that were solvable with a commercial branch-and-bound code, all values of $\alpha \in (0,1)$ yielded the minimum number of startups and machines. In view of this, in our implementation, rather than solve (P1') first and then (P3), we decided to solve (P3'), minimizing a weighted sum of the two objective functions directly by placing equal weight on each term (i.e., $\alpha = 1/2$).

4.4. Size of model (P3')

The size of the multi-period machine assignment model (P3') is governed by the number of variables and constraints in (1d) and (1e) which are functions of the number of machines, the number of operations, and

Table 4
Size of an instance of the machine assignment problem

Measures	MLOCR	DBCS	Manual
No. of machines, $ M $	8	25	120
No. of operations, $ N $	3	10	8
No. of periods, $ T $	48	48	48
No. of constraints	3966	24,264	114,971
No. of variables	576	7350	38,940
Time to solve LP (seconds)	0.51	11.37	681

the number of periods. Table 4 gives an indication of how large (P3') is, in practice. Also included are the computation times needed to solve the linear programming relaxation for machine groups MLOC, DBCS and Manual. These were obtained with the mixed-integer linear programming solver that comes with Xpress (Dash Optimization, 2002). Note that the size of the Manual group is determined by the maximum number of persons required in any time period. These workers are responsible for casing the mail, that is, placing each letter in a designated slot (or box) that is part of a cluster. In the test, equal weight was placed on the two objective function terms ($\alpha = 1/2$).

As discussed in Section 5, (P3') solves quickly with Xpress for instances that contain as many as 7000 variables and 50,000 constraints. However, the size of the problem grows sharply with the number of machines and operations. For the Manual group with a maximum of 120 workers, problem instances contain as many as 38,940 variables and 114,971 constraints. Compared to the DBCS group which is about 1/5 this size, the time spent on solving the LP relaxation increased dramatically from 11.37 seconds to 681 seconds, almost a factor of 60 increase even though the number of machines only increased by less than a factor of 5, from 25 to 120. Because the multi-period assignment problem must be solved quickly in the postal application, we felt that a heuristic was called for to reduce the worst-case run times.

4.5. An extension to Algorithm 1 to minimize the number of machines used per operation

Algorithm 1 allows total flexibility to incorporate other considerations in the multi-period assignment problem, such as balancing workloads among machines and limiting the number of machines used per operation. These considerations can be taken into account with a few simple modifications. We demonstrate this for the case of minimizing the number of machines used per operation.

First, as in Algorithm 1, at time period t , we require that an operation be assigned to machines that were assigned the same operation in the previous period $t - 1$ to guarantee the minimum number of startups. Second, when assigning an operation we give preference to machines that have been assigned to this operation in one or more previous periods.

In addition, whenever a tie exists, instead of selecting machines randomly, we choose machines whose assigned operations in previous periods are *least likely* to continue beyond the current period t . To quantify this idea, let $f(m)$ be a measure of the likelihood that the operations assigned to machine m in all previous periods will continue in the future. If this value is relatively high, then more likely machine m will be assigned to some of these operations in future periods. Thus, we would like to assign machine m last or leave it unassigned in the current period so that the operation that it is now performing will not be carried over and interfere with its assignments in future periods.

To estimate this likelihood for each machine m , let $N'(m, t)$ be the set of operations machine m was assigned during periods $1, \dots, t - 1$. The following merit functions were evaluated.

- (1) $f(m) = \sum_{n \in N'(m, t)} I(n)$ where $I(n) = 1$ if $r_n(\tau) > 0$ for any period $\tau \geq t$; 0 otherwise.
- (2) $f(m) = \sum_{n \in N'(m, t)} \sum_{\tau \geq t+1} r_n(\tau)$.
- (3) $f(m) = \sum_{n \in N'(m, t)} \max\{r_n(\tau) : \tau \geq t + 1\}$.

Function (1) records the number of operations that were assigned to machine m in previous periods and still need to be assigned in future periods. Function (2) calculates the number of machines required in future periods for operations assigned to machine m previously, and function (3) keeps track of the maximum number of machines required in any future period for operations assigned to machine m previously. These functions are defined for each machine m and are updated in each period t as the algorithm progresses. The second function (2) provided the best results in virtually all test cases, and was adopted in our implementation.

Algorithm 2 (Minimize_Startups_and_Operations_per_Machine).

Step 1: Set $r_n(0) = 0$ for all $n \in N$, evaluate the merit score $f(m)$ for all $m \in M$ and let $t = 1$.

Step 2: While ($t \leq 48$)

(2.1) For each operation n , do the following.

(a) Find the $r_n(t - 1)$ machines that were assigned operation n in the previous period.

(b) Rank the $r_n(t - 1)$ machines in ascending order according to their merit score $f(m)$ and assign operation n to the first $\min\{r_n(t), r_n(t - 1)\}$ of them and release the remaining machines from operation n .

(c) Let $r'(n) = \max\{0, r_n(t) - r_n(t - 1)\}$ be the number of unassigned jobs.

(2.2) For each operation n not fully covered, i.e., $r_n(t) - r_n(t - 1) \geq 0$, do the following.

(a) Find unassigned machines that have been assigned to operation n in previous periods $1, \dots, t - 1$. Suppose there are $r_p(n)$ of them.

(b) Rank them in ascending order of their merit score $f(m)$ and assign the first $\min\{r'(n), r_p(n)\}$ machines operation n .

(c) Update $r'(n) = r'(n) - \min\{r'(n), r_p(n)\}$ as the number of unassigned jobs.

(2.3) For each operation n that is still not covered, i.e., $r'(n) > 0$, do the following: find all remaining machines that are idle, rank them in descending order of their $f(m)$, select the first $r'(n)$ and assign them operation n .

(2.4) Update the merit score $f(m)$ for all $m \in M$ and set $t = t + 1$.

Step 3: Stop.

Of course, this algorithm is only a heuristic for (P3'). Both (P2') and (P3') are believed to be NP-hard so it is not likely that a polynomial time algorithm can be devised for either problem.

One possible extension to Algorithm 2 would involve the development of an exchange procedure to reduce the number of machines used per operation while keeping the number of startups constant. If we define a neighborhood as consecutive periods during which an operation is scheduled on a specific machine, improved solutions may be found by swapping machine assignments. Maintaining the minimum number of startups, however, is likely to limit the effectiveness of such a procedure.

5. Computational results

A set of experiments was conducted to evaluate the effectiveness of the proposed algorithms. As mentioned, both algorithms achieve the minimum number of startups so our main objective was to evaluate the number of different machines used per operation. As such, Algorithm 2, referred to as the heuristic, was the only one tested. For comparative purposes, we also solved problem (P3') with Xpress. In the computations, the preference weight α was set to 0.5 in each instance.

All models were implemented with Mosel, the modeling language provided by Dash Optimization, and solved with their general linear and integer programming software. The computations were performed on a 1.13 GHz Pentium III PC with 256 MB of RAM.

Input data in the form of number of machines required for each operation per period was generated from the equipment scheduler (Zhang and Bard, 2005) using data obtained from the Dallas facility. We ran that code for each day of the week and then solved the corresponding multi-period assignment problems.

Actually, the equipment scheduler solves a series of similar problems, with the final one having the objective of minimizing a weighted sum of the number of startups and length of machine operations. Accordingly, the schedule generated at the final stage usually contains many fewer startups than the solution

obtained at the initial stage. For the multi-period assignment problem, these schedules are relatively easy to handle and are referred to as *easy problems*, while the schedules generated at the initial stage are harder to handle and are referred to as *hard problems*.

The results for two representative machine groups, DBCS and Manual for both easy and hard problems, are presented below. Before discussing the results, though, we take a second look at the lower bounds on the number of startups and the number of machines per operation, respectively, derived in Section 4.

5.1. Lower bounds

1. $\sum_{n \in N} \sum_{t \in T} \max\{0, r_n(t) - r_n(t-1)\}$, lower bound on the number of startups.
2. $\sum_{n \in N} \max_{t \in T} \{r_n(t)\}$, lower bound on the number of machines per operation.

As discussed in the proof of Algorithm 1, the lower bound on the number of startups is the tightest bound possible. Both algorithms are designed to find it. The lower bound on the number of machines per operation, however, can be different than the optimum. Table 5 illustrates such a case. In the table, we see that the maximum number of machines for operations 1, 2 and 3 are each one. However, operation 2 has to be performed on two machines, m_1 and m_2 , because it is not possible to arrange operations 1 and 3 on a single machine due to their overlap. Thus, the lower bound is 3, while the optimal assignment is 4. It has been observed that the relaxed LP solution to (P3) always achieves both lower bounds.

5.2. Results for easy problems

Tables 6 and 7 present the number of startups and the number of machines used per operation obtained with the heuristic and the B&B code for DBCS and Manual, respectively. The DBCS group contains 25 machines while the size of the Manual group is a function of the day of the week. The columns labeled “time” are in units of seconds, and the columns labeled “# of nodes” refer to the number of nodes explored in the B&B tree to reach the IP solution. In Table 7, the column labeled “Size of Manual” indicates the maximum number of workers, $|M|$, needed on a particular day to perform the manual operations. This number is obtained from the equipment scheduler and is only provided as a point of reference. Recall that the size of the MILP is $O(|M| \cdot |N| \cdot |T|)$. In the experimental design, the B&B code was run until the first feasible solution was found and then for up to 10 additional minutes if necessary.

As we see in Tables 2 and 3, both the heuristic and the B&B code achieved the lower bounds in 13 of the 14 cases, indicating that the solutions obtained are provable optimal for all but one instance. The only discrepancy was on Saturday in the case of the DBCS where both methods indicated that 18 machines per operation were needed, compared to the lower bound of 17. The situation depicted in Table 5, however, was present on Saturday so the optimum actually is 18. On average, the heuristic was more than 100 times faster than the B&B code in the DBCS case, and nearly 2000 times faster in the Manual case. Note that the number of startups is not reported for the heuristic because the minimum number is guaranteed.

A second observation is that in 12 out of the 14 instances, the optimal solution was obtained at the root node of the B&B tree when the IP was solved. This confirms Corollary 1 and suggests that the LP relaxation gives a tight formulation. For the Saturday DBCS schedule, a feasible solution was obtained at the 18th

Table 5
A case where the second lower bound is not tight

t	1	2	3	4	5	6	7	8	9	10
m_1	2	2	2	0	3	3	3	3	3	0
m_2	0	1	1	1	1	1	0	2	2	2

Table 6
Comparative results for easy problem (DBCS)

Day	Lower bound		Heuristic		Branch and bound (P3')				
	# Of machines	# Of startups	# Of machines	Time (seconds)	# Of machines	# Of startups	LP time	# Of nodes	IP time
Mo	74	86	74	0.15	74	86	13	1	13
Tu	73	88	73	0.21	73	88	15	1	15
We	71	78	71	0.16	71	78	19	3	23
Th	77	89	77	0.14	77	89	18	1	18
Fr	75	82	75	0.18	75	82	21	1	21
Sa	17	40	18	0.15	18	40	4	18	36 ^a
Su	62	67	62	0.13	62	67	2	1	28

^a No improvement after 2 hours.

Table 7
Comparative results for easy problem (manual)

Day	Size of manual	Lower bound		Heuristic		Branch and bound (P3')				
		# Of machines	# Of startups	# Of machines	Time (seconds)	# Of machines	# Of startups	LP time	# Of nodes	IP time
Mo	120	131	135	131	0.28	131	135	504	1	504
Tu	107	118	129	118	0.37	118	129	771	1	771
We	108	122	128	122	0.37	122	128	657	1	657
Th	109	121	125	121	0.38	121	125	569	1	569
Fr	117	127	131	127	0.38	127	131	588	1	588
Sa	53	58	59	58	0.23	58	59	117	1	517
Su	59	65	67	65	0.24	65	67	132	1	87

node but did not improve after 2 hours of computations. The lower bound stayed the 17 and never increased during this time. This, we believe, is due to the symmetry in the model. Because all machines are identical, branching on one machine simply leads to a similar fractional solution in which another machine is assigned to process two or more operations concurrently.

A third observation is that an extremely large amount of time is needed to find a solution at the root node of the B&B tree for the Manual group. Similar results were obtained with CPLEX so we do not feel that the optimizer is the issue, but rather the problem size (see Table 4). If more than a few nodes have to be

Table 8
Comparative results for difficult problem (DBCS)

Day	Lower bound		Heuristic			Branch and bound (P3')			
	# Of machines	# Of startups	# Of machines	Gap%	Time (seconds)	# Of machines	LP time	# Of nodes	IP time
Mo	106	161	112	5.6	0.15	106	56	8	86
Tu	109	157	114	4.6	0.16	109	53	9	90
We	138	273	168	21.7	0.16	^a	105	^a	^a
Th	115	190	130	13.0	0.15	^a	181	^a	^a
Fr	159	300	178	11.9	0.15	^a	53	^a	^a
Sa	96	154	99	3.1	0.19	96	23	10	37
Su	102	146	103	0.9	0.18	102	4	3	5

^a No integer feasible solution in 30 minutes.

Table 9
Comparative results for difficult problem (manual)

Day	Size of manual	Lower bound		Heuristic			Branch and bound (P2')		
		# Of machines	# Of startups	# Of machines	Gap%	Time (seconds)	LP time	# Of nodes	IP time
Mo	268	396	830	409	3.3	0.98	706	^a	^a
Tu	198	336	627	365	8.6	0.68	746	^a	^a
We	236	322	667	328	1.9	0.81	1460	^a	^a
Th	229	375	824	392	4.5	0.75	1262	^a	^a
Fr	220	398	926	452	13.6	0.76	828	^a	^a
Sa	335	630	1407	635	0.8	1.10	838	^a	^a
Su	150	193	477	249	29.0	0.52	583	^a	^a

^a No solution beyond root node in 30 minutes.

explored, the time spent on solving the LP, coupled with the symmetry in the model, implies that standard B&B may not be practical. This was confirmed when we tried to solve the hard problem instances.

5.3. Results for hard problems

Table 8 gives the computational result for the hard problems associated with the DBCS group. The “Gap” column refers to the gap between the heuristic solution and the lower bound for the number of machines per operation. Once again, the heuristic and the B&B code always found the optimal number of startups so those results are not reported.

From the data in Table 8, we see that on average, the heuristic obtained a feasible solution within 8% of the lower bound for all seven problems within 0.2 second. For those cases in which it was possible to compare the results with the optimal solution, the average gap was only around 3.5%. The performance of the B&B code, on the other hand, was much less impressive. Feasible integer solutions were obtained in only three of the seven cases within the 30-minute time limit.

For the Manual group, the B&B code could not solve a single instance within the allotted time. The smallest problem resulted from the Sunday schedule, and for (P3') contains 279,203 rows, 119,528 columns, and 551,544 nonzero elements. After 40 minutes, no LP solution was uncovered so the computations were halted. To get an idea of the minimum number of machines required, we attempted to solve (P2') instead of (P3'). The former model contains roughly 1/2 the number of variables as the latter, but despite this reduction, the results were still disappointing.

Table 9 presents the computational results for our attempt to solve (P2') for the manual group. As can be seen, for the Sunday problem, it still took 583 second to solve the LP relaxation at the root node. What is more, no feasible solution beyond the root node was obtained within the 30 minutes allotted, so we decided to terminate the computations. Similar results were realized for the other cases.

The results for the heuristic, on the other hand, were quite satisfactory. Solutions were usually found within 1 second and, on average, were no more than 8.8% from the lower bound. Because the lower bound is often below the optimal solution, the optimality gap could actually be much smaller than indicated.

6. Discussion

The focus of this paper has been on a specialized version of a multi-period machine assignment problem that arose when scheduling equipment at US mail processing and distribution centers. Because startups are sequence independent and the assignment costs are unitary in the basic model, we were able to show that the corresponding integer program can be solved in polynomial time. A construction algorithm was

presented to find the minimum number of startups. For the more general case where the objective was to minimize both the number of startups and the number of machines used per operation, optimal solutions could no longer be obtained by solving the equivalent of a linear program. A heuristic, based on the construction algorithm was developed for this purpose.

The heuristic was compared with a commercial branch-and-bound code on two sets of data. For the easy problems, both methods performed well, although the heuristic was faster by two to three orders of magnitude. This is particularly important in an operational environment where many instances have to be solved within a matter of minutes. For the hard problems, the branch-and-bound code proved to be incapable of solving all but a few instances. As a consequence, the heuristic was selected for implementation as part of a decision support system currently under development for the US Postal Service.

References

- Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1989. Network flows. In: Nemhauser, G.L., Rinnooy Kan, A.H.G., Todd, M.J. (Eds.), *Handbooks in Operations Research and Management Science, Optimization*, vol. 1. North-Holland, Amsterdam, pp. 211–369.
- Aronson, J.E., 1986. The multiperiod assignment problem: A multicommodity network flow model and specialized branch and bound algorithm. *European Journal of Operational Research* 23 (3), 367–381.
- Bard, J.F., 1988. A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions* 20 (4), 382–391.
- Dash Optimization, 2002. *Xpress-Mosel: User Guide*, Englewood Cliffs, NJ.
- Gupta, B., 1995. A parallel multiperiod assignment algorithm. Ph.D. Dissertation, University of Georgia, Athens, GA.
- Kurz, M.E., Askin, R.G., 2001. Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research* 39 (16), 3747–3769.
- Maxon, S.L., Bhadury, J., 2001. An ms-excel implementation of a multi-period assignment problem with repetitive tasks. In: *Proceedings of the 13th Annual CSU-POM Conference*, California State University San Bernardino, pp. 39–48.
- Nemhauser, G.L., Wolsey, L.A., 1988. *Integer and Combinatorial Optimization*. John Wiley and Sons, NY.
- Tuytens, D., Teghem, J., Fortemps, P., Nieuwenhuyze, K.V., 2000. Performance of MOSA method for the bicriteria assignment problem. *Journal of Heuristics* 6, 295–310.
- Zhang, X., Bard, J.F., 2005. Equipment scheduling at USPS processing and distribution centers. *IIE Transactions on Scheduling and Logistics*.