M&SOM MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 10, No. 1, Winter 2008, pp. 24–42
ISSN 1523-4614 | EISSN 1526-5498 | 08 | 1001 | 0024

informs®
DOI 10.1287/msom.1060.0148
© 2008 INFORMS

Workforce Design with Movement Restrictions Between Workstation Groups

Jonathan F. Bard
Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, Austin, Texas 78712, jbard@mail.utexas.edu

Lin Wan
Amazon.com, Seattle, Washington 98104, linwan@amazon.com

This paper is concerned with the problem of determining the optimal size and composition of a permanent workforce needed to run a facility when demand is specified by a workstation group (WSG) for up to 24 hours a day, 7 days a week. For full-time employees, a solution is characterized by a bid job, which consists of a five-day-a-week schedule, a lunch break for all shifts, and a set of WSG task assignments for each of the half-hour periods in a shift. In contrast, each part-time employee may be given anywhere from one to six shifts during the week, and each shift may vary from four to eight hours in length. To facilitate supervision, all employees must be assigned to a home WSG, but when idle time exists in their schedules, they can be redeployed to other WSGs for a portion of the day.

One of the complicating and unique factors addressed in this paper is the existence of nonsymmetric movement restrictions between WSGs. For example, an employee whose home base is *A* may be permitted to perform tasks at *B*, but not vice versa. Because the full problem could not be reduced to a single model, a multistage solution approach was developed. In the first stage, an extended shift-scheduling problem is solved to determine the optimal number of employees and their shifts. The results are postprocessed in subsequent stages to obtain lunch breaks, days off, and task assignments under WSG movement restrictions.

In the implementation of the multistage approach, two alternatives were explored. The first was based on the idea of partitioning the WSGs into manageable clusters and then solving them in series. The second involved the direct solution of an integer programming formulation of the task assignment problem with home-base restrictions and WSG movement restrictions, but for a fixed workforce. An iterative scheme was used to adjust the size of the workforce until all constraints were satisfied and overall optimality was achieved. Testing was done with data provided by the U.S. Postal Service (USPS) mail processing and distribution center (P&DC) in Dallas. The computations showed that the second alternative always yielded the smaller workforce and was always able to find good solutions within 30 minutes.

Key words: staff scheduling; movement restrictions; decomposition; integer programming; workstation groups
History: Received: November 9, 2005; accepted: November 27, 2006. Published online in *Articles in Advance* December 11, 2007.

1. Introduction

Organizations with shrinking margins and strong competition have come to realize that effective personnel planning and scheduling are essential for long-term survival. Poor planning can lead to an oversupply of workers with too much idle time, or an undersupply with an attendant loss of business, especially in the service industry. Whether it is a call center, a manufacturing facility, a processing plant, or a hospital, management's goal is to find the best mix of employees to satisfy demand at minimum cost. What stands in the way of achieving this goal is a complex of labor laws, work rules, union agreements, and outdated company polices.

Many studies have shown that cross-training and the use of higher-skilled workers to fill in for lower-skilled workers when idle time exists in their schedules can yield substantial cost savings (e.g., see Bard 2004a, Malhotra and Ritzman 1994, Misra et al. 2004). Other management options that can be applied with some advantage include different days-off policies, variable start times, and the use of part-time flexible employees. Out of habit, though, it is common for organizations to adopt overly restrictive practices—such as providing weekly schedules with uniform start times—without fully examining the cost consequences.

In previous work, we developed an integrated methodology for long-term planning that was aimed at determining the optimal size and composition of a permanent workforce in high-volume factories (Bard et al. 2003). In the first stage, an extended shift-scheduling problem was solved to obtain the optimal number of shifts and the minimum number of full-time and part-time workers by skill type to meet expected demand. Postprocessors were then applied to fix days off, lunch breaks, and individual task assignments in half-hour increments. The final output was a set of individual weekly schedules commonly called *tours*. The purpose of this paper is to extend our methodology to include an overlooked aspect of personnel scheduling that arises when each employee must be assigned to a home base and restrictions exist on how he or she can be repositioned during a shift.

The motivation for this work derived from our involvement with the U.S. Postal Service (USPS) and their ongoing effort to improve the efficiency of mail-processing operations. Throughout the United States there are approximately 275 major processing and distribution centers (P&DCs) that serve as hubs in the mail and parcel distribution network. On a typical day, a medium-sized P&DC might receive as many as five million letters, 0.5 million flats, and thousands of parcels. To handle such large volumes, advanced equipment in the form of optical character readers, automated facer-cancellers, bar-code sorters (BCS), and material-handling systems have been uniquely designed to automate as much of the mail stream as possible.

P&DCs can be viewed as high-volume factories, running 24 hours a day, 7 days a week (Berman et al. 1997). They are staffed by a skilled workforce comprising full-time, part-time, and temporary employees whose schedules are subject to a wide range of negotiated restrictions. In the long run, the goal is to design the permanent workforce and to find weekly tours for each employee by skill category. In this paper, a tour includes the workdays, their length, the daily start time, the lunch break, the home base, and the task assignments (Bechtold and Brusco 1994, Billionnet 1999, Emmons and Burns 1991). At P&DCs, these specifications define a "bid job" (e.g., see Ritzman et al. 1976) and are at the center of our analysis. While workforce sizing and tour construction may be treated as separate problems, we define the tour-scheduling problem as a combination of the two (see Jarrah et al. 1994, Rekik et al. 2004). We also use the words *tour* and *bid job* interchangeably.

At P&DCs, the need to specify a period-by-period set of tasks for each worker gives rise to the *task assignment problem* (Ernst et al. 2004). The solution of this problem must strike a balance between the efficient use of the workforce and the need for managerial oversight. A more flexible workforce allows for a wider range of assignments with a corresponding reduction in labor costs. As workers transit from one location to another, though, productive time may be lost and supervision may become more difficult. In facilities where the physical layout limits movement or where union rules limit the job content, there may be additional restrictions on what an employee may be asked to do.

The primary contribution of this paper is the development and testing of two approaches for dealing with movement restrictions between workstation groups (WSGs), and the need to assign each employee to a home base when solving the tour-scheduling problem and constructing bid jobs. The first approach is based on the idea of converting a general description of the problem into a directed network in which each node represents a WSG (i.e., a group of similar machines) and each arc represents the permissible transitions. In the second approach, a tentative workforce is found, along with a schedule for each worker by solving a relaxation of the tour-scheduling problem. The solution serves as input to a second integer program whose objective is to minimize the uncovered demand over the planning horizon. Any shortages that are identified are handled in one of two ways: (1) either they are added back to the original demand, and the process is repeated until all requirements are met; or (2) the first approach is applied with the incremental demand as input.

Although the proposed methodologies were developed with the USPS application in mind, they are readily generalizable. With some minor adjustments to take into account time scale and operational-related constraints, they can be applied to any facility in which employee assignments must respect certain physical or logical boundaries.

In the next section, we provide an example to illustrate what is meant by movement restrictions and then give a formal definition of the problem. In §3, we describe the first solution approach, followed in §4 with the development of an integer programming model that can be used to obtain more robust assignments, but at much greater computational cost. It is also shown that the underlying problem is NP-hard. Our experimental results are presented in §5, where it is seen that most practical instances can be solved within less than 30 minutes, a reasonable amount of time for planning purposes.

2. Problem Definition

Given a set $J = \{\text{WSG}_1, \ldots, \text{WSG}_n\}$ of n WSGs and an $n \times n$ matrix R that embodies the movement restrictions between each pair (j, k) of elements in J, the overall problem is to find the minimum-size workforce required to meet the staffing demand over the planning horizon subject to tour scheduling and WSG movement restrictions. Here, $R = (r_{jk})$, where $r_{jk} = 1$ if a worker with home base $j \in J$ is permitted to move to $k \in J$, and 0 otherwise. An additional constraint imposed on the problem is that each worker must spend at least as much time at his home base as at any other WSG. For example, if worker 1 is assigned to WSG_1 for 40% of the week, WSG_2 for 15% of the week, and WSG_3 for 45% of the week, then WSG_3 must be his home base.

The USPS planning model, which is outlined in §2.2, was designed to help construct bid jobs for full-time regular (FTR) employees and to develop weekly schedules for part-time flexible (PTF) employees. In the model, the workday is divided into 48 periods, each 30 minutes long. A regular employee works a shift of a predefined length and may start during one of three intervals (used for accounting purposes only). This gives rise to 48 different shifts, each 17 periods ($8\frac{1}{2}$ hours) long, including the lunch break, for full-timers. For part-timers, there are generally 24 different start times and five different shift lengths, making 120 different part-time shift types in all. Allowable shift lengths are 8, 10, 13, 15, and 17 periods, including the breaks where applicable. Model options include two consecutive days off in a row for each bid job, the specification of a time band in which a full-timer must start each of his or her shifts during the week, a lunch break window, a minimum ratio of full-timers to part-timers, the assignment of 10-hour shifts four days a week, and the use of downgrading. By choice, overtime and temporary labor are excluded from the planning problem (see Bard et al. 2003 for the details, and Lin et al. 2000 for an alternative system).

While the solution to the tour-scheduling problem specifies the workdays and shift assignments for each employee, in the construction of bid jobs it is often necessary to specify how that person will be spending his or her time on a period-by-period basis. The corresponding task assignment problem (e.g., see Campbell and Diaby 2002, Ernst et al. 2004, Franz and Miller 1993, Wan 2005) is typically solved in a post-processing stage of the computations. When a hierarchy of skill categories exists, it may be possible to assign idle time in the schedule of a higher-skilled worker to cover demand of a lower-skilled worker. This is known as downgrading and can provide substantial cost savings (Bard 2004a, Bard and Purnomo 2005, Dawid et al. 2001). Both of the algorithms proposed in the next section make use of this option.

2.1. WSG Movement Restrictions

In P&DCs, a task assignment for a mail processor is associated with a specific WSG; i.e., a group of similar machines such as delivery bar-code sorters (DBCSs) or multiline optical character readers (MLOCRs) (see Table 1 for definitions). Although it is desirable that each employee be assigned to a single WSG (as well as to a specific machine within that WSG) for a full shift, this is not possible in most cases because equipment schedules, which are derived from mail arrival profiles, do not match shift lengths. Some operations might only be two or three hours long, while others may last up to 12 hours.

When skill substitution is permitted or when the same skills are used at different WSGs, a certain amount of movement is necessary to avoid excess idle time. A *good* assignment of tasks is characterized by as few switches among WSGs as possible; in other words, one that minimizes some function of the total number of switches over the day for each worker category.

Table 1 Definition of Worker Categories and Equipment in a P&DC

Worker category	Abbreviation	Equipment	Abbreviation
Types of mail processors:		Advanced facer-canceller system	AFCS
Flat sorting machine operator	P6-FSMO	Delivery bar code sorter	DBCS
General expeditor	P6-GE	Flat sorting machine	FSM
Parcel post distribution machine operator	P6-PPDMO	Input subsystem	ISS
Sack sorting machine operator	P6-SSMO	Manual operations	MANUAL
Mail processing clerk	P5-MPC	Multiline optical character reader	MLOCR
Flat sorting machine operator	P5-FSMO	Output subsystem	OSS
Parcel post distribution machine operator	P5-PPDMO	Remote encoding center	REC
Data conversion operator	P4-DCO		
Types of mail handlers:			
Mail handler	MH5		
Mail handler equipment operator	MH5-EO		
Mail handler technician	MH5-T		
Mail processing machine operator	MH5-MPMO		
Sack sorting machine operator	MH5-SSMO		
Mail handler	MH4		
Sack sorting machine operator	MH4-SSMO		

When demand is specified by WSG, the layout of a facility or a supervisor's wish to keep tight control over those in his or her area are two factors that may vastly complicate the design and use of the workforce, regardless of its inherent flexibility. If movement between all WSGs is possible, then the aggregate demand can be used and a single problem can be solved. If there are some restrictions, then the situation becomes much more challenging. For example, consider a facility comprised of three WSGs $(G1, G2, G3)$ with the following restrictions (see Figure 1(a)):

(i) $G1 \leftrightarrow G2$ (workers in G1 can transit to G2 and vice versa)

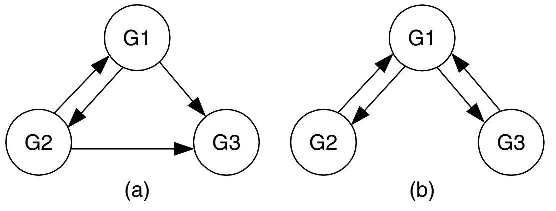
(ii) $G1 \rightarrow G3$ (workers in G1 can transit to G3)

(iii) $G2 \rightarrow G3$ (workers in G2 can transit to G3)

The scheduling problem for the facility can be approached sequentially by first merging the demand at nodes G1 and G2, and then finding a solution for the combined demand data. Any idle time in the solution for the workers assigned to G1 and G2 can be used to satisfy demand at G3 before the problem associated with G3 is solved.

Now consider the scheduling problem with the restrictions depicted in Figure 1(b). Combining the demand of all three WSGs is inappropriate because workers whose home base is G2 are not permitted to be assigned tasks at G3. If G1 and G2 are combined and the corresponding problem solved, idle time of workers whose home base is G1 can be allocated to satisfy demand at G3, but the opportunity to allocate idle time of workers whose home base is G3 to satisfy demand at G1 will be lost. This can lead to suboptimal solutions. An additional complication arises as a result of the requirement that each worker must spend a plurality of his or her time at his or her home base. Prior to our work, no mathematical formulation existed for this case, or the more general case in which the home base and the task assignments are to be determined concurrently.

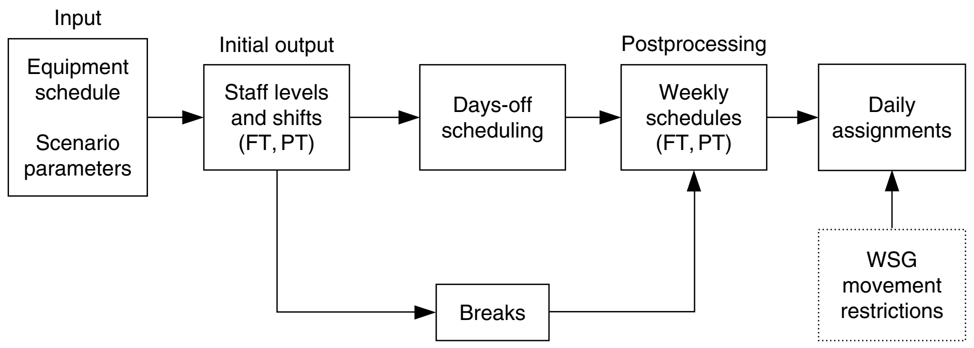
Figure 1 Examples of WSG Restrictions



2.2. Current System

A decision support system known as SOS has been developed for long-term planning at P&DCs. The major computations involve the solution of a series of

Figure 2 Schematic of the Computational Flow in SOS



large-scale integer programs (IPs) to determine overall workforce size and shift requirements, and the postprocessing of the results to construct bid jobs (see Figure 2). The objective is to minimize the cost of the permanent workforce. Principal inputs include demand in the form of a weekly equipment schedule and a set of parameter values that define the scenario under investigation (see Bard 2004b for a discussion of demand). When the user requests that *skill substitution* be applied, SOS has the ability to move people across WSGs during the solution of the task assignment problem. The hierarchical structure for doing this is embedded in the downgrading matrix shown in Figure 3. Cells with an "×" indicate that the skill identified on the far left in the corresponding row can substitute for the skill listed in the top row of the corresponding column (see Table 1 for skill definitions). The shaded cells indicate that as a general rule, no substitution is permitted between mail processors (P) and mail handlers (MH).

When the extended shift-scheduling problem is solved (second box in Figure 2), SOS does not distinguish WSGs within the same skill category, so in the postprocessing stage, workers are moved freely from one WSG to another (in any direction), notwithstanding the substitution matrix. For example, SOS will assign a worker from a DBCS group to an MLOCR group, or vice versa, when idle time exists, because any P5-MPC can operate either of these machines. The resultant task assignments, however, may violate the

Figure 3 Skill Substitution Matrix in SOS

Lower skill

Higher skill

	P6-FSMO	P6-GE	P6-PPDMO	P6-SSMO	P5-MPC	P5-FSMO	P5-PPDMO	P4-DCO	MH5	MH5-EQ	MH5-T	MH5-MPMO	MH5-SSMO	MH4	MH4-SSMO
P6-FSMO	N/A				×	×			▨	▨	▨	▨	▨	▨	▨
P6-GE		N/A			×		×		▨	▨	▨	▨	▨	▨	▨
P6-PPDMO			N/A		×				▨	▨	▨	▨	▨	▨	▨
P6-SSMO				N/A	×				▨	▨	▨	▨	▨	▨	▨
P5-MPC					N/A				▨	▨	▨	▨	▨	▨	▨
P5-FSMO					×	N/A			▨	▨	▨	▨	▨	▨	▨
P5-PPDMO					×		N/A		▨	▨	▨	▨	▨	▨	▨
P4-DCO								N/A	▨	▨	▨	▨	▨	▨	▨
MH5	▨	▨	▨	▨	▨	▨	▨	▨	N/A					×	
MH5-EQ	▨	▨	▨	▨	▨	▨	▨	▨		N/A				×	
MH5-T	▨	▨	▨	▨	▨	▨	▨	▨			N/A			×	
MH5-MPMO	▨	▨	▨	▨	▨	▨	▨	▨				N/A		×	
MH5-SSMO	▨	▨	▨	▨	▨	▨	▨	▨					N/A	×	
MH4	▨	▨	▨	▨	▨	▨	▨	▨						N/A	
MH4-SSMO	▨	▨	▨	▨	▨	▨	▨	▨						×	N/A

local movement restrictions, implying that the solution provided by the tour-scheduling model is infeasible. We now address this issue.

2.3. Model Development

In theory, virtually all planning and scheduling problems can be formulated as set covering-type models in which each column represents a feasible assignment. In practice, this is likely to produce problem instances with an impossibly large number of columns (e.g., see Bechtold and Jacobs 1990). SOS was designed around a constraint-based formulation that is used for the major computations (workforce sizing and shift determination) and is manageable up to the point of including the constraints associated with the task assignments and the WSG movement restrictions. The former are handled by one of several algorithms that do not take into account the locations of individual machines or the transit time between them. Hence, solutions may violate the WSG movement restrictions (lower far right box in Figure 2). We propose two approaches to ensure feasibility. To simplify the presentation, we will only address a single worker category, such as P5-MPC, with the understanding that the ideas are completely general.

3. Sequential Procedure

The simplest way to accommodate movement restrictions is to treat each WSG separately and solve the corresponding tour-scheduling problems in turn. Unfortunately, the aggregate solution is not likely to be very good because of the absence of any movement between WGSs, even when permitted. A slight modification where idle time is allocated appropriately after obtaining a solution for a particular WSG may lead to only a marginal improvement in the overall results. Of course, if the movement restrictions matrix R indicates that all WSGs are completely connected, implying that there are really no movement restrictions, then it is optimal to solve a single aggregate problem, postprocess the results to obtain the task assignments, and then fix the home base of each worker by determining where he or she spends the plurality of his or her time.

For the intermediate case, we begin by representing R as a directed network $G = (V, A)$ similar to those depicted in Figure 1. The first step is to generate clusters by combining nodes that are completely connected. For discussion purposes, a cluster may include a single node, but when two or more nodes are included, there will be two-way arcs between each pair in the graph.

At an intermediary stage in the aggregation process, let C be the set of clusters and denote the modified network by $\widehat{G} = (C, A)$. We now arbitrarily select a $c_1 \in C$ and scan each remaining cluster to see if one of them can be merged with c_1. A cluster $c_2 \in C\backslash\{c_1\}$ can be merged with c_1 if and only if every node $j \in c_1$ has a two-way link with every node $k \in c_2$. If such a c_2 is found, then all nodes in c_2 are merged into c_1 (i.e., $c_1 \leftarrow c_1 \cup c_2$) as well as the arcs associated with these nodes, and c_2 is deleted from $\widehat{G}$. The search is repeated starting with the new c_1 and continuing until no pair of clusters can be found that satisfies the merging conditions. When two clusters are merged to form c_1, all arcs between the nodes in c_1 and the nodes in the original graph G, excluding the nodes in c_1 (i.e., $V\backslash c_1$), are retained.

The next step is to identify clusters with outbound arcs only and place them in a queue Q. All such clusters are removed from $\widehat{G}$. It may be necessary to loop through $\widehat{G}$ several times, because removing c and arcs $(c, I(c))$, where $I(c)$ is the set of clusters that have inbound arcs originating from cluster c, may produce additional clusters with outbound arcs only.

When no more clusters can be removed from $\widehat{G}$, the search for loops begins. Placing the remaining K clusters in the set $C = \{c_1, \dots, c_K\}$ in arbitrary order, we start with, say, cluster c_1, and select a $c_2 \subset C\backslash c_1$ such that at least one arc in the set $\{(c_2, c_1)\}$ exists. The process is repeated, starting this time from c_2 until a loop is identified, i.e., until a cluster is selected for the second time. The loop does not necessarily have to include cluster c_1. Let $L = \{c_1, \dots, c_{k-1}, c_k\}$ be the ordered set of clusters in the loop, where $c_1 = c_k$. We now look for the particular cluster c^* in L that has the minimum number of inbound arcs from its immediate predecessor cluster, $p(c^*)$, in the loop and remove all such arcs from $\widehat{G}$; i.e., find $c^* \in \arg\min\{|\{(p(c), c)\}|\colon c \in L\}$ breaking ties arbitrarily, and remove arcs $\{(p(c^*), c^*)\}$ from $\widehat{G}$. This breaks the loop. If c^* in the resultant graph has outbound arcs only, then it is removed from $\widehat{G}$ and placed in Q.

As mentioned, multiple passes through $\widehat{G}$ may be required because removing c^* and its outbound arcs may produce additional clusters with outbound arcs only.

If there are still clusters that cannot be removed from the network, we continue to search for and break loops as described until all clusters are placed in Q. Let $\bar{A}$ be the set of arcs that are dropped when merging two clusters and breaking loops, and let $A^* = A\backslash\bar{A}$ be the set of remaining arcs. As desired, the graph $G^* = (Q, A^*)$ is a directed network without loops. The algorithm is now formally stated, followed by a justification of the reduction and loop-breaking components. An example to highlight each step is given at the end of the subsection.

Network Reduction Algorithm.

Input: Movement restrictions matrix R

Output: Queue Q of independent nodes and clusters

Notation: P = path vector, E = set of inbound arcs along a path, A_i = set of arcs joining cluster c_i to its immediate successor cluster c^* in path P, A^* = set of arcs in final directed network

Step 1 (Initialization). Use R to construct the directed graph $G = (V, A)$ and the sets of nodes $O(j)$ and $I(j)$ that have outbound links to and inbound links from node j, respectively, for all $j \in V$. Put $A^* = A$, $C = \varnothing$, and $Q = \varnothing$.

Step 2. (Cluster formation)

(1) For each $j \in V$, create a cluster $c_j = \{j\}$ and put $C \leftarrow C \cup \{c_j\}$.

(2) For $c \in C$, find a cluster $\hat{c} \in I(c)$ with two-way arcs between all $j \in c$ and all $k \in \hat{c}$. Put $c \leftarrow c \cup \hat{c}$, $C \leftarrow C\backslash\{\hat{c}\}$, $A \leftarrow A\backslash\{(c, \hat{c}), (\hat{c}, c)\}$, and $A^* \leftarrow A^*\backslash\{(c, \hat{c}), (\hat{c}, c)\}$. Repeat until no more merging is possible.

Step 3 (Cluster ordering).

(a) Let $\widehat{G} = (C, A)$ and construct sets $O(c)$ and $I(c)$ for all $c \in C$.

(b) (Independent clusters) For all $c \in C$, if $O(c) = \varnothing$, put $Q \leftarrow Q \cup \{c\}$, $C \leftarrow C\backslash\{c\}$, $A \leftarrow A\backslash\{(c, \hat{c})\text{:b}\hat{c} \in I(c)\}$. Repeat until there is no $c \in C$ such that $O(c) = \varnothing$. If $C \neq \varnothing$, go to Step 3(c); otherwise, stop.

(c) (Finding and breaking loops) Let $P = \varnothing$, $E = \varnothing$. Start from any $c \in C$ and let $c^* = c$.

(i) If $c^* \notin P$, then go to Step (ii), otherwise, go to Step (iii).

(ii) (Loop not found). $P \leftarrow (P, c^*)$. Pick any $c_i \in O(c^*)$, let $A_i = \{(c_i, c^*)\}$, and put $E \leftarrow E \cup \{A_i\}$. Let $c^* = c_i$, and go to Step (i).

(iii) (Loop found). Let $i^* \in \arg\min\{|A_i|\text{: } A_i \in E\}$. Put $A \leftarrow A\backslash A_{i^*}$, $A^* \leftarrow A^*\backslash A_{i^*}$, and go to Step 2(b).

It is straightforward to show the following.

Proposition 1. *If the network G has no clusters with outbound arcs only, then*

(1) *at least one loop exists in G, and*

(2) *the reduction algorithm is guaranteed to find a loop at Step* 3.

The algorithm terminates with $C = \varnothing$ and the ordered set Q. Also available are the sets $I(c)$ and $O(c)$, for all $c \in Q$ and A^*, which can be used to construct the graph $G^* = (Q, A^*)$. The final step is to solve the tour-scheduling and task assignment problems for each $c \in Q$ in the appropriate order. Any idle time that exists in a shift that is associated with a worker in cluster c is used to satisfy demand at nodes $j \in I(c)$ as long as the home-base constraint is not violated. Of course, idle time at leaf nodes cannot be assigned to other clusters.

In summary, the sequential procedure represents a decomposition of the full problem into a series of tour-scheduling problems whose solutions are guaranteed to be feasible to the WSG restrictions. After the workforce and corresponding tours are determined for each cluster with the methodology outlined in Figure 2, the respective task assignment problems are solved. A description of several algorithms designed for this purpose can be found in Wan (2005).

Figure 4 Movement Restrictions Network, for Example

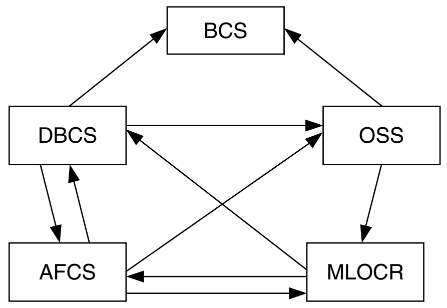
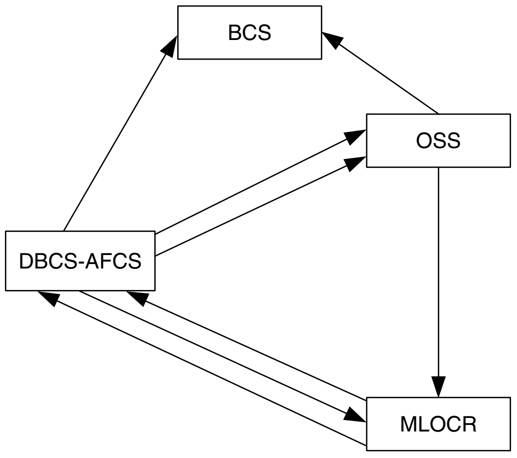


Figure 5 **Network After Clustering**



Example. The network in Figure 4 will be used to illustrate the network reduction algorithm. Each node corresponds to a WSG and the authorized movements are represented by the directed arcs. The abbreviations in Table 1 are used to identify the WSGs. The first step is to convert all nodes to clusters and to place all clusters with outbound arcs only in Q. None exist, so we go to Step 2 and begin the merging procedure. Two choices are available: Either combine DBCS and AFCS or combine AFCS and MLOCR. The first option is chosen, giving the reduced network shown in Figure 5. Further clustering is not possible because no three nodes are completely connected.

Going to Step 3, we now must break loops in $\widehat{G}$ and construct Q by ordering the four clusters {DBCS-AFCS, BCS, OSS, MLOCR}. At each iteration, the ordering process tries to find a cluster with no inbound links. If no such cluster exists, then one or more loops are present in the network. To identify a loop, we start from any cluster c and try to build a path by joining c with a cluster in $\Psi(c)$, and so on. If a cluster enters the path twice, then a loop has been found. Because every cluster has at least one inbound link, Proposition 1 guarantees that a loop exists and that the algorithm will find it.

Suppose we start from BCS. Then $O(\text{BCS}) =$ {DBCS-AFCS, OSS}, and either one can be chosen to be the next cluster in the path. Suppose we pick OSS, which gives the path BCS $\leftarrow$ OSS. The current cluster is OSS and $O(\text{OSS}) =$ {DBCS-AFCS}, which is added to the path to get BCS $\leftarrow$ OSS $\leftarrow$ DBCS-AFCS. Similarly, MLOCR is added next to give the path BCS $\leftarrow$ OSS $\leftarrow$ DBCS-AFCS $\leftarrow$ MLOCR. Note that $O(\text{MLOCR}) =$ {OSS, DBCS-AFCS}; we arbitrarily pick OSS, which is already in the path, so a loop has been found: OSS $\leftarrow$ DBCS-AFCS $\leftarrow$ MLOCR $\leftarrow$ OSS.

Referring to Figure 5, we see that there are two links from DBCS-AFCS to OSS, two links from MLOCR to DBCS-AFCS, and one link from OSS to MLOCR, so the loop is broken between OSS and MLOCR; i.e., the least number of links is removed. The resultant network $\widehat{G}$ is displayed in Figure 6(a).

At this point, we only need to check MLOCR to see if it has any inbound links, i.e., whether $O(\text{MLOCR}) = \varnothing$. Because this set is not empty, it cannot be removed from the network and placed in Q. Repeating the loop identification process at Step 3 yields DBCS-AFCS $\leftarrow$ MLOCR $\leftarrow$ DBCS-AFCS, which is broken by removing the link from DBCS-AFCS to MLOCR. The resultant network is shown in Figure 6(b), where we see that MLOCR now has no inbound links. Therefore, MLOCR is removed from $\widehat{G}$ along with its outbound arcs, and placed in the queue giving $Q =$ {MLOCR}. Figure 7(a) depicts the reduced network. The next cluster to be removed is DBCS-AFCS (see Figure 7(b)), followed by OSS, and then BCS. The final ordering in Q is {MLOCR, DBCS-AFCS, OSS, BCS}.

Figure 6 **Networks After Breaking Loops**

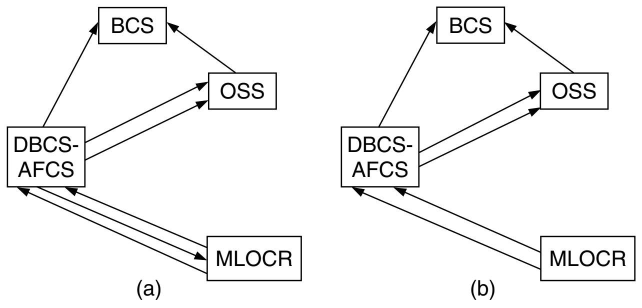
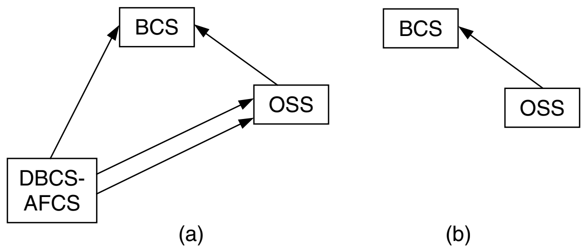


Figure 7 **Network After Removing the Two Clusters**



4. Iterative Procedure

Our second approach to dealing with the WSG restrictions begins by solving a relaxed version of the tour-scheduling problem that does not include the task assignment, movement restrictions, and home-base constraints (all but the two far-right boxes in Figure 2). The solution is used as input to an auxiliary IP that we call the WSG movement restriction assignment problem (WGAP). In the formulation of the WGAP, it is necessary to take into account the likelihood that the shift assignments obtained from the tour-scheduling problem will not be feasible to the WSG constraints. If this is the case, we iterate by modifying the demand and resolving the relaxed tour-scheduling problem.

A solution to the WGAP provides each worker with a home base and a task for each period in his or her schedule. The objective is to make these assignments in a way that minimizes the gap in coverage over the planning horizon. If the optimal objective function value is greater than zero, then some demand remains uncovered and the solution is not feasible to the WSG constraints. A new problem is then solved, with the gap as input. Two options are considered.

(1) The new demand d_{jp}^{new} in period p for WSG j is the old demand d_{jp}^{old} plus the gap s_{jp}; i.e., $d_{jp}^{\text{new}} = d_{jp}^{\text{old}} + s_{jp}$ for all $p \in P$, $j \in J$. The optimization process is repeated until a feasible solution is found.

(2) The new demand $d_{jp}^{\text{new}} = s_{jp}$ for all $p \in P$, $j \in J$ and the sequential procedure is called.

In the developments, we make use of the following notation:

Indices and Sets

i	index for workers; $i \in I$
j, k	index for WSG; $j \in J$
p	index for periods; $p \in P$
$P(i)$	set of periods that worker i is scheduled to be on duty during the week
$O(j)$	set of WSGs that have outbound links to WSG j
$I(j)$	set of WSGs to which a worker whose home base is WSG j can move

Parameters

d_{jp}	(demand) number of workers that are needed in WSG j during period p

Decision Variables

x_{ij}	(binary) one if WSG j is the home base of worker i, zero otherwise
y_{ijp}	(binary) one if worker i is assigned to WSG j in period p, zero otherwise
s_{jp}	uncovered demand (gap) in WSG j during period p

WGAP Model

$$\text{Minimize} \quad z = \sum_{j \in J} \sum_{p \in P} s_{jp} \tag{1a}$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1, \quad \forall\, i \in I \tag{1b}$$

$$\sum_{j \in J} y_{ijp} \le 1, \quad \forall\, i \in I,\ \forall\, p \in P(i) \tag{1c}$$

$$\sum_{i \in I} y_{ijp} + s_{jp} \ge d_{jp}, \quad \forall\, j \in J,\ \forall\, p \in P \tag{1d}$$

$$y_{ijp} \le x_{ij} + \sum_{k \in O(j)} x_{ik}, \quad \forall\, i \in I,\ \forall\, j \in J,\ \forall\, p \in P(i) \tag{1e}$$

$$\begin{aligned} & x_{ij} \in \{0,1\}, \quad \forall\, i \in I,\ \forall\, j \in J, \\ & y_{ijp} \in \{0,1\}, \quad \forall\, i \in I,\ \forall\, j \in J,\ \forall\, p \in P(i) \\ & s_{jp} \ge 0, \quad \forall\, j \in J,\ \forall\, p \in P. \end{aligned} \tag{1f}$$

The objective function in (1a) sums the unmet demand determined by constraint (1d). Constraint (1b) assigns each worker i to exactly one home base, while (1c) limits the number of WSGs to which i can be assigned to at most one in period p. The set $P(i)$ is obtained from the solution of the relaxed tour-scheduling problem and is equivalent to the shift assignments for i over the week. When a solution to the tour-scheduling problem satisfies the WSG constraints, $s_{jp} = 0$ for all j and p in (1d). This implies that all demand can be covered by the available workforce.

Constraint (1e) ensures that the movement restrictions specified by the matrix R and embodied in the set $O(j)$ are satisfied in a solution. If WSG j is the home base of worker i, which is the case when $x_{ij} = 1$, then worker i can be assigned to any WSG $k \in I(j) \cup \{j\}$ during each period $p \in P(i)$. To guarantee that a worker spends more time at his home base than at any other WSG, one more constraint is needed:

$$\sum_{p \in P(i)} y_{ijp} \ge \sum_{p \in P(i)} y_{ikp} - |P(i)|(1 - x_{ij}), \quad \forall\, i \in I,\ \forall\, j \in J,\ \forall\, k \in I(j) \backslash \{j\}. \tag{1g}$$

If a worker is required to spend at least 50% of his or her time at the home base, then (1g) should be replaced with

$$\sum_{p\in P(i)} y_{ijp} \ge \sum_{k\in I(j)} \sum_{p\in P(i)} y_{ikp} - |P(i)|(1-x_{ij}), \quad \forall\, i\in I,\ \forall\, j\in J, \quad (2)$$

where the first summation on the right-hand side of (2) can be extended to $k \in J\backslash\{j\}$ to strengthen the inequality.

If the inclusion of (1g) in the model makes the problem too unwieldy, a two-stage heuristic might be a reasonable way to proceed. In the first stage, the optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ to (1a)–(1f) would be found; in the second stage, $\mathbf{x}$ would be fixed at $\mathbf{x}^*$, and new values for $(\mathbf{y}^*, \mathbf{s}^*)$ would be found with (1g) included for each $i \in I$. This approach reduces the number of additional constraints by a factor of $O(|J|)$. We use a variation of this idea below.

Symmetry. In the task assignment problem, workers are indistinguishable from each other, so there can be a vast number of alternative optima. This kind of symmetry may seriously undermine the performance of a branch-and-bound algorithm, as evidenced by an extended search tree in which the same solutions appear repeatedly at different nodes. One way to deal with this difficulty is to augment the basic model with suitable symmetry-breaking hierarchical constraints (see Sherali and Smith 2001 for an extended discussion of symmetry). These constraints will tighten the LP formulation of the problem and therefore have the potential to significantly reduce the extent of the feasible region that must be explored during branch and bound. For the task assignment component of the problem, the following constraint

$$x_{ij} \ge x_{i+1,\,j} - \sum_{k\in J\backslash\{j\}} x_{i-1,\,k}, \quad i=2,\ldots,|I|-1,\ \forall\, j\in J \quad (3)$$

can be added to partially eliminate symmetric assignments of workers to home bases. The second term on the right-hand side of (3) is one when worker $i-1$ is assigned to a home base other than j. In this case, the constraint is redundant; otherwise, it only permits worker $i+1$ to be assigned to WSG j if i is also assigned to j. As a consequence, it is not possible for i and $i+2$ to be assigned to WSG j, but not $i+1$. Nevertheless, it does not rule out the case where i and $i+k$ are assigned to WSG j and not $i+1, i+2, \ldots, i+k-1$, where $k \ge 3$. To ensure that consecutive workers are assigned to the same home base, it would be necessary to expand constraint (3) by including logic variables for each of the prior workers and WSGs. This would add $O(|I|\cdot|J|)$ variables to the model.

An alternative approach to the objective in (1a) is to minimize the number of switches from one WSG to another, i.e., where worker i is assigned to WSG j and worker $i+1$ is assigned to WSG k. Let γ_{ij} be a nonnegative variable equal to one if worker $i+1$ is assigned to WSG j worker i is assigned to WSG $k \ne j$, and zero otherwise. We now add the following constraint to the model:

$$x_{ij} - x_{i+1,\,j} + \gamma_{ij} \ge 0, \quad i=1,\ldots,|I|-1,\ \forall\, j\in J,$$

and modify the objective function (1a) to be

$$\text{Minimize}\, z = \sum_{j\in J}\sum_{p\in P} s_{jp} + \sum_{i=1}^{|I|-1}\sum_{j\in J} \gamma_{ij}. \quad (1a')$$

Fortunately, γ_{ij} can be treated as a continuous variable because it will always be zero or one in an optimal solution. Also, it is not necessary to multiply the second term in (1a′) by a small positive constant to ensure that the number of uncovered periods is minimized before the number of switches, as in goal programming. Because the workers are indistinguishable from each other, there is always an optimal solution in which the minimum number of switches is achieved. A bound on this number is $\sum_{i,\,j}\gamma_{ij} \le |J|-1$.

WSG demand. If it is assumed that the number of workers assigned to a WSG is proportional to its demand, then we can add a constraint to the model to tighten the feasible region. Let $D_j = \sum_{p\in P} d_{jp}$ be the total demand for WSG j, and order the WSGs such that $D_{[j]} \ge D_{[j+1]}$, where $[j]$ is the WSG in position j. The following constraint ensures that the number of workers assigned to $[j]$ is greater than or equal to the number assigned to $[j+1]$

$$\sum_{i\in I} x_{i,\,[j]} \ge \sum_{i\in I} x_{i,\,[j+1]}, \quad \forall\, j\in J\backslash\{[J]\}. \quad (4)$$

Although (4) is not a valid inequality for WGAP, if including it in model (1) reduces the computational burden significantly, then the resultant degradation in the solution may be justified. Moreover, solving the problem with (4) will provide an upper bound that may be close to the optimal objective function value.

4.1. Complexity Issues

We now address the WGAP as represented by model (1), in which the objective is to minimize the number of uncovered periods. To see its complexity, we define the recognition version of the problem as follows.

Instance of WGAP. A finite number or workers m that must be assigned to one of n WSGs, a set of periods $P(i)$ that defines worker i's schedule, a set of restrictions $I(j)$ that limits the movement of workers assigned to WSG $j \in J$, a list of nonnegative integers d_{jp} specifying the demand for workers in period p for WSG j, and a list of nonnegative integers s_{jp} indicating the amount of demand not covered in period p at WSG j.

Question. Is there an assignment of workers to WSGs and then to periods within their schedule such that at least $\sum_{j\in J}\sum_{p\in P} d_{jp} - \sum_{j\in J}\sum_{p\in P} s_{jp}$ of the demand is covered?

Theorem 1. *The recognition version of WGAP is NP-complete in the strong sense.*

Proof. We start with an instance of the directed m-commodity integral flow problem (DmCIF) problem and show that it can be polynomially transformed into an instance of WGAP in which a worker can be assigned to multiple WSGs. The recognition version of DmCIF is defined on a directed graph $G = (V, A)$ with specific vertices s_i and e_i, capacity $c(a) \in Z^+$ for each $a \in A$, and requirements $R_i \in Z^+$, $i = 1, \ldots, m$. The following question is asked: Are there m flow functions f_i: $A \to Z_0^+$ such that

(a) for each $a \in A$, $f_1(a) + \cdots + f_m(a) \le c(a)$,

(b) for each $v \in V\backslash\{s, e\}$, flow is conserved at v, and

(c) for $i = 1, \ldots, m$, the net flow into e_i under flow $f_i(a)$ is at least R_i?

Even et al. (1976) showed that D2CIF is NP-complete in the strong sense by transformation from 3SAT and remains so when $s_1 = s_2$, $e_1 = e_2$ and when arcs are restricted to carry only one commodity.

To simplify things a bit, let $H_i = |P(i)|$ be the number of periods in worker i's schedule and assume that $\sum_{i\in I} H_i = \sum_{j\in J}\sum_{p\in P} d_{jp}$; that is, no worker has idle time (this assumption is made to obviate the need to define a dummy WSG to take up the slack). From a general instance of DmCIF, we construct the corresponding instance of WGAP depicted in Figure 8. In the network, there is one source node for each commodity (worker), which is connected to n successor nodes $J_1, \ldots, J_n$—one for each WSG. To avoid introducing m more nodes, we assume that source node i has external supply H_i.

Following each WSG node $j \in J$ is a subnetwork of size $|J| \times |P|$. Let the flow through this subnetwork represent the schedule of workers who are based at j (many arcs, such as those from the n WSGs to periods 47 and 48, are not shown). In the example, a worker whose home base is J_1 cannot be assigned to J_n.

A node in the subnetwork is denoted by (j, p), and each entering arc has capacity $c(a) = 1$ (not shown). These arcs are depicted in bold because they really represent up to m arcs, one for each worker. For each $i \in I$, we need individual arcs that go from each WSG node J_k to each subnetwork node (j, p) and from each node (j, p) to the other nodes in the subnetwork as long as $j \in O(J_k)$ and $p \in P(i)$; that is, as long as the WSG restrictions are satisfied and i is scheduled to work in period p.

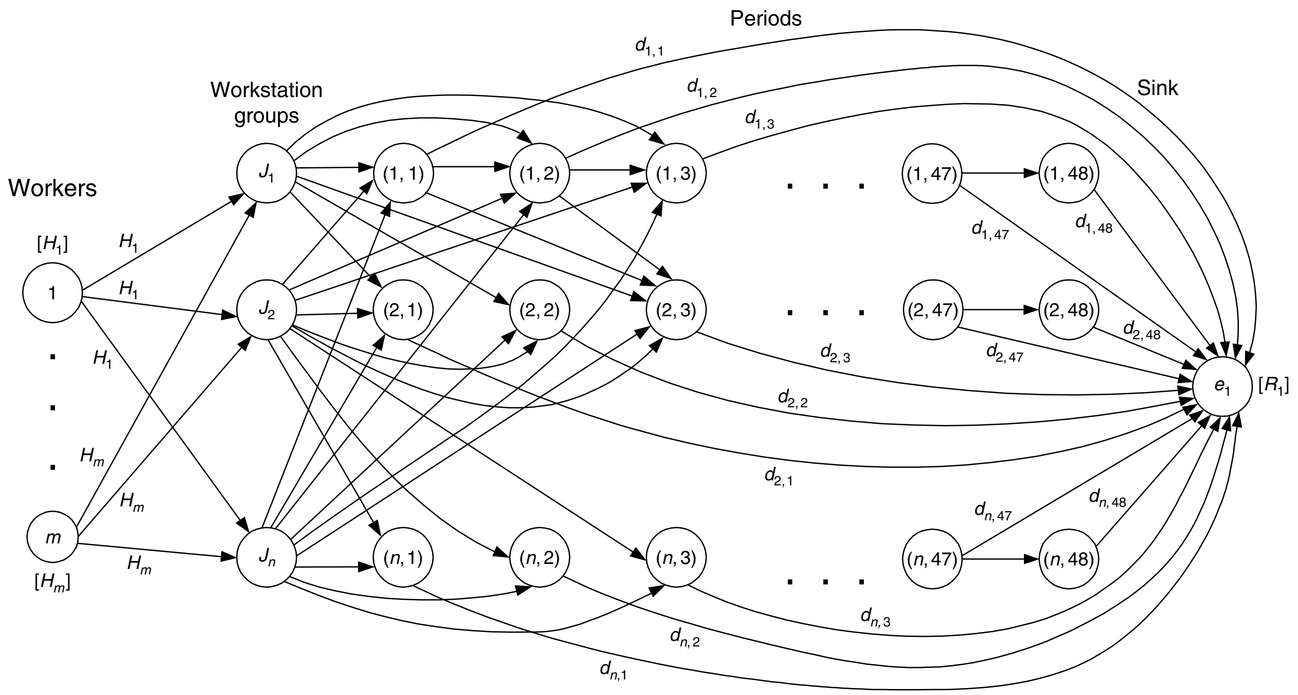
At the far right of Figure 8 there is a single sink node e_1 that is connected to each node in the subnetwork (not all arcs are shown). The capacity of the arc joining (j, p) to e_1 is d_{jp} and the capacity of each arc leaving source node i is H_i. Thus, the maximum flow into the sink node e_1 is $\sum_{j\in J}\sum_{p\in P} d_{jp}$. Note that we do not require that all the flow out of source node $i \in I$ go to only one WSG node $j \in J$. This is a relaxation of constraint (1g), which means that $\mathbf{x}$ is being treated as a continuous variable. To finish the description of the general WGAP, we set the parameter $R_1 =$ quantity of demand covered.

These developments lead to the following observations:

(1) If $R_1 = \sum_{j\in J}\sum_{p\in P} d_{jp}$, we are asking: Can all the demand be covered with the existing restrictions; i.e., can sufficient flow be sent through the network so that no period is left uncovered? By setting R_1 to a smaller value, say, $\sum_{j\in J}\sum_{p\in P} d_{jp} - \sum_{j\in J}\sum_{p\in P} s_{jp}$, we are equivalently asking: Can this portion of the demand be covered?

(2) There is always a solution in which there is no flow between any two nodes in the subnetwork. This follows because such flow does not contribute to the total amount of flow that arrives at e_1.

Figure 8 Multicommodity Flow Network Used in the Proof of Theorem 1



(3) The constructed instance of WGAP has a *yes* answer if and only if the recognition version of D*m*CIF does.

(4) The network in Figure 8 can be constructed in $O(|I| \times |J| \times |P|)$ time so the transformation is polynomial. Moreover, any candidate solution to WGAP can be evaluated in $O(|I| \times |P|)$ time, so it is in NP.

As a consequence, the version of the general WGAP where workers can be assigned to more than a single WSG is NP-complete. Because that problem is a relaxation of model (1a)–(1f), the result follows. □

We now consider what happens when the WSG assignments are fixed prior to scheduling the workers and the home-base plurality constraint is absent.

Corollary 1. *If x_{ij} is fixed for all $i \in I$, $j \in J$, then WGAP* (1a)–(1f) *reduces to a series of $|P|$ transportation problems.*

Proof. Fixing the **x** variables means that each worker is assigned to a home base. The general WGAP then reduces to

$$\text{Minimize} \quad z = \sum_{j\in J}\sum_{p\in P} s_{jp} \tag{5a}$$

$$\text{subject to} \quad \sum_{j\in J} y_{ijp} \le 1, \quad \forall\, i \in I, \forall\, p \in P(i) \tag{5b}$$

$$\sum_{i\in I} y_{ijp} + s_{jp} \ge d_{jp}, \quad \forall\, j \in J, \forall\, p \in P \tag{5c}$$

$$y_{ijp} \in \{0, 1\}, \quad \forall\, i \in I, \forall\, j \in J, \forall\, p \in P(i),$$

$$s_{jp} \ge 0, \quad \forall\, j \in J, \forall\, p \in P, \tag{5d}$$

where the **y** variables only exist if they satisfy constraint (1e).

Model (5a)–(5d) decomposes by p. To see how a transportation problem results for each $p \in P$, we identify a bipartite network that has one source node with unit supply for each i if $p \in P(i)$, one dummy source node denoted by zero with supply $\sum_{j\in J} d_{jp}$, and n destination nodes with demand d_{jp}, $j = 1, \ldots, n$. An arc exists between source node i and destination node j if worker i is permitted to move to WSG j. All such arcs have zero cost.

The supply at node 0 is used to cover any unmet demand in period p and the leaving arcs all have unit cost. The associated flow is represented by the variable s_{jp}. Minimizing the cost of satisfying all the demand gives rise to a transportation problem. □

Corollary 2. *When x_{ij} is fixed for all $i \in I$, $j \in J$, WGAP given by* (1g), (5a)–(5d) *remains NP-complete.*

Proof. When constraint (1g) is added to (5a)–(5d), it is no longer possible to decompose the problem by period. If worker i is assigned to WSG j (that is, $x_{ij}=1$), this constraint requires that the sum of the flow leaving WSG j and going to any other WSG in $J\backslash\{j\}$ be less than flow going to the nodes (j,p) in the subnetwork associated with worker i. The resultant problem is equivalent to what is called integral flow with bundles and is NP-complete in the strong sense (Garey and Johnson 1979).

To see the equivalence, assume that there are two WSGs and m workers, all of whom are assigned to WSG J_1 (this will be the source node). Let bundle B_i be the set of arcs leaving the source node J_1 and going to nodes (k,p) for $k\in J\backslash\{j\}$, $p\in P(i)$. Also, let B_{m+1} be the set of all arcs leaving the source node and B_{m+2} all the other arcs in the network. For $1\le i\le m$, we have $\sum_{a\in B_i} f(a)\le H_i/2$, for bundle $m+1$, we have $\sum_{a\in B_{m+1}} f(a)\le \sum_{i\in I} H_i/2$, and for bundle $m+2$, we have $\sum_{a\in B_{m+2}} f(a)\le\infty$. At the sink node e_1, the requirement R_1 is $\sum_{j\in J}\sum_{p\in P} d_{jp}-\sum_{j\in J}\sum_{p\in P} s_{jp}$. Thus, in this special case of Problem (1g), (5a)–(5d) has a solution with no more than $\sum_{j\in J}\sum_{p\in P} s_{jp}$ periods uncovered if and only if there is a function $f\colon A\to Z_0^+$ that satisfies the bundle requirements for the integral flow problem defined above. □

Because there is nothing in the proof of Corollary 2 that depends on worker movement restrictions $I(j)$, we have the following.

Corollary 3. *When x_{ij} is fixed for all $i\in I$, $j\in J$, and movement between WSGs is unrestricted; i.e., the set $I(j)=J$ for all $j\in J$, the task assignment problem given by* (1g), (5a)–(5d) *remains NP-complete.*

4.2. Solving the Integer Programming Formulation of WGAP

Initial testing indicated that instances of model (1) with more than about 70 workers and three WSGs with any number of restrictions could not be solved with CPLEX 9.0 within an hour. To generate feasible solutions within an acceptable amount of time, it was therefore necessary to design an approximation method. In the model, there are three sets of decision variables: $\mathbf{x}=(x_{ij})$, $\mathbf{y}=(y_{ijk})$, and $\mathbf{s}=(s_{jk})$. (Note that we did not include constraints (3) and (4) in our code.) When the $\mathbf{x}$ variables are fixed, implying that the home base of each worker has been decided, the model can be solved easily. Choosing the home base is a critical decision because it determines the permissible task assignments, which may dramatically affect the objective value.

We take a two-stage approach to the problem. In the first stage, we solve the LP relaxation of WGAP to obtain the solution $(\bar{x}_{ij},\ \forall\, i\in I,\ j\in J)$ and $j^*\in \arg\max\{\bar{x}_{ij}\colon j\in J\}$ for all $i\in I$. Ties in the determination of j^* are broken arbitrarily. For a given fraction $\rho\in[0,1]$, we then fix the home base of $\rho\times 100\%$ of the workers, beginning with those whose fractional solutions $\bar{x}_{ij^*}$ are the largest. Fixing all of the $\mathbf{x}$ variables would sacrifice too much in solution quality and has not been necessary in practice.

Variable Fixing Algorithm.

Input: Set of workers I, set of WSGs J, fixing ratio $0\le\rho\le 1$

Output: Set of home bases $B=\{b_i$, for all $i\in I\}$ for the workforce

Initialization: Let $\omega=\rho\times|I|$ be the number of workers to be fixed, set $b_i=-1$ for all $i\in I$, $L=\varnothing$, and $Z=\varnothing$.

Step 1. Solve model (1) as an LP to get $(\bar{x}_{ij},\ \forall\, i\in I,\ j\in J)$.

Step 2. For all $i\in I$, $z_i=\max\{\bar{x}_{ij}\colon j\in J\}$ and $Z\leftarrow Z\cup\{z_i\}$.

Step 3. Reorder all $z_i\in Z$ from largest to smallest; break ties arbitrarily.

Step 4. Identify the first ω elements in Z and put their indices into L.

Step 5. For all $i\in L$, set $b_i\in\arg\max\{\bar{x}_{ij}\colon j\in J\}$ and construct B.

If $b_i\neq -1$ in the output set B, then the home base of worker i is set such that $x^*_{ib_i}=1$. In the second stage of our procedure, model (1) is solved for $\mathbf{y}$ and $\mathbf{s}$ and the remaining $\mathbf{x}$ variables. The collective steps for obtaining a solution to the original problem are as follows.

Iterative Algorithm.

Input: Demand in form of equipment schedule $\{d_{jp}\colon \forall\, j\in J,\ p\in P\}$, movement restrictions matrix R, value of logical parameter SEQ, threshold parameter ρ_0 for calling the variable fixing algorithm, and all other data elements that define the long-term planning model embodied in SOS.

Output: Size and composition of permanent workforce W^*, including bid jobs, home base $\{(x^*_{ij})\colon \forall\, i,j\}$, and task assignments $\{(y^*_{ijp})\colon \forall\, i,j,p\}$

Step 1 (Relaxed solution). Solve the extended shift-scheduling component of SOS to generate the workforce W_{SOS} and the shifts for each worker. Put $W^* \leftarrow W_{\text{SOS}}$.

Step 2 (Find home base and task assignments). Set up model (1) using W_{SOS}, $\{d_{jp}\colon \forall\, j, p\}$, and R as input.

If $(|I| \times |G| \le \rho_0)$, then set $\rho = 0$; otherwise, set $\rho \in [0.4, 0.6]$.

Call Variable Fixing Algorithm to get home bases B.

Solve model (1) with B to get optimal home bases, task assignments, and uncovered demand: $\{(x^*_{ij})\colon \forall\, i, j\}$, $\{(y^*_{ijp})\colon \forall\, i, j, p\}$, $\{(s^*_{jp})\colon \forall\, j, p\}$.

If $U \equiv \{(s^*_{jp})\colon \forall\, j, p\} = \varnothing$, stop; otherwise, go to Step 3.

Step 3. If (SEQ $=$ ⟨true⟩), put $d^{\text{new}}_{jp} \leftarrow s_{jp}$ for all $p \in P$, $j \in J$ and call the sequential procedure to get W_{SEQ}. Put $W^* \leftarrow W^* \cup W_{\text{SEQ}}$, update $\{(x^*_{ij})\colon \forall\, i, j\}$ and $\{(y^*_{ijp})\colon \forall\, i, j, p\}$, and stop.

Otherwise, put $d_{jp} \leftarrow d_{jp} + s_{jp}$ for all $p \in P$, $j \in J$, and go to Step 1.

At Step 1, the relaxed tour-scheduling problem is solved without WSG movement restrictions to get a tentative permanent workforce, W_{SOS}. The home-base assignments and task assignments are made at Step 2, where the variable fixing algorithm is called if the size of the problem, as measured by the number of workers times the number of WSGs $(|I| \times |G|)$, is greater than some threshold ρ_0. We settled on $\rho_0 = 200$ as a guideline. If there is any uncovered demand, then we go to Step 3 and use either the sequential procedure to find the number of additional workers required to satisfy that demand, or return to Step 1 with the original demand augmented by the uncovered demand, and repeat the entire process.

In the development of the algorithm, we tried several variations, including the replacement of the demand variables s_{jp} with shift variables, and an adaptive strategy for setting the fraction ρ in the variable fixing algorithm. In the case of the former, the objective was to minimize the number of additional shifts needed to satisfy the uncovered demand. In the case of the latter, no specific rule worked best. In general, we found that values of ρ smaller than 0.4 led to instances that were almost as difficult as when no x_{ij} variables were fixed; for values larger than 0.6, we found that the solution quality was not much better than obtained by fixing all the x_{ij} variables.

Example (Cont'd.). Consider again the network in Figure 4. When the iterative algorithm is applied, the size of the workforce found by SOS in Step 1 is $W_{\text{SOS}} = 97$. The corresponding IP has 1,529 variables and 1,120 constraints, and terminated with a 1.5% optimality gap after the five-minute time limit was reached. Using the 97 workers as input, model (1) has 27,671 variables and 32,907 constraints. Its LP relaxation was solved at Step 2 in 58 seconds, yielding an objective function value of zero.

For a fixing rate $\rho = 0.6$, model (1) is re-solved as an IP with 60% of the **x** variables fixed, giving a problem with 24,913 variables and 18,694 constraints. The solution was found after about 25 minutes with an optimality gap of 0%, and indicated that seven periods were uncovered. At Step 3, the sequential procedure is called to generate additional workers to handle the demand in U. In about a second, a solution was found with $W_{\text{SEQ}} = 1$, so the total workforce needed to operate the facility is $W^* = W_{\text{SOS}} + W_{\text{SEQ}} = 98$. By way of comparison, when the sequential procedure was used by itself, a solution of $W^* = 111$ was found in about eight minutes.

5. Computational Experience

To assess the performance of the two procedures, a series of tests was conducted using data provided by USPS Dallas P&DC. For each of the three data sets, four closely related scenarios were generated and compared. All scenarios had the same number of

Figure 9 Movement Restrictions Networks for Data Set 1

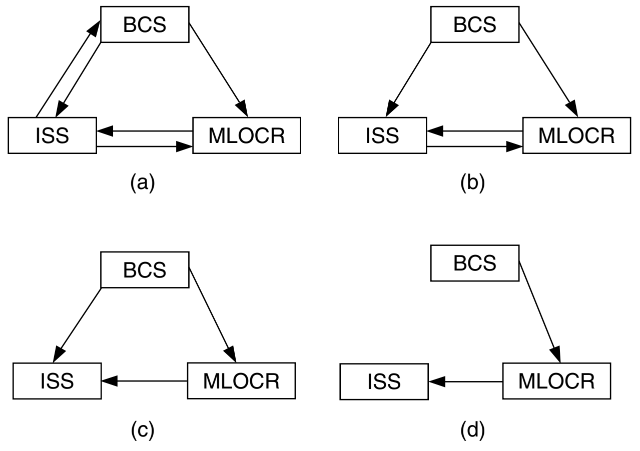
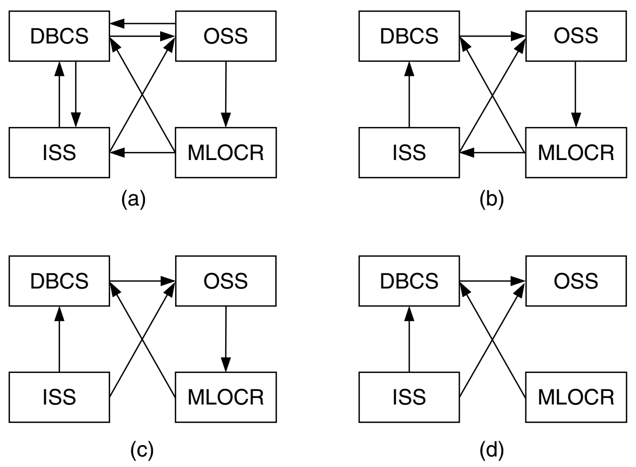


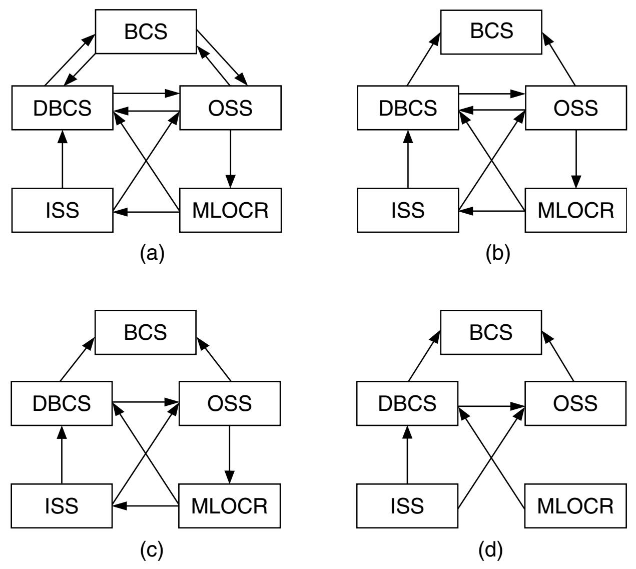
Figure 10 **Movement Restrictions Networks for Data Set 2**



WSGs, but different restriction matrices *R*. The networks for data set 1 are shown in Figure 9 and are displayed in descending order according to the number of links in each. Similarly, the networks for data sets 2 and 3 are shown in Figures 10 and 11, respectively.

All computations were performed on a PC with dual Xeon 1.8 GHz CPU, 1 GB memory, running SuSE Linux 9.0. The implementation was done in Java SDK 1.3, which calls CPLEX 9.0 to solve the IPs. The barrier option was used at the root node of all search trees to solve the first LP-relaxation.

Figure 11 **Movement Restrictions Networks for Data Set 3**



5.1. General Results

Table 2 gives the tour-scheduling solutions found by SOS without the WSG movement restrictions; i.e., the WSG networks are completely connected. In this case, the aggregate demand can be used to solve the tour-scheduling problem, and it will always be possible to postprocess the results to obtain a feasible solution. When restrictions are present, these solutions provide a lower bound (LB) on the cost of the workforce. Although the LB is the same for all scenarios in a particular data set because the demand is the same for each scenario, its quality is a function of the number of links in the corresponding network and may be different for each scenario. When running SOS, the FTR/PTF ratio was set to be ≥ 4.

The staff-scheduling results obtained from the sequential and iterative procedures are given in Table 3. In the case of the iterative algorithm, we set SEQ = $\langle$true$\rangle$ at Step 3 because this always proved to be the better choice. The column *Number of workers* indicates the total number of employees needed to satisfy all the demand when the WSG restrictions are enforced. The next two columns indicate how the workforce is broken down with respect to FTRs and PTFs. By design, each FTR was scheduled for exactly 40 hours, while the PTFs were assigned up to 24 hours per week on average, with their number of workdays and shift lengths varying in the solution. This is the reason why the staffing costs are different among the various scenarios even though the workforce composition is the same. For data set 2, for example, the solutions generated by the sequential procedure for scenarios a, c, and d all call for 63 FTRs and 15 PTFs, but the total staffing costs differ by a fraction of a percent.

The quality of the solutions in Table 3 can be judged, in part, by their distance from the LB provided by the SOS solutions in Table 2. From the data in Column 6, the average gap between the solutions found by the sequential procedure and the LB

Table 2 **LB for Each Data Set**

Data set	Number of workers	Number of FTRs	Number of PTFs	Total staffing cost ($)	Idle time (%)	Solution time (sec)
1	205	165	40	204,539	11	305
2	72	58	14	71,922	9.8	302
3	45	36	9	44,701	11	304

Table 3 Results for Sequential and Iterative Procedures

	Sequential procedure						Iterative procedure					
Scenarios	Number of workers	Number of FTRs	Number of PTFs	Staffing cost ($)	Gap with LB (%)	Idle time (%)	Number of workers	Number of FTRs	Number of PTFs	Staffing cost ($)	Gap with LB (%)	Idle time (%)
Data set 1												
a	221	178	43	220,427	7.77	17.4	205	165	40	204,539	0	11.0
b	229	184	45	23,521	9.28	18.3	205	165	40	04,539	0	11.0
c	235	189	46	229,996	12.45	20.6	205	165	40	204,539	0	11.0
d	235	189	46	230,159	12.53	20.7	215	174	41	214,866	5.05	15.0
Data set 2												
a	78	63	15	77,355	7.55	16.0	72	58	14	71,922	0	9.8
b	80	65	15	79,171	10.08	17.8	74	60	14	74,170	3.13	12.4
c	78	63	15	77,157	7.28	15.7	75	61	14	75,294	4.69	12.4
d	78	63	15	77,297	7.47	15.9	75	61	14	75,294	4.69	13.5
Data set 3												
a	50	41	9	49,842	11.50	19.9	45	36	9	44,701	0	11.0
b	54	44	10	52,699	17.89	24.0	45	36	9	44,701	0	11.0
c	54	45	9	53,484	19.65	25.0	45	36	9	44,701	0	11.0
d	55	45	10	53,449	19.57	24.9	53	44	9	53,693	20.12	24.1

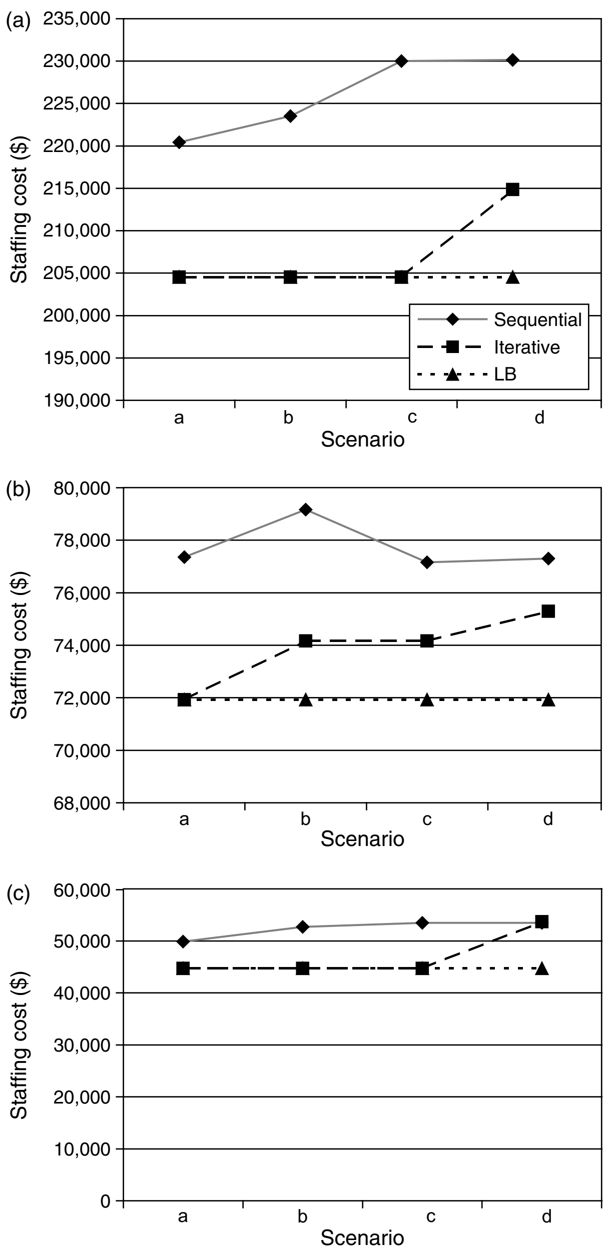
is computed to be about 12%. In general, as the number of links in the networks shown in Figures 9–11 decreases, the gap increases, although not uniformly. What can be observed from the data is that the ability to merge WSGs into clusters helps to improve the solution quality. In particular, see scenarios a and b for data set 1, and scenario a for data set 3. For data sets 1 and 3, no merging is possible for scenarios c and d. For data set 2, merging is only possible for scenario a, but no advantage is gained, at least with respect to scenarios c and d.

The solutions found by the iterative procedure were significantly better than those found by the sequential procedure. Using a combination of a 1% optimality gap and a 30-minute time limit as the stopping criteria, and variable fixing fractions of 0.6, 0.6, and 0.2, respectively, for the three data sets, the next-to-last column in Table 3 indicates that seven out of the 12 instances reached their LB when model (1) was solved. Thus, the optimal solution was obtained at Step 2 without requiring any additional computations. For these scenarios, the data in Column 7 represent the true optimality gap for the sequential procedure. Of course, when the *Gap with LB* reported in Table 3 is greater than zero, we have no way of confirming whether the current solution actually minimizes the cost of the workforce, the original objective. Figure 12 plots the results for the two procedures.

For the seven scenarios solved to optimality, it might appear that the WSG movement restrictions played no part in the problem. The number of workers and the cost, however, do not give the full picture. The fact that there is roughly 10% idle time in the LB solutions (see Table 2) suggests that there are multiple optima to the unrestricted task assignment problem. The one found by the algorithm built into SOS rarely, if ever, satisfied the WSG restrictions, hence the need to solve model (1). In the remaining five scenarios, the gap between the iterative solution and the LB ranged from approximately 3% to 20%. In only one scenario did the sequential procedure do better.

Another observation that can be made about the iterative solution is that it is highly correlated with the number of links in the network—the fewer the links, the larger the gap. Fewer links mean more restrictions, so it is less likely that the relaxed solution will be feasible when the movement restrictions are considered in model (1). Situations in which the sequential procedure might do better occur when the difference between the SOS solution and the iterative solution is large, which would be evidenced by a large amount of uncovered demand at Step 2. In particular, when there are few links in the network, the optimal size of the workforce may be far from the SOS solution, so uncovered demand will be high. Because the corresponding instances of model (1) are more difficult

Figure 12 Comparison of Sequential and Iterative Procedures



to solve for these scenarios, the solutions obtained may not be optimal, especially when a high fraction of variables is fixed (i.e., ρ close to 1). In any case, the workers added at Step 3 usually have a very large amount of idle time in their schedules, which suggests that the sequential approach may provide for a smaller workforce.

5.2. Variable Fixing Results

The computational results obtained by attempting to solve model (1) with both CPLEX and the variable fixing algorithm are displayed in Table 4. For all scenarios and all values of ρ, a 30-minute time limit was placed on the operations performed by CPLEX, which included a call to its built-in feasibility heuristic every 10 nodes. When the variable fixing algorithm was used, additional time was allowed at Step 1 of the iterative algorithm for solving the LP (which could take several minutes) and at Step 3 for running the sequential procedure (which took a few seconds at most).

The data in Columns 2–4 indicate that large-scale instances are the norm even when there are relatively few restrictions in the network. Nevertheless, Column 7 shows that seven out of the 12 scenarios were solved to optimality within 30 minutes by CPLEX. A 0% gap was always achieved, with the feasible solution always being found by the CPLEX heuristic. In the same amount of time, a feasible solution with about a 90% gap was obtained for scenario d in data set 3. For the four remaining scenarios, CPLEX was not able to find an integer-feasible solution. The corresponding LP bounds were all zero or close to zero, and closing the (relatively large) gaps proved difficult.

The variable fixing rates for the different data sets are listed in Column 10. For larger instances, it is generally more difficult to find optimal or even feasible solutions, so more variables must be fixed to make them tractable. As shown in Column 11, the variable fixing (V-F) algorithm found the optimal solution for the same seven scenarios that were solved by CPLEX (i.e., when $\rho = 0$). With respect to solution time, variable fixing does not provide any advantage for these cases because the LP-relaxation needs to be solved first to decide which variables to fix. The real advantage is apparent for the remaining cases where good feasible solutions were found. For scenario d in data set 3, for example, the number of uncovered periods fell from 169 to 60.

For the variable fixing algorithm, the fraction ρ must be decided in advance. If this value is too low, the resultant problem may be too difficult to solve; if it is too high, the quality of the solution may suffer because many good assignments may be ruled out. Table 5 gives the solutions for different values of ρ for data set 2. The results for the other two data sets are

Table 4 Performance of Variable Fixing Algorithm for Model (1)

Scenario	Number of columns	Number of rows	Number of non-zeros	LP objective value	LP time (sec)	CPLEX objective value	CPLEX time (sec)	Optimal gap (%)	Fix rate	V-F solution	Solution time (sec)
Data set 1											
a	32,883	25,304	167,779	0	12.4	0	173.0	0	0.6	0	703.7
b	30,798	23,718	145,067	0	12.1	0	167.4	0	0.6	0	175.9
c	30,769	33,600	159,940	0	27.3	0	151.2	0	0.6	0	145.4
d	30,770	37,671	159,732	9.9	26.2	—	1,800.0	—	0.6	34	1,881.6
Data set 2											
a	15,603	14,497	109,186	0	10.3	0	1,772.5	0	0.6	0	1,265.1
b	15,651	19,515	113,230	0	18.5	—	1,800.0	—	0.6	5	1,856.4
c	15,651	19,488	108,963	0	15.8	—	1,800.0	—	0.6	7	1,835.2
d	15,651	19,416	98,685	0	19.8	—	1,800.0	—	0.6	15	2,026.9
Data set 3											
a	12,607	12,359	102,250	0	13.3	0	618.9	0	0.2	0	626.2
b	12,611	15,115	102,830	0	14.3	0	1,024.2	0	0.2	0	1,029.1
c	12,611	15,070	94,716	0	17.1	0	1,324.4	0	0.2	0	1,659.3
d	12,611	15,003	84,649	13.8	20.9	169	1,808.4	89.45	0.2	60	1,834.8

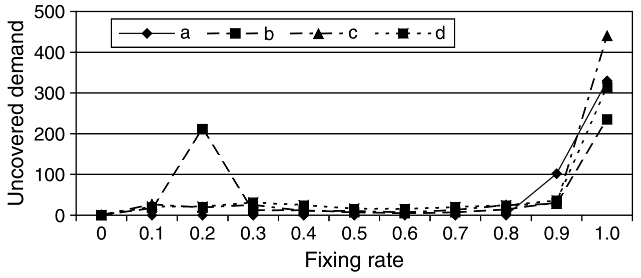
Table 5 Influence of Fixing Rate on Uncovered Demand

	Value of fixing parameter, ρ										
Scenario	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
a	0	0	0	0	0	0	0	0	0	102	330
b	—	18	211	12	12	7	5	7	14	31	235
c	—	27	19	26	10	10	7	13	25	28	440
d	—	21	21	31	25	16	15	20	24	36	314

not shown because most of the instances were solvable with $\rho = 0$.

Figure 13 plots the uncovered demand as a function of the fixing rate ρ. As expected, solution quality deteriorates rapidly for large values of ρ, say, above 0.8. When the fixing rate is low, the solutions are good for all scenarios except b when $\rho = 0.2$. Although this appears to be an anomaly, it becomes much more difficult, in general, to find good solutions as the problem size grows, so setting ρ too low may yield poor results. For this data set, the best solutions were obtained by setting $\rho = 0.6$ for all instances. However, the optimal fixing rate is not necessarily the same for all scenarios and problem sizes. In our experience, however, solution quality was relatively stable for ρ between 0.4 and 0.6.

Figure 13 Parametric Analysis of Variable Fixing Fraction, ρ



6. Conclusions

Repositioning workers during the day to increase their productivity is often constrained by a combination of physical barriers, union agreements, and supervisory preferences. The most common way of dealing with these restrictions is to ignore them and then use either overtime or temporary employees to ensure that all demand is covered. In this paper, two procedures were presented that directly take into account the need to limit the movement of workers when assigning them tasks over the day. Extensive computational testing showed that the iterative procedure with variable fixing provided the better results. Problem instances with hundreds of employees and up to five WSGs were solved to (near) optimality within 30 minutes.

Although these results are promising, we would still like to be able to solve larger instances with the same degree of accuracy. One alternative approach would be to develop a column generation scheme based on decomposing the problem by either WSG or employee. We have had some success for the case in which the workforce is known and, hence can be treated as input. Another area for future research concerns the design of a postprocessor for reducing the number of shifts and workers. Initial attempts at implementing a tabu search algorithm for this purpose were not successful because of the complexity of the neighbor definition and the fact that before a full-time employee can be eliminated, all five of his or her shifts must be converted to idle time.

References

Bard, J. F. 2004a. Staff scheduling in high volume service facilities with downgrading. *IIE Trans. Scheduling Logist.* **36**(10) 985–997.

Bard, J. F. 2004b. Selecting the appropriate input data set when configuring a permanent workforce. *Comput. Indust. Engrg.* **47**(4) 371–389.

Bard, J. F., H. W. Purnomo. 2005. A column generation-based approach to solve the preference scheduling problem for nurses with downgrading. *Socio-Econom. Planning Sci.* **39**(3) 193–213.

Bard, J. F., C. Binici, A. H. deSilva. 2003. Staff scheduling at the United States Postal Service. *Comput. Oper. Res.* **30**(5) 745–771.

Bechtold, S. E., M. J. Brusco. 1994. Working set generation methods for labor tour scheduling. *Eur. J. Oper. Res.* **74** 540–551.

Bechtold, S. E., L. W. Jacobs. 1990. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Sci.* **36**(11) 1339–1351.

Berman, O., R. C. Larson, E. Pinker. 1997. Scheduling workforce and workflow in a high volume factory. *Management Sci.* **43**(2) 158–172.

Billionnet, A. 1999. Integer programming to schedule a hierarchical workforce with variable demands. *Eur. J. Oper. Res.* **114** 105–114.

Campbell, G. M., M. Diaby. 2002. Development and evaluation of an assignment heuristic for allocating cross-trained workers. *Eur. J. Oper. Res.* **138**(1) 9–20.

Dawid, H., J. König, C. Strauss. 2001. An enhanced rostering model for airline crews. *Comput. Oper. Res.* **28** 671–688.

Emmons, H., R. N. Burns. 1991. Off-day scheduling with hierarchical worker categories. *Oper. Res.* **39**(3) 484–495.

Ernst, A. T., H. Jiang, M. Krishnamoorthy, D. Sier. 2004. Staff scheduling and rostering: A review. *Eur. J. Oper. Res.* **153**(1) 3–27.

Even, S., A. Ital, A. Shamir. 1976. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.* **5** 691–703.

Franz, L. S., J. L. Miller. 1993. Scheduling medical residents to rotations: Solving the large-scale multi-period staff assignment problem. *Oper. Res.* **41**(2) 269–279.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

Jarrah, A. I. Z., J. F. Bard, A. H. deSilva. 1994. Solving large-scale tour scheduling problems. *Management Sci.* **40**(9) 1124–1144.

Lin, C. K. Y., K. F. Lai, S. L. Hung. 2000. Development of a workforce management system for a customer hotline service. *Comput. Oper. Res.* **27**(10) 987–1004.

Malhotra, M. K., L. P. Ritzman. 1994. Scheduling flexibility in the service sector: A postal case study. *Production Oper. Management* **3** 100–117.

Misra, S., E. J. Pinker, R. A. Shumsky. 2004. Salesforce design with experience-based learning. *IIE Trans. Logist. Scheduling* **36**(10) 941–952.

Rekik, M., J.-F. Cordeau, F. Soumis. 2004. Using Benders decomposition to implicitly model tour scheduling. *Ann. Oper. Res.* **128** 111–133.

Ritzman, L. P., L. J. Krajewski, M. J. Showalter. 1976. The disaggregation of aggregate manpower plans. *Management Sci.* **22**(2) 1372–1380.

Sherali, H. D., J. C. Smith. 2001. Improving discrete model representations via symmetry considerations. *Management Sci.* **47**(10) 1396–1407.

Wan, L. 2005. Staff planning and scheduling in the service industry. Doctoral dissertation, Graduate Program in Operations Research & Industrial Engineering, The University of Texas, Austin, TX.