The Dissertation Committee for Lin Wan Certifies that this is the approved version of the following dissertation:

Staff Planning and Scheduling in the Service Industry: An Application to US Postal Service Mail Processing and Distribution Centers

Committee:

Jonathan F. Bard, Supervisor

J. Wesley Barnes

John Hasenbein

Erhan Kutanoglu

Leon Lasdon

Staff Planning and Scheduling in the Service Industry: An Application to US Postal Service Mail Processing and Distribution Centers

by

Lin Wan, B. Eng.; B. Law; M. Eng.

Dissertation

Presented to the Faculty of the Graduate School of

University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2005

This dissertation is dedicated to my grandfather.

Acknowledgments

There are many people that have played important role in supporting me in writing this dissertation. First and foremost is Dr. Jonathan F. Bard, who not only served as my advisor but also encouraged and challenged me throughout my doctoral study. He patiently guided me through the dissertation process and helped me to pursue my ideas.

My thanks also go to all other members of my doctoral committee, Dr. Wesley Barnes, Dr. John Hasenbein, Dr. Erhan Kutanoglu, and Dr. Leon Lasdon, for their invaluable comments and suggestions.

The friendship of Hadi Purnomo is much appreciated and has led to many interesting and good-spirited discussions relating to this research. I am also grateful to my friends Guzin Bayraksan and Yang-Chi Chen for their help and support.

I would also like to thank Planmatics Inc. for providing support for my study on staff scheduling in USPS. Mr. Andres Granados, Ms. Nilufer Uyanik and Mr. Robert Carignan are especially thanked for providing all the data, documents, and their opinions on the research. I also want to thank Zhan Goloborodko, Shankar Dhanaraj, and Mehmet Gurbuz for helping me to implement the results of my research and perform tests.

Last, but not least, I would like to thank my wife Lily for her support and love in the past few years. I am also grateful to my parents, Huiyi and Meilan, for their endless love, care, and support.

Staff Planning and Scheduling in the Service Industry: An Application to US Postal Service Mail Processing and Distribution Centers

Lin Wan, PhD

University of Texas at Austin, 2005

Supervisor: Jonathan F. Bard

This research addresses weekly personnel planning and scheduling problems that arise at various service facilities staffed by full-time and part-time employees. In response to demand fluctuations, expected leave, training assignments, and other contingencies, weekly adjustments are often required to better match available personnel with demand over the planning horizon. Unlike manufacturing where uniform 8-hour shifts are the rule, service organizations may experience several busy periods during the day that do not fit a standard shift. In such cases, supervisors must adjust employee schedules by assigning overtime, increasing the number of part-time hours, and calling in temporary workers. The situation is complicated by union contracts, labor rules, and company policies.

To find solutions that can be implemented in a real-world environment, a two-phase approach was developed. In the first phase, the adjustment problem is formulated as a large-scale integer program and solved to generate the adjusted shift schedules. In the second phase, the shift schedules are post-processed to provide daily task assignments for each worker.

An integrated model that combines the shift scheduling and task assignment is also proposed to incorporate base group requirements and movement restrictions. Since only relatively small problems could be solved by commercial solver, two decomposition heuristics—network splitting and column generation—were designed to deliver good feasible solutions in a more timely manner. In conjunction with this problem, the impact of the workgroup restrictions on long-term staff planning was also investigated.

An analysis of the problems is presented for an application involving weekly and long-term scheduling at a mail processing and distribution center. The results indicate that high quality solutions can be obtained within a reasonable amount of time.

Table of Contents

Page

List of Tables

List of Figures

Chapter 1

Introduction

A distinguishing characteristic of organizations in the service industry is that customer demand varies sharply from one period to the next. Hour-to-hour and day-to-day fluctuations make staff scheduling much more difficult than in manufacturing where demand, for the most part, is steady and predictable. In general, the goal of management is to find the best mix of hourly employees so that demand is satisfied at minimum cost. The problem is complicated in the service industry by labor laws, union contracts, and local company policies.

The purpose of this research is to investigate the issues surrounding staff scheduling in organizations that face changing demand patterns that peak for short periods during the day. Examples of such organizations include hospitals, restaurants, airlines, and call centers as well as mail processing and distribution centers (P&DCs), the application of interest. In particular, the United States Postal Service (USPS) is the third largest employer in the U.S. with over 800,000 operational and clerical personnel on its payroll. When a letter is posted, its first stop is the local P&DC where it is cancelled, sorted, and then dispatched to either the addressee or another P&DC. More than a third of the USPS workforce is needed to run the 275 P&DCs nationwide, a number that has remained steady despite the decline in mail volume and the introduction of advanced technology in the form of optical character readers, barcode sorters, and computerized material handling systems.

Mail processing centers are like high volume factories that run 24 hours a day, 7 days a week, and are staffed by a skilled workforce comprising full-time, part-time, and casual employees. On a typical day, a medium facility might receive as many as 5 million letters, 0.5 million flats, and thousands of parcels. The analysis in this research begins with a fixed workforce whose composition has been determined to meet long-term performance and budgetary goals. For the USPS, the sizing of the career workforce is a

well-studied problem, primarily because of the constant pressure it faces to be financially self-supporting. Over the last 25 years, several comprehensive analyses have been undertaken to determine optimal staffing levels as well as the optimal equipment configurations; e.g., see Bard et al. (1993), Berman et al. (1997), Jarrah et al. (1994), Showalter et al. (1977). Most recently, the USPS has embarked on a new effort to streamline its operations and reduce the number of employees nationwide. At the heart of this effort is a decision support system that includes a long-term staff planning module [called the Scheduling Optimization System (SOS)] developed by Bard et al. (2003). In this research, a second module is designed to make weekly adjustments to the staff schedule to account for changes in demand, seasonal factors, and planned leave.

1.1 Hierarchy of Staff Planning Problems

Typical issues in personnel scheduling include days-off assignments, lunch break determination, leave management, part-time flexible labor and casual labor management, overtime allocation, personnel assignment to various work centers, as well as the difficulties associated with large fluctuations in hourly and daily demand. Some variations of the basic problem include a non-homogeneous workforce in terms of skill categories, various labor requirements such as maximum work stretches, break definitions, off-days and off-weekend policies (Aykin 1996, Bechtold and Jacobs 1990, Burns and Carter 1985, Emmons 1985), and management considerations such as customer priority, service standards, start time rules, and objective function definitions (Beaumont 1997, Mason et al. 1998).

In the most general sense, workforce planning problems can be viewed temporally and decomposed along the time axis (see Fig. 1.1). This suggests a hierarchical analytic approach. In the long run, the goal is to find the optimal size of the workforce as well as regular weekly schedules or tours for each employee in a given skill category. This means specifying the work days, their length, the daily start time, and the lunch break. At processing and distribution centers these specifications constitute a “bid job” (Jarrah et al. 1994).

At the next level in the hierarchy, given the regular workforce, adjustments must be made on a weekly basis to account for planned absenteeism and expected departures from average demand. To accomplish this, critical resources must be tracked and evaluated. The goal is to provide weekly schedules that balance overtime and the use of part-time labor so that all requirements are met at minimum cost and at minimum deviation from the bid jobs. This can be thought of as a replanning problem and is the focus of this research.

Finally, at the day-to-day level, supervisors must deal with unplanned absenteeism, machine breakdowns, and unexpected spikes in demand; i.e., uncertainty. This is a real-time scheduling problem that falls under the heading of *control*. In the airline industry where bad weather, for example, can occasion flight delays and cancellations (Bard et al. 2001, Clausen et al. 2001), the goal is to get back on track as soon as possible at minimum cost and with minimum deviation from the original schedule.

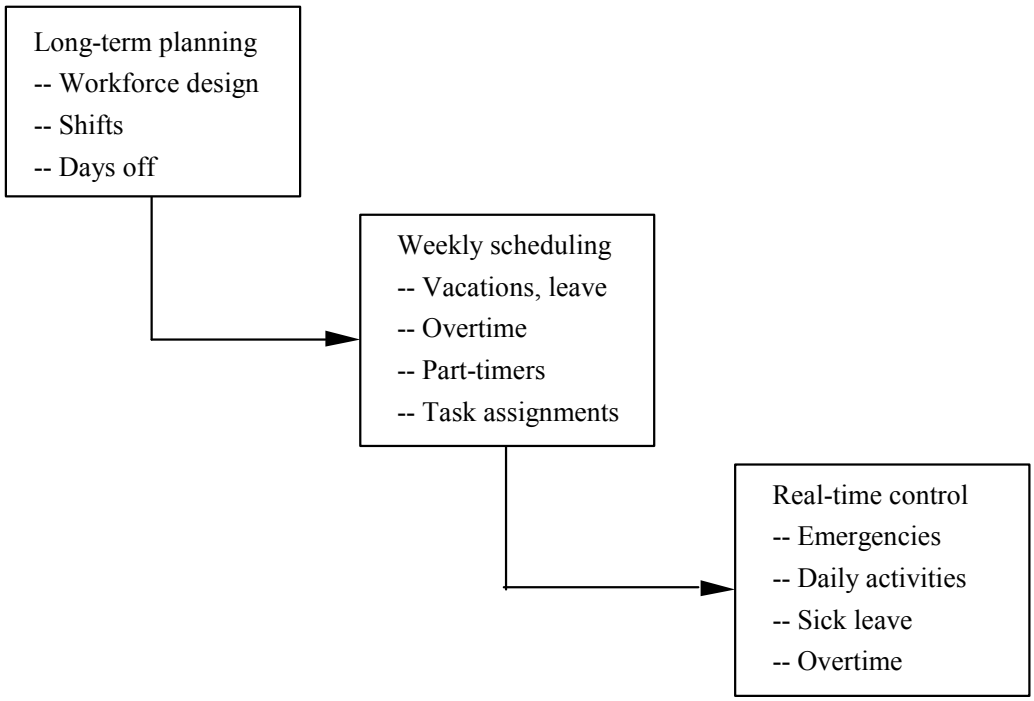


Figure 1.1. Three Levels of Personnel Scheduling

1.2 Components of Weekly Staff Scheduling

Several related problems surrounding the weekly staff scheduling are investigated in this research, including weekly shift scheduling, task assignment, and staff scheduling with workstation group (WSG) restrictions.

1.2.1 Weekly Shift Scheduling

Weekly shift scheduling needs to produce weekly schedules or tours for each employee in the workforce. Three problems must be solved to construct *tours*. The base problem is called the "shift-scheduling" problem. The objective of this problem is to find the optimal crew-size and the optimal work schedule for every member of the crew. A *shift* is a set of consecutive time periods within a workday and the *shift length* is the total amount of time it covers. In this study, the work period of every employee is referred to as his *shift* and the problem of finding the optimal work schedule is referred to as the problem of finding the optimal *shift lengths*. The shift-scheduling problem starts by defining a set of shifts, their start-times and their lengths, and then determines the number of employees that should be assigned to each shift so as to satisfy the demand in each period.

The second part of the problem is named as the "days-off" problem. Every worker needs to be given some days off in a workweek. The number of off-days or the characteristics of these days (weekend days, weekdays or combinations thereof) varies according to the organization and the type of industry in which it operates. The organization needs to provide enough slack within its workforce throughout the week to satisfy the off-day requirements that have been taken into account while determining the optimal workforce size. The implicit modeling of the "days-off" problem is possible with the addition of extra constraints.

The third part of the problem is called the "breaks" problem. Almost every shift, except those that are shorter than a specific length, deserves a lunch break. Labor contracts determine the latest start time and the duration of this break. To assign lunch breaks a *break window,* which is a set of possible periods for the break in every shift, is first created and workers are provided with a lunch break within their break windows. In

order not to use a worker in those periods in which he gets his break, there should be enough capacity to cover for him. Therefore, the model has to provide the necessary amount of slack, and also make sure that everybody is taking his break in the prescribed window. This means that the "shift-scheduling" problem also must take the break allowance into account while assigning employees to shifts. The implicit modeling of break allowances for each employee is possible with the use of additional variables and constraints. However, the additions to the model will only guarantee that there are a sufficient number of idle periods for every worker, which can be assigned as breaks. Determining who takes which period off is a separate assignment problem called the "break assignment" problem.

1.2.2 Task Assignment

At the weekly level, it is also common to specify individual task assignments for each shift in fixed time increments. In many industries this is an easy problem to solve because workers spent the majority of their day in one location. When demand varies from one time period to the next or when different products are assembled in a flexible environment, it is necessary to reposition the workforce periodically to maximize the use of both equipment and manpower. However, this presents two minor difficulties. First, it is easier to supervise a stationary workforce than a mobile one, and second, it is impossible to avoid the loss of some productivity due to travel time and the reorientation that takes place after each move.

These issues give rise to what is called the *task assignment problem* (e.g., see Ernst et al. 2004). The objective is to construct daily schedules for each member of the workforce that minimize the weighted sum of transitions between WSGs while ensuring that all demand is satisfied.

1.2.3 Workstation Group Restrictions

To match the layout of a facility or to ease the management of the workforce, constraints are often placed on the movement of employees between different WSG even though the

same skills are required at each. A WSG is usually associated with a certain location or job activity and it is common to assign every employee a base WSG or home base where they must spend the plurality of their time. Nevertheless, when the workforce is homogeneous or when higher skilled employees can be downgraded to work at lower skilled jobs, it is desirable to reassign them to other WSGs when idle time exists in their schedules. Such movement may be subject to restrictions, though, as a result of, say, physical distance or other special requirements. For example, it may be possible to reposition an employee whose home base is WSG *A* to work in WSG *B* when he or she is idle, but not in WSG *C*. The presence of these types of restrictions vastly complicates the weekly staff scheduling problem.

1.3 Guide to This Dissertation

The contents of this dissertation are arranged as follows: Chapter 2 extensively reviews the literature related with staff scheduling in general and the specific problems that were the focus of this research. In the following three chapters, the details of the weekly shift scheduling, task assignment, and weekly staff scheduling with WSG restrictions are presented respectively, including the models, the solution approaches, and the computational results. Since the workstation group restrictions are also present in long-term staff planning, the long-term scheduling problem with workgroup restrictions is investigated in Chapter 6. Chapter 7 summarizes the dissertation and suggests future work. The problems discussed in the different chapters are related because all of them were motivated by a staff scheduling project associated with USPS mail processing and distribution centers. However, each of the problems has other independent applications, so each of the chapters (Chapters 3-6) has its own set of notation and formulations.

Chapter 2

Literature Review

Typical issues in staff scheduling include days-off assignments, break assignments, leave management, and the use of part-time and casual labor. Overtime allocation and personal preferences are also management concerns. Variations of the basic personnel scheduling problem include non-homogeneous workforce in terms of skill, various labor requirements such as maximum work stretches, break definitions, off-days and off weekend policies, as well as maximum and minimum workforce constraints, start time regulations, and objective function definitions (e.g., see Nanda and Browne 1992). Although the literature on staff planning and scheduling is vast, most of research has concentrated on the long-term problems associated with shift scheduling. Both exact and heuristic methods have been developed for problems arising in such industries as transportation, healthcare, and retail services. Ernst et al. (2004) provide a state-of-the-art review.

2.1 Tour Scheduling Problem

In the last two decades, major breakthroughs in modeling and computation techniques have led to more integrative methodologies for solving the tour scheduling problem. The components of this problem include shift and days-off scheduling, break and task assignments, and in some cases, overtime allocation and individual preference considerations.

Burns and Carter (1985) were the first to provide a comprehensive solution to the days-off assignment problem. They derived a set of lower bounds on the workforce size that took into account days-off requirements as well as the requirement for A out of B weekends off. Their results assumed a maximum work stretch of six consecutive days for each employee. In related work, Alfares (1997) proposed an efficient algorithm for assigning two consecutive days off to employees in tour scheduling problems. He first

developed lower bounds on the workforce size and then introduced them as additional constraints in a linear programming model. This was sufficient to ensure integer solutions.

The implicit modeling of breaks was first proposed by Bechtold and Jacobs (1990) who derived three constraints that collectively ensured feasible break assignments. Aykin (1996) used a similar objective function for the shift scheduling problem, extending the model to allow for multiple, rather than single, breaks and break windows. His approach called for a new set of decision variables to represent every possible combination of breaks. The resultant model was significantly smaller than its predecessors.

Jarrah et al. (1994) were the first to address the days-off scheduling and shift scheduling in a unified manner. They presented a new methodology for solving the large-scale combined shift and days-off scheduling problem when the labor requirements span less than 24 hours per day. They begin with an integer programming formulation and then introduce a set of aggregate variables and related cuts. When the aggregate variables are fixed the original problem decomposes into seven sub-problems (one for each day of the week) that are much easier to solve. Solutions were obtained for problems with up to 1,400 integer variables and 1,500 constraints. The approach was further refined by Bard et al. (2003).

Brusco and Jacobs (1998) presented a 2-stage solution procedure for the restricted start time tour scheduling problem, which may be described as the determination of appropriate subsets of shift starting times for full-time and part-time employees, as well as the assignment of employees to tours associated with these starting times. To find solutions, they developed a construction/improvement heuristic and a three-stage procedure for reducing the density of the A-matrix. Computational experience was presented for a large set of real-world problems that contained on the order of 1344 pure integer variables, 192 binary variables, and 867 constraints.

Beaumont (1997) took a more expansive view and included worker availability, the maximum number of workers who can start at the same time, the relative efficiency

of a worker, the cost of making a customer wait, annual leave factors, the expected number of jobs an employee can complete in each period, the number of contractors, and the maximum number of jobs that can be done in each period, as parameters. His model also permitted a limited amount of queueing of customers, but ultimately was too large to be of practical value.

Berman et al. (1997) proposed a Markovian network model to determine permanent workforce requirements and daily assignments in a high volume factory, while simultaneously scheduling the flow of work. Because of the complicated nature of the workplace and the variety of labor rules that had to be considered, they formulated the problem as a linear program and used expected values as input. The primarily purpose of the model was for long-term planning rather than schedule generation.

Looking at more advanced computational approaches, Brusco (1998) evaluated the performance of dual all-integer cutting planes for solving the tour scheduling problem. He showed that a cutting plane enhanced by an LP objective cut and a sophisticated row selection rule improved solution times with respect to a commercial branch and bound code.

With weekly scheduling in mind, McManus (1977) investigated how best to allocate overtime in the British Post Office. His main goal was to identify the optimal level of staffing for given fluctuations in daily demand. By making assumptions about how the workload changes from day to day, he was able to derive a series of optimal scheduling rules. The approach starts with an estimate of the daily workload distribution for a given level of staffing and then determines the need for overtime. This research significantly extends this approach by considering a mix of options for dealing with periods of both high and low demand. In reality, it is necessary to take into account the myriad rules and regulations that make standard shift scheduling methods intractable.

Berman and Larson (1993) introduced the idea of 'just-in-time personnel' when trying to configure a workforce in an environment characterized by high absenteeism and daily workload variability. A system that makes use of just-in-time personnel attempts to meet its labor requirements at minimum cost by reducing both excess worker inventory

and worker shortages. The analysis assumed three types of equally skilled workers: full-timers, part-timers and on-call temporaries whose workload varied randomly according to a normal distribution from day to day. Shortages due to random absenteeism were managed with overtime and the use of temporaries. An algorithm specially tailored for the problem was developed to find the optimal workforce composition.

Lewis et al. (1998) model an administrative office as a closed queuing network and work towards the allocation of a given number of workers across tasks. The allocation component of the problem was formulated as a nonlinear integer program with the assumption that all tasks in the office are interdependent rather than independent or serial. The proposed solution approach first identifies the bottleneck workstation and then allocates workers optimally.

The literature on the midterm adjustment problem is more limited with most of the work focusing on nurse and crew rostering (e.g., see Burke et al. 2004, Dawid et al. 2001). In a call center application, Caprara et al. (2003) developed a column generation approach that relied on solving a simplified network subproblem to find feasible schedules.

Others, such as Easton and Rossin (1997), have proposed similar models when overtime is the only option for meeting spikes in demand. They argue that the increasing per capita labor expenses have forced service sector employers to increase the use of overtime and decrease the use of part-time labor. They evaluated the effects of alternative overtime staffing and scheduling policies on critical performance measures, such as total labor costs, labor utilization and workforce size, and found that even small amounts of premium pay for overtime provided significant savings. An important conclusion was that the ideal workforce size and proportion of overtime allocated for a given scheduling policy seems to be relatively insensitive to changes in per capita labor costs.

2.2 Task Assignment Problem and WSG Restrictions

While much has been written on the classical assignment problem and its variants, little research exists on the type of assignment problem addressed in this research. The task

assignment problem can be viewed as a special case of the multidimensional assignment problem. Early on, Pierskalla (1968) formulated the three-dimensional assignment problem as an IP and solved it with branch and bound. More recently, Gilbert and Hofstra (1988) categorized certain higher dimensional assignment models and showed that the general class was NP-hard.

At the mid-term planning level, Mukherjee and Gilbert (1997) considered the problem of scheduling instructors in executive development programs run by universities and other institutions. They developed a 0-1 integer programming formulation but the many restrictions and the dynamic nature of the environment led to an unsolvable model. As an alternative, they developed four heuristics based on Lagrangian relaxation. Under various conditions, each was shown to be fast, accurate and reasonably suitable for both random and real problems that required the scheduling of up to 547 classes.

Applications of the assignment problem somewhat related to this work can be found in the field of transportation. Hall and Lotspeich (1996), for example, present a multi-commodity network flow model for lane assignment on an automated highway, where the commodities represent trip destinations (i.e., exit onto ramps). A static formulation was given that did not consider the time distribution of demand. Apart from the normal network flow constraints, the model incorporated bundle constraints to account for traffic that enters, exits and passes through each lane within a segment of the highway. The objective was to maximize total flow, subject to a fixed origin/destination pattern expressed on a proportional basis. Tests were conducted for highways with up to 80 segments, 20 destinations and 5 lanes.

Another generic problem that is closely related is the multi-period assignment problem. Miller and Franz (1993) studied the problem of assigning medical residents to training rotations and clinic stints. The objective was to maximize the residents' schedule preferences while meeting the hospital's training goals. A decision support system was designed to review naturally occurring infeasibilities due to the complexity of the scheduling problem and to make decisions about altering conflicting constraints.

Aronson (1986) developed a branch and bound algorithm for the multi-period assignment problem by transforming it into a multi-commodity network flow problem. His model included the cost of assigning a person to an activity in each time period as well as the cost of transferring a person from one activity to another in successive time periods. Arc capacities prohibited the assignment of an activity to more than one person in each period while the pure minimum cost network flow structure limited the assignment of a person to no more than one activity in each time period. An exact algorithm was proposed to find solutions based on solving a relaxation of the multi-commodity network flow problem obtained by eliminating the mutual capacity constraints. The relaxation led to a series of shortest path subproblems whose solutions were used to establish branching rules and to provide a lower bound on the original objective function.

To the best of the author's knowledge, there has been little if anything published on the WSG restrictions problem. There has been some work, though, on cross-training and the use of higher skilled workers to fill in for lower skilled workers when idle time exists in their schedules (e.g., see Bard 2004a, Malhotra and Ritzman 1994, Misra et al. 2004).

Campbell and Diaby (2002) developed an assignment heuristic for allocating cross-trained workers to multiple departments at the beginning of a shift. Each worker had different qualifications with respect to each department. The problem was formulated as a variant of the generalized assignment problem with a concave objective function that measured department preferences. The authors present a comparison of their linear approximation heuristic with a greedy approach and a Lagrangian heuristic.

With regard to postal operations, most of the existing research has centered on tour construction. In an early study, Showalter et al. (1977) developed a simple building heuristic aimed at specifying shift start times, work center assignments, and mail class responsibility for each employee. Assignments were made subject to a number of constraints, including days off requirements, work center capacities, mail arrival volumes, and mail flow patterns through the system. For more recent work, see Bard et al. (2003),

Berman et al. (1997), and Malhotra et al. (1992).

Despite the abundant research on the various components of staff scheduling, most concentrated on long-term problems associated with tour scheduling. Models that address weekly adjustment problem are very limited. In addition, WSG restrictions are also absent in the literature. This research tries to incorporate this practical feature into the general framework of weekly staff scheduling. Although models in this research are based on P&DC operations, the results should be applicable to a wide range of applications.

Chapter 3

Weekly Shift Scheduling

In long-term staff planning, the goal is to find the optimal crew-size and the optimal shift schedule while allowing for the days-off and break requirements. The solution of this stage yields the minimum number of employees for each shift necessary to satisfy the demand in each period of each day. The demand requirements assumed in this model are representative of a typical weekly demand. In essence, long-term planning is needed to fix the workforce levels. The demand data used for the long-term analysis are really average values and are likely to vary from week-to-week and day-to-day. Moreover, vacations, sick leave and other types of absenteeism reduce the size of the regular workforce on a daily basis. Re-planning is necessary to accommodate more accurate estimates of weekly demand and planned leave. To do this, the long-term model must be modified to account for the available workforce and expected demand.

3.1. The Weekly Scheduling Model

The weekly scheduling problem is to develop a weekly staffing plan that makes the most efficient use of the current workforce size augmented by the use of part-time flexibles, casuals and overtime. To provide every worker with a descriptive schedule of his own, this solution needs to be integrated with a post-processing algorithm to pick the periods that would be taken as breaks for each employee. Attempts would also be made to develop a pre-processing algorithm to handle leave.

3.1.1 Model Components

The workforce in a P&DC is composed of full-timer regulars (FTRs), part-time regulars (PTRs), part-time flexibles (PTFs), and casuals (CAS). A regular employee has a predetermined start time for every working day. Flexible employees and casuals are not necessarily given a 5-day a week schedule, but are called in when needed. Nevertheless,

the goal is to provide each PTF with at least a minimum number of hours per week in order to promote job satisfaction and retention. Unlike casuals, PTRs and PTFs receive benefits and are considered career employees. Generally speaking, all employees prefer a constant start time from one day to the next, if for no other reason than fluctuating schedules make it difficult to establish a stable personal regime.

A full-timer works 8½ consecutive hours, which includes a ½ hour allowance for a lunch break (in reality, he is off the clock for the ½ lunch). A part-timer, on the other hand, may be assigned one of a variety of possible shift lengths. Note that a *shift* is a set of consecutive time periods within a workday and the *shift length* is the total amount of time it covers. In this study, lengths from 4 to 12½ hours (including the lunch breaks where applicable) are considered. All employees working more than 6 hours per day must be given a ½ hour lunch break.

3.1.2 Notation and Formulation

At the weekly level, the goal is to derive schedules for each full-time and part-time employee that minimize personnel costs subject to bid job assignments and contractual agreements. For FTRs, overtime that can vary from 1 to 4 hours on a normal working day and up to 8 hours on a day off. This is further discussed in the next subsection.

The notation used in the formulation of the model are presented below. The extensive list of symbols is primarily due to the need to distinguish the various categories of workers.

Indices

- d days of the week; $d = 1,\ldots,7$
- t time periods during a day; $t = 1,\ldots,48$
- k full-time regular, part-time regular and part-time flexible employees, $k \in K$
- s shifts associated with the regular workforce or part-time flexibles, $s \in S(k,d)$; for casuals, $s = 1,\ldots,n^C$

Parameters

c^1_{kds} cost when employee k works shift s on day d (~$30/hr for a regular employee and $25/hr for a flexible part-time employee)

c^2_k penalty overtime hourly rate for regular employee k (twice the straight rate)

c^3_k regular overtime hourly rate for part-time employee k (one-and-a-half the straight rate)

c^4_k penalty overtime hourly rate for part-time employee k (twice the straight rate)

c^5_s cost for casual shift s (~$12/hr)

H_{kdst} 1 if shift s on day d covers period t for employee k; 0 otherwise

C_{st} 1 if casual shift type s covers period t; 0 otherwise

D_{dt} demand for period t on day d

l_s length of part-time shift s

o_{ks} amount of overtime associated with shift s for employee k

$PD^{\min}_k$ minimum number of days per week that must be assigned to part-time worker k (= 2 for PTFs)

$PD^{\max}_k$ maximum number of days per week that can be assigned to part-time worker k (= 6 for PTFs)

$PH^{\min}_k$ minimum number of hours per week that must be assigned to part-time worker k

$PH^{\max}_k$ maximum number of hours per week that can be assigned to part-time worker k

e earliest period a break can begin for any of the permissible shifts

q latest period a break can begin for any of the permissible shifts

Sets

K	set of all employees, $K=K^{\mathrm{F}} \cup K^{\mathrm{P}} \cup K^{\mathrm{L}}$
K^{F}	set of full-time regular employees
K^{P}	set of all part-time regular employees
K^{L}	set of part-time flexible employees
$W(k)$	set of days employee k is scheduled to work as defined by his or her bid job
$\bar{W}(k)$	set of days employee k is off (complement of $W(k)$)
$E(d)$	set of regular employees that are not scheduled to work on day d
$S(k,d)$	set of shifts that employee k is permitted to work on day d
$\hat{S}(k,d)$	set of overtime shifts that employee k is permitted to work on day d
B_{kr}	$\{j$: break window for regular shift j for employee k lies entirely between period r and $q\}$
B_r^C	$\{j$: break window for casual shift j lies entirely between period r and $q\}$
F_{kr}	$\{j$: break window for regular shift j for employee k lies entirely between periods e and $r\}$
F_r^C	$\{j$: break window for casual shift j lies entirely between periods e and $r\}$
T_{kd}	set of shifts that regular employee k is permitted to work on day d that require a break
T^C	set of casual shift types that require a break
M	set of initial periods of the break windows, in ascending order
N	set of final periods of the break windows, in ascending order

Decision variables

x_{kds}	(binary) 1 if employee k works shift s on day d; 0 otherwise

ν_{ds} number of casual shifts of type s required on day d

β_{dt} total number of breaks initiated in period t on day d

γ_{kd} amount of penalty overtime (double time) worked by employee k on day off d

δ_{kd} (binary) 1 if d is the first off day for employee k; 0 if d is the second day off

μ_k number of hours that part-time employee k works in a week that exceeds 40 but occurs during the first 8 hours of a shift

τ_k number of hours that part-time employee k works in a week that exceeds 56 but occurs during the first 10 hours of a shift

Model

Minimize $$z = \sum_{d=1}^{7}\sum_{k\in K}\sum_{s\in S(k,d)} c^1_{kds}x_{kds} + \sum_{d=1}^{7}\sum_{k\in E(d)} c^2_k\gamma_{kd} + \sum_{k\in K^P\cup K^L} c^3_k\mu_k + \sum_{k\in K^P\cup K^L} c^4_k\tau_k + \sum_{d=1}^{7}\sum_{s=1}^{n^C} c^5_s\nu_{ds} \quad (3\text{-}1a)$$

subject to

$$\sum_{k\in K}\sum_{s\in S(k,d-1)} H_{k,d-1,s,t+48}x_{k,d-1,s} + \sum_{k\in K}\sum_{s\in S(k,d)} H_{kdst}x_{kds} + \sum_{s=1}^{n^C} C_{s,t+48}\nu_{d-1,s} + \sum_{s=1}^{n^C} C_{st}\nu_{ds} - \beta_{d-1,t+48} - \beta_{dt} \ge D_{dt},\ d = 1,\ldots,7;\ t = 1,\ldots,48 \quad (3\text{-}1b)$$

$$\sum_{t=e}^{r}\beta_{dt} - \sum_{k\in K}\sum_{s\in F_{kr}} x_{kds} - \sum_{s\in F_r^C}\nu_{ds} \ge 0,\ \ \forall\ r\in N;\ d = 1,\ldots,7 \quad (3\text{-}1c)$$

$$\sum_{t=r}^{q}\beta_{dt} - \sum_{k\in K}\sum_{s\in B_{kr}} x_{kds} - \sum_{s\in B_r^C}\nu_{ds} \ge 0,\ \ \forall\ r\in M;\ d = 1,\ldots,7 \quad (3\text{-}1d)$$

$$\sum_{k\in K}\sum_{s\in T_{kd}} x_{kds} + \sum_{s\in T^C}\nu_{ds} - \sum_{t=e}^{q}\beta_{dt} = 0,\ \ d = 1,\ldots,7 \quad (3\text{-}1e)$$

$$\sum_{s\in S(k,d)} x_{kds} = 1,\ k\in K^F\cup K^P;\ d\in W(k) \quad (3\text{-}1f)$$

$$\sum_{s \in S(k,d)} x_{kds} \le 1, \; k \in K^{\mathrm{L}}; \; d = 1,\ldots,7 \text{ or } k \in K^{\mathrm{F}} \cup K^{\mathrm{P}}; \; d \in \bar{W}(k) \tag{3-1g}$$

$$PD_k^{\min} \le \sum_{d=1}^{7} \sum_{s \in S(k,d)} x_{kds} \le PD_k^{\max}, \; k \in K^{\mathrm{P}} \cup K^{\mathrm{L}} \tag{3-1h}$$

$$PH_k^{\min} \le \sum_{d=1}^{7} \sum_{s \in S(k,d)} l_s x_{kds} \le PH_k^{\max}, \; k \in K^{\mathrm{P}} \cup K^{\mathrm{L}} \tag{3-1i}$$

$x_{kds} \in \{0,1\}$, $\nu_{ds} \ge 0$ and integer, $\gamma_{kd} \ge 0$ and integer,

$$\mu_k \ge 0, \; \beta_{dt} \ge 0, \; \forall \; k,d,s,t \tag{3-1j}$$

The objective function (3-1a) minimizes the total weekly cost of the existing workforce. For FTRs and PTRs, the cost coefficient c_{kds}^1 is a function of the particular employee *k*, the day of the week *d*, whether or not *d* is a day off, and the shift *s* worked. The definition of the set $S(k,d)$ allows for straight and overtime shifts, each with appropriate costs. Permissible shifts for regular employee *k* depend on whether he is scheduled to work on day *d*, and are determined from his bid job. Time-and-a-half is paid for the first two hours of overtime and double time is paid for the second two hours on a scheduled day. Similarly, PTFs can work shifts of various lengths, but because a shift is not associate with a particular day, only the cost of part-time shift *s* for employee *k*, c_{kds}^1, must be specified. To simplify the formulation, c_{kds}^1 is still used for part-time shifts even though it is not a function of the day *d*.

The second term in the objective function adds penalty overtime for FTR shifts assigned on off days. The rule is that all hours worked above 8 on the first day off as well as all hours worked on the second day off are paid at twice the normal rate. The variable γ_{kd} captures the amount of overtime that is entitled to double pay. The constraints needed to enforce this rule are discussed below.

The overtime rules for PTRs and PTFs are slightly different because these employees are not required to have 2 days off every week. In addition to the general requirement that all hours worked in a day that exceed 8 be counted as overtime, it is also

required that all hours worked in a week that are above 40 be counted as overtime, and all hours that are above 56 be counted as penalty overtime. The first requirement is included in the definition of the cost coefficient c^1_{kds}. To account for the second, it is necessary to introduce the variable μ_k, which is defined as the cumulative number of non-overtime hours that exceed 40 hours in a week, and τ_k, which is defined as the cumulative number of non-penalty hours that exceed 56 hours in a week. For an FTR, μ_k and τ_k are always 0 but they can be positive for part-time employees. The third and fourth terms in the objective function represent the cost of these overtime hours.

When the weekly demand is much higher than the demand used by SOS to determine the optimal workforce size, shortages may exist. In this case, casuals are hired to fill in. The last term in the objective function is the total cost associated with the use of casual shifts to cover the excess demand.

Regarding the constraints, (3-1b) assures that the net workforce is sufficient to cover the demand for each period, each day of the week. The net workforce is the total number of part-time and full-time employees whose shift definitions cover a specific period *t*, less those who have a break during period *t* or *t* + 48 if the shift started on the previous day. The 0-1 matrices (**H** and **C**) filter out employees and shifts that do not cover the period under consideration. Because some shifts will actually spill over to the next day, it is necessary to adjust the indexing scheme so that it goes beyond 48 periods. A corresponding number of additional break variables, β_{dt}, are required to implement this approach. If τ is the last period in a break window of an 8½-hour shift, then $\beta_{dt}, \equiv 0$ for $t > 47 + \tau$.

To account for breaks, in fact, three more constraints are needed. The first, (3-1c), is referred to as the *forward pass* constraint by Bechtold and Jacobs (1990). It assures that the total number of breaks initiated from period *e* up to a given period *r* exceeds the total number of employees who should have taken their breaks by that period. The employees included in the constraint are those whose break windows are fully covered through *r*, but not the ones who have the option of a break in some future period.

The second constraint (3-1d) is referred to as the *backward pass* constraint and ensures that the total number of breaks that are initiated from some specific period r through the end of the day (or until the last period that can be taken as a break, which is period q) exceeds the number of employees who are entitled to a break during this interval. In other words, there should be sufficient breaks in the future to satisfy the break requirement for the rest of the day.

These two constraints are needed to provide every employee with a 1-period break, but they are not sufficient to enforce the requirement that exactly one break be assigned to each shift entitled to one. Furthermore, they do not limit the break assignments to their respective ranges. Constraint (3-1e), which is known as the balance equation, is needed to ensure that every shift is assigned a break and that it is within its permitted window. To allow for spillover from one day to the next, it is necessary to allow the elements in the sets M and N to extend beyond period 48.

The set $S(k,d)$ contains all the shifts that employee k is permitted to work on day d. When employee k is either a FTR or a PTR, he would ordinarily work the shift associated with his bid job. To allow for overtime, if d is one of the 5 scheduled work days, $S(k,d)$ would include shifts that extend the assigned shift up to, say, 4 hours. If d is an off day, $S(k,d)$ might include a range of shifts with different lengths and start times. To further increase the options, earlier start time than specified by the bid job are considered. At the USPS, the union contract allows a shift to start 4 hours earlier than normal provided that overtime is paid for the additional hours. Including all possibilities greatly increases the size of $S(k,d)$.

With this in mind, constraint (3-1f) ensures that one and only one shift is assigned to regular employee k each day of the week while constraint (3-1g) ensures that each PTF is given at most one shift per day, and the same for regular employees on their off-days. The next two inequalities, (3-1h) and (3-1i), limit the weekly schedule for part-time employee k to at most $PD_k^{\max}$ days and $PH_k^{\max}$ hours, respectively. Lower bounds can also be placed on days and hours worked per week.

The model does not guarantee that each PTF will be given a 5- or 6-day a week schedule. In fact, when volume is low, some PTFs may be given a light schedule or maybe not even be called in at all during the week. In general, the number of variables required to account for PTFs and CAS is much greater than the number for the regular workforce because there are many more part-time shift options. This number increases by a factor of four when overtime shifts are considered.

Variable restrictions are given in (3-1j). Note that β_{dt} will be integral in an optimal solution so it is not necessary to impose this requirement explicitly.

3.1.3 Overtime Constraints

Rules for assigning overtime are somewhat complicated but, in general, the USPS limits it to no more than 6% of the total hours worked by career employees. The 6% does not refer to a maximum in any week but to an annual average. Nevertheless, this value is used as the default in computations.

To model regular and penalty overtime, let OTX_k be the maximum number of overtime hours permitted per week for employee k (= 20), let ODX_k be the maximum number of scheduled days of overtime for employee k (= 4), let OT_{ratio} be the fraction of the total hours worked that can be overtime (= 0.06), let q_s be the number of regular hours associated with shift s, i.e., $q_s = l_s$ if $l_s \le 8$, otherwise $q_s = 8$, and let p_s be the number of non-penalty hours associated with shift s, i.e., $p_s = l_s$ if $l_s \le 10$, otherwise $q_s = 10$. The following constraints need to be added to the model.

$$\sum_{d=1}^{7} \sum_{s \in \hat{S}(k,d)} o_{ks} x_{kds} + \mu_k \le OTX_k, \quad k \in K \tag{3-1k}$$

$$\mu_k \ge \sum_{d=1}^{7} \sum_{s \in S(k,d)} q_s x_{kds} - 40, \quad k \in K^P \cup K^L \tag{3-1l}$$

$$\tau_k \ge \sum_{d=1}^{7} \sum_{s \in S(k,d)} p_s x_{kds} - 56, \quad k \in K^P \cup K^L \tag{3-1m}$$

$$\sum_{d \in W(k)} \sum_{s \in \hat{S}(k,d)} x_{kds} \le ODX_k, \quad k \in K \tag{3-1n}$$

$$\sum_{k\in K}\sum_{d=1}^{7}\sum_{s\in\hat{S}(k,d)} o_{ks}x_{kds} + \sum_{k\in K^P\cap K^L}\mu_k \le OT_{\text{ratio}}\sum_{k\in K^R}\sum_{d=1}^{7}\sum_{s\in S(k,d)} l_s x_{kds} \tag{3-1o}$$

$$\sum_{s\in\hat{S}(k,d)} o_{ks}x_{kds} \le \gamma_{kd} + 8\delta_{kd},\ d \in \bar{W}(k), k \in K^F \tag{3-1p}$$

$$\sum_{d\in\bar{W}(k)} \delta_{kd} \le 1,\ k \in K^F \tag{3-1q}$$

$$\delta_{kd} \in \{0, 1\}, d \in \bar{W}(k), k \in K^F \tag{3-1r}$$

An upper bound on weekly overtime for each employee is enforced by constraint (3-1k). Constraint (3-1*l*) counts the additional overtime hours, μ_k, for each part-time employee *k*. If PTF *k* works 7-hour shifts for 6 days, for example, then $\mu_k = 2$. Similarly, Constraint (3-1m) counts the additional penalty overtime hours, τ_k, for each part-time employee *k*. At the USPS, there is an upper bound on the number of days that an employee can work overtime in a week. This is taken into account by constraint (3-1n). Constraint (3-1o) ensures that the percentage of overtime hours with respect to the total scheduled hours associated with the regular workforce does not exceed the limit denoted by OT_{ratio}. Finally, constraints (3-1p) and (3-1q) determine which of the two days off for FTR employee *k*, if either, should be considered the first day and which should be considered the second day from the point of view of minimizing penalty overtime. The binary variable δ_{kd} along with constraint (3-1r) are used to determine these designations. Because μ_k and τ_k will always be integral in an optimal solution, they can be treated as continuous.

What is missing from the formulation is a constraint on the use of casuals. In general, the USPS limits the number of casuals to no more than 5.9% of the total workforce over the year, excluding the December holiday season. Rather than trying to include a surrogate for such a constraint in the model, it is accommodated by setting the cost coefficient c_s^5 to an arbitrarily large value ($40/hour).

3.2. Solution Methodology

Model (3-1) is a large-scale mixed-integer linear program (MILP) whose degree of difficulty depends mainly on the number of career employees, the shift definitions, the length of the planning horizon, and the number of time periods per day. For problems as complex as staff scheduling at P&DCs, it is rare that all factors can be included in a single model. In this case, once a solution is found to the MILP, individual weekly schedules must be constructed. This means assigning breaks to each shift more than 6 hours in length. For casuals, this also means combining shifts into 1- to 6-day tours. Lastly, each worker must be given a set of tasks to perform during each ½-period he or she is enrolled in a shift. In some case, one or more persons may be idle for a number of periods.

The components of the methodology are depicted in Fig. 3.1. The procedures used at each step are discussed below.

3.2.1 Solving the MILP

Initial attempts to solve model (3-1) with CPLEX were mostly successful except for those cases in which a large number of casuals were required and the actual demand differed sharply from the demand used to derive the permanent workforce. Success was judged by convergence to within 1% of optimality prior to reaching a 1-hour time limit. To deal with the difficult cases, a "target" solution strategy was developed, which involves constructing a feasible solution from the linear programming relaxation solution. This is done by defining an optimization problem whose objective is to minimize the sum of the absolute deviations of the integer component of the solution from a target solution given by the LP relaxation. Loosely speaking, the objective is to find a feasible solution as "close" as possible to the LP solution.

Target solutions have been used mostly in nonlinear programming to find good starting points (e.g., see Cai et al. 2001). With regard to integer programming, their use is believed to create an asymmetry in the problem structure that helps reduce the size of

the branch and bound tree but there have been few studies to support this. The algorithm follows.

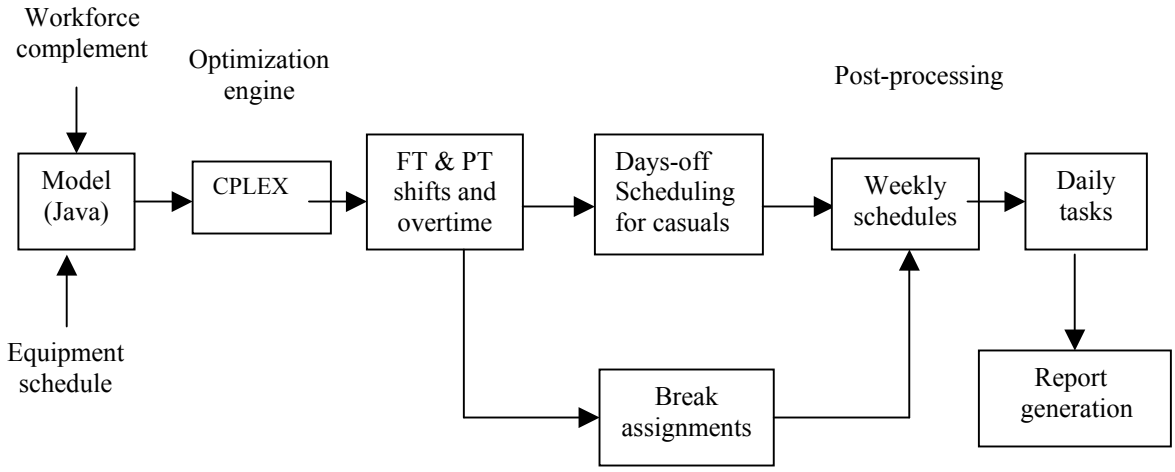


Figure 3.1. Computational Steps in Solution Methodology

Target_Heuristic

Input: Permanent workforce composition and days-off schedule; demand requirements for each ½-hour period in the week; ε arbitrarily small positive number.

Output: Shift assignments for full-time, part-time, and casual employees, including overtime allocations for full-timers.

Step 1: Solve the LP relaxation of the original problem (3-1). Denote the solution by $X_{kds}^{LP} = \left(x_{kds}^{LP}, \beta_{dt}^{LP}, v_{ds}^{LP}, \gamma_{kd}^{LP}, \delta_{kd}^{LP}, \mu_{k}^{LP}, \tau_{k}^{LP}\right)$ for all k, d, s.

Step 2: Solve the following MILP.

$$\text{Minimize } z = \sum_{d=1}^{7} \sum_{k \in K} \sum_{s \in S} \Delta_{kds} + \varepsilon \left(\sum_{d=1}^{7} \sum_{k \in K} \sum_{s \in S(k,d)} c^1_{kds} x_{kds} + \sum_{d=1}^{7} \sum_{k \in \bar{E}(d)} c^2_k \gamma_{kd} \right.$$

$$\left. + \sum c^3_k \mu_k + \sum c^4_k \tau_k + \sum_{d=1}^{7} \sum_{s=1}^{n^C} c^5_s \nu_{ds} \right) \qquad \text{(3-2a)}$$

subject to (3-1b) – (3-1o) and

$$\Delta x_{kds} \geq X_{kds} - X^{LP}_{kds} \qquad k \in K;\ d = 1,\ldots,7;\ s \in S \qquad \text{(3-2b)}$$

$$\Delta_{kds} \geq X^{LP}_{kds} - X_{kds} \qquad k \in K;\ d = 1,\ldots,7;\ s \in S \qquad \text{(3-2c)}$$

$$\Delta_{kds} \geq 0 \qquad k \in K;\ d = 1,\ldots,7;\ s \in S \qquad \text{(3-2d)}$$

where S is the set of all possible shifts that may be worked by at least one employee. Note that Δ_{kds}, X_{kds} and X^{LP}_{kds} are 7-component vectors and really should be written out component-wise in (3-2).

Constraints (3-2b) and (3-2c) together define $\Delta_{kds} = \left| X_{kds} - X^{\mathrm{LP}}_{kds} \right|$, the L_1-norm. The objective (3-2a) is to minimize a combination of the deviations (term 1) and the original objective function (term 2). As in goal programming, term 1 serves as the primary objective function and term 2 as the secondary objective, which is designed to restrict the current search to a neighborhood of the solution to model (3-1). Test results are reported in the next section for model (3-2) as well as two other schemes. In particular, setting the target in (3-2a) to the nearest integer $\left[X^{LP}_{kds} \right]$ is also tried, as well as solving the original problem with only the casual variables relaxed and using the resultant values as the target.

3.2.2 Break Assignments

Each shift longer than 6 hours is ensured a break within its prespecified break window by Eqs. (3-1c) – (3-1e), but the actual period during which the break occurs is not specified by the solution. There is a 5-period window centered in the middle of the shift. The assignment of a break to a period is essentially a transportation problem so a bi-partite

network can be used as a post-processor. In fact, all that is really needed is a feasible solution since no priorities are given for any of the periods with respect to a shift. Thus all arcs in the network have the same cost and bounds, implying that the objective function is constant.

Rather than set up a transportation model to solve the problem, a simple greedy heuristic was used. The first step is to array the breaks and shifts in ascending order. For the first break in the array, all eligible shifts are identified along with the latest period in their break window that is eligible. If possible, the assignment is made to the shift whose latest eligible period is the last period in its break window. Ties are broken arbitrarily, and if it is not possible to make the assignment to the last period of any shift, the next to last period is considered, and so on. The process is repeated until all breaks are assigned. The forward and backward equations, (3-1c) and (3-1d), guarantee the feasibility of this approach.

3.2.3 Days-off and Weekly Scheduling

In its current form, model (3-1) does not have sufficient constraints to guarantee that each casual can be given a weekly schedule with 5 working days and 2 days off. If this is desirable, constraints of the following form must be added to the formulation.

$$\theta_s \geq \tfrac{1}{5} \sum_{d=1}^{7} v_{ds} , \quad s = 1,\ldots,n^C \tag{3-3a}$$

$$\theta_s \geq v_{ds} , \quad s = 1,\ldots,n^C;\ d = 1,...,7 \tag{3-3b}$$

Here, the integer variable θ_s is the number of casuals who work shift s during the week. For a given s, Eq. (3-3a) says that the number of shifts in a week must be less than or equal to 5 times the number of casuals who work that shift. Eq. (3-3b) says that the number of casuals must be greater than or equal to the number of shifts scheduled for any day d. The Burns-Carter (1985) algorithm is used to construct the weekly schedules with v_{ds} as the demand on day d. In practice, however, (3-3a) – (3-3b) are too restrictive, and must be relaxed to allow the start time of a casual to vary from one day to the next. This

is further discussed by Bard et al. (2003) as are the constraints needed to guarantee each worker two days off in a row, if this is a requirement.

For this research, a heuristic is used to string together shifts to form a 5- or 6-day schedule of 39 hours or less. The only restriction is that the start times of a casual's daily shifts fall within a 6-hour time band. In the heuristic, the casual shifts are grouped by time band and arrayed in nondecreasing order of their start time. The first available shift on day 1 is selected. On each subsequent day, the next available shift is selected until a 5-day schedule is obtained. If no shifts remain on a particular day, the next day will be searched until one can be found.

3.2.4 Daily Task Assignments

The demand requirements specified by the parameter D_{dt} in Eq. (3-1b) represent an aggregation of demand on day d for time period t. In a P&DC, clerks fall into one of several categories, the most prevalent being *automation*. A clerk working in automation can run MLOCRs, BCSs, or perform manual casing. The post-processing problem is to assign each employee a series of tasks in ½-hour increments so that the number of transitions from one machine group to another is minimized over a shift. By constructing a network in which each node represents a time period - machine group combination with demand D_{dt}, it is possible to formulate the problem as variant of a minimum cost, multi-commodity network flow program, where each worker is a separate commodity. The inputs to the problem are the weekly schedules including breaks and days off, and the daily demand by period and machine group. The output is a period-by-period machine group assignment for each worker. The detail of the task assignment problem will be discussed further in next chapter.

3.3. Experimental Design and Analysis

A series of tests was performed to determine the tractability of the weekly scheduling model (3-1) and the effectiveness of the target heuristic (3-2a) – (3-2d). The main purpose is to evaluate the computational effort involved in finding high quality solutions

and the cost implications associated with seasonal fluctuations in demand. All input data were provided by the Dallas P&DC.

The first group of tests was designed to check the response of the model when the permanent workforce was fixed and the demand was varied from 85% to 125% of the baseline. The second group of tests was intended to evaluate the target heuristic. Several different options were compared. The third group was aimed at assessing the impact of leave on system performance, as measured by overtime hours, number of casuals, and total cost. In this analysis, the permanent workforce was reduced in 5% increments starting with the baseline and ending with a 20% reduction.

Before running the model it was necessary to determine the size and composition of the permanent workforce. This was done by running the long-term planning model now being used by the USPS called the scheduling optimization system (SOS). For an average week, SOS takes the demand for labor by workstation as input and determines the optimal mix of FTRs, PTRs, and PTFS along with bid jobs for all FTRs and PTRs, and nominal schedules for PTFs. It does not consider the use of casuals or overtime.

In each scenario investigated, SOS was first run with the following parameter settings: no requirement for consecutive days off, a 6-hour start time window for each FTR tour, a lunch break for all shifts greater than 6 hours, FT/PT ratio ≥ 4, a 6-hour start time window for all PTF shifts (one every ½ hour), no option for the use of PTRs, and no overtime or casuals. In the first set of experiments, the staffing demand was generated with a complementary system known as the equipment scheduling optimizer (ESO) developed by Zhang and Bard (2005). Using P&DC volume arrival profiles and end-of-run reports for a typical week, EOS produces equipment schedules and staffing requirements in ½-increment by skill category. In general, workers are classified as either mail processors or mail handlers, and each subgroup is scheduled separately. This analysis focused primarily on the P5-MPCs who are mainly responsible for running the automation equipment; i.e., various types of bar-code sorters and optical character readers. The mail processors in this category form one of the largest subgroups in a facility.

Table 3.1 identifies the seven scenarios used to evaluate changes in demand. The first four were derived using a two-step process. First, a baseline was determined by running SOS to fix the composition of the permanent workforce in terms of FTRs and PTFs. Second, the weekly scheduling optimization system (hereafter referred to as WSO) was run for each of the four different levels of demand shown in the third column of the table. These values are expressed as a percentage of the baseline defined by scenario 2. Scenarios 5, 6 and 7 reflect the staffing levels determined by SOS for the corresponding demand requirements, similarly given in the third column. Scenario 2 is repeated after Scenario 5 for ease of reference. To generate the demand data, the mail volume arrival profiles were modified by the appropriate percentages and then ESO was run. The last two columns in the table give the size of the permanent workforce for each scenario.

Table 3.1. Model Size for First Group of Tests for Skill Category P5-MPC

Scenario	Staff (SOS)	Demand (ESO)	No. of variables	No. of constraints	Density of A-matrix	No. of FTR	No. of PTF
Fixed workforce size							
1	100%	85%	57,895	2861	1.72%	120	30
2	100%	100%	57,895	2861	1.72%	120	30
3	100%	115%	57,895	2861	1.72%	120	30
4	100%	125%	57,895	2861	1.72%	120	30
Variable workforce size							
5	85%	85%	53,515	2738	1.79%	112	27
2	100%	100%	57,895	2861	1.72%	120	30
6	115%	115%	61,519	2975	1.67%	128	32
7	125%	125%	68,767	3195	1.57%	144	36

All computations were performed on a PC with a P-4 2.4G CPU, 512mb memory, running Windows XP. The implementation was done in Java SDK 1.3, which calls CPLEX 8.1 to solve the integer programs (see Appendix A for a description of the interface). Concert Technology was used to set up the model.

For FTR k, the set $S(k,d)$ consisted of the nine shifts associated with his bid job, while the set $\hat{S}(k,d)$ included eight overtime shifts of length 9, 9½ , 10, …,12½ hours

each day $d \in W(k)$. For PTF k, $S(k,d)$ consisted of 10 shift types of length 4, 4½, 5, 5½, 6, 6½, 7, 7½, 8 and 8½ hours, each with twelve different starting times set a half hour apart. Identical shifts were included in the set $S(k,d)$ on each day d. The start time of the first shift was set to the earliest start time of all the shifts assigned to PTF k by the long-term scheduler, SOS. The twelve starting times correspond to a 6-hour band. Average wage rates for the P5-MPCs at the Dallas facility were used to compute the cost coefficients for the FTRs and PTFs; the cost coefficients for the casuals were based on an hourly rate slightly above that of penalty overtime to ensure that overtime and PTF hours are used before casual hours. Finally, all PTFs were given at least 2 days per week, and between 10 and 39 hours.

3.3.1 Analysis of First Set of Experiments – Benefit of Weekly Adjustments

Tables 3.2a and 3.2b report the scheduling results for the first set of experiments. The second column in Table 3.2a gives the costs produced by WSO, while the third column gives the percentage increase or decrease with respect to the baseline, scenario 2. The fourth column lists the costs produced by SOS for those scenarios where it was possible to perform the computations. Recall that SOS is used to generate the optimal workforce for a given level of demand. In scenario 1, 3 and 4, the workforce was fixed at the level obtained by SOS for 100% demand so there is no output to report in column 4. In all cases, the values listed represent the 'true' costs based on a rate of $12/hr for casuals rather than the exaggerated rate, which, as mentioned is used in the model to ensure that casuals are called in only after all other options are exhausted.

Because the SOS solution is always feasible to model (3-1), the slight improvement evidenced by WSO was expected and serves to verify, in part, that the computations are being performed correctly. For the first scenario, the limited amount of overtime used (12 hours) replaces several part-time shifts. No penalty overtime or casual hours are used in either the first or second scenario. The large number of total idle hours is due to a combination of the 15% reduction in demand, the inability to change the bid jobs, and the requirement to give each PTF the minimum of either 2 days or 10 hours of

work a week. Idle time increases a bit in scenarios 3 and 4 because more workers are required to meet the increased demand. Idle rates shown in Table 3.2b remain about the same. Although it would be more efficient to first meet some of the increased demand with casual hours rather than overtime or PTF hours, the cost structure does not permit this.

Table 3.2a. Basic Staffing Results for First Group of Tests for Skill Category P5-MPC

Scenario	True weekly cost	Percent w.r.t. baseline	Weekly cost (SOS)	Total regular FTR hours	Total PTF hours	Total regular overtime hours	Total penalty overtime hours	Total casual hours	Total idle hours
Fixed workforce size									
1	$149,193	97.6%	--	4800	567.5	11.5	0.5	0.0	1074.5
2	$152,888	100.0%	$153,609	4800	749.0	0.0	0.0	0.0	554.0
3	$174,697	114.3%	--	4800	873.5	303.5	28.5	46.5	589.0
4	$183,509	120.0%	--	4800	923.5	297.5	65.0	552.5	576.5
Variable workforce size									
5	$140,054	91.6%	$141,780	4480	584.5	2.0	0.0	0.0	761.5
2	$152,888	100.0%	$153,609	4800	749.0	0.0	0.0	0.0	554.0
6	$164,092	107.3%	$164,788	5120	823.5	7.5	0.0	0.0	488.0
7	$183,476	120.0%	$184,311	5760	870.0	12.5	0.0	0.0	580.5

Table 3.2b. Normalized Staffing Results for First Group of Tests for Skill Category P5-MPC

Scenario	Average PTF hours	Average regular overtime hours	Average penalty overtime hours	Average idle hours	Overtime usage	Casual usage	Idle rate (WSO)	Idle rate (SOS)
Fixed workforce size								
1	18.92	0.10	0.00	7.16	0.22%	0.00%	19.97%	--
2	24.97	0.00	0.00	3.69	0.00%	0.00%	9.98%	10.50%
3	29.12	2.53	0.24	3.66	5.49%	0.77%	9.73%	--
4	30.78	2.48	0.54	2.10	5.46%	8.32%	8.68%	--
Variable workforce size								
5	21.65	0.02	0.00	5.48	0.04%	0.00%	15.03%	16.30%
2	24.97	0.00	0.00	3.69	0.00%	0.00%	9.98%	10.50%
6	25.73	0.06	0.00	3.05	0.13%	0.00%	8.20%	8.80%
7	24.17	0.09	0.00	3.23	0.19%	0.00%	8.74%	9.40%

Scenarios 5, 6, and 7 indicate the benefits of being able to adjust the permanent workforce on a weekly basis as the demand changes. Comparing scenarios 1 and 5, for example, it can be seen from Table 3.1 that the optimal complement of FTRs and PTFs is 112 and 27, respectively, for a 15% decrease in demand. From the data in Table 3.2a, it is observed that the corresponding cost savings is $9149 or 6.13% per week. Similar results were obtained when the demand was increased by 15%, but the cost saving became very small when the increase was 25%. In this case, there were much more casual hours, which is relatively cheap, in scenario 4 used to satisfy the additional demand, while in scenario 7, more FTRs and PTFs were used. As expected, no penalty overtime and no casuals are used in scenarios 5, 6 and 7; the small amount of regular overtime appearing in the solution replaces a few part-time shifts.

If few or no casuals are used when the staffing level is fixed, as is the case in scenarios 1 and 3, the costs are always higher than when the staffing level is allowed to vary to better match the demand. As soon as the number of casual hours becomes more than a few percentage points of the total, however, a fixed workforce size produces about the same cost. This can be seen by comparing scenarios 4 and 7.

Table 3.2b gives a slightly different perspective of the results. Comparing the first four scenarios, it is obvious that when demand is low, part-timers are assigned fewer hours than in the long-term schedule (18.92 hr vs. 24.97 hr on average), and when demand goes up, part-timers are assigned more and more work until the 39-hour limit is reached. The percentage of workers who are actually assigned 39 hours is 0, 0, 6.7%, and 30%, respectively. For scenarios 3 and especially 4, the use of regular overtime, penalty overtime, and casual hours becomes significant. The 6% overtime limit is almost reached in both cases. The number of penalty hours, however, is always very small because of the 6% limit on total overtime hours does not provide much opportunity for double overtime.

Table 3.3 presents the computational results for the first set of experiments. A 10-minute time limit was placed on all runs, but the program was halted if a 1% optimality gap was reached before then. In all cases, CPLEX's default settings were used except

that the MIP emphasis was set to 'feasibility' and the frequency of heuristic was set to once per 15 nodes. The solution times listed were obtained from calls to the internal clock of the PC by the Java code. Although CPU times would have been preferable, CPLEX does not have a system call for this purpose.

In Table 3.3, the 'IP solution' column lists the actual objective values obtained from CPLEX when the exaggerated costs were used for the casual wage rate. The 'IP – LP gap' column reports the percentage gap between the IP solution and the LP solution found at the first node of the branch and bound tree. In general, this gap is extremely small, implying that the LP relaxation is tight. This partially explains why problems of such size can be solved so quickly.

Table 3.3. Computational Results of First Group of Tests

Scenario	IP solution	Solution time (sec)	Node best solution found†	LP solution	LP solution time (sec)	IP – LP gap	Optimality gap
Fixed workforce size							
1	$149,193	20.1	0	$148,221	6.3	0.65%	0.65%
2	$152,888	37.0	60	$152,683	4.7	0.13%	0.13%
3	$177,858	143.5	330	$176,180	9.4	0.94%	0.94%
4	$221,079	219.2	345	$219,174	33.5	0.86%	0.86%
Variable workforce size							
5	$140,054	21.4	0	$139,797	4.5	0.18%	0.18%
2	$152,888	37.0	60	$152,683	4.7	0.13%	0.13%
6	$164,092	37.0	0	$163,770	12.9	0.20%	0.20%
7	$183,476	32.6	0	$183,211	15.6	0.14%	0.14%

† In each case, the solution was found by CPLEX's heuristic; node 0 is the root node in the search tree.

In four of the seven scenarios, the optimal solution was found at the first node of the tree, and in all cases, it was the optimum (within the 1% tolerance). In fact, all feasible solutions were uncovered with CPLEX's heuristic using a frequency setting of once per 15 nodes, and somewhat surprisingly, were always within the 1% optimality gap. This is shown in the last column of the table. The reason why the last two columns

differ slightly is because after solving the LP relaxation, CPLEX adds cuts at node 0 and then re-solves the augmented problem. In general, the lower bound increases due to the cuts, but the resultant solution is rarely if ever integral. At this point, the heuristic is called in an attempt to find a feasible solution. As an example, 14 GUB cover cuts, 17 cover cuts, and 19 Gomory fractional cuts were added at node 0 for scenario 1.

3.3.2 Comparative Results for Target Heuristic

The IP-LP gap reported in Table 3.3 indicates that the LP bound is very tight. A closer examination of the raw output data revealed that while the number of integer variables (after CPLEX presolve) in each problem instance is more 50,000, the number of variables with a fractional value in an LP solution is always under 500. This suggests that there may be many very good integral solutions within the neighborhood of the LP solution and offers an explanation why in some cases (i.e., when the workforce is varied to match the change in demand, as in scenarios 5, 6, 7) optimal solutions are found by CPLEX's heuristic at the first node. When the workforce remains fixed and the demand changes, however, it becomes much harder for the heuristic to find feasible solutions so more time is needed for convergence. This has motivated the development of an optimization-based heuristic aimed at reducing the computational effort by trying to find a good feasible solution in the neighborhood of the LP solution. As mentioned, three slightly different approaches have been investigated.

Method 1: First solve the LP relaxation. Then, using the fractional solution as a target, minimize the sum of the deviations of the IP solution from the LP solution.

Method 2: Instead of using the LP solution as the target, round each variable in the LP solution to the nearest integer, and use this integer solution as the target in the second step.

Method 3: Relax the integrality constraints for the shift variables, ν_{ds}, corresponding to casual workers, solve the resulting mixed-integer linear program, and then use the solution of the relaxation as the target in second step.

To gauge performance, the three methods were used to solve the problems associated with scenarios 1 – 4. Table 3.4a lists the objective function values and solution times found by CPLEX using a 1% optimality gap as the stopping criterion. The last two columns report the results for the first feasible solution found in the process. The percentage under each entry is the ratio of that number to the corresponding number in the 'Optimal solution' column.

Table 3.4a. CPLEX Results

Scenario	Optimal solution		First feasible integer solution	
	Objective value	Solution time (sec)	Objective value	Solution time (sec)
1	$149,193	20	$149,193	20
	--	--	100.00%	100.00%
2	$152,888	37	$154,844	25
	--	--	101.28%	66.16%
3	$177,858	144	$180,809	135
	--	--	101.66%	94.18%
4	$221,079	219	$226,059	62
	--	--	102.25%	28.24%

Table 3.4b. Solutions with Approximation Methods

Scenario	Method 1		Method 2		Method 3	
	Objective value	Solution time (sec)	Objective value	Solution time (sec)	Objective value	Solution time (sec)
1	$149,169	45	$149,525	40	$149,337	35
	99.98%	223.85%	100.22%	197.47%	100.10%	172.78%
2	$153,112	44	$154,213	41	$153,639	76
	100.15%	119.36%	100.87%	109.75%	100.49%	204.73%
3	$179,372	53	$180,615	74	$178,658	368
	100.85%	36.64%	101.55%	51.52%	100.45%	256.39%
4	$223,654	57	$223,152	70	$223,790	306
	101.16%	26.12%	100.94%	32.16%	101.23%	139.76%

Table 3.4b presents the results obtained with the approximation methods in the same format. As shown, method 3 has the worst performance, primarily because it requires the solutions of two mixed-integer programs rather than one LP and one IP as is the case with the two others. Looking at the results for scenarios 3 and 4, which are the more difficult instances, method 1 appears to be the better choice compared to method 2.

With regard to the easier problems, the statistics in Table 3.4a indicate that the optimal solution for scenario 1 is found at node 1 in 20 seconds and for scenario 2 at node 61 in 37 seconds. Although method 1 finds very good solutions to these problems in a short amount of time, it can never beat CPLEX when the optimum is found at node 1, nor is likely to do as well when the branch and bound tree is small. Nevertheless, for the harder problems in scenarios 3 and 4, CPLEX required over 300 nodes just to find feasible solutions, while method 1 found good solutions quickly. In fact, the performance of method 1 was stable in all cases tested. This is further evidenced by the results reported in the next section.

3.3.3 Analysis of Third Set of Experiments – Staff Shortages

The third set of experiments was designed to simulate the impact of a reduction in personnel for reasons such as sick leave, vacations or training. The scenarios were generated by (randomly) removing a fixed percentage of the permanent workforce determined by running SOS. Table 3.5 identifies the five cases examined for the following three skill categories: P5-MPCs, P5-DCs and P5-FSMOs. Reductions ranged from 5 to 20%. In each case, the input demand data reflected the equipment schedules produced for the Dallas P&DC using their tool called SiteMeta rather than ESO data. Because the input data for running ESO is only available for letter mail, it can only be used at this time to generate equipment schedules for P5-MPCs. Therefore, SiteMeta was used to provide schedules for this phase of the analysis.

As an aside, a comparison of scenario 2 in Table 3.1 with scenario 8 in Table 3.5 suggests the extent of staff reductions that is achievable when SiteMeta is replaced by ESO. When SiteMeta is used to produce the equipment schedule for P5-MPCs, the

results indicate that a total of 189 FTRs and 47 PTFs are required to run the facility. When ESO is used, the comparable numbers are 120 FTRs and 30 PTFs, respectively, a 36% drop in each category.

Table 3.5. Model Size for Third Group of Tests -- Three Skill Levels

Scenario		Leave rate	Demand (Dallas)	No. of variables	No. of constraints	Density of *A*-matrix	No. of FTR	No. of PTF
P5-MPC	8	0	100%	88,963	3815	1.33%	189	47
	9	5%	100%	84,319	3683	1.38%	179	46
	10	10%	100%	80,167	3551	1.42%	171	45
	11	15%	100%	76,279	3430	1.47%	160	39
	12	20%	100%	72,127	3298	1.52%	146	40
P5-DC	13	0	100%	129,091	5036	1.02%	278	69
	14	5%	100%	123,127	4849	1.06%	262	65
	15	10%	100%	116,407	4561	1.13%	258	62
	16	15%	100%	110,443	4464	1.15%	253	58
	17	20%	100%	104,479	4277	1.20%	204	57
P5-FSMO	18	0	100%	51,439	2678	1.83%	107	26
	19	5%	100%	48,607	2601	1.88%	100	26
	20	10%	100%	47,023	2535	1.92%	95	19
	21	15%	100%	44,191	2458	1.97%	94	21
	22	20%	100%	41,359	2381	2.02%	71	20

Table 3.6a and 3.6b give the staffing results for all 15 scenarios. Scenarios 8, 13, and 20 correspond to the baseline for the respective skill categories. For example, the weekly cost for P5-MPCs obtained by running SOS was $242,488 while the same cost obtained with WSO was $241,139, a difference of $1349. This 0.56% reduction is due to the increased flexibility available in WSO with respect to the use of overtime and part-time hours. As shown, reducing the size of the permanent workforce does not noticeably increase the cost associated with the P5-MPCs and may even reduce it, suggesting that the original complement of FTRs and PTFs for this category was too large. Similar results were obtained for the P5-DCs and P5-FSMOs. As the percentage reduction in the workforce increases, the overall costs generally remain even because better use is made

of PTF hours. For the scenarios that specify 20% reductions, the corresponding solutions call for a large increase in the number of casual hours which happen to cost much less than either FTR or PTF hours.

Table 3.6a. Basic Staffing Results for Third Group of Tests -- Three Skill Levels

Scenario		True weekly cost	Percent of baseline	Weekly cost (SOS)	Total regular FTR hours	Total PTF hours	Total regular overtime hours	Total penalty overtime hours	Total casual hours	Total idle hours
P5-MPC	8	$241,139	100.00%	$242,488	7560	1177.5	7.0	0.0	0.0	1099.0
	9	$240,352	99.67%	--	7200	1443.5	58.5	1.0	0.0	1057.5
	10	$243,705	101.06%	--	6800	1614.0	231.0	20.0	0.0	1019.5
	11	$247,404	102.60%	--	6440	1548.5	473.5	36.0	90.5	943.0
	12	$237,507	98.49%	--	6040	1476.0	402.5	73.5	548.5	895.0
P5-DC	13	$376,019	100.00%	$378,759	11120	1517.5	19.5	0.0	0.0	1511.0
	14	$373,731	99.39%	--	10560	1851.5	91.0	12.0	30.0	1398.5
	15	$377,654	100.43%	--	10000	2047.5	308.0	55.0	56.0	1320.5
	16	$382,632	101.76%	--	9440	2093.5	597.5	105.5	128.0	1218.5
	17	$369,699	98.32%	--	8880	2052.0	556.5	140.5	616.0	1099.0
P5-FSMO	18	$146,528	100.00%	$146,417	4280	665.0	0.0	0.0	0.0	307.0
	19	$147,355	100.56%	--	4080	801.5	39.5	14.5	0.0	297.5
	20	$150,179	102.49%	--	3840	892.5	141.0	44.5	4.5	284.5
	21	$150,375	102.63%	--	3640	853.0	233.5	53.0	118.0	259.5
	22	$143,565	97.98%	--	3440	778.0	157.5	111.5	383.0	232.0

The average results shown in Table 3.6b are also in line with expectations. For the more extreme cases investigated, the overtime percentage is near the 6% limit and the use of casuals is critical for meeting the demand. It is worth noting that as the number of workers decreases, the use of part-time hours goes up dramatically, approaching the 39-hour limit in most cases. On another matter, the high idle rates in the last column for the P5-MPCs and P5-DCs suggest that either the equipment schedule is not very good, or that it is not possible to fit shifts to demand efficiently for these data sets.

Table 3.6b. Average Staffing Results for Third Group of Tests -- Three Skill Levels

Scenario		Average PTF hours	Average over-time hours	Average penalty hours	Average idle hours	Over -time usage	Casual usage	Idle rate (WSO)	Idle rate (SOS)
P5-MPC	8	25.05	0.04	0.00	4.66	0.08%	0.00%	12.57%	13.20%
	9	32.81	0.33	0.01	4.72	0.68%	0.00%	12.15%	--
	10	38.43	1.36	0.12	4.81	2.90%	0.00%	11.77%	--
	11	38.71	2.94	0.22	4.29	5.93%	1.05%	10.98%	--
	12	38.84	2.67	0.49	2.96	5.57%	6.42%	10.48%	--
P5-DC	13	21.99	0.07	0.00	4.35	0.15%	0.00%	11.94%	12.80%
	14	28.05	0.34	0.05	4.15	0.82%	0.24%	11.15%	--
	15	33.02	1.23	0.22	4.08	2.91%	0.45%	10.59%	--
	16	35.48	2.53	0.45	3.80	5.69%	1.04%	9.85%	--
	17	36.64	2.51	0.63	2.71	5.69%	5.03%	8.98%	--
P5-FSMO	18	25.58	0.00	0.00	2.31	0.00%	0.00%	6.21%	6.10%
	19	33.40	0.39	0.14	2.36	1.09%	0.00%	6.03%	--
	20	37.19	1.47	0.46	2.35	3.77%	0.09%	5.78%	--
	21	38.77	2.57	0.58	1.88	5.85%	2.41%	5.30%	--
	22	38.90	1.83	1.30	1.25	5.52%	7.86%	4.76%	--

It is also investigated how method 1 performs on the more difficult instances. Table 3.7 presents the computational results for the scenarios corresponding to staffing reductions of 10%, 15% and 20%. The 'CPLEX' column and the 'First feasible solution' column respectively list the objective function values and solution times obtained when model (3-1) was solved directly. Again, the percentage under each entry is the ratio of the associated value with the corresponding value in the 'CPLEX' column. As shown, the target heuristic always find a feasible solution within 2% of the optimum within a fraction of the time. In addition, the solution times associated with method 1 do not increase dramatically as the difference between the staffing level and demand grows, a phenomenon observed when model (3-1) is solved. For the difficult problems, then, the target heuristic appears to be an acceptable compromise, especially when time is an important factor.

Table 3.7. Comparison of CPLEX with Approximation Method 1

Scenario	CPLEX		First feasible solution		Method 1	
	Objective value	Solution time (sec)	Objective value	Solution time (sec)	Objective value	Solution time (sec)
10	$243,705	733.2	$244,229	591.0	$245,211	169.9
	--	--	100.22%	80.61%	100.62%	23.18%
11	$253,558	3193.2	$255,582	158.0	$256,728	224.7
	--	--	100.80%	4.95%	101.25%	7.04%
12	$274,805	1528.1	$276,489	336.9	$278,102	163.2
	--	--	100.61%	22.05%	101.20%	10.68%
15	$381,462	361.9	$386,329	344.2	$382,424	395.6
	--	--	101.28%	95.12%	100.25%	109.32%
16	$391,336	445.0	$391,336	445.0	$391,722	257.0
	--	--	100.00%	100.00%	100.10%	57.75%
17	$411,587	694.2	$413,553	649.6	$413,142	191.8
	--	--	100.48%	93.58%	100.38%	27.62%
20	$150,485	112.4	$150,485	112.4	$152,003	51.4
	--	--	100.00%	100.00%	101.01%	45.73%
21	$158,399	1444.5	$158,880	161.4	$160,984	91.3
	--	--	100.30%	11.17%	101.63%	6.32%
22	$169,609	528.6	$171,110	334.3	$172,261	191.8
	--	--	100.88%	63.24%	101.56%	36.28%

Chapter 4

Task Assignment Problem

Having solved the weekly shift scheduling problem, each full-time regular (FTR) and part-time flexibles (PTF) worker must be given a daily assignment of tasks for each day he or she is scheduled to work. Generally, these tasks are associated with a specific machine or workstation. It is desirable, but not mandatory, that each worker be assigned to a single machine for a full shift. In most cases, however, this is not possible 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 run up to 12 hours.

A "good" assignment of tasks is characterized by as few switches among machines as possible; that is, one that minimizes some function of the total number of transitions over the day for each worker category. It is not possible to take transitions into account in the weekly staff-scheduling model that determines the daily shift for each worker because demand is specified by worker category only.

4.1 Problem Description

In defining the problem, the following assumptions are made.

1. Each shift longer than 6 hours includes a ½-hour lunch break.
2. Demand requirements for a particular category of worker (e.g., clerks, mail handlers) can be partitioned into a finite number of groups so the problem decomposes by worker category.
3. All demand must be satisfied.
4. Shifts can spill over from one day to the next so the problem does not decompose by day.
5. The assignment of workers to a specific workgroup in numbers greater than the demand in any particular period represents idle time.

In the solution to the weekly scheduling problem, a sufficient number of shifts are generated to guarantee that assumption 3 can be met. Also, if there is only one workgroup, there is no need to make assignments because all transitions between machines in a workgroup have zero cost. In the more general case, when there are, say, m workgroups, the following result holds.

Theorem 4.1 The task assignment problem is NP-hard in the strong sense.

Proof. It will be shown that the quadratic assignment problem (QAP) can be polynomially transformed into the task assignment problem (TAP). In particular, Sahni and Gonzalez (1976) showed that a version of QAP with only $x_j x_{j+1}$ nonlinear terms is NP-complete by transformation from Hamiltonian circuit. Their proof is valid for the model that is being considered as well.

To define the QAP of interest, let n be the number of workers, t the number of time periods over which assignments are to be made, and $m(p)$ the number of jobs in period p, where $n \geq m(p)$ for $p = 1, \ldots, t$. Let c_{jp}^k be the cost of assigning worker k to job j in period p and s_{jlp}^k the cost of transferring worker k from job j in period p to job l in period $p + 1$. The objective is to minimize the cost of assigning workers to jobs over the planning horizon, where the decision variable $x_{jp}^k = 1$ if worker k is assigned job j in period p, 0 otherwise. This leads to the following QAP:

$$\text{Minimize} \sum_{p=1}^{t} \sum_{k=1}^{n} \sum_{j=1}^{m(p)} c_{jp}^k x_{jp}^k + \sum_{p=1}^{t-1} \sum_{k=1}^{n} \sum_{j=1}^{m(p)} \sum_{l=1}^{m(p+1)} s_{jlp}^k x_{jp}^k x_{l,p+1}^k \tag{4-1a}$$

$$\text{subject to} \sum_{k=1}^{n} x_{jp}^k = 1, \; j = 1, \ldots, m(p); \; p = 1, \ldots, t \tag{4-1b}$$

$$\sum_{j=1}^{m(p)} x_{jp}^k \leq 1, \; k = 1, \ldots, n; \; p = 1, \ldots, t \tag{4-1c}$$

$$x_{jp}^k = 0 \text{ or } 1, \; k = 1, \ldots, n; \; j = 1, \ldots, m(p); \; p = 1, \ldots, t \tag{4-1d}$$

The objective function in (4-1a) sums the cost of each assignment. Eq. (4-1b) ensures that all jobs are covered in every time period, and Eq. (4-1c) prevents a worker from being assigned more than one job in each period. Note that it is an easy matter to convert (4-1c) to an equality by adding $n - m(p)$ dummy jobs with zero cost coefficients in each period p to obtain the equality form of the model.

Now, given any instance of QAP, construct the following instance of TAP. Let there be n workers, t time periods, and m workgroups, where the mth workgroup corresponds to either the lunch break or the time when a worker is off duty. Let d_{jp} be the demand for workers associated with workgroup j in period p and set $m(p) = \sum_{j=1}^{m} d_{jp}$. For the mth workgroup, d_{mp} is the number of workers whose shifts do not cover period p plus those who are scheduled for lunch during period p. Let s_{jlp}^{k} be the cost of transferring worker k from job j in period p to job l in period $p + 1$. If jobs j and l are in the same workgroup or if either is in workgroup m, then $s_{jlp}^{k} = 0$. Also, let the cost of worker k being assigned to a job j in workgroup m during his designated lunch break and when he is not scheduled to work be $c_{jp}^{k} = 0$; at all other times p let $c_{jp}^{k} = \mu$, where μ is an arbitrarily large constant.

It is not difficult to see that an optimal solution to QAP with these cost coefficients can be transformed into an optimal solution to TAP in linear time. All jobs will be covered by exactly one worker, and in periods where there are more workers then jobs, the surplus workers will be assigned to a job in workgroup m. In all other periods, the arbitrarily large value of μ assures that they will be assigned to a workgroup other than m. Sorting out whether a worker is at lunch or idle is straightforward.

Similarly, a solution to TAP can be translated directly into a solution of QAP. Moreover, the simple observation that any assignment for TAP can be checked for feasibility in $O(nt)$ time implies that it is in NP. This leads to the conclusion that the workgroup assignment problem is NP-hard. ■

Mail processing environment. Each day, mail arriving at a P&DC undergoes the following three high-level operations: (i) if a stamp exists, it is cancelled, (ii) the address is read and a bar code is sprayed on the envelop, and (iii) each piece of mail is sorted to its final destination. Four types of automation equipment are used for these activities: (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). WSGs are typically formed by combining similar pieces of equipment located in the same area; e.g., all AFCSs may constitute a single workgroup. When a large number of machines of the same type are present, they may be partitioned into several workgroups.

4.2. Model Formulation

The task assignment problem has several different generic interpretations. It can be viewed as a variant of the minimum cost multi-commodity network flow problem in which the individual workers are the commodities.

4.2.1 Basic Model

Let n be the number of workers, m the number of workgroups, and t the number of time periods. For each worker $k \in K$ a directed subgraph $G_k = \{N_k, A_k\}$ is defined with node set N_k and arc set A_k. Each node $i \in N_k$ represents a WSG - time period pair, which collectively, form a rectangular grid, or t-layered network. For completeness, it is necessary to add source and sink nodes (s_k and t_k) for each worker. In all, $|N_k| = 2 + tm$. Each arc $(i, j) \in A_k$ represents a permissible transition from one workgroup to another in adjacent time periods and has "cost" c_{ij}^k, which may be viewed more appropriately as a penalty. At a minimum, this value includes the cost of switching from workgroup i to workgroup j but may also include the cost of being assigned to j. It is assumed that cost is not a function of time.

Figure 4.1 depicts a "complete" graph for one worker for 336 periods or 1 week. The numbers in square brackets adjacent to the nodes represent demand d_{jp}, while the

numbers inside the nodes denote the workgroup j and the period p, respectively. In reality, it is only necessary to include nodes that correspond to the shifts that worker k is assigned over the week. Accordingly the nodes that correspond to the predetermined lunch breaks and the periods between shifts can be omitted. This greatly reduces the size of N_k.

The full graph G_k for each worker k has the same node set except for s_k and t_k. The arc sets, however, are all disjoint. The full graph for the problem is $G = (N, A) = \bigcup_{k \in K} G_k = \left(\{N_1 \cup \cdots \cup N_n\}, \{A_1 \cup \cdots \cup A_n\}\right)$ and is likely to be very sparse because arcs only exist between nodes in successive periods, except for the two situations just mentioned. That is, when there is a lunch break in period p, the nodes and arcs associated with period p can be skipped for this worker and new arcs added that join the nodes in period $p - 1$ to the nodes in period $p + 1$. Similarly, when a shift ends in period p and the next shift starts in period $p + q$, the intermediary arcs and nodes can be skipped.

If a supply of 1 is specified at the source node s_k and a demand of 1 at the sink node t_k for all $k \in K$, the task assignment problem is to find a minimum cost flow from each s_k to each t_k such that the demand, d_{jp}, at each intermediate node jp is satisfied. When $d_{jp} = 0$ for all nodes but s_k and t_k, the overall problem decomposes into n shortest path problems, one for each worker.

For the TAP to be feasible, it must have $\sum_{j=1}^{m} d_{jp} \le n$ for all p. When $\omega(p) \equiv n - \sum_{j=1}^{m} d_{jp} > 0$, $\omega(p)$ workers in period p will be idle. The only remaining modeling issue concerns the initial conditions. There are two ways to view the overall problem, which leads to two different formulations. If a new solution is required every week because the demand changes, it is necessary to take into account the last assignment of each worker k in the last period ($t = 336$) of the previous week and define the arc costs, $c^k_{s_k, j}$, accordingly. Assuming that there is no transition cost when a shift starts, if worker k is off in period 336, $c^k_{s_k, j} = 0$ for $j = 1,\ldots,m$; otherwise, $c^k_{s_k, j}$ will reflect the cost of going

from the workgroup he is assigned period 336 in the previous week to perhaps a new workgroup period 1 in the current week.

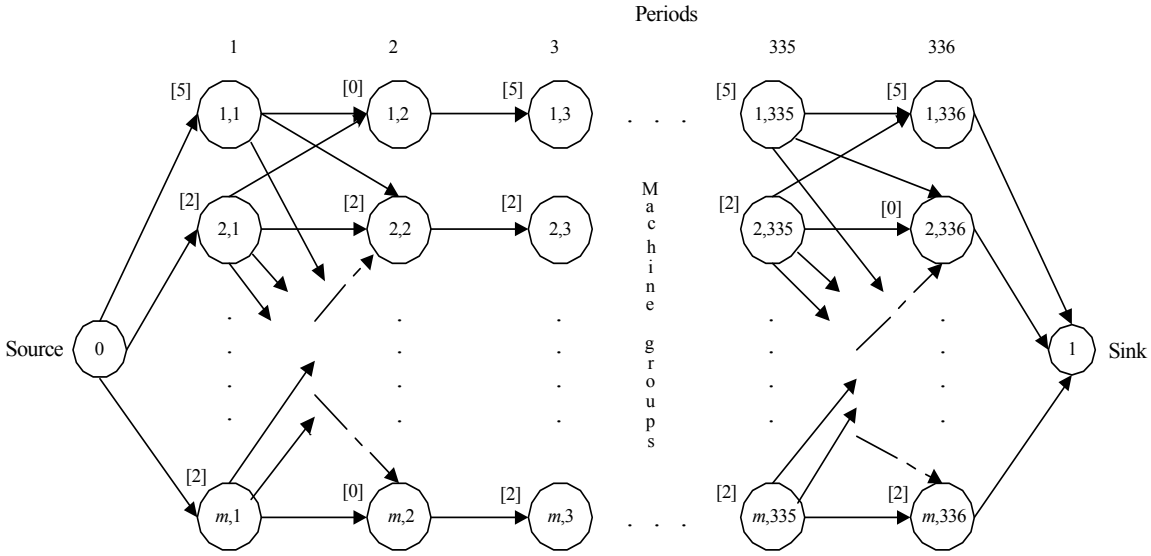


Figure 4.1. Multi-period, Single-commodity Network Representation for a Week

Alternatively, if the demand remains constant from week to week, then the problem only has to be solved once. In this case, all source and sink nodes would be removed, arcs would be added between the nodes associated with the last working period and the nodes associated with the first working period, and the solution would be interpreted as a circulation thereby obviating the need to account for initial conditions. The model presented below reflects the first view because it is more general. In the developments, the following notation is used, which is somewhat different than the notation previously defined.

Indices

i, j = indices for nodes

k = index for workers

Sets

I = set of nodes

K = set of workers

$A(k)$ = set of nodes corresponding to the periods during the week that worker k is on duty

$F(k,i)$ = set of nodes that are immediate successors of node i for worker k

$P(k,i)$ = set of nodes that are immediate predecessors of node i for worker k

Parameters

c_{ij}^k = cost of a transition from node i to node j for worker k

D_i = demand at node i

Decision variables

x_{ij}^k = (binary) equal to 1 if node i is immediate predecessor of node j for worker k; 0 otherwise

Model

$$\text{Minimize} \quad z = \sum_{k \in K} \sum_{i \in I} \sum_{j \in F(k,i)} c_{ij}^k x_{ij}^k \tag{4-2a}$$

$$\text{subject to} \quad \sum_{j \in F(k,i)} x_{ij}^k - \sum_{j \in P(k,i)} x_{ji}^k = 0, \text{ for all } k \in K,\ i \in A(k) \tag{4-2b}$$

$$\sum_{j \in F(k,s_k)} x_{s_k j}^k = 1, \text{ for all } k \in K \tag{4-2c}$$

$$\sum_{j \in P(k,t_k)} x_{j t_k}^k = 1, \text{ for all } k \in K \tag{4-2d}$$

$$\sum_{k \in K} \sum_{j \in F(k,i)} x_{ij}^k \geq D_i, \text{ for all } i \in I \tag{4-2e}$$

$$x_{ij}^k = 0 \text{ or } 1, \text{ for all } k \in K,\ i \in A(k),\ j \in F(k,i) \tag{4-2f}$$

The objective function (4-2a) minimizes the total transition costs for the workforce during the week. The coefficient c_{ij}^k is defined for each employee k and appropriate nodes i and j. It is positive only if i and j are in different workgroups. For the calculations, if i and j correspond to successive working periods, set $c_{ij}^k = 1$; if there is a lunch break period between nodes i and j, $c_{ij}^k = 0.5$; and if there is a shift break between i and j, $c_{ij}^k = 0.1$. Alternative definitions are possible. For example, if it were more

desirable for worker k to be assigned to workgroup g_1 than g_2, c_{ij}^k could be adjusted so that solutions were biased towards g_1.

The inherent network structure of the problem is embodied in the definition of the set $A(k)$ which is of size $O(mt)$. Constraint (4-2b) requires conservation of flow at each node, while Eqs. (4-2c) and (4-2d) ensure that exactly one unit of flow is allowed in the network for each commodity from source to sink. Constraint (4-2e) ensures that all the demand is met and constraint (4-2f) places binary restrictions on the variables.

Model (4-2) is not a pure minimum cost multi-commodity network flow problem for at least two reasons (see Garey and Johnson 1979 for a definition). First, the network on which it is defined has a special structure; second, such problems require that the flow on each arc be restricted to some upper bound, $u(i,j)$. Although it would be possible to split each node i in model (4-2) into two nodes [say, i_1 and i_2], join them by a single arc, and then associate the demand D_i with the arc capacity [say, $u(i_1,i_2)$], the sense of the inequality in constraint (4-2e) would be in the wrong direction. Nevertheless, several special cases arise when some workers have the same schedule.

Proposition 4.1 Let $K_1 \cup K_2 \cup \cdots \cup K_b = K$ be a partition of K into b sets such that the each worker in K_k, $k = 1, \ldots, b$ has the same schedule. Then $k \in K$ can be replaced with $k = 1, \ldots, b$ in model (4-2) and Eqs. (4-2c) and (4-2d) can be replaced with

$$\sum_{j \in F(k,s_k)} x_{s_k j}^k = |K_k|, \text{ for all } k = 1, \ldots, b \tag{4-3a}$$

$$\sum_{j \in F(k,t_k)} x_{j t_k}^k = |K_k|, \text{ for all } k = 1, \ldots, b \tag{4-3b}$$

where $x_{ij}^k = 1$ now denotes a transition from node i to node j for a worker in set K_k.

Proof: Eq. (4-3a) and (4-3d) represents the aggregation of Eq. (4-2c) and (4-2d) for all workers in a particular set K_k so any point that is feasible to the new formulation is feasible to model (4-2). The right-hand side of (4-3a) and (4-3b) indicates that there must be $|K_k|$ units of flow in the corresponding aggregated network. Because the $|K_k|$ workers

are indistinguishable from each other, any feasible solution to (4-2b), (4-3a), (4-3b), (4-2e), (4-2f) obtained after replacing $k \in K$ with $k = 1, \ldots, b$ will be feasible to (4-2b) – (4-2f); that is, the flow in the aggregated network maps directly into the flows in the individual networks. ■

Note that if the two workers have the same shift schedule but different lunch breaks, Proposition 1 is not valid. If the two networks were combined, it might happen that one worker took two lunch breaks and the other none, or that they both took lunch breaks in the same period rather than different periods. Either situation could lead to an incorrect solution.

Proposition 4.2 When each worker k has the same schedule, model (4-2) can be transformed into a pure min-cost flow problem.

Proof: When all the workers have the same schedule, they are indistinguishable from one another so the index k ca be dropped from the formulation and set $A(k) = I$, $s_k = s$, $t_k = t$, $F(k, i) = F(i)$, and $P(k,i) = P(i)$. To guarantee that there are n units of flow in the network, the right-hand sides of (4-2c) and (4-2d) is set to n. Finally, to ensure that all demand is met the node splitting idea is used and a lower bound is set on the arc that joins the two new nodes, say, nodes i_1 and i_2, to D_{i_1}. Letting I' be the set of nodes in the new network and E be the set of arcs that join the split nodes, model (4-2) becomes

$$\text{Minimize } z = \sum_{i \in I'} \sum_{j \in F(i)} c_{ij} x_{ij} \tag{4-4a}$$

$$\text{subject to } \sum_{j \in F(i)} x_{ij} - \sum_{j \in P(i)} x_{ji} = 0, \text{ for all } i \in I' \setminus \{s,t\} \tag{4-4b}$$

$$\sum_{j \in F(s)} x_{sj} = n \tag{4-4c}$$

$$\sum_{j \in P(t)} x_{jt} = n \tag{4-4d}$$

$$x_{ij} \geq D_i, \text{ for all } (i,j) \in E \tag{4-4e}$$

$$x_{ij} \geq 0 \text{ and integer, for all } i \in I', \ j \in F(i) \tag{4-4f}$$

which is a pure min-cost network flow problem. ■

When solving integer programs with exact methods, it is often helpful to add valid inequalities that remove symmetry in the problem definition (Sherali and Smith 2001). The goal is to prevent fractional solutions that look different but are actually reflections of each appearing at different nodes in the branch and bound tree. For the TAP, symmetry is present when two or more workers have the same schedule for all or part of the week. The following result addresses this issue.

Proposition 4.3 (Symmetry) For the TAP defined for a single week only, assume that workers k and $k + 1$ have the same schedules and lunch breaks from period q through period 336. Let $I_p = \{i_{p1},\ldots,i_{pm}\}$ ($p = q,\ldots,336$) represent the set of nodes in the network during period p and let $J_p = I_{p+s}$, where $p + s$ is the first working period after p. Then the following constraints are valid for model (4-2) for all values of $p = q,\ldots,336$ that correspond to working periods.

$$x^k_{i_{pg} j_{ph}} \geq \sum_{l=h}^{m} x^{k+1}_{i_{pg} j_{pl}}, \ \forall \ i_{pg} \in I_p; \ j_{ph} \in J_p; \ g,h = 1,\ldots,m \tag{4-5}$$

Proof: For period p, when workers k and $k + 1$ are at node i_{pg} and worker k transits from workgroup g to workgroup h, constraint (4-5) restricts worker $k + 1$ to transitions from workgroup g to workgroup h or higher. This prevents workers k and $k + 1$ from switching assignments in period $p + s$ and hence breaks the symmetry. Because k and $k + 1$ have exactly the same schedule from period p through the end of the week they appear in the model to be identical from p forward, so no advantage can be gain by interchanging their assignments after node i_{pg}. This validates the introduction of constraint (4-5). ■

Corollary 4.1 For the TAP defined as a circulation, a sufficient for constraint (4-4) to be valid for model (4-2) is that workers k and $k + 1$ have exactly the same schedule for the entire week.

4.2.2 Model with Idle Time and Lunch Breaks

The objective of the task assignment problem is to minimize the number of transitions between different workgroups for all employees. The following are the most common types of transitions:

1. immediate, from one WSG to another
2. after lunch break (different WSG before and after lunch)
3. after idle period (different WSG before and after idle period)
4. between shifts (changing WSGs)

The first is the most undesirable because it involves a change of location, supervisor, and job content for the employee. The others are less costly as measured by time and inconvenience. The easiest way to represent this preference structure is to assign unique costs to the coefficients in (4-2a). This approach is possible for transitions after the lunch break, between workgroups, and between shifts because the schedule of each worker is known in advance. However, the idle periods cannot be identified until a solution is found.

In the network description of model (4-2), each node is associated with a workgroup-time period pair (j,p). To accommodate idle time explicitly, when the number of workers who are available in time period p is greater than the total demand in period p, the corresponding node is devided in two creating a *work node* and an *idle node*. In a solution, if a worker is assigned to the former, it means that he is active in the period under consideration; if he is assigned to the latter, he is idle.

In the multi-commodity network, these two nodes are distinguished only by the costs on their leaving arcs. If nodes i and j are in different workgroups and i is an idle node, then arc (i, j) corresponds to a transition after an idle period and should be assigned

a cost that reflects this type of transition. Note that it is not necessary for a worker to move to the idle node in another workgroup because the worker can always choose moving to the idle node in current workgroup without increasing the total cost. So the corresponding arcs can be removed except when there is a lunch break or a shift break right after the associated period. Because the size of model (4-2) is proportional to the square of the number of workgroups, the augmented model is roughly twice larger than the original model when the unnecessary arcs are omitted.

A second extension of model (4-2) involves the option of assigning lunch breaks along with the tasks. In the USPS application, the lunch break is part of the bid job and is scheduled in advance by SOS. In practice, management has some discretion in rescheduling the lunch break to better match the current day's supply and demand of labor. For an 8-hour shift, the break can be assigned any time within a 3-hour window starting two hours into the shift.

Because a transition after the lunch break is more acceptable than an immediate transition, the rescheduling option may yield better results. To build this into the model, it would be necessary to add a lunch break node in each time period covered by the break window of a shift. As was the case for idle periods, one additional node would be needed for each workgroup. This construction is based on the assumption that the transition cost is a function of the workgroup assigned just prior to the lunch break. If there is no transition cost after the lunch break, it could be modeled as just another workgroup. To guarantee that one and only one break is assigned to each shift, the following constraint should be added to model (4-2):

$$\sum_{i \in E(k,d)} \sum_{j \in F(k,i)} x_{ij}^{k} = 1, \text{ for all } k \in K,\ d \in C(k) \qquad \text{(4-2g)}$$

where $E(k,d)$ is the set of lunch break nodes in the break window of worker k on day d and $C(k)$ is the set of days in which worker k requires a lunch break.

In the remaining sections, the focus is on the explicit representation of idle time only. Because the lunch breaks can be treated in a similar manner, they don't affect the nature of the problem or the solution methodology.

4.3. Solution Methodology

The first algorithm developed for the TAP at P&DCs was a greedy heuristic (referred to as the SOS heuristic) that viewed the week as a circular array of 336 (= 48×7) ½-hour periods. In this design, no starting point exists so every element in the array always has a successor (all integers ≥ 0 are valid indices and all indices whose modulo 336 give the same number represent the same element). The two data structures used by the heuristic are as follows.

1. A list of workstations with corresponding demand. The demand is represented by an array of 336 integers that give the number of workers required for each period of the week. The workstations are grouped in sections or workgroups.
2. A list of shifts satisfying all demand for the given week. Each shift is associated with a single worker and has a starting period (a value between 0 and 335), a length, and possibly a lunch period. Shifts are grouped by starting period giving rise to 336 <shift/period> lists; the shift/period list[p] contains all the shifts starting at period <p>. During execution, a shift starting at <p> may be assigned to a workstation for a duration <d> which is less than the shift length <l>. In this case, the partially assigned shift is added to the shift/period list[$p + d$] for further assignment during the remaining <$l - d$> periods. Thus, each shift/period list contains two types of shifts: (1) unassigned shifts that can be assigned to a workstation in any section, and (2) partially assigned shifts that should be assigned, if possible, to a workstation in the same section.

The first step is to compute the period of the week with least demand. Let

$$p_{\min} = \operatorname{argmin}\left\{\sum_{j=1}^{m} d_{jp} : p = 0,...,335\right\}$$

The greedy SOS heuristic cycles through the circular array of 336 shift/period lists, starting at $p_{\min}$. For each period, an assignment is give to all the shifts on the shift/period list thereby emptying it. However, each shift not completely full is added to a later shift/period list. Processing the last few lists re-introduces shifts to the first few lists so it

is necessary cycle through the circular array until all lists are empty. Additional data structures are needed to track partial assignments and unsatisfied demand; additional logic is needed to determine which workstation to select for the next shift assignment in light of the objective of minimizing the number of transitions. The details are available from the authors.

In an effort to improve on this heuristic, two separate approaches that can be combined into a single methodology have been developed. Each is discussed below.

4.3.1 Delayed Idle Period Assignment and Daily Decomposition Algorithm

When idle time is included in model (4-2) explicitly, the number of nodes in the multi-commodity network doubles. This leads to a linear increase in the number of constraints, which is $O(nmt)$, and a quadratic increase in the number of variables, which is $O(nm^2t)$. Computational experiences have shown that only very small instances can be solved optimally at this level of generality. As a compromise, model (4-2) is solved as originally described and schedule the idle time in a post-processing phase. Although this approach should be viewed as a heuristic, the often negligible impact of idle time suggests that the resultant solutions should be near-optimal.

The problem of assigning idle time in the post-processing phase can be modeled as an integer program but for the reason just mentioned, a greedy approach is taken. In particular, when there is a surplus of workers in a period it must be decided which of them should be considered active and which should be considered idle. To resolve the issue, it is noted that the only way that the objective function can be reduced at this point is by inserting an idle period immediately before or after a transition from one workgroup to another in a worker's schedule. This means that the priority for assigning idle time in a particular period should be given to those workers who have immediate transitions right before or after the period in which a surplus of idle time exists. If no workers have such schedules, then the idle periods can be assigned arbitrarily.

Figure 4.2 illustrates the types of transitions that might occur for a worker during the week. The numbers in the boxes represent the workgroup, "L" denotes a lunch break,

and “R” indicates an unscheduled period. Given the following cost structure: immediate transition = 1.0, transition after lunch = 0.5, transition after idle time = 0.5, transition between shifts = 0.1, suppose that workgroup 4 has a demand of three during period 24, and that the solution to model (4-2) assigns all four workers to that workgroup. As shown in the figure, worker 1 has a lunch break in the preceding period with a cost of 0.5, while worker 2 is scheduled to begin a new shift at this time (in the example, worker 2’s assignment in the last period of the previous shift does not affect the results). Therefore, no benefit can be gained by allowing either of them to be idle in period 24. The same can be said for worker 3 who is assigned to workgroup 4 during periods 23, 24 and 25. Making period 24 an idle period will not affect the cost of his schedule because he is assigned to workgroup 4 in periods 23 and 25. As such, no cost will be incurred by the transition from the idle period.

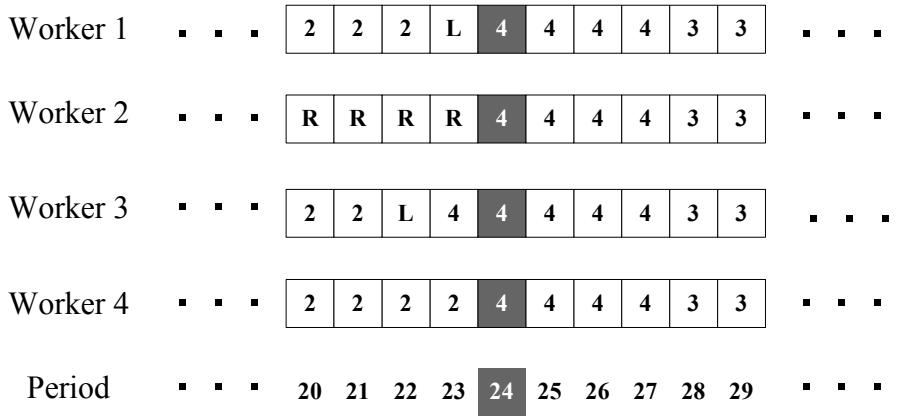


Figure 4.2. Various Types of Transitions for Given Schedules

The best choice is to assign the idle time to worker 4 who has an immediate transition in period 23. Because the transition will now be after an idle period rather than a workgroup, the objective function will decrease from its current value by the difference between the two costs; that is, cost reduction = 1 – 0.5 = 0.5. Summarizing, the cost for each worker before idle time is assigned is given by the vector (1.5, 1.0, 1.5, 2.0); the costs after are (1.5, 1.0, 1.5, 1.5).

The algorithm used in the post-processing phase to assign idle time is given in Fig. 4.3.

```
Let W_i = set of workers assigned to node i, and let p_i = time period associated with i ∈ I.
Set no_priority = false
For (all nodes i ∈ I){
   While (|W_i| > D_i) {
      For (all workers k ∈ W_i){
         If (k has a transition right before or after period p_i or no_priority = true){
            Assign k idle time during period p_i
            W_i ← W_i \ {k}
            Continue
         }
      }
      If (all workers have been checked at node i and W_i ≠ ∅) {
         no_priority = true
      }
   }
}
```

Figure 4.3. Heuristic for Assigning Idle Time

A second method to reduce the problem size is to decompose model (4-2) into seven subproblems, one for each day. When this is done, the solution from the first day provides the initial conditions for the second day, and so on. These conditions are a function of the shifts that spill over from one day to the next, where day 7 can be thought of as day 0. Because the number of arcs in the network (and hence the number of variables) grows linearly with the number of periods t in the planning horizon, the complexity of the integer program (4-2) grows exponentially with t.

Although solving the daily subproblems separately cannot be expected to yield the optimal solution for the week, the decomposition approach, combined with delayed idle period assignment, does yield high quality solutions quickly for instances with up to 300 workers and 7 workgroups. Larger instances require a smaller grid than a day if solutions are to be obtained in a reasonable amount of time. Combining these two ideas

leads to even greater reductions in problem size and run times without much degradation in solution quality, as shown in the section on computational results.

4.3.2 Tabu Search

Tabu search (Glover and Laguna 1997) is a powerful meta-heuristic that has been used successfully in solving many types of large-scale optimization problems that have resisted exact methods. In the computations, the neighborhood of the incumbent is explored and the best feasible solution found becomes the new incumbent, regardless of quality. To prevent returning to the same local solution within a fixed number of iterations, recently visited points are placed on a tabu list.

The essential components of the procedure are the (1) initial solution, (2) neighborhood definition, (3) tabu list, and (4) search strategy. Although there has been continued debate as to whether a good initial solution leads to greater overall efficiency, computational experiences have shown that the better the initial solution, the better the results, at least for the TAP under reasonable run time limits. Three options were investigated: (1) the solution obtained with the SOS heuristic, (2) the solution obtained with the delayed idle period assignment and daily decomposition (DIPA&DD) algorithm, and (3) the solution obtained by solving a series of shortest route problems (SSRA). The first two have already been discussed; the third is explain below.

Initial Solution with Shortest Route Algorithm

An implication of Proposition 4.2 is that when the demand constraints (4-2e) are relaxed, model (4-2) can be decomposed into n independent pure network flow problems with integral solutions. Because each network is cyclic and contains only one unit of flow, the problem is equivalent to finding the shortest loop in the network. CPLEX was used to solve this problem.

Of course, ignoring the demand constraints and solving all n flow problems separately is not likely to produce a feasible solution to (4-2). To achieve feasibility, a simple greedy algorithm is used, in which the workers, arbitrarily ordered, are scheduled

in series (see Fig. 4). For the first worker, the shortest route problem is solved and the demand associate with each node in the solution is reduced by 1. When the demand at a node reaches 0, it is removed from the network. The process is repeated until all workers have been scheduled.

```
For (all workers k ∈ K){
    Set up the single-commodity network shown in Fig. 1 for current worker k.
    For (all nodes i ∈ I) {
        If (remaining demand at node i = 0){
            Remove node i and all arcs connected to it from the network.
        }
    }
    Solve the shortest loop problem to get the schedule for current worker k
    For (all nodes i ∈ I) {
        If (node i is on the schedule of worker k){
            Deduct 1 from the remaining demand at node i.
        }
    }
}
```

Figure 4.4. Sequential Shortest Route Method

Neighborhood Definition

A neighborhood is defined as a set of points that can be reached from the current solution by performing one or more exchanges, which characterize a move. The algorithm proceeds from an initial feasible solution or incumbent to a new solution by searching the current neighborhood for the "best" solution. Depending on the strategy used, the new solution may or may not be feasible.

For the task assignment problem, a neighborhood is defined as all feasible points that can be reached by a swap move that switches the positions of two workers in the same period. To ensure that only neighborhood points that are feasible and different than the current solution are considered, some restrictions must be placed on a move. First, at least two workers must be scheduled in the period under consideration or no swap can take place. Second, the selected workers must be assigned to different workgroups in the chosen period otherwise the exchange will have no effect on the solution.

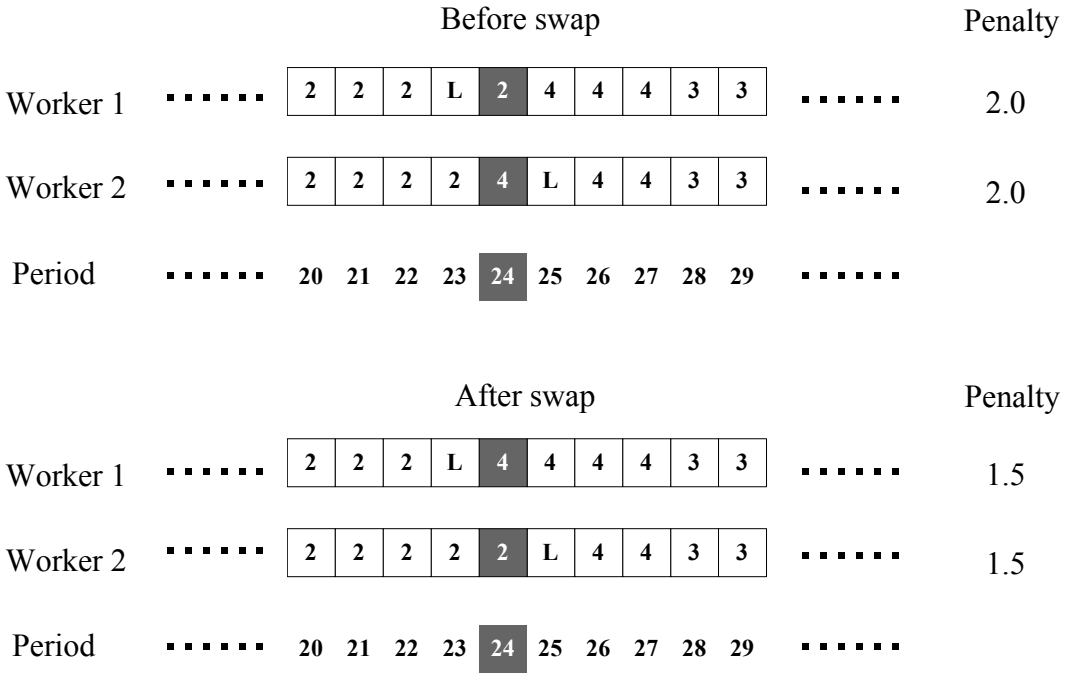


Figure 4.5. Example of Swap Move

Figure 4.5 gives an example of a swap move for workers 1 and 2 in period 24. This is an admissible move because they are assigned to different workgroups in this period. Before the swap, both workers have two immediate transitions in this part of schedule, which induces a penalty of 2.0 for each. After the swap, the first immediate transition in both cases is replaced by a transition after lunch, which brings the penalty down to 1.5 for each worker. Therefore, the value of the swap (*move_value*) is –1.0. In the algorithm, all feasible moves are considered and the one with the smallest value is chosen, giving the new incumbent.

The size of the neighborhood is $O(n^2t)$, where n is the number of workers and t is the number of periods, so it is reasonable to conduct an exhaustive search. A second advantage is that every feasible point in the solution space can be reached if enough iterations are performed. Nevertheless, a significant amount of effort may still be required to conduct one neighborhood search for t fixed even though the size of the

neighborhood is only quadratic in n. Finally, it should be mentioned that more complex neighborhoods were considered, including three way swaps and multiple period swaps, but they proved to be more time consuming and no more effective than the one selected.

Tabu List and Aspiration Criterion

The fundamental process that tabu search uses to transcend local optimality is to make certain moves forbidden (tabu) on a temporary basis. The process is operationalized with a tabu list whose length determines how many iterations a certain move is forbidden. Each entry on the list is a combination of a worker and a period; i.e., (worker i, period p). During execution, any move that leads to a solution that includes a combination on the list is disallowed. After the current neighborhood is searched and a new incumbent is identified, the two workers associated with the move and time period in which it occurs are added to the tabu list.

Nevertheless, the tabu status of a move can be overridden when a certain aspiration criterion is satisfied. In this implementation, a move on the tabu list is accepted if it leads to a solution better than the best solution found so far.

Search Strategy

There are two important characteristics for a successful tabu search procedure: exploration and exploitation. Exploration is the ability of the algorithm to diversify the search throughout the solution space, while exploitation is the ability of the algorithm to intensify the search in the areas of the solution space that show promise. In practice, long-term memory in conjunction with the short-term memory provided by the tabu list, are used to implement the exploration strategy. The long-term memory in this algorithm is in the form of an $n \times 336$ matrix M, where an element M_{ip} represents the number of times the pair (worker i, period p) has been involved in a move. Any move associated with this pair is additionally penalized in accordance with the value of $M_{ip.}$ For example, if a candidate swap calls for workers i and j to change positions in period p, then the

actually value used in the neighborhood comparisons is *move_value* + 0.1(M_{ip}+M_{jp}) rather than *move_value* alone.

As mentioned, the size of the neighborhood is polynomial in *n* but still large for most real-world instances. To reduce the computational effort, it is common to limit the search to a certain number of randomly selected candidates within a neighborhood. The certain number, *candidate_num*, is a parameter that is set adaptively according to the size of the problem and the progress of the algorithm, and represents the exploitation strategy. In this implementation, it oscillates between two values: *smallscan* and *bigscan*. Whenever an improved solution is encountered, *candidate_num* is increased to *bigscan* to allow the neighborhood of the solution to be searched more thoroughly; it is reset to *smallscan* after 200 consecutive iterations without improvement.

Algorithm Description

Figure 4.6 highlights the major components of the tabu search procedure. The first step is to generate an initial feasible solution with one of the methods described above. The results are saved as both the *current solution* and the *best solution* found so far; the corresponding objective function value is used to initialize the data element *best_cost*. The *move_value* and the *tabu_list* are also initialized.

Given an initial solution, the algorithm iterates until a prespecified limit (*max_iterations*) is reached or there is no improvement in *max_no_improve* iterations. The move to be made at a given iteration is found by computing the sum of *move_value* and *move_penalty* of all candidate swaps in the neighborhood of the current solution. Because it is the transition cost that is being minimized, the best candidate move is the one associated with the smallest sum. A move is admissible if (1) it is not tabu or (2) its tabu status can be overridden by the aspiration criteria. The *best_move* is then performed and the tabu data structures are updated. The best overall solution (*best_soln*) is updated if the current value of the total cost is less than the objective value of the incumbent.

The data structures used to describe the heuristic are as follows.

- *num_iterations*: current iteration number

- *max_iterations*: limit on total number of iterations that can be performed
- *curr_cost*: sum of the cost of transitions for all workers in a week (objective function value)
- *best_cost*: best total transition cost for all workers obtained so far (incumbent objective function value)
- *curr_soln*: current solution obtained after executing the best move
- *best_soln*: best solution obtained so far (incumbent
- *move_value*: difference in the cost of a transition before and after the swap of the two workers being considered in a certain period
- *best_move_value*: lowest *move_value* among all candidate moves
- *move_penalty*: additional penalty imposed by the long-term memory *freq_matrix M*; when workers *i* and *j* are selected two switch positions in period *p*, *move_penalty* = $0.1(M_{ip}+M_{jp})$
- *best_move_penalty*: *move_penalty* for the best move of all candidate moves
- *no_improve*: number of consecutive iterations during which no better solutions are found
- *max_no_improve*: maximum number of consecutive iterations permitted during which no better solutions are found: 5,000 for all data sets
- *tabu_size*: total number of iterations for which a move is held tabu: 20 in this implementation
- *tabu_list*: short-term memory function that stores recent moves that are forbidden in subsequent iterations
- *freq_matrix*: long-term memory function that records the number of times that each possible move is revisited
- *candidate_num*: number of candidate moves that are scanned during each iteration (can be one of two values, *smallscan* = $n^2/100$ and *bigscan* = $n^2/50$, depending on the situation)
- *candidate move*: solution that results when two workers assigned to different workgroups in the current period swap position

- *admissible move*: candidate move that either satisfies the aspiration level criterion or is not tabu

```
1. Generate initial feasible solution and save it as curr_sol and best_sol.
2. Evaluate curr_cost and set best_cost = curr_cost.
3. Initialize tabu_list and freq_matrix.
4. Set candidate_num = smallcan.
5. Set no_improve = 0.
6. Do{
        best_move_value = ∞
        for (all candidate moves)
        {
                if (move_ status ≠ tabu or move_ value is improving)
                {
                        if (move_value + move_penalty < best_value +
best_move_penalty)
                        {
                                best_move_value = move_value
                                best_move_penalty = move_penalty
                                best_move = current_move
                        }
                }
        }
        execute best_move [ update current solution by swapping ]
        curr_cost = curr_cost + best_move_value
        update tabu_list
        update freq_matrix
        if (curr_cost < best_cost)
        {
                best_cost = curr_cost
                best_soln = curr_soln
                candidate_numb = bigscan
                no_impove = 0
        }else{
                no_impove = no_improve + 1
        }
    } While (num_iterations < max_iterations or no_improve < max_no_improve)
```

Figure 4.6. Tabu search procedure

4.4. Computational Experience

The various algorithms and initialization procedures developed for solving the task assignment problem were tested using data obtained from the Boston P&DC. Table 4.1 lists each approach. All algorithms were coded in Java and run on a Linux workstation with dual Xeon 1.8G CPUs and 1 gigabyte of memory. (Our licensing agreement, however, did not allow for parallel processing.) CPLEX 9.0 was used to solve the embedded IPs and LPs. For the IPs, setting the CPLEX emphasis parameter to "feasibility" rather than "optimality" worked best, as did setting the heuristic frequency parameter to every 15 nodes.

Table 4.1. Approaches Investigated

Methodology	Abbreviation
Solve model (4-2) with idle period nodes	EXACT
Delayed idle period assignment	DIPA
Delayed idle period assignment and daily decomposition	DIPA&DD
Tabu search	TS
Greedy SOS heuristic	SOS
Sequential shortest route algorithm	SSRA

The data sets used in the computations are shown in Table 4.2 and reflect the range of problem sizes that might be encountered in a large facility. Five different worker categories are included with anywhere from 17 to 311 workers and 3 to 28 workgroups. This diversity should allow draw some general conclusions to be drawn from the results.

Of the two groups of data sets in Table 4.2, the instances associated with the first group are relatively small and can be solved to optimality with EXACT within 1 hour. For EXACT, DIPA and DIPA&DD, a 1% optimality gap was used as the stopping criterion for small data sets to ensure that high quality solutions were found within an acceptable amount of time. For discussion purposes, the solution provided by EXACT will be termed the "optimal solution" and used as the reference point for evaluating the performance of the other algorithms.

The instances associated with the second group of data sets are larger than those in the first and could not be solved to within 10% of optimality using a 2-hour stopping criterion. In many cases, not even a lower bound could be found after several hours of computations because their LP relaxations could not be solved. Nevertheless, it is still useful to compare the results obtained with the other methods to get a relative understanding of their performance.

Table 4.2. Data Sets for Computational Experiments

Problem no.	Worker category	Number of workers (n)	Number of workgroups (m)
Small data sets			
1	P5-MPC	17	5
2	P5-FSMO	34	7
3	P5-PPDMO	45	3
4	P5-MPC	85	3
5	P5-MPC	93	3
6	P5-MPC	105	3
7	P5-MPC	116	4
8	MH5-EO	68	10
Large data sets			
9	P5-MPC	197	4
10	MH4	33	10
11	P5-MPC	222	6
12	P5-MPC	311	6
13	P5-MPC	288	7
14	MH4	171	28

Table 4.3 lists the number of variables, number of constraints, and number of nonzero elements in the A matrix of the IP associated with model (4-2) for all data sets. The first set of statistics includes idle time nodes in the formulation and the second does not. The exclusion of idle time nodes reduced the dimensions of these problems by approximately 50%. The symmetry constraints (4-5) were not included in the testing because the size of the LPs and not the number of nodes in the branch and bound trees was the limiting factor. Adding these constraints would have only aggravated the situation.

Table 4.3. Size of Weekly Problem for Model (4-2)

Data Set	Model (4-2) with idle periods			Model (4-2) without idle periods		
	No. of Variables	No. of constraints	No. of nonzeros	No. of variables	No. of constraints	No. of nonzeros
Small data sets						
1	27,736	9,186	68,049	13,077	5,520	29,598
2	20,417	7,493	51,330	7,246	3,211	17,317
3	23,345	11,417	55,870	13,385	7,660	31,134
4	97,722	42,088	221,875	49,825	27,395	113,404
5	123,823	51,548	281,878	57,741	31,443	131,372
6	154,363	62,356	351,605	66,825	35,981	151,850
7	186,408	70,563	427,638	90,504	44,284	204,464
8	110,852	37,390	249,636	50,041	26,252	112,399
Large data sets						
9	380,172	137,358	866,920	168,261	78,686	377,552
10	132,335	27,321	331,790	80,899	21,481	195,628
11	1,059,070	259,894	2,328,239	624,624	186,411	1,351,720
12	1,387,969	248,252	3,073,161	647,369	209,292	1,405,198
13	1,223,668	308,335	2,733,587	718,946	219,333	1,558,397
14	7,482,642	547,204	36,518,862	3,217,536	383,043	16,433,488

4.4.1 Results for Small Data Sets

The objective function values and solution times obtained with all methods for the small instances are reported in Table 4.4. When EXACT and DIPA were used, a 1-hour time limit was set for solving model (4-2) for the week. For DIPA&DD, the time limit was set to 5 minutes for each single-day problem. When tabu search was used, the SOS solution served as the starting point, and "no improvement in 5,000 consecutive iterations" served as the stopping criterion.

The EXACT solutions provided the baseline against which the other solutions were compared. In Table 4.4, the percentage under each "objective value" and "solution time" entry represents the ratio between the EXACT result and results obtained from the respective heuristics. For example, the best solution provided by DIPA&DD for data set 4 is 53.9, which is 115.91% of the optimal solution, 46.5. Similarly, the time for DIPA&DD to find this solution was 4.0 seconds, which is 1.51% of the time used by EXACT.

Because of the need to perform "what if" analysis in the real operating environment, the EXACT solution times are acceptable for only the very small data sets. Empirically speaking, any data set that has an *nm* value larger than 1,000 cannot be solved by EXACT, and most of the time, even the LP relaxation cannot be solved within several hours. In fact, the optimal solution is found by CPLEX build-in heuristic at the first node of branch and bound tree for most problems, except problem 3 and 8. Most of the computational effort went into solving the LP relaxations, generating cuts, and performing build-in heuristics.

Table 4.4. Computational Result of Small Data Sets

Data set	EXACT		DIPA		DIPA&DD		TS	
	Objective value	Solution time (sec)	Objective value	Solution time (sec)	Objective value	Solution time (sec)	Objective value	Solution time (sec)
1	72.5	56.3	78.8	11.4	78.7	3.2	83.8	118.7
	100.00%	100.00%	108.69%	20.25%	108.55%	5.68%	115.59%	210.83%
2	94.6	9.0	98.8	3.6	99	5.2	98.5	245.2
	100.00%	100.00%	104.44%	40.18%	104.65%	58.04%	104.12%	2736.61%
3	68.6	12.5	78.3	8.5	74.2	3.8	94.1	71.6
	100.00%	100.00%	114.14%	68.00%	108.16%	30.40%	137.17%	572.80%
4	46.5	265.0	48.5	30.6	53.9	4.0	60.7	110.2
	100.00%	100.00%	104.30%	11.55%	115.91%	1.51%	130.54%	41.58%
5	74.3	547.9	81.6	59.7	88.6	6.8	95.4	115.0
	100.00%	100.00%	109.83%	10.90%	119.25%	1.24%	128.40%	20.99%
6	81.3	1762.5	85.2	55.5	86.8	5.2	108.2	245.1
	100.00%	100.00%	104.80%	3.15%	106.77%	0.30%	133.09%	13.91%
7	89.9	2722.7	97.1	126.7	103.3	12.7	141.4	175.2
	100.00%	100.00%	108.01%	4.65%	114.91%	0.47%	157.29%	6.43%
8	27.2	3268.2	28.0	484.5	36.1	15.9	43.7	173.8
	100.00%	100.00%	102.94%	14.82%	132.72%	0.49%	160.66%	5.32%
Avg	100.00%	100.00%	107.14%	21.69%	113.86%	12.26%	133.36%	451.06%

The size of model (4-2) without idle period nodes is about 50% of the size of the exact model, and thus can be solved much faster. On average, DIPA found solutions that were only 7.14% above the optimum, on average, in approximately 1/5 of the time. For

the small data sets, this method is effective but cannot be relied on as the problem size increases, as discussed in the next section. DIPA&DD is more efficient in terms of the solution time, which is under 13% of the time required by EXACT, on average, while objective function values remained within 14% of the optimum. Finally, tabu search showed no advantage with respect to solution quality for small instances. In part, this was due to the relatively poor quality of the initial solution provided by the SOS heuristic. This issue is further examined below.

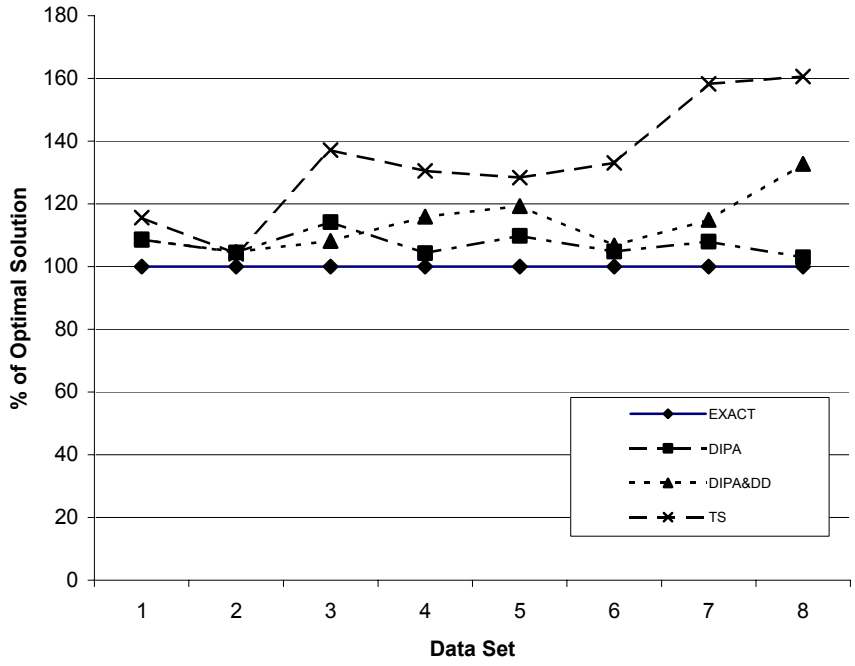


Figure 4.7. Objective function comparisons for small data sets

The graphs in Figs. 4.7 and 4.8 plot the performance of the different methods for the small data sets. From Fig. 4.7, it is easy to see that relative solution quality with respect to the objective function value remains the same for most instances. EXACT gives the best results in general, while DIPA consistently provides solutions that are very close to the optimum. The DIPA&DD solutions are close to or even better than those obtained with DIPA for data sets 1, 2, 4 and 6, but are noticeably worse for data sets 3, 5,

7 and 8. The tabu search results were consistently inferior to those obtained with the other methods.

Figure 4.8 depicts relative run times. For the first three problems, tabu search took much longer than the other methods due to the nonproductive time required to satisfy the termination criteria "no improvement in 5,000 consecutive iterations." The corresponding points are out range. For the remaining problems, tabu search was more competitive but still not as effective. Of the IP approaches, DIPA&DD provided the fastest times, EXACT the slowest times, and DIPA was somewhere in between. The relative run time for tabu search decreased steadily as the problem size increased, and eventually dropped below that of DIPA for data set 8.

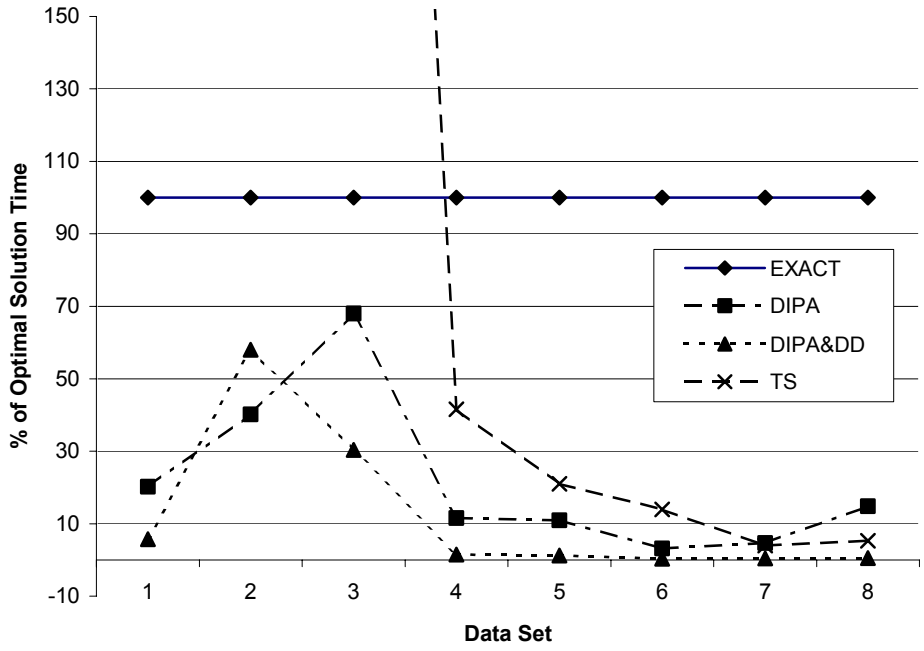


Figure 4.8. Run time comparisons for small data sets

4.4.2. Results for Large Data Sets

Table 4.5 presents the computational results obtained with DIPA, DIPA&DD, TS and SOS for the large data sets. The SOS results are included to provide a baseline for the more difficult instances, and reflect the performance of the system now in use at P&DCs.

Because none of the LP relaxations were solvable by EXACT, no lower bounds are available.

The relative performance of the four methods tested was the same as was observed for the small data sets. DIPA provided the best solutions but could solve only the first two instances; i.e., data sets 9 and 10. Using a 20-minute time limit for each single-day problem, DIPA&DD was able to find relatively good solutions in acceptable time, except for the largest instance – data set 14. None of the daily LP relaxations could not be solved within the allotted time.

The solutions found by tabu search, again, were not as good as those provided by DIPA or DIPA&DD, but it did manage to improve the initial SOS solution by an average of 13.15%. As a consequence of the randomness inherent in the search strategy, no correlation between problem size and computational effort was evident.

Whether the additional time required by any of the proposed methods to improve the initial solution can be justified, depends on the time available for the analysis. In the current system, the SOS heuristic runs in a matter of seconds. This represents a negligible amount time when compared with the options being considered here.

Table 4.5. Computational Results for Large Data Sets

Data Set	DIPA		DIPA&DD		TS		SOS objective value
	Objective value	Solution time (sec)	Objective value	Solution time (sec)	Objective value	Solution time (sec)	
9	52.7	2468.2	61.2	20.2	102.8	753.5	119.8
10	181.9	5785.8	185.4	141.6	200.3	105.8	223.3
11	–	–	300.4	1105.2	421.3	1002.3	451.9
12	–	–	323.0	153.4	440.4	4139.5	551.2
13	–	–	415.6	2948.6	556.7	2008.3	657.2
14	–	–	–	–	815.0	801.7	929.0

4.4.3 Initializing Tabu Search

As opposed to implicit enumeration methods, metaheuristics such as tabu search, require a feasible solution to get started. Although this might be viewed as a disadvantage, when

coupled with other methods, they can be used in a complementary way to improve upon results obtained from those methods. In addition, because the computations can be stopped at any point, metaheuristics are often the best alternative for solving real-world problems that are intractable and not subject to decomposition.

Table 4.6 reports the improvements obtained with tabu search when started from the "final" solutions given in Table 4.5 for DIPA, DIPA&DD and SOS. Also included are the results obtained when started with the SSRA solutions. In each case, the final objective value, the run time, and the percentage improvement are listed.

The first observation is that the better the quality of the initial solution, the less improvement provided by tabu search. This is seen in the bottom row of Table 4.6 and is consistent with intuition; i.e., poorer solutions leave more room for improvement, although more improvement usually means that more time must be spent on the search. A second observation is that while tabu search can improve a solution quite a bit, the quality of the final solution is a function of the quality of the starting solution. For example, tabu search improved the SSRA solutions by 43.84% on average, but only improved the DIPA solutions by 1.82%. Despite this huge difference, the objective function values at termination were respectively 2.7 and 1.3 times greater when the tabu search was started with the SSRA solutions for data sets 9 and 10 than when it was started with the DIPA solutions.

Table 4.6. Tabu Search Started from Different Solutions

Data set	DIPA			DIPA&DD			SOS			SSRA		
	Obj. value	Time (sec)	Improve-ment	Obj. value	Time (sec)	Improve-ment	Obj. value	Time (sec)	Improve-ment	Obj. value	Time (sec)	Improve-ment
9	52.2	258.5	0.95%	60.7	273.4	0.82%	102.8	753.5	14.19%	140.2	1317.9	61.18%
10	177.0	62.8	2.69%	179.3	92.1	3.29%	200.3	105.8	10.30%	232.8	175.4	29.11%
11	–	–	–	292.8	929.6	2.53%	421.3	1002.3	6.77%	684.2	2420.2	42.85%
12	–	–	–	307.2	1897.1	4.89%	440.4	4139.5	20.10%	572.4	1253.8	51.97%
13	–	–	–	407.2	1586.5	2.02%	556.7	2008.3	15.29%	897.9	1723.5	38.19%
14	–	–	–	–	–	–	815.0	801.7	12.27%	1036.3	1200.3	39.75%
Avg	–	–	1.82%	–	–	2.71%	–	–	13.15%	–	–	43.84%

Chapter 5

Weekly Staff Scheduling with Workstation Group Restrictions

As described in previous two chapters, the weekly staff planning problem can solved in a two-phase manner if WSG restrictions are not required. First, a weekly shift scheduling problem is solved for the entire facility to decide the work days, their length, the daily start times, and the lunch breaks for each employee. Second, the task assignment problem is solved to derive individual job assignments for each employee by period over the planning horizon. Sufficient constraints are included in the shift scheduling model to ensure overall feasibility. When WSG restrictions are part of the input, the two-step approach fails because it is not possible to incorporate them in the full model.

5.1. Problems Description and Formulation

As mentioned, most of the mail arriving daily at a P&DC is processed by automated equipment that includes delivery bar-code sorters (DBCSs), multiline optical character readers (MLOCRs), the input subsystem (ISS), and the output subsystems (OSS). This equipment is arranged in WSGs and operated by different categories of employees. To develop a model, let $G = \{\text{WSG}_1,\ldots,\text{WSG}_\nu\}$ be the set of ν WSGs and let $R = (r_{gh})$ be a $\nu \times \nu$ matrix that embodies the authorized movements between each pair (g, h) of elements in G, where $r_{gh} = 1$ if a worker with home base $g \in G$ is allowed to move to $h \in G$, and 0 otherwise. An additional constraint is that each employee must spend at least as much time at his home base as at any other WSG.

The matrix R can be represented graphically as a directed network. Figure 5.1 shows an example in which the nodes are the WSGs, the numbers in square brackets refer to the number of workers whose home base is that WSG, and the arcs show the allowed

movement. The absence of a link between two nodes, such as from BCS to OSS, implies that a worker based at BCS cannot be repositioned to OSS even when there is idle time in his schedule. In contrast, a worker whose home base is ISS can be sent to DBCS or OSS as needed, but must spend a plurality of his time at ISS. It is important to note that such a worker, once repositioned at OSS, for example, cannot then be assigned to BCS. This constraint is one of the factors that greatly increase the difficulty of the scheduling problem.

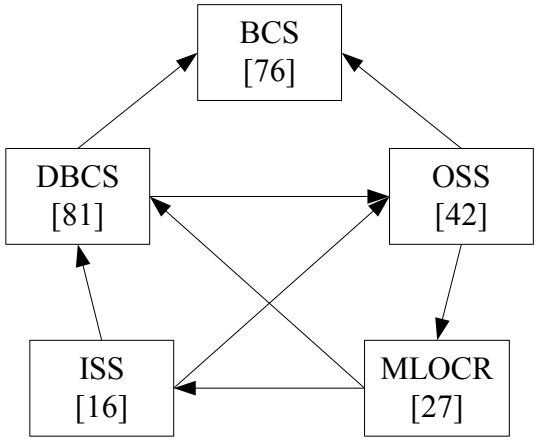


Figure 5.1. Example of Movement Restrictions Network

5.1.1 WSG restrictions

In general, there are two types of restrictions. The first type is imposed by the skill requirement of each WSG. An employee can move from his home base to another WSG only if he is qualified to perform the operation at the new WSG. Figure 5.2 identifies the 18 skill levels in a P&DC, along with the permissible downgrading options. An "×" in a cell means that a worker with the higher skill indicated on the left can substitute for a worker with the lower skill indicated at the top, but not vice versa. For example, a P6-FSMO flat sorting machine operator has greater skill than a P5-MPC mail processing clerk, and so can be downgraded to perform the tasks associated with P5-MPC job assignments.

The second type of restriction is managerial related and is typically a function of

the facility layout. Even when two WSGs require the same skill level, movement between them may be prohibited, or restricted to one direction only, due to physical barriers or transit considerations. Irrespective of the rationale, both types of restrictions are treated the same in the following model.

Higher Skill \ Lower 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				X	X									
P6-GE		N/A			X		X								
P6-PPDMO			N/A		X										
P6-SSMO				N/A	X										
P5-MPC					N/A										
P5-FSMO					X	N/A									
P5-PPDMO					X		N/A								
P4-DCO								N/A							
MH5									N/A					X	
MH5-EQ										N/A				X	
MH5-T											N/A			X	
MH5-MPMO												N/A		X	
MH5-SSMO													N/A	X	
MH4														N/A	
MH4-SSMO														X	N/A

Figure 5.2. Permissible Skill Downgrading

5.1.2 Notation and Formulation

The weekly staff scheduling model with WSG restrictions shares most notations and constraints with the weekly shift scheduling model (3-1) because it also embodies the full set of labor union, legal, and organization constraints specified by the USPS. In addition, new variables and constraints are needed to take WSG restrictions into account. The additional notation and the new model are presented as below.

Indices

g workstation groups, $g \in G$

e (dummy) idle WSG

Parameters

D_{dtg} demand of WSG g for period t on day d

$g(k)$ home base of employee k

$G(k)$ WSGs to which employee k can be assigned other than his home base

Sets

$K(g)$ set of employees who can be assigned to group g

$S(k,d,t)$ set of shifts that employee k is permitted to work on day d that cover period t

$B(k,d,s)$ set of periods in the break window of shift s on day d for employee k

S_c set of casual shifts; each shift is 6 hours

$S_c(t)$ set of casual shifts that cover period t

Decision variables

y_{kdtg} (binary) 1 if employee k is assigned to WSG g for period t on day d; 0 otherwise

θ_{dsg} number of casual shifts of type s assigned on day d for WSG g

Model

Minimize $$z = \sum_{d=1}^{7}\sum_{k\in K}\sum_{s\in S(k,d)} c^1_{kds}x_{kds} + \sum_{d=1}^{7}\sum_{k\in E(d)} c^2_k\gamma_{kd} + \sum_{k\in K^P\cup K^L} c^3_k\mu_k$$

$$+ \sum_{k\in K^P\cup K^L} c^4_k\tau_k + \sum_{d=1}^{7}\sum_{s\in S_c}\sum_{g\in G} c^5_s\theta_{dsg} \qquad (5\text{-}1a)$$

subject to Eq. (3-1f)-(3-1r).

$$\sum_{k\in K(g)} y_{kdtg} + \sum_{s\in S_c(t)} \theta_{dsg} \ge D_{dtg},\ d = 1,\ldots,7;\ t = 1,\ldots,48;\ g \in G \qquad (5\text{-}1b)$$

$$\sum_{g\in G} y_{kdtg} = \sum_{s\in S(k,d,t)} x_{kds},\ k \in K,\ d = 1,\ldots,7;\ t = 1,\ldots,48 \qquad (5\text{-}1c)$$

$$\sum_{t\in B(k,d,s)} y_{kdte} \ge x_{kds},\ k \in K,\ s \in S(k,d),\ d = 1,\ldots,7 \qquad (5\text{-}1d)$$

$$\sum_{d=1}^{7}\sum_{t=1}^{48} y_{kdtg(k)} \ge \sum_{d=1}^{7}\sum_{t=1}^{48} y_{kdtg},\ k \in K,\ g \in G(k) \qquad (5\text{-}1e)$$

$$y_{kdtg} \in \{0,1\},\ \theta_{dtg} \ge 0 \text{ and integer},\ \forall\ k,d,t,g \qquad (5\text{-}1f)$$

The objective function (5-1a) minimizes the total weekly cost of the existing workforce. It is almost identical to (3-1a) except that the casual shift variable θ_{dsg} is

defined for each WSG. The full explanation of each item in the objective function can be found in Section 3.1.2.

Regarding the constraints, (5-1b) ensures that the net workforce is sufficient to cover the demand of each WSG for each period, each day of the week. The net workforce is the total number of part-time and full-time employees who are assigned to WSG g in period t, plus the number of casual shifts that are required when demand exceeds supply. To ensure that the model does not grow without bounds, only 6-hour casual shifts are considered. Constraint (5-1c) connects the shift assignment variables x and task assignment variables y. Employee k can be assigned a task in period t only if he is assigned to a shift that covers this period.

To account for breaks, (5-1d) guarantees that an employee spends at least one period in the idle WSG during the break window of a shift that requires a break. Constraint (5-1e) enforces the home base restriction for each employee. The left-hand side represents the number of periods that employee k spends at his home base $g(k)$, which must be greater than or equal to the number of periods he spends elsewhere. The set $G(k)$ is derived from the matrix R. Variable constraints are defined in (5-1f).

5.1.3 Workforce Priorities

The staff scheduling problem addressed in this paper involves not only a multi-skilled workforce, but cost-sensitive worker categories with different hourly rates for FTRs, PTRs, PTFs, overtime and casuals. Although a manager would generally be driven to minimize overall cost when constructing weekly schedules, labor agreements and company policies may require that work be assigned in an order that conflicts with this objective. For example, the contract between the USPS and its craft union prohibits departures from the bid job assignments unless overtime is paid. The schedules of flexible workers, however, can be adjusted with wide latitude to match demand. In some situations, overtime can be allocated only when no more PTF hours are available; in all cases, casuals can be called in only when there are no other options.

If the cost structure is consistent with the priority ordering of the organization, then model (5-1) will always provide the least cost solution and the priority requirements will be satisfied automatically. In this case, casuals actually have the lowest hourly rates but also have the lowest priority. To enforce priority requirements in general, it is necessary to replace all the cost coefficients in (5-1a) with artificial, scaled values. Suppose there are P wage categories and each is assigned a priority level p ($0 \leqslant p \leqslant P - 1$), where the smaller the level the higher the priority. If a variable associated with priority p in (1) has unit cost c_0, then the penalized cost is set for this variable to $c_0 \times 5^p$.

5.1.4 Need for Decomposition

The proposed model integrates the shift scheduling and task assignment problems, and increases in size in proportion to the number of workers $|K|$ and the number of WSGs $|G|$ for $|T|$ and $|D|$ fixed. For a moderate size problem with 200 workers and 4 WSGs, model (1) – (18) includes about 200,000 integer variables and 100,000 constraints, and is essentially unsolvable with a commercial code. The experience with CPLEX 8.1 shows that more than 30 minutes just to solve the LP relaxation. To deal with this limitation, two decomposition approaches will be introduced.

5.2. Network Splitting

The size of model (5-1) is $O(|K|\cdot|G|\cdot|D|\cdot|T|)$. The simplest and most natural idea is to decompose it by WSG and solve the weekly scheduling problem for each WSG separately. This strategy, however, removes the permissible links between WSGs so the combined solution is likely to be far from optimal (some flow can be maintained by downgrading idle time, but testing showed that this does not much benefit).

As an alternative, the WSGs can be partitioned into several mutually exclusive subsets, $G = \cup_\kappa G_\kappa$, depending on the value of $|K| \times |G|$, and solve the resultant problems in turn. The goal is to preserve as many links in the movement restriction network $N = (G, A)$ as possible while ensuring that feasible solutions to the original problem can be

obtained quickly. As part of the procedure, idle time will be downgraded from one subset to another, as permitted by the matrix R.

To find the best way to split the network N, a second IP is solved sequentially to create the subsets. At iteration κ, the WSGs selected for subset G_κ are removed from the network along with all arcs between G_κ and $G \setminus G_\kappa$ in either direction.

Before presenting the model, some additional notations are introduced

Indices and parameters

g, h	indices for WSGs
r_{gh}	1 if there is a directed link from WSG g to WSG h; 0 otherwise
n_g	number of employees whose home base is WSG g
C_L	lower bound of the size of the subset
C_U	upper bound of the size of the subset
$W_{\max}$	maximal number of WSGs that can be selected for a subset

Decision variables

χ_g	(binary) 1 if WSG g is selected for the current subset; 0 otherwise
ψ_{gh}	(binary) 1 if WSG g is selected and WSG h is not; 0 otherwise

Model

$$\textit{Minimize } \mathrm{z} = \sum_{g \in G} \sum_{h \in G} \left(0.5 n_g r_{gh} + n_h r_{hg}\right) \psi_{gh} \tag{5-2a}$$

$$\text{subject to} \quad C_L \le \left(\sum_{g \in G} n_g \chi_g \right) \sum_{h \in G} \chi_h \le C_U \tag{5-2b}$$

$$\sum_{g \in G} \chi_g \le D_{\max} \tag{5-2c}$$

$$\chi_g - \chi_h \le \psi_{gh}, \ \forall\, g,h \in G, \ \ g \ne h \tag{5-2d}$$

$$\chi_g, \ \psi_{gh} \in \{0,1\}, \ \forall\, g,h \in G, \ \ g \ne h \tag{5-2e}$$

The objective in (5-2a) is to minimize the total weighted sum of the links affected when the subset is identified. To see how this works, suppose that WSG g is selected to

be in the current subset and WSG h is not, so $\psi_{gh} = 1$ and the links between g and h are removed. In general, removing a link will lead to a suboptimal solution but determining the actual degree of degradation is difficult. Because the possibility of movement along link (g,h) is related to the number of workers whose home base is WSG g, this number, n_g, is used to weight the variable ψ_{gh} in the objective function. On the other hand, the outbound links associated with the selected WSGs are not as critical as the inbound links because idle time will be used to partially satisfy the demand of the remaining WSGs. To take this into account, the weights on the outbound links are discounted by 50%.

The number of workers included in each subset is controlled by constraint (5-2b). The upper bound, C_U, is intended to limit the size of model (5-1) that will eventually be solved to derive weekly schedules, while the lower bound, C_L, is used to avoid unnecessarily small subsets and the continuing need to split the network. This constraint assumes a quadratic form that reflects the proportionality of problem with the product of the total number of workers and the number of WSGs in a subset. Of course, all quadratic terms can be linearized by introducing auxiliary variables and constraints as follows: replace the term $\chi_g \chi_h$ with the binary variable ξ_{gh} and add $\xi_{gh} \le \chi_g$, $\xi_{gh} \le \chi_h$, and $\xi_{gh} \ge \chi_g + \chi_h + 1$ to the formulation. The 0-1 integer linear program that results can be solved easily with a commercial code.

Because problem difficulty is more dependent on the number of WSGs than on the number of workers, constraint (5-2c) is included to limit the size of G_κ. Constraint (5-2d) coupled with the minimization objective, guarantees that ψ_{gh} equals 1 if and only if WSG g is selected and WSG h is not. The χ and ψ variables are defined to be binary in (5-2e).

After solving the linearized version of (5-2), a check is made to see whether the remaining WSGs satisfy the upper bound constraint (5-2b). If not, the network is split again. All subsets are placed in a queue Q and solved in the same order in which they were generated.

Network_Splitting_Algorithm

Input: Set of WSGs G, movement restriction matrix $R = (r_{gh})$, number of workers n_g for all $g \in G$, bounds C_L and C_U on the size of the subsets, and bound D_{max} on the number WSGs in a subset

Output: Queue Q comprised of subsets $G_\kappa \subseteq G$, such that $G_\kappa \cap G_l = \varnothing$ and $G = \cup_\kappa G_\kappa$

Step 0: (Initialization) Set up the movement restriction network $N = (G, A)$, where $A = \{(g, h) : r_{gh} = 1 \text{ for all } g \neq h \in G\}$; set $Q = \varnothing$ and $\kappa = 1$.

Step 1: If $|G| = 1$ or $\sum_{g \in G} n_g \leq C_U$, put $Q \leftarrow Q \cup G$ and stop; otherwise go to Step 2.

Step 2: Set up network splitting IP (5-2) and solve to get $\left(\chi^\kappa, \psi^\kappa\right)$.

2a. Let $G_\kappa = \{g : \chi_g^\kappa = 1 \text{ for all } g \in G\}$ and put $Q \leftarrow Q \cup \{G_\kappa\}$

2b. Put $G \leftarrow G \setminus G_\kappa$, $A \leftarrow A \setminus \{(g, h) : g \in G_\kappa \text{ or } h \in G_\kappa\}$, and $\kappa \leftarrow \kappa + 1$; go to Step 1.

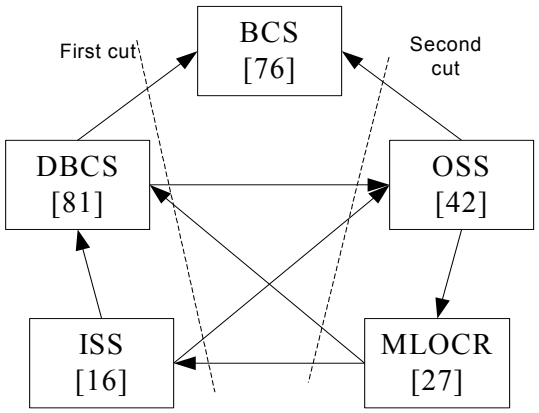


Figure 5.3. Results from Network Splitting Algorithm

IP (5-2) for the network given in Figure 5.1 contains 45 variables and 83 constraints. Using parameter values of $C_L = 100$ and $C_H = 200$, CPLEX is able to find a solution in less than a second. The subset that is identified is G_1 = {DBCS, ISS}, which is placed in Q. After updating the network and solving, the subset generated is G_2 =

{MLOCR, OSS}. Because only one WSG remains, the algorithm terminates with Q = ({DBCS, ISS},{MLOCR, OSS}, {BCS}) . Figure 5.3 shows the two cuts.

5.3. Column Generation Heuristic

An interesting observation about model (5-1) is that it decomposes by worker when the demand (5-1b) and overtime ratio (3-1o) constraints are dropped. This suggests that it may be possible to solve much larger instances by applying Dantzig-Wolfe (D-W) decomposition (i.e., column generation) to the LP relaxation of (5-1) and then branch and price to solve the IP (Wolsey 1998). The advantage of this approach is twofold: first, the LP bound obtained at the root node of the search tree with D-W is usually much better than the bound obtained by solving the LP relaxation of the original model; and second, much less memory is required with D-W because the subproblems (one for each worker here) are solved separately. The subproblem results are used to construct the *master problem*, which typically has relatively few constraints but potentially a large number of columns. In this case, the master problem consists of the demand and overtime ratio constraints.

Although memory limitations are not a concern with D-W, there is still a need to solve, what turns out to be, an extremely large master problem. For model (5-1), trying to use CPLEX to solve the master problem as an IP proved fruitless for all but the smallest instances, so the idea of obtaining exact solutions with branch and price was ruled out. Nevertheless, it was found that by using column generation in a limited way, good feasible solutions can be identified in reasonable amount of time. When the subproblems are solved iteratively, they produce individual weekly schedules that are used to populate the master problem. In the formulation discussed presently, the master problem is designed to select one schedule per worker in a way that satisfies the overtime ratio constraint and as much demand as possible. Because all uncovered demand can be assigned to casuals, a feasible solution can be readily generated from the current columns of the master problem.

5.3.1 Master Problem

In D-W decomposition, each column in the master problem represents a complete weekly schedule for a worker, including the shift assignment for each workday, the break period, and the task assignment by WSG for each period in the week. Let c be the index for columns and let all other indices and parameters be the same as previously defined. In addition, the following notations are introduced.

Parameters and sets

s_k^c	staffing cost of schedule associated with column c for worker k
o_k^c	overtime hours in schedule associated with column c for worker k
l_k^c	total number of working hours in schedule associated with column c for worker k
X_{kdtg}^c	mapping parameter, 1 if schedule c associated with worker k covers period t on day d in group g, 0 otherwise
$C(k)$	set of columns associated with worker k
M	large number

Decision variables

λ_k^c	(binary) 1 if column c associated with worker k is selected, 0 otherwise

Master problem ($\mathcal{MP}$)

$$\text{Minimize } z = \sum_{k\in K}\sum_{c\in C(k)} s_k^c \delta_k^c + M\sum_{d\in D}\sum_{s\in S_c}\sum_{g\in G}\theta_{dsg} \tag{5-3a}$$

$$\text{subject to } \sum_{k\in K}\sum_{c\in C(k)} X_{kdtg}^c \lambda_k^c + \sum_{s\in S_c(t)}\theta_{dsg} = D_{dtg},\ \forall\ d\in D,\ t\in T,\ g\in G \tag{5-3b}$$

$$\sum_{k\in K}\sum_{c\in C(k)} o_k^c \lambda_k^c \le OT_{\text{Ratio}}\sum_{k\in K}\sum_{c\in C(k)} l_k^c \lambda_k^c \tag{5-3c}$$

$$\sum_{c\in C(k)} \lambda_k^c = 1,\ \forall\ k\in K \tag{5-3d}$$

$$\lambda_k^c \in \{0,1\},\ \forall\ c,k,\ u_{dtg}\ge 0 \text{ and integer } \forall\ d,t,g \tag{5-3e}$$

The objective in (5-3a) is to minimize the total staffing cost. Because casuals generally have the lowest priority, the second term is multiplied by a big *M*. Constraints (5-3b) and (5-3c) are the equivalent of (5-1b) and (3-1o), respectively, but written in terms of the schedule variables λ_k^c instead of the original variables (x_{kds}, γ_{kd}, μ_k, τ_k). The parameters $\left(X_{kdtg}^c\right)$ are derived from the solution of the pricing subproblems, which are presented next. The convexity constraint (5-3d) ensures that that exactly one scheduled is selected for each worker.

5.3.2 Pricing Subproblem

The intent of the subproblem, which is defined in part by constraints (5-1c) – (5-1f), (3-1f) –(3-1n) and (3-1p) – (3-1r), is to identify "promising" columns for $\mathcal{MP}$. In the context of linear programming, a promising column is one with a negative reduced cost. When $\mathcal{MP}$ is re-solved with this column included, the new solution should be an improvement over the current solution. If such a column exists for worker *k*, then it can be found by minimizing a generic representation of the reduced cost for that worker. To formulate the objective function for subproblem *k*, let α_{dpg}^1 be the dual variable associated with the demand constraint (5-3b), α^2 the dual variable associated with the overtime ratio constraint (5-3c), and α_k^3 the dual variable associated with the convexity constraint (5-3d). For each worker *k*, the reduced cost for column *c* in $\mathcal{MP}$ is

$$\overline{s}_k^c = s_k^c - \sum_{d\in D}\sum_{t\in T}\sum_{g\in G}\alpha_{dtg}^1 X_{kdtg}^c - \alpha^2\left(OT_{\text{ratio}}\, l_k^c - o_k^c\right) - \alpha_k^3 \qquad (5\text{-}4)$$

To put (5-4) into a form that can be used as an objective function, the coefficients $\left(s_k^c, X_{kdtg}^c, l_k^c, o_k^c\right)$ must be expressed in terms of the original problem variables. Making the appropriate substitutions and removing the column index *c* gives

$$\overline{s}_k = \sum_{d=1}^{7}\sum_{s\in S(k,d)} c_{kds}^1 x_{kds} + \sum_{d=1}^{7} c_k^2 \gamma_{kd} + c_k^3 \mu_k + c_k^4 \tau_k - \sum_{d\in D}\sum_{t\in T}\sum_{g\in G}\alpha_{dtg}^1 y_{kdtg}$$

$$-\alpha^2\left(OT_{\text{ratio}}\sum_{d\in D}\sum_{s\in S(k,d)} l_s x_{kds} - \sum_{d\in D}\sum_{s\in S(k,d)} o_{ks}x_{kds} - \mu_k\right) - \alpha_k^3 \qquad (5\text{-}5)$$

The first four components in (5-5) represent the weekly cost for worker k, while the remaining components are the adjustments imposed by the $\mathcal{MP}$ dual values. In formulating the subproblems, all the constraints in the original formulation (5-1) except (5-1b) and (3-1o) are retained. However, several of those constraints decompose by worker type and so are only included when applicable; e.g., constraint (3-1f) is for regular workers, while (3-1g) is for flexible workers.

Subproblem k ($\mathcal{SP}_k$)

$$z_{\text{SP}}^k = \text{Minimize}\{\,\overline{s}_k : \text{(5-1c)} - \text{(5-1f)},\ \text{(3-1f)} - \text{(3-1n)},\ \text{and}$$
$$\text{(3-1p)} - \text{(3-1r) for } k \text{ fixed}\} \qquad (5\text{-}6)$$

Although (5-6) can generally be solved to within 1% of optimality by CPLEX in less than 1 second, some cases take up to 10 minutes. When a problem instance contains several hundred workers, this can be excess, so to prevent spending too much time generating columns, the solution time of each subproblem is limited to 0.5 seconds. With this restriction, it is possible that the solution to (5-6) will be suboptimal, or not even integral, at termination. If the reduced cost associated with a feasible solution is negative, then the corresponding column can still be added to $\mathcal{MP}$; if no feasible solution is found or the reduced cost of the feasible solution is nonnegative (this usually happened only when there were a few columns in $\mathcal{MP}$), then no columns are generated by the subproblem. Because of the large number of subproblems, early termination of (5-6) was not a limiting factor in optimizing $\mathcal{MP}$.

5.3.3 Initial Columns, Column Management, and Feasible Solutions

Solving the LP relaxation of the master problem with D-W can be accelerated when the process is initiated with good columns. In this case, each worker has a regular weekly

schedule provided by SOS, so the corresponding columns, call them $\left\{\left(s_k^1, X_{kdtg}^1, l_k^1, o_k^1\right) : \forall\ k,d,t,g\right\}$, can be used to start the computations. When the weekly demand $\left\{D_{dtg} : \forall\ d,t,g\right\}$ is close to the demand used by SOS to determine the permanent workforce, these columns are close to optimal and provide much better convergence than realized with a phase I procedure.

For problems with a large number of workers, the number of columns in $\mathcal{MP}$ increases rapidly, eating up memory and reducing computational efficiency. To avert this situation, the following column management procedure has been implemented. Each time $\mathcal{MP}$ is re-solved, the basis status of each column is checked and the columns with an "at lower" status for five consecutive iterations are removed. The status "at lower" means that the variable is nonbasic at its lower bound. With this procedure, the number of columns increases much more slowly and remains stable after a dozen or so iterations. Approximately 10 columns per worker was the norm.

Most applications of column generation exhibit what is called a "tailing off effect" where improvement slows as more columns are added. It is also experienced in this problem, so rather than trying to solve the LP master to optimality, the D-W computations is stopped if convergence did not occur within 10 – 30 minutes, depending on the problem size. Based on the LP solution at that point, the three columns with highest fractional λ_k^c values were then selected for each $k \in K$ and the corresponding version of (5-3) was solved as an IP to get a feasible solution to (5-1). Even with only 3 $\times$ $|K|$ columns for the λ variables, though, the resultant problem may still be beyond an exact solution due to the large number of θ variables as well as the large number of demand constraints in (5-3b).

5.3.4 Heuristic for IP Master Problem

If the overtime (OT) ratio constraint (5-3c) in $\mathcal{MP}$ is momentarily ignored, a set-partition/set-covering-type problem with SOS constraints (5-3d) results. This observation suggests that it may be possible to find good feasible solutions of much larger instances

with a set-covering heuristic. The one proposed by Chvatal (1979) was adapted for this purpose. The basic idea is to rank the columns according to their benefit-to-cost ratio and pick the one with the highest ratio. The costs and benefits are then recomputed and the process is repeated until all demand is covered. The idea can be combined with a randomized procedure, such as GRASP (see Feo and Bard 1989), with just a bit more effort.

When D-W terminates, the first step is to compute the benefit-to-cost ratio of each column in $\mathcal{MP}$, including the columns that represent casual shifts. If column c for worker k covers the demand of group g in period t of day d, then the marginal benefit of selecting that column is $1/D_{dtg}$. The total benefit is the sum of the marginal benefits over all days, periods and groups, i.e., $\sum_{d,t,g} x^c_{kdtg} / D_{dtg}$. The ratio is computed as: $r^c_k = \left(\sum_{d,t,g} x^c_{kdtg} / D_{dtg}\right) / s^c_k$. Next a restricted candidate list consisting of the n_{list} highest ratio values is constructed and one is selected randomly at the current iteration ($n_{\text{list}} = 5$ was used in the implementation). The corresponding column is added to the solution and the demand D_{dtg} is updated for all d, t, g. Because only one column can be selected for each worker in accordance with constraint (5-3d), all other columns for that worker are removed from $\mathcal{MP}$.

To ensure that constraint (5-3c) is satisfied, the OT ratio associated with the current solution is calculated after a column is selected and, if it is above 6%, only columns with an OT ratio less than or equal to 6% will be considered in the next round. Otherwise, all remaining columns are eligible. Note that with this scheme, it is possible that the final OT ratio will be slightly higher than 6% in some cases.

The process is repeated until all the demand is covered, implying that the selected columns constitute a feasible solution. The entire process is repeated a predetermined number of times to see whether a better solution can be found.

5.3.5 Post-processor

A weekly schedule for each worker consists of as set of shift assignments and complementary days off, overtime, and task assignments by WSG. Each of these components is encoded either directly or indirectly in the parameters $\left(X^c_{kdtg}\right)$ that define the columns in $\mathcal{MP}$. Because of the flexibility in assigning tasks, however, it may be possible to improve the results by removing some of the casual shifts once the tours are determined. This is done by solving the task assignment problem separately with a post-processor.

The task assignment problem can be viewed as a reduced version of model (5-1) in which the shift variables (x_{kds}, γ_{kd}, μ_k, τ_k) are fixed in accordance with the set of tours found by the IP heuristic. As such, the working periods are known for each employee. Let $T(k, d)$ be the set of working periods for employee k on day d and let $s(k, d)$ be the shift specified for employee k on day d. To minimize the number of casual shifts required for the week, the following IP is used to reassign tasks.

$$\theta^* = \text{ Minimize } \sum_{d\in D}\sum_{s\in S_c}\sum_{g\in G}\theta_{dsg} \tag{5-7a}$$

$$\text{subject to } \sum_{k\in K(g)} y_{kdtg} + \sum_{s\in S_c(t)} \theta_{dsg} \geq D_{dtg},\ d \in D,\ t \in T, g \in G \tag{5-7b}$$

$$\sum_{g\in G} y_{kdtg} = 1,\ t \in T(k, d), k \in K, d \in D \tag{5-7c}$$

$$\sum_{t\in B(k,d,s(k,d))} y_{kdte} \geq 1,\ k \in K, d \in D \tag{5-7d}$$

$$\sum_{d\in D}\sum_{t\in T} y_{kdtg(k)} \geq \sum_{d\in D}\sum_{t\in T} y_{kdtg},\ k \in K,\ g \in G(k) \tag{5-7e}$$

$$y_{kdtg} \in \{0,1\},\ \theta_{dsg} \geq 0 \text{ and integer } \forall\ k,d,t,g,s \tag{5-7f}$$

Model (5-7) is still quite large but much easier to solve than the original problem. Computational experiences indicated that in most cases, it could be solved by CPLEX within 10 minutes using a 1% stopping criterion. Regardless of optimality, though, the

value of θ^* in (5-7a) was always less than the comparable value in (5-3a) found by the IP heuristic.

5.3.6 Details of Column Generation Algorithm

Following is the combined procedure that includes the solution of the master problem $\mathcal{MP}$, the subproblems $\mathcal{SP}_k$, the IP heuristic, and the post-processor.

Input: Permanent workforce K along with their bid jobs, updated weekly demand $\{D_{dtg} : \forall\ d, t, g\}$, workgroup restrictions R, remaining parameters and sets that define model (5-1)

Output: Weekly schedules for the workforce $S^* = \left\{s_k^* : k \in K\right\}$, where $s_k =$ $\left\{\left(x_{kds}, y_{kdtg}, \gamma_{kd}, \delta_{kd}, \mu_k, \tau_k\right) : d \in D, t \in T, g \in G(k), s \in S(k,d)\right\}$

Step 0: (Initialization) Let $C(k)$ be the set of columns for worker k and let $S(k)$ be the set of schedules corresponding to columns in $C(k)$. Let s_k^0 be the long-term schedule of employee k determined by SOS, and set $C(k) = \{0\}$ and $S(k) = \left\{s_k^0\right\}$. Set up $\mathcal{MP}$ (5-3) with initial columns $C(k)$ for all $k \in K$.

Step 1: (Column generation) Let ν_k^c be the number of iterations that column c associated with employee k has been "at lower" basis status. Set $\nu_k^c = 0$.

1a. Solve the LP relaxation of $\mathcal{MP}$.

1b. For all $k \in K$ and $c \in C(k)$, if the basis status is "at lower," then put $\nu_k^c \leftarrow \nu_k^c + 1$; otherwise, put $\nu_k^c \leftarrow 0$. If $\nu_k^c \geq 5$, then put $C(k) \leftarrow C(k) \setminus \{c\}$ and $S(k) \leftarrow S(k) \setminus \left\{s_k^c\right\}$

1c. For all $k \in K$, solve $\mathcal{SP}_k$ (5-6). Let z_{SP}^k be the optimal objective function value and let s_k^c be the column that corresponds to the solution. If $z_{\text{SP}}^k < 0$, put $C(k) \leftarrow C(k) \cup \{c\}$ and $S(k) \leftarrow S(k) \cup \left\{s_k^c\right\}$.

1d. If $z_{SP}^{k} \geq 0$ for all $k \in K$ or time limit is reached, go to Step 2. Otherwise, go to Step 1a.

Step 2: (IP Heuristic)

2a. For all $k \in K$, let $\bar{C}(k)$ and $\bar{S}(k)$ be the set of columns and schedules, respectively, for worker k selected for the IP master. Set $\bar{C}(k) = \varnothing$, $\bar{S}(k) = \varnothing$.

2b. Let $i = \text{argmax}\left\{\lambda_k^c : c \in C(k)\right\}$. Put $\bar{C}(k) \leftarrow \bar{C}(k) \cup \{i\}$, $\bar{S}(k) \leftarrow \bar{S}(k) \cup \left\{s_k^i\right\}$, and $\lambda_k^c = 0$. If $\left|\bar{C}(k)\right| = 3$, go to Step 2c; otherwise repeat.

2c. For all $k \in K$, remove all columns $c \in C(k) \setminus \bar{C}(k)$ from $\mathcal{MP}$.

2d. Solve $\mathcal{MP}$ as an IP with columns $\bigcup_{k \in K} \bar{C}(k)$. Call the solution $\lambda^* = \left(\lambda_k^{c*} : k \in K\right)$ and denoted by $c^* \in \bar{C}(k)$, the corresponding columns.

2e. Extract s_k^{c*} from $\bar{S}(k)$ and set $s_k^* = s_k^{c*}$ for all $k \in K$.

Step 3: (Post-processing)

3a. Construct $T(k, d)$ and $s(k, d)$ from s_k^{c*} for all $k \in K$ and $d \in D$, and set up the task assignment problem (5-7).

3b. Solve to get $y^* = \left(y_k^*\right)$ and update the value of the corresponding components in s_k^* for all $k \in K$. Return S^*.

The tour found in Step 2 and the task assignments found in Step 3 represent the solution to the original problem. While several alternatives to Steps 1 and 2 were explored in the development stages, they did not lead to any noticeable improvement, and thus their results are not include in the computational report. One example was the application of a GRASP, as described in Section 5.3.4, to solve $\mathcal{MP}$ using all the columns in the final LP solution rather than just three for each worker.

5.4. Experimental Results

A series of tests was performed to evaluate the tractability of the models and the effectiveness of the network splitting and column generation heuristics in finding high quality solutions. Three data sets were used corresponding to small, medium, and large instances, respectively. With a few exceptions, only those associated with the small data set could be solved directly with CPLEX. For the large data set instances, CPLEX was not able to find any feasible integer solutions after several hours of computations.

Recall that in weekly scheduling at P&DCs, the size of the workforce and the bid jobs are fixed, so overtime, extended part-time hours, and casuals are the options available to meet changing demand. For each data set, five related scenarios were investigated, each corresponding to a different level of demand with respect to the baseline. They are denoted by 80%, 90%, 100%, 110%, and 120%, respectively, where the "percentage" refers to the ratio of the actual weekly demand to the long-term demand used by SOS to determine the permanent workforce. The former is obtained by increasing or decreasing the mail arrival volumes by the given percentages and then generating a new equipment schedule using the model developed by Zhang and Bard (2005). In other words, the actual demand is not a period-by-period fixed multiple of the long-term demand.

In general, problems are easier to solve when the demand is close to or lower than the baseline; i.e., the 100% scenario. In these cases, the optimal schedules are either close to the bid job assignments or can be obtained simply by reducing part-time hours. When the demand is above the baseline, the full set of options may be needed, making the problem much more difficult to solve. Interesting, even when the demand is set to 100%, it is not always possible for either the network splitting algorithm or column generation algorithm to find the optimal solution (which is the solution provided by SOS) due to the presence of WSG restrictions. When the long-term problem is solved by SOS, the solution includes the home base designation; when the weekly problem is solved, the home base is fixed so additional constraints exist. The feasible region that results when the network splitting algorithm is applied is likely to be tighter than the feasible region

associated with the original problem so the optimal solution may not be feasible. The fact that the column generation algorithm stops short of solving the full master problem as an IP implies suboptimality.

All data used in the testing were provided by the Dallas P&DC. The algorithms were coded in Java and run on a Dell PC with a P4-2.53G processor and 512mb of memory. CPLEX 8.1 was used to solve all LPs and IPs included in the procedures. For the latter computations, the MIP emphasis option was set to 'feasibility' and the built-in feasibility heuristic was called at every 5 nodes in the search tree. Finally, it should be mentioned that many more problem instances than those presented here were solved during the development phase of the project. Because the results were invariant, the summary is limited to examining the effect of model size on computational performance.

5.4.1 Small Data Set

The movement restriction network for the small data set is shown in Figure 5.4, which is the same as the network in Figure 5.1 but with the BCS node and connecting arcs removed. There are four WSGs and 80 workers, 65 being FTRs and 15 being PTFs. The number of workers in each WSG is given in the square brackets.

In general, the size of a problem instance varies with the number for workers $|K|$, the number of workgroups $|G|$, and the connectivity of the matrix R. Therefore, each instance arising from the same data set should have the same dimensions. Table 5.1 lists the size of the small data set problems after being tightened by CPLEX's preprocessor. Even with only 80 employees, they are already overwhelmingly large, with over 65,000 variables and 35,000 constraints.

Table 5.1. Small Data Set Problem Sizes

Demand level	Number of variables	Number of constraints	Number of non-zeros	Density of A matrix
80%	66,524	35,831	834,678	0.04%
90%	67,199	36,045	838,183	0.03%
100%	67,163	36,009	837,793	0.03%
110%	66,934	35,957	837,229	0.03%
120%	67,358	36,081	838,499	0.03%

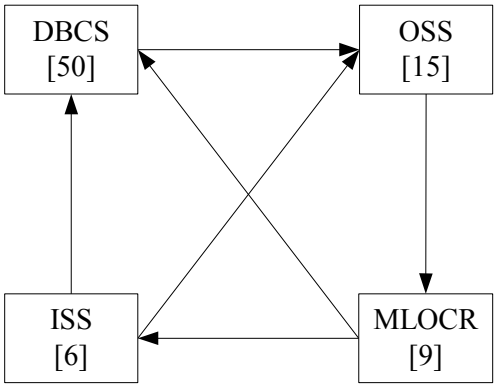


Figure 5.4. Movement Restriction Network for Small Data Set

Table 5.2 displays the computational results obtained by solving (5-1) directly with CPLEX. A 1-hour time limit and a 1% optimality gap were used as the stopping criteria for all scenarios, which are listed in the first column. The second column gives the IP solution and the third column the solution time in seconds. The next column identifies the node in the search tree at which the best (optimal) was found. A "+" signifies that the built-in heuristic found the solution. The column "Opt. gap" gives the percentage gap between the feasible solution at termination and the best lower bound using the artificial cost coefficients discussed in Section 5.1.3. In all experiments, the order was part-time hours, (full-time and part-time) overtime, then casuals. As expected, instances with 100% or less demand were easily solved to within 1% of optimality in less than 65 seconds. In contrast, when the demand level was 10% above the baseline, more than 11 minutes were required, and when it was 20% above the baseline, the best that could be achieved was a 7.39% gap after the full hour.

Table 5.2. Computational Results Obtained with CPLEX for Small Data Set

Demand level	IP solution	Solution time (sec)	Node best solution	Opt. gap	Weekly cost	Average PTF hours	OT ratio	CAS ratio	Idle rate
80%	$77,170	24.6	0+	0%	$77,170	11.87	0%	0%	27.61%
90%	$78,071	29.4	0+	0%	$78,071	14.20	0%	0%	20.30%
100%	$78,141	64.4	0+	0.09%	$78,141	14.47	0%	0%	16.77%
110%	$80,610	689.2	500+	0.53%	$80,610	19.40	0%	0%	14.84%
120%	$98,852	3609.7	735+	7.39%	$83,108	23.20	0.81%	0%	13.41%

Because artificial cost coefficients are used in the objective function, the value of the solution reported in column 2 of Table 5.2 does not reflect the true costs, which are given in column 6. For the first four scenarios, these costs exactly match the IP value because none of the options associated with the penalized costs is included in the solutions. In particular, PTF hours in column 7 are well below the maximum, and both the OT ratio (percentage of overtime with respect to total working hours) in column 8 and the CAS ratio (percentage of casual hours with respect to total working hours) in column 9 are 0%. As desired, overtime kicks in only when the PTF hours are nearly exhausted. For the 120% scenario, the average PTF hours approach the limit of 24 hours/week and overtime is used to satisfy extra demand. No casuals are needed. The idle rate reported in the last column is the percentage of total idle hours with respect to total working hours, and is seen to decrease with demand.

The network splitting and column generation algorithms were also tested on the small data set instances in order to compare the quality of their solutions with the solutions obtained with CPLEX The results are reported in Table 5.3. When the network splitting algorithm was run, it produced the following two subsets: {DBCS, OSS} and {AFCS, MLOCR}. Not surprisingly, this partition is different than one produced by the first cut shown in Figure 5.3.

Using a time limit 10 minutes for each subset, the full problem can again be solved quickly when the demand is low, but is much more difficult for the 110% and 120% scenarios (see column 3). The solution quality reported in column 2 also deteriorates rapidly. Column 4 gives the percentage gap with respect to the solution found by CPLEX, which goes as high as 357%. A comparison of the true weekly costs, though, gives a much more favorable picture because the gap is only 4% on average. The remaining four statistics are in line with the results in Table 5.1.

In contrast, the column generation algorithm is not noticeably affected by the level of demand. The majority of time reported in column 3 is used to solve the LP relaxation of $\mathcal{MP}$, which, if left unchecked takes more than an hour due to the tailing off effect. As a consequence, a time limit of 10 minutes was placed on the D-W

computations for the small data set instances. The LP solution obtained at that point was always within 5% of the optimal LP solution. In Step 2 of the algorithm, a 5-min time limit was imposed when solving $\mathcal{MP}$ as an IP. CPLEX required anywhere from a fraction of a second to several seconds to solve each subproblem (5-6), $\mathcal{SP}$.

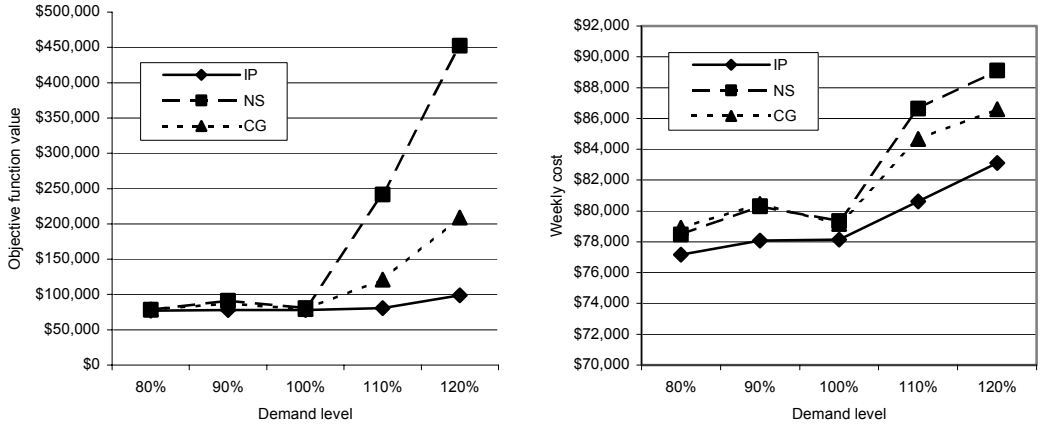
From column 3 in Table 5.3, it can be seen that the column generation heuristic invariably required more time than the network splitting algorithm, especially for the easy cases where the demand is low. This relative inefficiency is overshadowed by the improved solution quality, with the difference increasing as the demand increased. On average, the gap between the true weekly cost obtained with CPLEX and the column generation heuristic was 3%.

Table 5.3. Computational Results for Small Data Set Obtained with Heuristics

Demand level	Obj. value	Solution time (sec)	Gap with IP	Weekly cost	Average PTF hours	OT ratio	CAS ratio	Idle rate
Network splitting								
80%	$78,469	30.2	1.68%	$78,469	15.4	0%	0%	28.70%
90%	$91,299	28.8	16.94%	$80,297	17.8	0.80%	0%	22.15%
100%	$80,857	47.8	3.48%	$79,339	17.4	0.14%	0%	17.91%
110%	$241,503	625.6	199.59%	$86,648	20.6	4.80%	0.20%	19.06%
120%	$452,386	616.1	357.64%	$89,105	22.6	5.86%	0.64%	17.15%
Column generation								
80%	$78,901	766.8	2.24%	$78,902	16.6	0%	0%	29.16%
90%	$87,149	933.6	11.63%	$80,453	20.9	0%	0.15%	22.91%
100%	$79,171	678.2	1.32%	$79,171	17.4	0%	0%	17.79%
110%	$120,904	940.5	49.99%	$84,674	23.3	2.26%	0.49%	18.32%
120%	$209,035	932.2	111.46%	$86,605	23.3	3.15%	2.15%	16.42%

When demand is low, the solutions found by either heuristic are still comparable with the optimal solutions found by CPLEX. For the high demand scenarios, the gap gets quite large primarily because of the use of scaled cost coefficients. A detailed examination of the heuristic solutions revealed that the regular workforce was not used as efficiently as possible due to the decomposition. As a consequence, it was necessary to cover the increased demand with overtime, part-time hours, and casuals, which have

lower priority than the regular workforce and hence higher costs. Ideally, the lower priority options are used only after the regular workforce is fully engaged.



(a) (b)

Figure 5.5. Comparison of Costs for Small Data Set

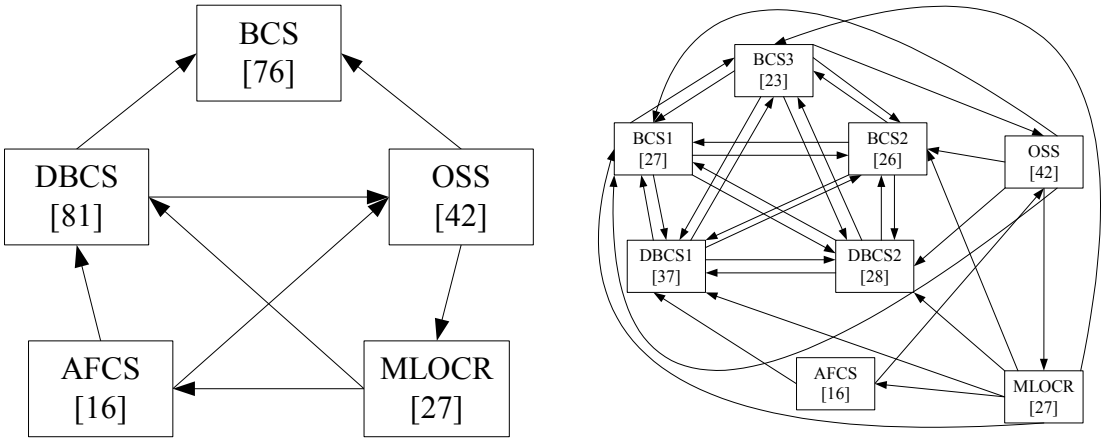
Even with cost coefficient scaling, it was still difficult to find solutions with either heuristic that strictly followed the priority order. For network splitting, while the priorities are satisfied in each subset by design, they were rarely satisfied in the aggregate because of the way the demand is distributed among the WSGs. On the other hand, the column generation heuristic can accommodate the priority order of all options except for the use of casuals, which are not included in the subproblems. In all other cases, the subproblem solutions take into account the priorities and hence produce columns for $\mathcal{MP}$ with appropriately weighted cost coefficients. When $\mathcal{MP}$ was solved as an IP at Step 2, however, it was occasionally necessary to use casuals despite their high costs. The reasons were twofold. First, the columns generated by D-W did not always contain sufficient overtime or part-time hours. Second, the LP solution rarely required overtime even when demand was high, so the three best columns almost never included it. The results would probably have been the same if all the columns were used at Step 2.

Without a full branch-and-price implementation, suboptimality can be expected for virtually all data sets above the baseline.

Figures 5.5a and 5.5b plot the IP solution and the actual weekly costs, respectively, as a function of demand. A close examination of the graphs reveals that the differences are much less dramatic than indicated by the IP objective function values when viewed separately.

5.4.2 Medium and Large Data Sets

The value of the heuristics is not obvious for the small data set instances because CPLEX can find optimal or good solutions in a relatively small amount of time. For larger problems, however, CPLEX was not able to find feasible solutions within several hours of computations. To further test the heuristics, two additional data sets were investigated. The medium data set is associated with the network in Figure 5.6a (the same as in Figure 5.1) and consists of 242 workers and 5 WSGs. The large data set consists of 228 workers and 8 WSGs. The accompanying movement restriction network is displayed in Figure 5.6b.



(a) (b)

Figure 5.6. Movement Restriction Networks for Medium and Large Data Sets

The network splitting algorithm divided the medium problems into 3 subgroups as mentioned, and the large problems into 4 subgroups in the following order: {AFCS, OSS}, {BCS2, MLOCR}, {DBCS1, DBCS2}, {BSC1, BCS3}. The solution time for each subgroup was again limited to 10 minutes. For the column generation algorithm, a time limit for solving the LP relaxation with D-W at Step 1 was set at 20 and 30 minutes, respectively, for the two data sets. These values were halved to 10 and 15 minutes apiece at Step 2 for solving $\mathcal{MP}$ as an IP.

Table 5.4. Medium and Large Data Set Problem Sizes

Demand level	Number of variables	Number of constraints	Num of non-zeros	Density of *A* matrix
Medium data set				
80%	202,138	104,861	2,608,147	0.01%
90%	205,375	134,635	2,816,463	0.01%
100%	206,830	105,744	2,629,689	0.01%
110%	207,437	105,784	2,632,936	0.01%
120%	207,667	105,904	2,634,778	0.01%
Large data set				
80%	258,170	107,332	2,876,676	0.01%
90%	272,722	108,799	2,941,095	0.01%
100%	275,478	108,875	2,952,205	0.01%
110%	276,972	108,981	2,959,689	0.01%
120%	280,109	109,121	2,973,273	0.01%

The problem sizes for both data sets are listed in Table 5.4 and are seen to be extremely large on all measures except the density of the *A* matrix. CPLEX could only find an optimal solution to the medium 80% instance in less than an hour (1771 sec with a 0.78% optimality gap) and to the medium 100% instance in 4005 sec with a 2.21% optimality gap. For the 110% and 120% instances in the large data set, almost an hour was needed just to solve the LP relaxation.

The computational results for the medium and large data sets are summarized in Tables 5.5 and 5.6, respectively. The results are consistent with what was observed for the small data set. In general, both heuristics can find feasible solutions in the time allotted, but the gap between the solution and the LP lower bound is excessive in many

cases, especially when the demand is high. In terms of solution quality, the column generation heuristic found better feasible solutions than the network splitting algorithm in most cases, as reported in column 7, 'Weekly cost.' However, the network splitting algorithm runs quite a bit faster, especially for the low demand scenarios.

Table 5.5. Computational Results for Medium Data Set

Demand level	LP solution	LP time (sec)	Obj. value	Total time (sec)	Gap with LP	Weekly cost	Average PTF hours	OT ratio	CAS ratio	Idle rate
Network splitting										
80%	$231,817	83.8	$233,769	110.8	0.84%	$233,769	13.3	0%	0%	28.11%
90%	$234,292	118.3	$301,327	704.7	28.61%	$240,477	18.2	0.32%	0.05%	22.32%
100%	$233,634	139.8	$351,443	736.2	50.42%	$238,594	16.5	0.33%	0.09%	18.04%
110%	$242,108	175.3	$687,049	1885.5	183.78%	$253,736	23.1	2.29%	0.30%	18.52%
120%	$293,491	277.0	$1,361,464	1856.9	363.89%	$263,747	22.6	4.80%	0.68%	16.63%
Column generation										
80%	$231,817	83.8	$250,869	1954.6	8.22%	$238,965	17.9	0%	0.09%	29.99%
90%	$234,292	118.3	$316,606	1956.4	35.13%	$242,049	19.7	0.14%	0.55%	23.20%
100%	$233,634	139.8	$237,559	1438.0	1.68%	$237,559	16.7	0%	0%	17.81%
110%	$242,108	175.3	$411,687	1996.4	70.04%	$261,154	22.7	4.24%	0.44%	22.29%
120%	$293,491	277.0	$490,992	1979.5	67.29%	$286,098	23.2	5.12%	0.59%	17.18%

Table 5.6. Computational Results for Large Data Set

Demand level	LP solution	LP time (sec)	Obj. value	Total time (sec)	Gap with LP	Weekly cost	Average PTF hours	OT ratio	CAS ratio	Idle rate
Network splitting										
80%	$220,563	1002.5	$298,229	262.0	35.21%	$229,942	16.5	1.99%	0.00%	26.37%
90%	$224,702	1249.8	$776,023	1411.4	245.36%	$240,845	19.3	4.17%	0.16%	20.97%
100%	$223,382	897.6	$943,712	1854.9	322.47%	$244,605	21.0	4.46%	0.35%	18.63%
110%	$254,689	3408.0	$2,362,614	1927.7	827.65%	$247,973	21.9	4.71%	1.56%	16.02%
120%	$371,012	3275.8	$4,903,786	2450.0	1221.73%	$253,829	22.5	5.16%	3.78%	14.60%
Column generation										
80%	$220,563	1002.5	$255,144	3471.6	15.68%	$226,872	19.9	0.00%	0.23%	26.46%
90%	$224,702	1249.8	$337,256	3681.1	50.09%	$233,836	22.4	1.10%	0.76%	23.18%
100%	$223,382	897.6	$225,661	3105.5	1.02%	$225,661	19.0	0.00%	0.00%	13.67%
110%	$254,689	3408.0	$438,428	4976.9	72.14%	$251,993	23.1	5.05%	0.49%	16.02%
120%	$371,012	3275.8	$1,082,380	3914.1	191.74%	$258,978	23.9	5.72%	4.15%	16.18%

Another advantage of the column generation heuristic is that it does a better job at solving the problem when the actual demand is close to or equal to the long-term demand. This is because the bid job of each worker is used to initialize $\mathcal{MP}$, and some or all of these columns can be used to construct a good feasible solution. For both the

medium and large data sets, no overtime or casuals were used for the 100% scenario.

Regarding the priority order, little more can be said than it is difficult for either method to enforce it exactly. For the high demand scenarios, the results are very close to the desired order. For the lower demand scenarios, a curious observation was made. Depending on the algorithm and the particular data set, both the LP solution in column 2 and weekly cost corresponding to the IP solution in column 7 were higher for the 80% and 90% scenarios than for the 100% scenario. This can explained by a combination of suboptimality and a mismatch between the hourly demand patterns that are associated with the 80% and 90% scenarios and the bid jobs of the permanent workforce. In the latter case, the implication is that it may be better to maintain the same equipment schedule that was used to construct the permanent workforce than to adjust it when the weekly mail arrival volume falls.

5.4.3 Column Generation Post-processor

As discussed in Section 5.3.5, a post-processor is used to reduce the number of casual shifts found by the IP heuristic. Table 5.7 gives the column generation results before and after post-processing, which are seen to be quite impressive. Columns 2 and 3 indicate that quite a few casual shifts were eliminated at Step 3 of the algorithm. In general, this was almost always the case because many different task assignments exist for each one-day schedule. In other words, the full $\mathcal{MP}$ may have columns with identical one-day shift patterns but with different task assignments for each worker. Because the LP relaxation of $\mathcal{MP}$ is not solved to optimality, the column with the best task assignment may not be generated, which leaves room for improvement during post-processing.

In addition, the casual shifts have the lowest priority among all the scheduling options but have the largest cost coefficients so the objective function value in (5-3a) decreases markedly as casual shifts are eliminated. The output data in column 6 shows that the objective function value decreased by approximately 45% on average. After rescaling the cost coefficients, however, the true weekly cost only decreased by an average of 1.5%.

Table 5.7. Effectiveness of Post-processing

Demand level	No. of casual shifts		Objective function value			Weekly cost		
	Before	After	Before	After	Decrease	Before	After	Decrease
Small data set								
80%	9	0	$159,902	$78,902	50.66%	$79,550	$78,902	0.81%
90%	18	1	$242,399	$87,149	64.05%	$81,677	$80,453	1.50%
100%	0	0	$79,171	$79,171	0%	$79,171	$79,171	0%
110%	31	3	$377,404	$120,904	67.96%	$86,690	$84,674	2.33%
120%	44	13	$504,535	$209,035	58.57%	$88,837	$86,605	2.51%
Medium data set								
80%	14	2	$364,869	$250,869	31.24%	$239,829	$238,965	0.36%
90%	37	11	$576,856	$316,606	45.12%	$243,921	$242,049	0.77%
100%	0	0	$237,559	$237,559	0%	$237,559	$237,559	0%
110%	65	7	$943,437	$411,687	56.36%	$265,330	$261,154	1.57%
120%	75	10	$1,082,743	$490,993	54.65%	$290,778	$286,098	1.61%
Large data set								
80%	17	4	$379,644	$255,144	32.79%	$227,808	$226,872	0.41%
90%	108	11	$1,225,256	$337,256	72.47%	$240,820	$233,836	2.90%
100%	0	0	$225,661	$225,661	0%	$225,661	$225,661	0%
110%	126	8	$1,507,179	$438,429	70.91%	$260,489	$251,993	3.26%
120%	286	70	$3,075,880	$1,082,380	64.81%	$274,530	$258,978	5.66%

Chapter 6

Long-term Staff Planning with WorkStation Group Restrictions

In previous chapters, the problems surrounding weekly staff scheduling are investigated. In the weekly stage, i.e. mid-term, the permanent workforce is given as an input from the long-term scheduling, which determines the optimal size and composition of a permanent workforce. The long-term scheduling problem has been studied intensively and a decision support system known as SOS has been developed for long-term planning at P&DCs. However, the workstation group restrictions have never been addressed in any related researches. The purpose of this chapter is to investigate the impact of WSG restrictions on the long-term staff scheduling.

6.1. Problem Definition

For a specific P&DC, given a set $J = \{WSG_1,\ldots,WSG_n\}$ of n workstation groups and an $n \times n$ matrix R that embodies the movement restrictions between each pair (j,k) of elements in J, the workstation group restriction-assignment problem (WGAP) is to find the minimum size workforce required to meet the staffing demand over the planning horizon. 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 highlighted below, was designed to construct bid jobs for full-time regular employees and to develop weekly schedules for part-time flexible 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 (8½ hrs) 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. for an alternative system (2000)].

While the solution to the tour scheduling problem specifies the work days and shift assignments for each employee, 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, 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.

6.1.1 Workstation Group Movement Restrictions

In P&DCs, task assignments are generally associated with a specific machine or WSG, such as a DBCS or MLOCR (see Table B.1 for definitions). Although it is desirable that each employee be assigned to a single machine 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 run 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’

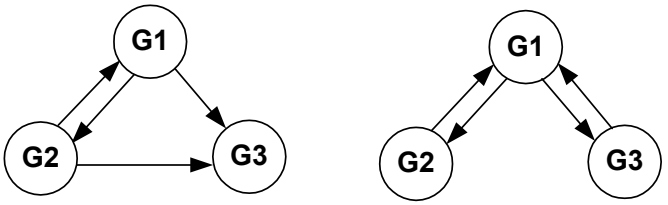
assignment of tasks is characterized by as few switches among WSGs as possible; that is, one that minimizes some function of the total number of switches over the day for each worker category.

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 6.1a).

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)



(a) (b)

Figure 6.1. Examples of workstation group restrictions

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 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 by Figure 6.1b. 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 due to the requirement that each worker must spend a plurality of his time at his home base. No mathematical formulation exists for this case, or the more general case in which the home base and the tasks assignments must be determined concurrently.

6.1.2 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 large-scale integer programs (IPs) to determine overall workforce size and shift requirements, and the post-processing of the results to construct tours (see Figure 6.2). 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 5.2. 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 B.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).

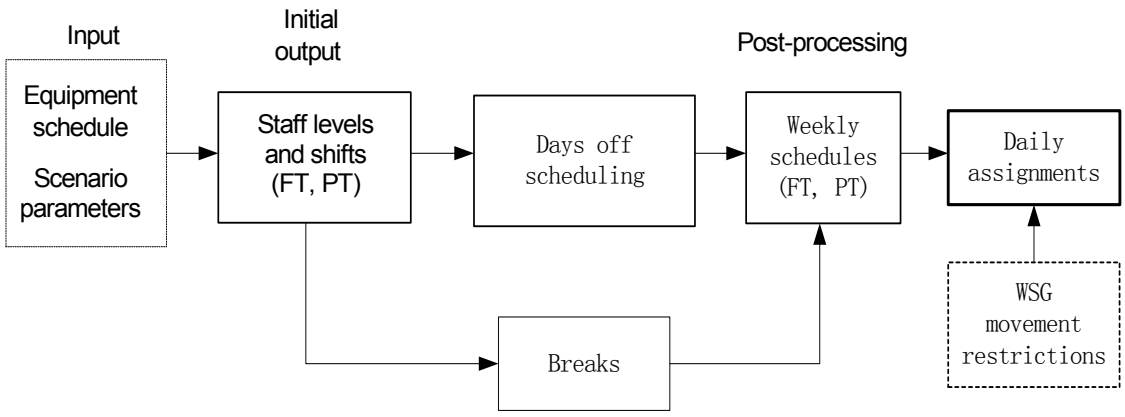


Figure 6.2. Schematic of the computation flow in SOS

When the shift scheduling problem is solved, SOS does not distinguish WSGs within the same skill category, so in the post-processing 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 local movement restrictions, implying that the solution provided by the tour scheduling model is infeasible. This issue is addressed now.

6.1.3 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). A constraint-based formulation has been used for the tour scheduling problem which is manageable up to the point of including the constraints associated with the task assignments and the WSG movement restrictions. The former are handled with a heuristic that does not take into account the locations of individual machines or the transit time between them. Hence, solutions may violate the WSG movement restrictions. Two approaches are proposed to ensure feasibility. To simplify the presentation, only a single worker category will be addressed, such as P5-MPC, with the understanding that the ideas are completely general.

6.2. 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 due to 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 only lead to a marginal improvement in the overall results. Of course, if the movement restrictions matrix R

indicates that all WSGs are completely connected, then it is optimal to solve a single aggregate problem, post-process the results to obtain the task assignments, and then fix the home base of each worker by determining where he spends the plurality of his time.

For the intermediate cases, the procedure begins by representing R as a directed network $G = (V, A)$ similar to those depicted in Figure 6.1. 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 $\hat{G} = (C, A)$. A $c_1 \in C$ is selected arbitrarily and each remaining cluster is scanned to see if one can be merged with c_1. A cluster $c_2 \in C \setminus \{c_1\}$ 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 $\hat{G}$. 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., $V \setminus c_1$), 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 $\hat{G}$. It may be necessary to loop through $\hat{G}$ 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 $\hat{G}$, the search for loops begins. Placing the remaining K clusters in the set $C = \{c_1,\ldots,c_K\}$ in arbitrary order, the step starts with, say cluster c_1, and select a $c_2 \subset C \setminus c_1$ 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 $L = \{c_1,\ldots,c_{k-1},c_k\}$ be the ordered set of clusters in the loop, where

$c_1 = c_k$. The particular cluster c^* in L that is being looked for is the one that has the minimum number of inbound arcs from its immediate predecessor cluster, $p(c^*)$, in the loop and remove all such arcs from $\hat{G}$; i.e., find $c^* = \text{argmin}\{|\{(p(c), c)\}| : c \in L\}$ and remove arcs $\{(p(c^*), c^*)\}$ from $\hat{G}$. This breaks the loop. If c^* in the resultant graph has outbound arcs only, then it is removed from $\hat{G}$ and placed in Q. As mentioned, multiple passes through $\hat{G}$ 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, the process continue to search for and break loops as described until all clusters are placed in Q. Let $\overline{A}$ be the set of arcs that are dropped when merging two clusters and breaking loops, and let $A^* = A \setminus \overline{A}$ 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 $c \leftarrow c \cup \hat{c}$, $C \leftarrow C \setminus \{\hat{c}\}$, $A \leftarrow A \setminus \{(c, \hat{c}), (\hat{c}, c)\}$, and $A^* \leftarrow A^* \setminus \{(c, \hat{c}), (\hat{c}, c)\}$. Repeat until no more merging is possible.

Step 3: (Cluster ordering)

a. Let $\hat{G} = (C, A)$ 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 $Q \leftarrow Q \cup \{c\}$, $C \leftarrow C \setminus \{c\}$, $A \leftarrow A \setminus \{(c, \hat{c}) : \hat{c} \in I(c)\}$. Repeat until there is no $c \in C$ such that $O(c) = \varnothing$. If $C \neq \varnothing$, go to Step 3c, 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, else go to Step iii.

ii. (Loop not found). $P \leftarrow (P, c^*)$. Pick any $c_i \in I(c^*)$, 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 $i^* = \text{argmin}\{|A_i| : A_i \in E\}$. Put $A \leftarrow A \setminus A_{i^*}$, $A^* \leftarrow A^* \setminus A_{i^*}$, and go to Step 2b.

It is straightforward to show the following.

Proposition 6.1. 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 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.

Example. The network in Figure 6.3 will be used to illustrate the algorithm. Each WSG is represented by a node and the authorized movements are represented by the directed arcs. The abbreviations in Table B.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 exists so the process goes 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 6.4. Further clustering is not possible because no three nodes are completely connected.

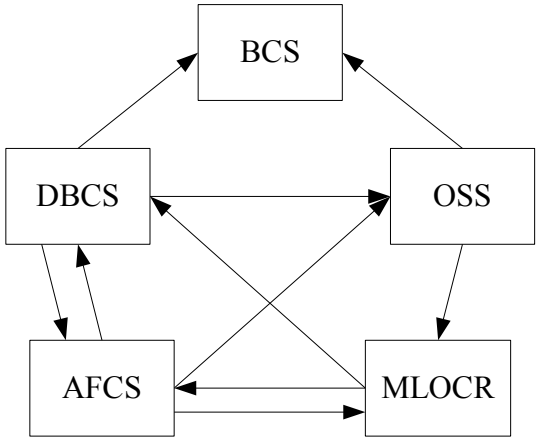


Figure 6.3. Movement restrictions network for example

Going to Step 3, loops must be broken in $\hat{G}$ 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 clusters exist, then one or more loops are present in the network. To identify a loop, the search starts from any cluster c and tries 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 6.1 guarantees that a loop exists and that the algorithm will find it.

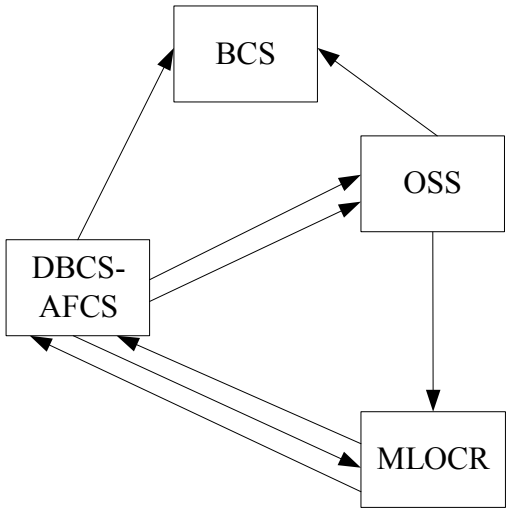


Figure 6.4. Network after clustering

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

Referring to Figure 6.4, it can be seen 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 $\hat{G}$ is displayed in Figure 6.5(a).

At this point, only MLOCR needs to be checked to see if it has any inbound links; i.e., whether O(MLOCR) = ∅. Since 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 ← MLOCR ← DBCS-AFCS, which is broken by removing the link from DBCS-AFCS to MLOCR. The resultant network is shown in Figure 6.5(b) where MLOCR now has no inbound links. Therefore, MLOCR is removed from $\hat{G}$ along with its outbound arcs, and placed in the queue giving Q = {MLOCR}. Figure 6.6(a) depicts the reduced network. The next cluster to be removed is DBCS-AFCS (see Figure 6.6b), followed by OSS, and then BCS. The final ordering in Q is {MLOCR, DBCS-AFCS, OSS, BCS}.

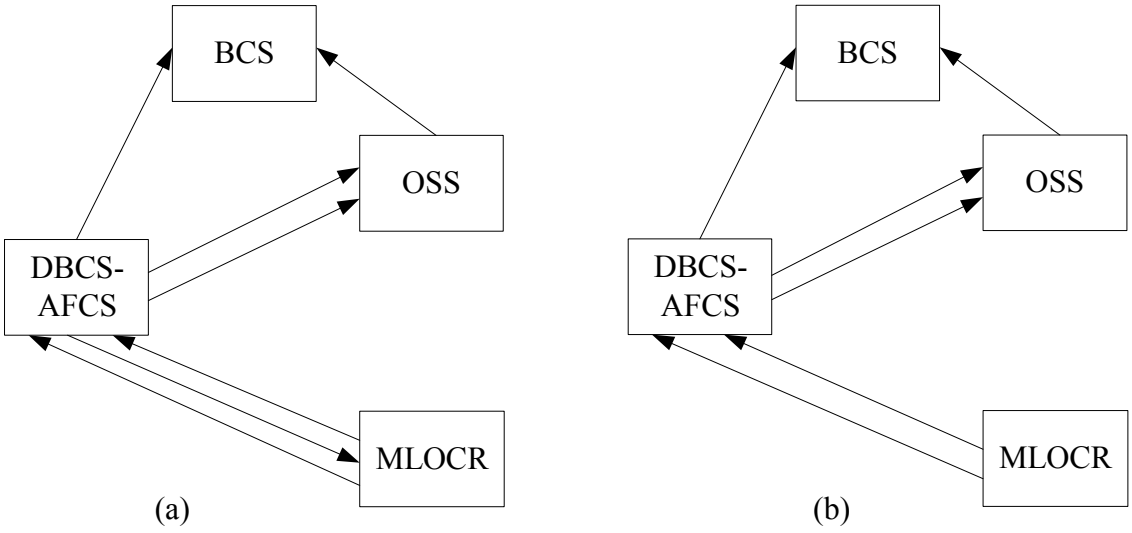


Figure 6.5. Networks after breaking loops

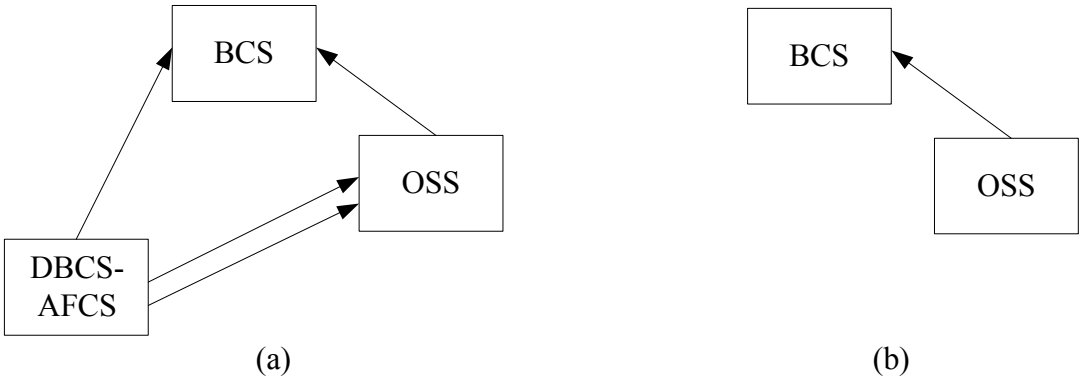


Figure 6.6. Network after removing the two clusters

6.3. Iterative Procedure

A second approach to solving WGAP is to formulate it as an optimization problem. Some compromise is needed, though, because an aggregate solution to the tour scheduling problem (see Figure 6.2) in terms of workforce size and shift assignments, is not likely to be feasible to the WSG constraints. Recall that the aggregate solution is always feasible to the task assignment constraints so that problem can be solved with a post-processor.

In iterative procedure, the WSG constraints are ignored and a solution to the relaxed tour scheduling problem is obtained. a second IP is then solved, which assigns each worker in the solution to a home base as well as to a set of tasks over each shift that is included in his or her tour. The objective is to minimize the number of uncovered periods. If the optimal objective function value is greater than zero, then the solution is not feasible to the WSG constraints and a new problem is solved with this demand 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 shortage 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 obtained.
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, the following notation is used.

Indices and sets

i	index for workers; $i \in I$
j, k	index for workstation groups; $j \in J$
p	index for periods; $p \in P$
$P(i)$	set of periods that worker i is scheduled to be on duty during the week
I(j)	set of WSGs that have inbound links to WSG j
$O(j)$	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) 1 if WSG j is the home base of worker i, 0 otherwise

y_{ijp} (binary) 1 if worker i is assigned to WSG j in period p, 0 otherwise

s_{jp} uncovered demand in WSG j during period p

Model

$$\text{Minimize } z = \sum_{j \in J} \sum_{p \in P} s_{jp} \tag{6-1a}$$

$$\text{subject to} \quad \sum_{j \in J} x_{ij} = 1, \ \forall \ i \in I \tag{6-1b}$$

$$\sum_{j \in J} y_{ijp} \leq 1, \ \forall \ i \in I, \ \forall \ p \in P(i) \tag{6-1c}$$

$$\sum_{i \in I} y_{ijp} + s_{jp} \geq d_{jp}, \ \forall \ j \in J, \ \forall \ p \in P \tag{6-1d}$$

$$y_{ijp} \leq x_{ij} + \sum_{k \in I(j)} x_{ik}, \ \ \forall \ i \in I, \ \forall \ j \in J, \ \forall \ p \in P(i) \tag{6-1e}$$

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

$$s_{jp} \geq 0, \ \ \forall \ j \in J, \ \forall \ p \in P \tag{6-1f}$$

The objective function in (6-1a) sums the unmet demand determined by constraint (6-1d). Constraint (6-1b) assigns each worker i to exactly one home base, while (6-1c) 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 (6-1d). This implies that all demand can be covered by the available workforce I.

Constraint (6-1e) ensures that the movement restrictions specified by the matrix R and embodied in the set $I(j)$ 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 $k \in$

$O(j) \cup \{j\}$ 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:

$$\sum_{p \in P(i)} y_{ijp} \geq \sum_{p \in P(i)} y_{ikp} - |P(i)|\left(1 - x_{ij}\right), \quad \forall\, i \in I, \quad \forall\, j \in J, \ \forall\, k \in O(j) \setminus \{j\} \qquad \text{(6-1g)}$$

If a worker is required to spend at least 50% of his time at the home base, then (6-1g) should be replaced with

$$\sum_{p \in P(i)} y_{ijp} \geq \sum_{k \in O(j)} \sum_{p \in P(i)} y_{ikp} - |P(i)|\left(1 - x_{ij}\right), \quad \forall\, i \in I, \quad \forall\, j \in J \qquad \text{(6-2)}$$

where the first summation on the right-hand side of (6-2) can be extended to $k \in J \setminus \{j\}$ to strengthen the inequality.

If the inclusion of (6-1g) 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 (**x***, **y***, **s***) to (6-1a) – (6-1f) would be found; in the second stage, **x** would be fixed at **x*** and new values for (**y***, **s***) would be found with (6-1g) included for each $i \in I$. This approach reduces the number of additional constraints by a factor of $O(|J|)$. a variation of this idea is used 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. 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

$$x_{ij} \geq x_{i+1,j} - \sum_{k \in J \setminus \{j\}} x_{i-1,k}, \quad i = 2,\ldots,|I|-1, \quad \forall\, j \in J \qquad \text{(6-3)}$$

can be added to partially eliminate symmetric assignments of workers to home bases. The second term on the right-hand side of (6-3) is 1 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 $i + 1$, $i + 2$,…, $i + k - 1$, where $k \geq 3$. To ensure that consecutive workers are assigned to the same home base, it would be necessary to expand constraint (6-3) 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 (6-1a) 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 1 if worker $i + 1$ is assigned to WSG j and worker i is assigned to WSG $k \neq j$, and 0 otherwise. The following constraint is added to the model

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

and modify the objective function (6-1a) to be

$$\text{Minimize } z = \sum_{j \in J} \sum_{p \in P} s_{jp} + \sum_{i=1}^{|I|-1} \sum_{j \in J} \gamma_{ij} \qquad \text{(6-1a')}$$

Fortunately, γ_{ij} can be treated as a continuous variable because it will always be 0 or 1 in an optimal solution. Also, it is not necessary to multiple the second term in (6-1a') by a small positive constraint 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 $\sum_{i,j} \gamma_{ij} \leq |J| - 1$.

WSG demand. If it is assumed that the number of workers assigned to a WSG is proportional to its demand, then a constraint can be added 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 $D_{[j]} \geq D_{[j+1]}$, 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 $[j+1]$.

$$\sum_{i \in I} x_{i,[j]} \geq \sum_{i \in I} x_{i,[j+1]}, \quad \forall j \in J \setminus \{[J]\} \tag{6-4}$$

Although (6-4) is not a valid inequality for WGAP, if including it in model (6-1) reduces the computational burden significantly, then the resultant degradation in the solution may be justified. Moreover, solving the problem with (6-4) will provide an upper bound that may be close to the optimal objective function value.

6.3.1 Complexity Issues

The WGAP addressed below is the one represented by model (6-1) in which the objective is to minimize the number of uncovered periods. To see its complexity, the recognition version of the problem is defined 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 limit 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 6.1 The recognition version of WGAP is NP-complete in the strong sense.

Proof. The proof starts 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 D*m*CIF 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 $f_i : A \rightarrow Z_0^+$ such that

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

(b) for each $v \in V \setminus \{s, e\}$, 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 D*m*CIF the corresponding instance of WGAP is constructed as depicted in Figure 6.7. 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, it is assumed 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 $i \in I$ individual arcs needed are those 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.

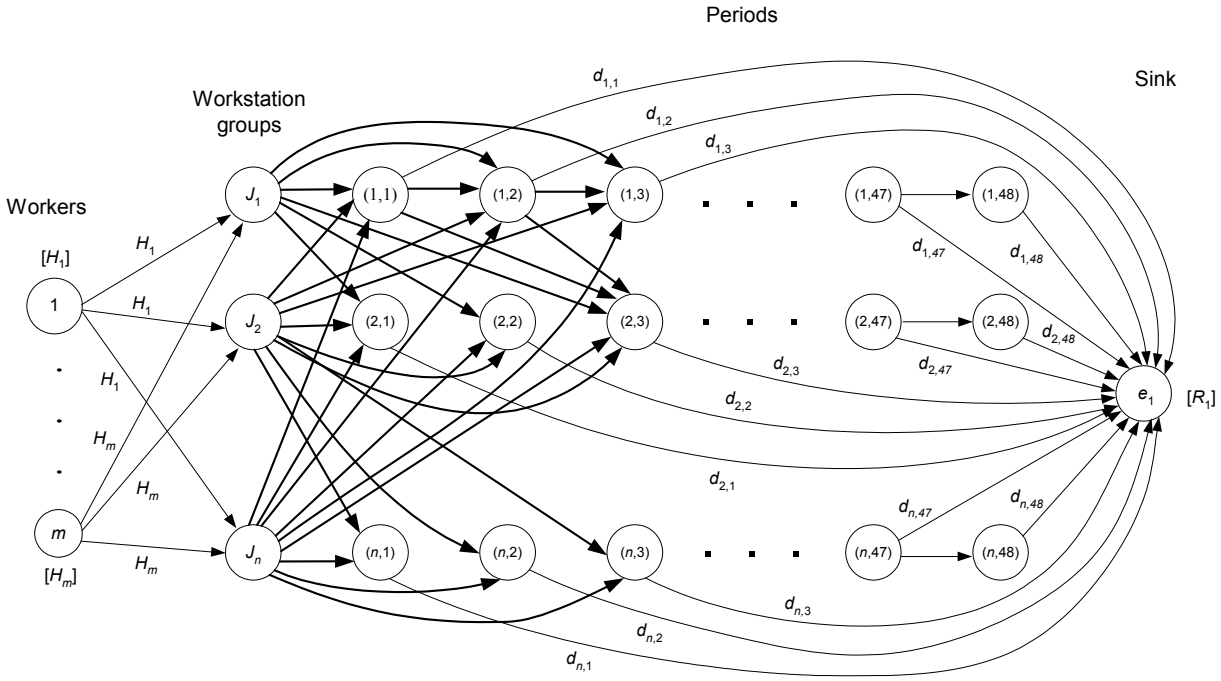


Figure 6.7. Multicommodity flow network used in proof of Theorem 6.1

At the far right of Figure 6.7 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 it is not required 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 **x** is being treated as a continuous variable. To finish the description of the general WGAP, set the parameter R_1 = quantity of demand covered.

These developments lead to the following observations.

1. If $R_1 = \sum_{j \in J} \sum_{p \in P} d_{jp}$, a question should be asked: 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, $\sum_{j\in J}\sum_{p\in P} d_{jp} - \sum_{j\in J}\sum_{p\in P} s_{jp}$, the equivalent question: 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.
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 6.7 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 (6-1a) – (6-1f) the result follows. ■

Next question is what happens when the WSG assignments are fixed prior to scheduling the workers.

Corollary 6.1 If x_{ij} is fixed for all $i \in I, j \in J$, then WGAP (6-1) 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

$$\text{Minimize } z = \sum_{j\in J}\sum_{p\in P} s_{jp} \tag{6-5a}$$

$$\text{subject to } \sum_{j\in J} y_{ijp} \le 1,\ \forall\ i \in I,\ \forall\ p \in P(i) \tag{6-5b}$$

$$\sum_{i\in I} y_{ijp} + s_{jp} \ge d_{jp},\ \forall\ j \in J,\ \forall\ p \in P \tag{6-5c}$$

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

$$s_{jp} \geq 0, \quad \forall\, j \in J,\ \forall\, p \in P \tag{6-5d}$$

where the **y** variables only exist if they satisfy constraint (6-1e).

Model (6-5) decomposes by p. To see how a transportation problem results for each $p \in P$, a bipartite network is identified, which has one source node with unit supply for each i if $p \in P(i)$, one dummy source node denoted by 0 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 6.2 When x_{ij} is fixed for all $i \in I, j \in J$, WGAP given by (6-1g), (6-5) remains NP-complete.

Proof. When constraint (6-1g) is added to (6-5), 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 $1 \leq i \leq m$, $\sum_{a \in B_i} f(a) \leq H_i / 2$, for bundle $m + 1$, $\sum_{a \in B_{m+1}} f(a) \leq \sum_{i \in I} H_i / 2$, and for bundle $m + 2$, $\sum_{a \in B_{m+2}} f(a) \leq \infty$. 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, this special case of problem (6-1g), (6-5) 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 $f: A \to Z_0^+$ that satisfies the bundle requirements for the integral flow problem defined above. ■

Because there is nothing in the proof of Corollary 6.2 that depends on worker movement restrictions $I(j)$, the following is true.

Corollary 6.3 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 (6-1g), (6-5) remains NP-complete.

6.3.2 Solving the Integer Programming Formulation of WGAP

Initial testing indicated that problem instances with more than about 70 workers and 3 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 model (6-1), there are three sets of decision variables: $\mathbf{x} = (x_{ij})$, $\mathbf{y} = (y_{ijk})$, and $\mathbf{s} = (s_{jk})$. (Note that constraints (6-3) and (6-4) are not included in the implemetation.) When the **x** variables are fixed, implying that the home base of each worker has been decided, the model can be solved very easily. Choosing the home base is a critical decision because it determines the permissible task assignments, which may dramatically affect the objective value.

A two-stage approach is taken to the problem. In the first stage, the LP relaxation of WGAP is solved to obtain the solution $\left(\bar{x}_{ij},\ \forall\ i \in I,\ j \in J\right)$ and $j^* = \text{argmax}\left\{\bar{x}_{ij} : j \in J\right\}$ for all $i \in I$. For a given fraction $\rho \in [0,1]$, the home bases of $\rho \times 100\%$ of the workers are then fixed, beginning with those whose fractional solutions $\bar{x}_{ij^*}$

are the largest. Fixing all of the **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 $B = \{b_i, \text{ for all } i \in I\}$ 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 $\left(\overline{x}_{ij},\ \forall\ i \in I,\ j \in J\right)$.

Step 2: For all $i \in I$, $z_i = \max\left\{\overline{x}_{ij} : j \in J\right\}$, 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 $b_i = \text{argmax}\left\{\overline{x}_{ij} : j \in J\right\}$ and construct B.

If $b_i \neq -1$ in the output set B, then the home base of worker i is set such that $x^*_{ij} = b_i$. In the second stage of the iterative procedure, model (6-1) is solved for **y** and **s** and the remaining **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 $\{d_{jp} : \forall\ j \in J, p \in P\}$, movement restrictions matrix R, value of logical parameter SEQ, threshold parameter ρ_0 for calling variable fixing algorithm, and all other data elements that define the long-term planning model SOS

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

Step 1: (Relaxed solution) Solve the shift scheduling component of SOS to generate the workforce W_{SOS}, and put $W^* \leftarrow W_{SOS}$.

Step 2: (Find home base and task assignments) Set up model (6-1) using W_{SOS}, $\{d_{jp} : \forall j, p\}$, 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: $\left\{\left(x_{ij}^*\right): \forall i, j\right\}$, $\left\{\left(y_{ijp}^*\right): \forall i, j, p\right\}$, $\left\{\left(s_{jp}^*\right): \forall j, p\right\}$.

If $U \equiv \left\{\left(s_{jp}^*\right): \forall j, p\right\} = \varnothing$, stop; otherwise go to Step 3.

Step 3: If (SEQ = <true>), then put $d_{jp}^{new} \leftarrow s_{jp}$ for all $p \in P, j \in J$ and call the sequential procedure to get W_{SEQ}. Put $W^* \leftarrow W^* \cup W_{SEQ}$, update $\left\{\left(x_{ij}^*\right): \forall i, j\right\}$ and $\left\{\left(y_{ijp}^*\right): \forall i, j, p\right\}$, 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 long-term tour scheduling problem is solved without WSG movement restrictions to get a tentative permanent workforce, W_{SOS}. The home base 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. As a guideline, ρ_0 is usually set to be around 200. If there is any uncovered demand, then 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, several variations are tried, 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, it is 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, it is also 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 6.3. When the iterative algorithm is applied, the size of the workforce found by SOS in Step 1 is $W_{SOS} = 45$. The IP problem solved by SOS has 1529 variables and 1120 constraints. The optimality gap is 1.3% after 302 seconds. At Step 2, model (1) is set up with W_{SOS}. The model has 27,748 columns and 12,611 rows, and LP relaxation is solved in 310 seconds with 0 objective value. The fix rate ρ is set to 0.6, and model (1) is resolved with 60% of the x variables fixed to generate the set of uncovered demand U. The problem is solved in about 10 minutes, and the result shows that the number of uncovered demand is 2. At Step 3, the sequential procedure is called to generate additional workforce to cover demand U, and the additional workforce $W_{SEQ} = 1$. So, the total workforce needed to operation the facility is $W^* = W_{SOS} + W_{SEQ} = 46$.

6.4. 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 WSGs but a different restriction matrices R. The networks for data set 1 are shown in Figure 6.8 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 6.9 and 6.10, respectively.

All computations were performed on a PC with dual Xeon 1.8G CPU, 1gb memory, running SuSE Linux 9.0. The implementation was done in Java SDK 1.3,

which calls CPLEX 9.0 to solve the integer programs. The barrier option was used at the root node of all search trees to solve the first LP relaxation.

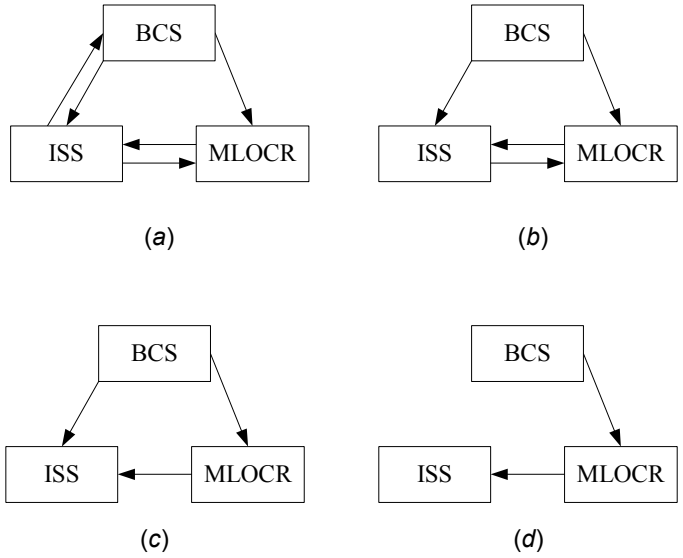


Figure 6.8. Movement restrictions networks for data set 1

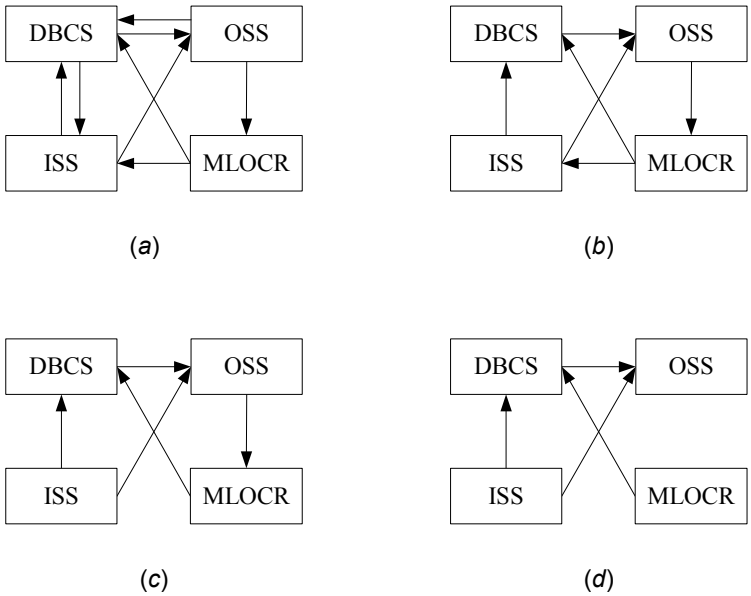


Figure 6.9. Movement restrictions networks for data set 2

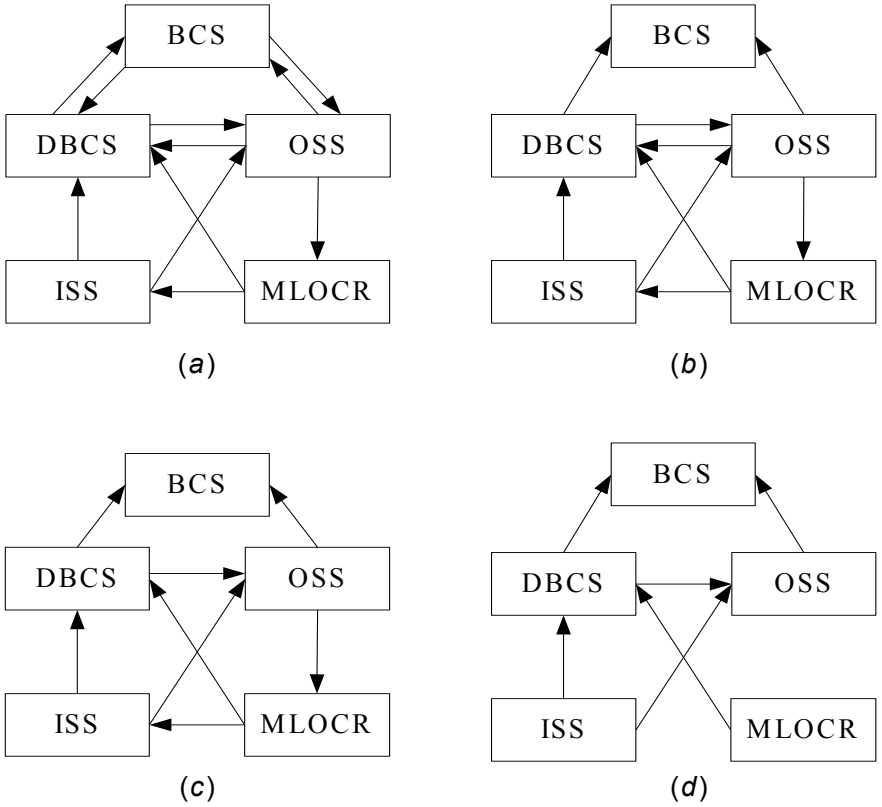


Figure 6.10. Movement resections networks for data set 3

6.4.1 General Results

Table 6.1 gives the tour scheduling solutions found by SOS when the WSG movement restrictions are omitted from WGAP, i.e., the WSG networks are completely connected. In this case, the aggregate WSG demand can be used to solve the tour scheduling problem, and it will always be possible to post-process the results to obtain a feasible solution. When restrictions are present, these solutions provide a lower bound on WGAP. Although the lower bound 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, no PTRs were allowed and the FTR/PTF ratio was set to be ≥ 4.

Table 6.1. Lower Bound 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

The staff scheduling results obtained from the sequential and iterative procedures are given in Table 6.2. In the case of the iterative algorithm, SEQ is set to be <true> at Step 3 because this always proved to be the better choice. The column "No. 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 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 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 total staffing the costs differ by a fraction of a percent.

The quality of the solutions in Table 6.2 can be judged, in part, by their distance from the lower bound provided by the SOS solutions in Table 6.1. From the data in column 6, the average gap between the solutions found by the sequential procedure and the lower bound is computed to be about 12%. In general, as the number of links in the networks shown in Figure 6.8 – 6.10 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*.

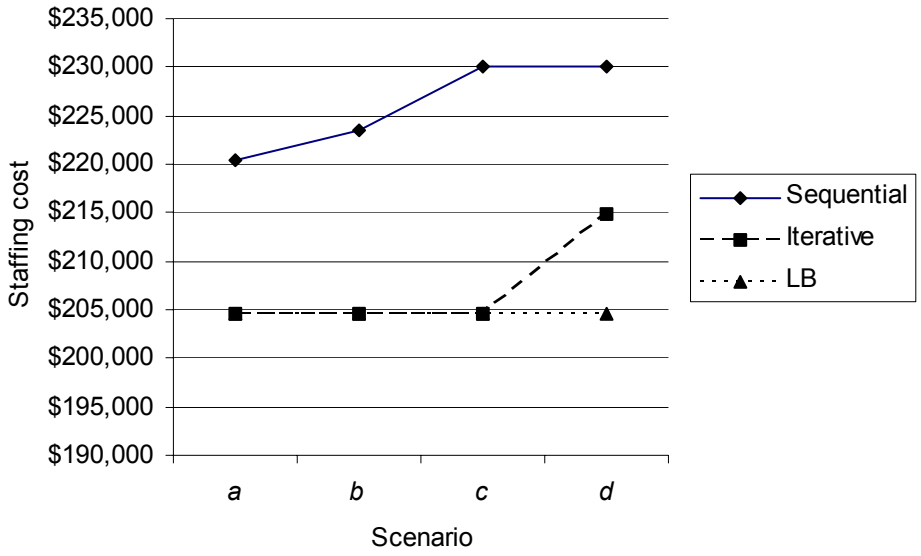
Table 6.2. Results for Sequential and Iterative Procedures

Scena-rios	Sequential procedure						Iterative procedure					
	No. of workers	No. of FTRs	No. of PTFs	Staffing cost	Gap with LB	Idle time	No. of workers	No. of FTRs	No. of PTFs	Staffing cost	Gap with LB	Idle time
Date set 1												
a	221	178	43	$220,427	7.77%	17.4%	205	165	40	$204,539	0%	11.0%
b	229	184	45	$223,521	9.28%	18.3%	205	165	40	$204,539	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	214866	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%

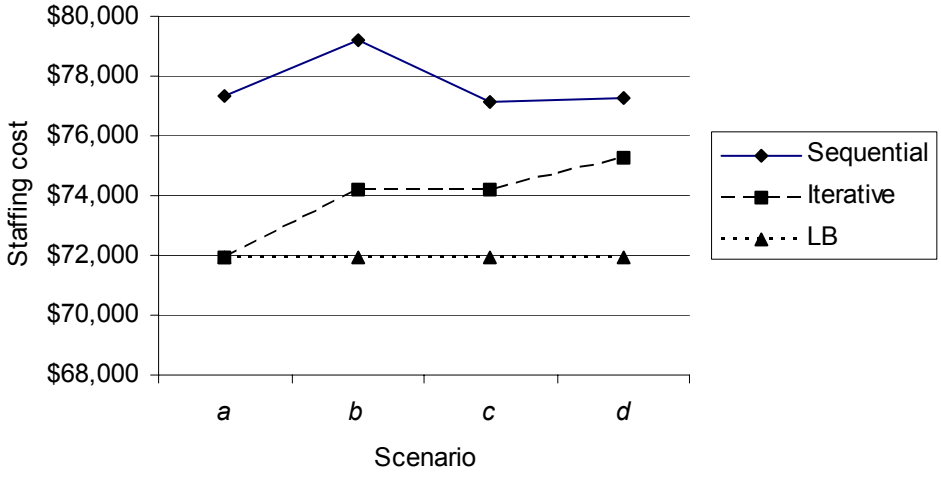
The solutions found by the iterative procedure were significantly better than those found by the sequential procedure. Using a 1% optimality gap or 30 minutes as the stopping criteria, and variable fixing fractions of 0.6, 0.6, and 0.2, respectively, for the three data sets, the next to last column in Table 6.2 indicates that seven out of the 12 instances reached their lower bound (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 represents the true optimality gap for the sequential procedure. Of course, when the "Gap with LB" reported in Table 6.2 is greater than 0, there is no way of confirming whether the current solution actually minimizes the cost of the workforce, the original objective. Figure 6.11 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 6.1) suggests that there are multiple optima to the

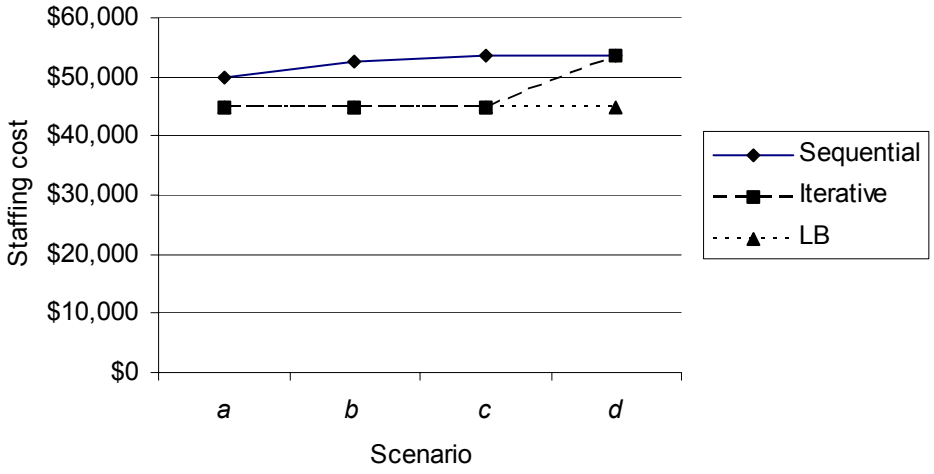
unrestricted task assignment problem. The one found by the algorithm built into SOS rarely, if ever, satisfied the WSG restrictions, hence the need to solve model (6-1). In the remaining five scenarios, the gap between the iterative solution and the lower bound ranged from about 3% to 20%. In only one scenario did the sequential procedure to better.



(a)



(b)



(c)

Figure 6.11. Comparison of sequential and iterative procedures

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 (6-1). 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 (6-1) are more difficult 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 a smaller workforce.

6.4.2 Variable Fixing Results

The computational results obtained by attempting to solve model (6-1) with both CPLEX and the variable fixing algorithm are displayed in Table 6.3. In all cases, a 30-minute

time limit was placed on CPLEX for all values of ρ and CPLEX's built in heuristic was called 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, 3 and 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 0 or close to 0, 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. Tables 6.4 gives the solutions for different values of ρ for data set 2. The results for the other two data sets are not shown because most of the instances were solvable with $\rho = 0$.

Table 6.3. Performance of Variable Fixing Algorithm for Model (6-1)

Scen-ario	No. of columns	No. of rows	No. of non-zeros	LP obj. value	LP time (sec)	CPLEX obj. value	CPLEX time (sec)	Opt. 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	–	1800.0	–	0.6	34	1881.6
Data set 2											
a	15,603	14,497	109,186	0	10.3	0	1772.5	0%	0.6	0	1265.1
b	15,651	19,515	113,230	0	18.5	–	1800.0	–	0.6	5	1856.4
c	15,651	19,488	108,963	0	15.8	–	1800.0	–	0.6	7	1835.2
d	15,651	19,416	98,685	0	19.8	–	1800.0	–	0.6	15	2026.9
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	1024.2	0%	0.2	0	1029.1
c	12,611	15,070	94,716	0	17.1	0	1324.4	0%	0.2	0	1659.3
d	12,611	15,003	84,649	13.8	20.9	169	1808.4	89.45%	0.2	60	1834.8

Table 6.4. 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
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

Figure 6.12 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 necessary the same for all scenarios and problem sizes. According to the computational experience, though, solution quality was relatively stable for ρ between 0.4 and 0.6.

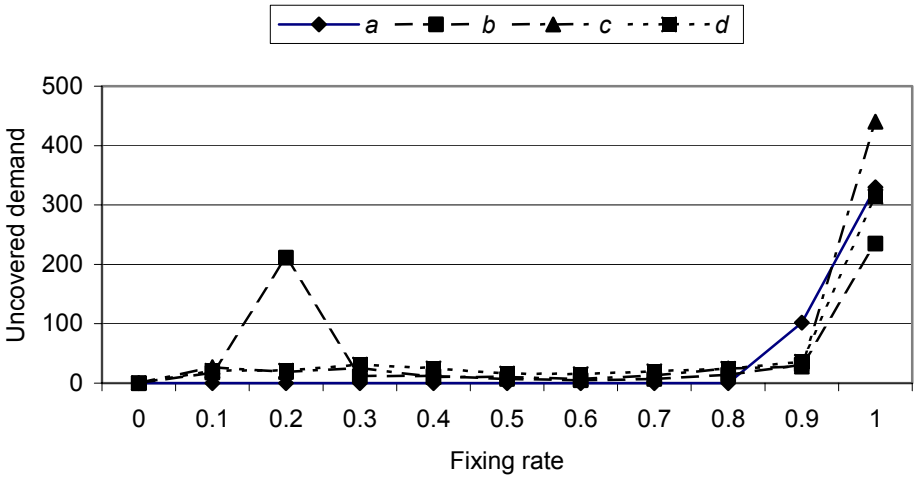


Figure 6.12. Parametric analysis of variable fixing fraction, ρ

Chapter 7
Summary, Future Work, and Conclusions

This research addressed the issues surrounding staff scheduling in organizations that face changing demand patterns that peak for short periods during the day. The results validate the use of optimization as a decision aid for managers who must continually adjust their staffing levels in light of changing demand and absenteeism. In the service industry, short-term adjustments play a critical role in both minimizing operational costs and meeting business objectives. For the USPS, these are ever-present concerns due to declining mail volumes, a fixed permanent workforce, and limited scheduling flexibility. Strict labor laws and union agreements prevent management from altering the size of the permanent workforce or even modifying the bid jobs. As a consequence, they must try to exploit the other options available to them such as the use of overtime, the extension of part-time hours, and the use of casuals as a last resort.

A new model was developed and tested for the weekly shift scheduling problem in Chapter 3. The analysis indicated that instances two to three times larger than any of those described in the literature can be solved with the proposed methodology, well within 5 – 10 minutes in most cases. This represents a sizable leap in capability, and is expected to result in tens of millions of dollars in annual saving for the USPS when implemented nationwide over the next three years. With respect to the performance of the feasibility heuristic, it is believed that one of the reasons why convergence was so rapid is because the search is indirectly limited to a small neighborhood of the target solution. That is, a local rather than a global solution is being sought. For large-scale integer programs, however, minimizing some measure of the deviation from a target may be more difficult than solving the original problem directly because of the additional constraints and variables needed to linearize the deviation terms. Nevertheless, one of the benefits offered by this type of approach is that it allows the user to strike a balance between the computational effort required to solve large-scale MILPs and the quality of solutions obtained. For practical purposes, when the actual demand is close to the

planned demand, solving the problem directly with a commercial code is recommended. For the more difficult cases, the target heuristic embodied in method 1 looks like the better choice.

In Chapter 4, both exact and heuristic methods were developed and tested for solving the task assignment problem. Included were a delayed idle period assignment algorithm, a daily decomposition algorithm, and tabu search. The computational results indicated that the exact method is only practical for very small instances, while daily decomposition can reliably provide near-optimal solutions for problems of moderate size. The advantages of tabu search only came into play for the large instances that were not solvable with the other methods.

As part of the analysis, the effectiveness of tabu search was also investigated when started from different feasible solutions. The results underscored the importance of good initial solutions, at least for the TAP. Although tabu search has the ability to transcend local optimality, it was found that an excessive amount of time was needed to make up the difference in objective function values when started from a poor solution compared with starting from a good solution.

In Chapter 5, a new model was presented for the weekly staff scheduling problem with workstation group restrictions. Finding optimal solutions with a commercial code proved difficult for all but relatively small instances. The size of the integer program that had to be solved was considerably larger than existing formulations due to the need to account for the position of each worker during each period of the week.

In light of this difficulty, two heuristics were developed that were shown to provide good feasible solutions in a reasonable amount of time. The first splits the movement restriction network into manageable pieces and then solves the scheduling problem for each piece in turn. The second takes advantage of problem structure and uses Dantzig-Wolfe decomposition to generate good weekly schedules for each worker. Feasible solutions were then constructed by solving a restricted version of the D-W master problem to optimality. Although the network splitting algorithm found feasible

solutions in less time, the solution quality of the column generation heuristic was consistently better.

The focus of this research was mainly on the weekly scheduling problems. However, the workstation group restrictions discussed in Chapter 5 is also commonly encountered in long-term staff planning. In Chapter 6, two procedures were developed as part of the process of constructing a permanent workforce to directly account for the need to limit the movement of workers when assigning them tasks during the day. 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 workstation group and each arc represents the permissible flow. In the derivation of the network, some compromise was needed to obtain feasible solutions. In the second approach, a relaxed solution to the shift scheduling problem is computed which 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 with one of two ways: either they are added back to the original demand and the process is repeated until all requirements are satisfied, or the first approach is applied with the incremental demand as input. Extensive computational testing showed that the iterative procedure with variable fixing provided the better results. Problem instances with hundreds of employees and five workstation groups were solved to (near) optimality within 30 minutes in most cases.

Future Work

The biggest challenges for this research are due to the full set of labor union, legal, and organization constraints specified by the USPS, and the intimidating sizes of the optimization problems encountered. More advanced computational approaches might be worth trying.

Other approaches to solving model (4-2) exactly that might be worth investigating include branch and price and Lagrangian relaxation. Both of these methodologies have the potential to exploit the fact that when the demand constraint (4-2e) is removed from the model, the TAP decomposes into n shortest route problems, one for each worker.

Although neither branch and price nor Lagrangian relaxation will provide better lower bounds than the LP relaxation, they are likely to reduce the overall computational effort because the accompanying subproblems would be much smaller than the full LP relaxation. With some additional computations, they may also provide good feasible solutions for initializing tabu search or some other metaheuristic.

The priority in which scheduling options should be used to meet demand was discussed in Section 5.1.3. Neither heuristic presented in Chapter 5 could guarantee strict enforcement in all cases. With this in mind, areas of future research might include the development of more robust methods for generating individual schedules, the use of intelligent heuristics for solving a larger-scale master problem, and a more adaptive approach to splitting and then partially reconstructing the network to allow more links to be considered in a solution.

Although the results shown in Chapter 6 are promising, from a methodological point of view, it is still desired to be able to solve larger instances with the same degree of accuracy. One possible approach would be to develop a column generation scheme based on decomposing the problem by either workstation group or employee. Another area for future research concerns post-processing the solution with the goal of 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 shifts must be converted to idle time.

Appendix A

Graphical User Interface

Figure A-1 presents a snapshot of the graphical user interface (GUI) designed as the front for the WSO. It is written in java to be web-enabled and consists of six tabs: 'Shift Types,' 'Workstations,' 'Schedule,' 'Wages,' 'Movement,' and 'Model Runs.' A brief explanation of each follows:

Shift Types: specify input requirements related to the shifts that are to be used in developing the weekly schedule; e.g., shift durations, union and legal restrictions, overtime considerations, and start time bands.

Workstations: user is given the option to either import the equipment schedule generated by ESO or enter a new one; the schedule is transformed into the demand requirements for the weekly model – Eq. (1b).

Schedule: user imports the long-term staffing schedule generated by SOS; this schedule serves as the baseline for the weekly model.

Wages: define or select pay rate data sets (cost coefficients) for different worker categories such as P5-MPC, P5-DC and different classes, i.e., full-time, part-time, overtime, and casual.

Movement: specify workgroup restrictions to be imposed on some or all worker categories; i.e., limit the movement between specified workgroups or workstations.

Model Run: submit and solve new problem; report results.

The use of the application is straightforward. The first five tabs are designed for data and the sixth to run the model. The weekly schedules produced for each FTR, PTF and selected casuals are displayed under the 'Model Runs' tab when the computations terminate. It is also possible to export the weekly schedules to an Excel file.

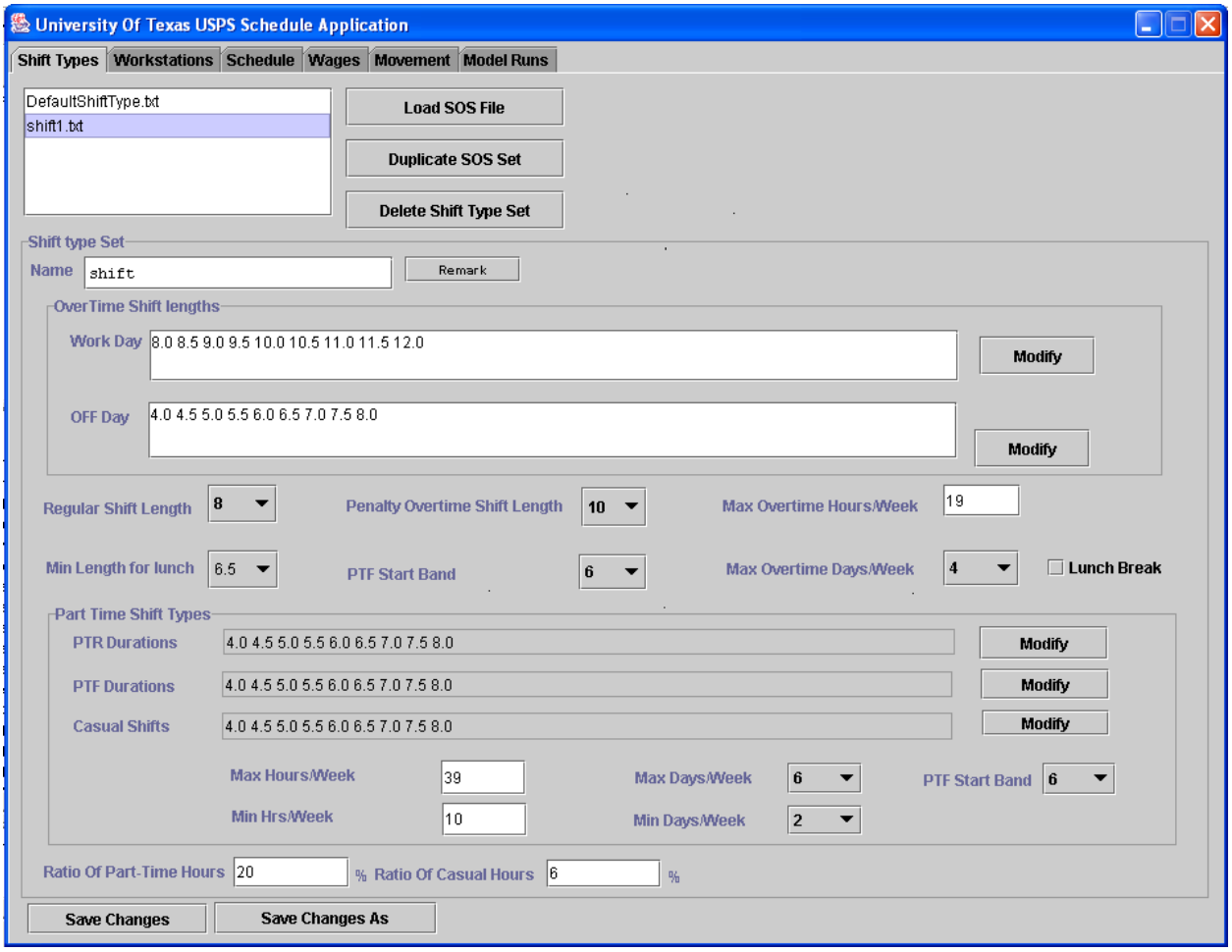


Figure A.1. Graphical user interface for WSO

Appendix B

Definition of Worker Categories and

Equipment in a P&DC

Table B.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	Barcode sorter	BCS
General expeditor	P6-GE	Delivery barcode sorter	DBCS
Parcel post distribution machine operator	P6-PPDMO	Flat sorting machine	FSM
Sack sorting machine operator	P6-SSMO	Input subsystem	ISS
Mail processing clerk	P5-MPC	Manual operations	MANUAL
Flat sorting machine operator	P5-FSMO	Multi-line optical character reader	MLOCR
Parcel post distribution machine operator	P5-PPDMO	Output subsystem	OSS
Data Conversion Operator	P4-DCO	Remote encoding center	REC
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		

Bibliography

Alfares, H.K. 1997. An Efficient Two-Phase Algorithm for Cyclic Days-Off Scheduling. *Computers & Operations Research* 25(11) 913-923.

Aronson, J.E. 1986. The Multi-Period Assignment Problem: A Multi Commodity Network Flow Model and Specialized Branch and Bound Algorithm. *European Journal Of Operational Research* 23 367-381.

Aykin, T. 1996. Optimal Shift Scheduling with Multiple Break Windows. *Management Science* 42(4) 591-602.

Bard, J.F. 2004a. Staff Scheduling in High Volume Service Facilities with Downgrading. Working paper, Graduate Program in Operations Research & Industrial Engineering, The University of Texas, Austin.

Bard, J.F. 2004b. Selecting the Appropriate Input Data Set When Configuring a Permanent Workforce. Working paper, Graduate Program in Operations Research & Industrial Engineering, The University of Texas, Austin.

Bard, J.F., Binici, C. and deSilva, A.H. 2003. Staff Scheduling at the United States Postal Service. *Computers & Operations Research* 30(5) 745-771.

Bard, J.F., deSilva, A.H., Feo, T.A. and Wert, S.D. 1993. Design of Semi-Automated Mail Processing Facilities. *IIE Transactions on Design & Manufacturing* 25(4) 88-101.

Bard, J.F. and Feo, T.A. 1991. An Algorithm for the Manufacturing Equipment Selection Problem. *IIE Transactions* 23(1) 83-92.

Bard, J.F. and Purnomo, H.W. 2005. A Column Generation-Based Approach to Solve the Preference Scheduling Problem for Nurses with Downgrading. Working paper, Graduate Program in Operations Research & Industrial Engineering, The University of Texas, Austin.

Bard, J.F., Yu, G. and Argüello, M.F. 2001. Optimizing Aircraft Routings in Response to Groundings and Delays. *IIE Transactions on Operations Engineering* 33(10) 931-947.

Beaumont, N. 1997. Scheduling Staff Using Mixed Integer Programming. *European Journal of Operational Research* 98(3) 473-484.

Bechtold, S.E. and Jacobs, L.W. 1990. Implicit Modeling of Flexible Break Assignments in Optimal Shift Scheduling. *Management Science* 36(11) 1339-1351.

Berman, O. and Larson, R.C. 1993. Optimal Workforce Configuration Incorporating Absenteeism and Daily Workload Variability. *Socio-Economic Planning Sciences* 27(2) 91-96.

Berman, O., Larson, R.C. and Pinker, E. 1997. Scheduling Workforce and Workflow in a High Volume Factory. *Management Science* 43(2) 158-172.

Brusco, M.J. 1998. Solving Personnel Tour Scheduling Problems Using the Dual All-Integer Cutting Plane. *IIE Transactions on Operations Engineering* 30(9) 835-844.

Brusco, M.J. and Jacobs, L.W.1998. Personal Tour Scheduling when starting time restrictions are present. *Management Science* 44 (4) 534-547.

Burke, E.K., Causmaecker, P.D., Vanden Berghe, G. and Van Landeghem, H. 2004. The State of the Art of Nurse Rostering. *Journal of Scheduling* 7(6) 441- 499.

Burns, R.N. and Carter, M.W. 1985. Work Force Size and Single Shift Schedules with Variable Demands. *Management Science* 31(5) 599-607.

Cai, X., McKinney, D.C. and Lasdon, L.S. 2001. Piece-By-Piece Approach to Solving Large Nonlinear Water Resources Management Models. *Journal of Water Resources Planning and Management* 127:6 363-368.

Caprara, A., Monaci, M. and Toth, P. 2003. Models and Algorithms for a Staff Scheduling Problem. *Mathematical Programming Series B* 98 445-476

Campbell, G.M. and Diaby, M. 2002. Development and Evaluation of an Assignment Heuristic for Allocating Cross-Trained Workers. *European Journal of Operational Research* 138 (1) 9-20.

Chvatal, V. 1979. A Greedy Heuristic for the Set-covering Problem. *Mathematics of Operations Research* 4(3) 233-235.

Clausen, J., Hansen, J., Larsen, J. and Larsen, A. 2001. Disruption Management. *OR/MS Today* 28(5) 40-43.

Dawid, H., Konig, J. and Strauss, C. 2001. An Enhanced Rostering Model for Airline Crews. *Computers & Operations Research* 28 671-688.

Easton, F. F. and Rossin, D.F. 1997. Overtime Schedules for Full Time Service Workers. *Omega* 25(3) 285-299.

Emmons, H. 1985. Work-force Scheduling with Cyclic Requirements and Constraints on Days Off, Weekends Off, and Work Stretch. *IIE Transactions* 17(1) 8-15.

Ernst, A.T., Jiang, H., Krishnamoorthy, M. and Sier, D. 2004. Staff Scheduling and Rostering: A Review of Applications, Methods and Models. *European Journal of Operational Research* 153(1) 3-17.

Even, S., Ital, A. and Shamir, A. 1976. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal on Computing* 5 691-703.

Feo, T. A. and Bard, J. F. 1989. Flight Scheduling and Maintenance Base Planning. *Management Science* 35(12) 1415-1432.

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

Gilbert, K.C. and Hofstra, R.B.1988. Multidimensional Assignment Problems. *Decision Science* 19 306-321.

Glover, F. and Laguna, M. 1997. Tabu Search, Kluwer Academic Publishers: Boston. MA.

Hall, R.W. and Lotspeich, D. 1996. Optimized Lane Assignment on an Automated Highway. *Transportation Research*, *Part C: Emerging Technology* 4(4) 211-229.

Jarrah, A.I.Z., Bard, J.F. and deSilva, A.H.1994. Solving Large-Scale Tour Scheduling Problems. *Management Science* 40(9) 1124-1145.

Lewis, L.H., Srinivasan, A., and Subramanian, E.1998. Staffing and Allocation of Workers in an Administrative Office. *Management Science* 44(4) 548-570.

Malhotra, M.K. and Ritzman, L.P. 1994. Scheduling Flexibility in the Service Sector: A Postal Case Study. *Production and Operations Management* 3 100-117.

Malhotra, M.K., Ritzman, L.P., Benton, W.C. and Leong, G.K.1992. A Model for Scheduling Postal Distribution Employees. *European Journal of Operational Research* 58 374-385.

Mason, A.J., Ryan, D.M. and Panton, D.M. 1998. Integrated Simulation, Heuristic and Optimization Approaches to Staff Scheduling. *Operations Research* 46(2) 161-175.

McManus, I.M. 1977. Optimum use of Overtime in Post Offices. *Computers & Operations Research* 4(4) 271-278.

Miller, J.L. and Franz, L.S. 1993. Scheduling Medical Residents to Rotations: Solving the Large-Scale Multi-Period Staff Assignment Problem. *Operations Research* 41(2) 269-279.

Misra, S., Pinker, E.J. and Shumsky, R.A. 2004. Salesforce Design with Experience-based Learning. *IIE Transactions on Logistics & Scheduling* 36(10) 941-952.

Mukherjee, A.K. and Gilbert, K.C. 1997. Lagrangian Heuristics for Instructor Scheduling in Executive Development Programmes. *Journal of Operations Research Society* 48(4) 373-382.

Nanda, R. and Browne, J.1992. Introduction to Employee Scheduling, Van Nostrand Reinhold: New York. NY.

Pierskalla, W.P. 1968. The Multidimensional Assignment Problem. *Operations Research* 15(2) 422-431.

Sahni, S. and Gonzalez, T. 1976. P-complete Approximation Problems. *Journal of the Association for Computing Machiner*y 23 555-565.

Sherali, H.D. and Smith, J. C. 2001.Improving Discrete Model Representations via Symmetry Considerations. *Management Science* 47(10) 1396–1407.

Showalter, M.J., Krajewski, L.J. and Ritzman, L.P. 1977. Manpower Allocation in US Postal Facilities: A Heuristic Approach. *Computers & Operations Research* 4(4) 257-269.

Wolsey, L.A. 1998. Integer Programming. John Wiley & Sons. New York. NY.

Zhang, X. and Bard, J.F. 2005. Equipment Scheduling at Mail Processing and Distribution Centers. Working paper, Graduate Program in Operations Research & Industrial Engineering, The University of Texas, Austin.

VITA

Lin Wan was born in Wuhan, China on April 16,1975, the son of Huiyi Wan and Meilan Lei. After completing his work at No.1 High School Affiliated to Huazhong Normal University, Wuhan, China, in 1993, he entered Huazhong University of Science and Technology in Wuhan , China. He received the degrees of Bachelor of Engineering and Bachelor of Law from Huazhong University of Science and Technology in June 1997. During the following year, he was employed as a student adviser in Huazhong University of Science and Technology. In September 1998, he entered the Graduate School of Huazhong University of Science and Technology and got his Master Degree of Engineering in June 2001.

In August 2001, he started his doctoral study in Operations Research and Industrial Engineering at University of Texas, Austin.

Permanent Address: 33 Caimao St., Apt. 1-2

Wuhan, Hubei, China, 430063