CHAPTER 64

Personnel Scheduling

RICHARD N. BURNS
BCW Consulting Limited

1. INTRODUCTION

Personnel scheduling involves two stakeholders: management, who need to have the work done, and workers, who need to have working conditions that contribute positively to their quality of life. In recent years, management has recognized that the objectives of the two groups need not be mutually exclusive. Good working schedules help to make happy employees and increase the length of time a person stays with the employer.

Personnel scheduling problems can be separated into different categories, based on the types of shifts to be worked, the employment contract working conditions clauses, and the pattern of the quantity of work needing to be done. The objective of this chapter is to give an overview of the types of personnel scheduling problems that arise and an introduction to some of the solution methods available.

2. BASIC STEPS FOR PERSONNEL SCHEDULING

Generally, there is a definite set of steps to complete the process of personnel scheduling. These steps will be discussed in more detail in this section, but the first step is to determine what work needs to be done.

Handbook of Industrial Engineering: Technology and Operations Management, Third Edition.
Edited by Gavriel Salvendy

2.1. The Quantity of Work to Be Done

The quantity of work to be done must be defined by the appropriate time period, which could be by as little as a 15-minute interval for cashiers and as long as a day for paper mills. For continuous process industries, such as chemical plants and some mining processes, the definition of what is needed is relatively easy to determine. Even so, the work must be defined by the hour and day of the year to allow for maintenance and shutdowns for holidays. There are many other industries that also have a well-known stable demand for work profile—for example, prisons, long-term health care facilities, and many manufacturing systems, such as assembly lines. At the other end of the scale are situations such as retail outlets and telephone operators, that have a demand that varies constantly during the day, and from day to day, and from season to season. Many of the organizations with such a fluctuating demand pattern have a detailed data bank of historic data, usually by the 15-minute interval. These data can be used to predict the work requirements for future time periods.

Hospitals have an expected amount of work for each day. However, many hospitals apply a patient classification system to all patients daily to calculate the actual amount of work needed for that day. This last-minute determination of the work requirement is necessary and unavoidable, even though it makes personnel scheduling very difficult.

Managing the demand to try to smooth out the fluctuations in work required is an important part of personnel scheduling. Examples of such efforts are visible in the reduced prices for matinees at theaters, early-bird specials at restaurants, and discounted long-distance calls for off hours. In manufacturing, longer, less complicated tasks are often done at night to reduce the number of supervisory and support staff needed. It is this ability to manage or alter the work required that can make the scheduling steps iterative. If the current demand proves too difficult to schedule, then this first step can be revisited to try to alter the demand. Once the concept of managing demand is acknowledged, the organization will often find suitable ways of making the demands change to match the ability to satisfy the demand.

2.2. Determining Staffing Requirements

Even when the work required is known, the process of determining how many people are needed can be very difficult. The first step is to break down the work required into shifts, with the number of people needed for each shift. The whole process depends on several factors and will be discussed in some detail in the subsequent sections. The criteria involved are as follows: what shift lengths are to be considered, what contract conditions apply, how many part-time workers are to be used, what sickness rate can be expected, what vacation and statutory holiday conditions apply, and what are the capabilities of the staff available. This last point illustrates that the process may be iterative because the worker availability may necessitate a change in the number of workers needed.

2.3. Determine Personnel Availability

There are, of course, the obvious categories of workers, such as full-time workers, regular part-time workers, casual workers, temporary workers, and labor pools. These categories can be expanded. For example, there may be full-time workers who only work weekends. There can be other full-time workers who only work specific shifts such as nights. In the part-time worker category, there may be two workers doing job sharing so that they are scheduled as if they were one full-time worker. In most cases, regular part-time workers are associated with specific jobs and times of availability. Casual workers also are often associated with specific jobs, while others can do many jobs. Maternity leave, long-term sick leave and other leaves, as well as vacations and requests for time off that have been granted must be acknowledged in determining who is available to work. In the last few years, many industries have employed more part-time employees to reduce costs and increase flexibility in scheduling. For example, the prison systems and schools now use more part-time staff that before. The availability of all the people must also include the number of hours of work available, the shifts, and the days of work available.

2.4. Developing Work Schedules

There is one fundamental approach to scheduling that needs to be considered when developing work schedules. In the past, many of the approaches have been to determine how many workers are needed in each time period and then assign people to the jobs. The emphasis has been from the management point of view of ensuring that there are enough people available to get the work done. This is not the best point of view or the approach that should be taken. The workers are at work, on average, 40 hours per week, even if they get paid for 37½ or some other figure. This leaves 128 hours of off time in each week for the workers. To the workers, the time off is the most important time. This is the time that they have to spend with family and friends, to indulge in hobbies, and just to relax. In developing schedules, the whole approach should be to make the time off for the workers the best time possible. A secondary goal, and one that is relatively easy to achieve, is to ensure that the right number of workers is available at all times.

For example, a worker does not like to have a weekend off to spend with the family, where they work the night shift Friday night which starts at 11:30 p.m. Friday and ends at 7:30 a.m. Saturday. By the time they get home from work on Saturday morning, have something to eat, and get at least seven hours of sleep, the day is gone. Although the shift schedule might show Friday night working, Saturday and Sunday off, and return to work on the day shift Monday, it is not a desirable time off.

Once the schedule has been built, it is relatively easy to assign workers to different lines of the schedule.

3. TYPES OF PERSONNEL SCHEDULING PROBLEMS

There are many ways of categorizing shift scheduling problems. Some of the choices are based on the demand patterns, such as stable demand during a shift vs. variable demand during a shift, or demand only during a single shift or multiple shifts. Another way of considering problems is by the way workers are scheduled, such as scheduling individuals vs. crew scheduling. Within each category there are numerous subcategories, and recent research has made progress on these problems.

3.1. Single Shift

For the most basic single shift problem, where there is only demand for the day shift for five days a week, the problem does not generally create difficulties. All workers can be scheduled to work Monday to Friday with every weekend off. Recently, many companies have changed this system to have flexible hours for the workers, providing that they spend at least a given number of hours in a specified time period, such as between 10 a.m. and 4 p.m. Again, no difficulty arises, and the same holds for summer hours, where people work 10-hour shifts for four days a week.

The more difficult problems occur when the daily demand for workers extends over a longer period than eight hours. In most retail stores, banks, and service outlets, such as clinics and repair depots, the hours of operation may be as many as 14 or 16 hours per day. Even operating rooms in major hospitals are used from early in the morning to 6 or 7 in the evening. A second condition that adds a whole new level of difficulty, is if the daily demand is for more than five days per week. Now the problem involves giving workers suitable weekends off with a required frequency.

Most retail outlets have developed sophisticated databases that track the historic sales. These stores are usually open long hours, seven days a week, and have a demand for workers that varies both by the day and by the 15-minute interval within a day. Although this is still a single-shift problem, the increased difficulty is shown when building schedules for these cases. Fortunately, part-time workers are used extensively, but even so, experience has shown that there is a great deal of discontent with the final weekly schedules that are posted. Section 4 of this chapter will address how to improve schedules for the time-varying demand problem.

3.2. Multiple Shifts Required

Multiple-shift scheduling problems have many added difficulties:

- Limiting the number of shift changes in a given time period
- Having a given amount of time off between shift changes
- Preceding weekends off and vacations with a suitable shift
- Balancing the less desirable shifts among workers
- Limiting the number of successive days that some shifts are worked
- Using worker seniority when assigning shifts if required by the contract
- Choosing the mix of shifts, such as 8-hour, 10-hour, and 12-hour shifts
- Balancing the paid hours worked

3.2.1. Crew Scheduling

In many continuous process industries, such as mines, steel, and chemical companies, the workers are scheduled by crew. The author has completed many consulting contracts for such companies, but has also turned down many more contracts for the very simple reason that one recurring problem could not be solved. The problem that arises time and again, but that is not solvable, is the request to help with the scheduling to reduce the amount of overtime being paid. Consider the mathematics of the situation. A crew is needed to be working continuously, seven days a week. This means that there are 7 days of 24 hours for a total of 168 hours of work needed. The company will have four crews rotating shifts to provide the coverage. Each crew provides 40 hours of work for a total of 160 hours. The shortfall of 8 hours each and every week is made up by the same four crews, so there must be 8 hours of overtime each week, which is an average of 2 hours of overtime each week for every crew. The mathematics seems obvious, but the request for help in removing this overtime arises a startling number of times. For most continuous process companies, there is no help in

reducing this overtime, but in a later section a set of suggested solutions for mining companies will be discussed.

3.2.2. Shift Scheduling for Individuals

Police, hospital workers, prison workers, all emergency systems, telephone workers, and many other seven-day-a-week operations schedule individuals rather than crews. These are the situations allowing for the most flexibility in scheduling, and also requiring complicated methods. They are also the source of most of the discontent with schedules because the workers expect better schedules than they receive, or better than may be possible. The following are some of the features not already mentioned above that make for discontent:

- Working too many days in a row too often
- Having too many single days off
- Perceiving inequities in schedules among workers
- Having too few shifts if a part-time worker
- Not having requested time off
- Not having a fair share of statutory holidays off

3.3. Hierarchical Workforce Requirements

Examples are perhaps the best way of illustrating these problems. For example, a prison system may require 20 guards for a shift, and at least one must be a qualified dog handler. In a hospital unit there may need to be eight nursing staff on the day shift, with at least six of them being four-year graduate nurses. The balance can be either nurses, nurse's aides, or nursing assistants.

Calculating the number of people needed and scheduling them is a little more difficult if there is a shortage of qualified people in some categories. Scheduling can be made easier by hiring the right mix of people or by training the people to achieve a desirable mix of worker skills. In recent years, many contract concessions have been given by unions to assist with this cross-training of workers.

3.4. Part-Time Workers

Part-time workers have become the mainstay for many industries. Various reasons exist for having part-time workers, only some of which pertain to scheduling, as can be seen in the following list:

- The full-time complement allowed is not enough to cover all the required workers, so part-time workers are scheduled into the master schedule to make up the deficit.
- People call in sick, so part-time workers are called at the last minute to fill the vacant spot in the schedule.
- A statutory holiday occurs and some full-time workers are given the day off; part-time workers are scheduled in advance for this holiday.
- Vacations, training time, business trips, long-term sickness, and other reasons for using a part-time worker are known in advance when the schedule is created.
- There is a fluctuation in demand during the day, or during the week that needs to be accounted for using part-time workers. In many cases, the need for part-time workers is known in advance, but in others, additional people will need to be called at the last minute.

Regular part-time workers are staff that are associated with the employer and used on a regular basis. In many instances these employees will fall under a union or employer contract. Some of the conditions prevailing in contracts are as follows:

- The person must be available for a given number of hours per week.
- The person must work at least a required number of weekends.
- The person must have at least a specified frequency of weekends off.
- The person cannot be required to work more than a specified number of days in a row.
- When part-time workers are needed, they are to be asked if they are available in some specified order, such as seniority, or equitably with the others.
- When called to work, the shift length must be at least as long as a specified time.

Many of the conditions above drive the way a schedule can be built. Very good records must be kept to ensure that the conditions are satisfied, as penalties often apply. The contacting of part-time

people to get them to commit to a specific job assignment is very time consuming and costly. For nurse scheduling, software packages are available on the Internet that will assist in this task.

Casual part-time workers are not usually employed under a set of contract conditions. For example, in nursing, a hospital will have a list of nurses who would like some work but who are not on the regular part-time list. These nurses are called only after the list of regular part-time nurses has been exhausted. They will work a full shift, in one of a short list of units that depends on the person's skills and experience. Retail stores and fast-food outlets use many casual workers, including a large number of students. This scheduling also takes a lot of time because of the information base that must be referred to when scheduling and updated as a schedule is used. Some of the information required is similar to that of the regular part-time workers. The following is a partial list:

- The person's usual available hours
- The number of hours desired by the person
- The skills that the person has and the jobs he or she qualifies for
- The seniority of the person
- The number of hours that the person has worked in a rolling window of time
- The person's immediate past schedule record
- The means of contacting the person
- How many recent requests to work were refused or not answered
- Any short-term unavailability times requested by the person for personal reasons
- The average number of hours given to people in the same category of workers to ensure equitable assignment of work time

Many grocery stores and fast-food stores leave the scheduling of part-time workers to be done manually by administration. While they try to do a good job, it is time consuming and inequitable schedules can inadvertently arise. Such schedules can lead to discontent and employee unrest. A good system must be set up for this task.

Temporary workers are only part-time in that they are not permanent workers. They will usually be summer students or people hired to fill a job that is being reserved for someone who will be absent for a longer period of time. Long-term sickness and maternity leaves are simple examples of the need for temporary workers. Temporary workers will be scheduled just as if they are full-time workers, subject to all the rules for the full-time workers they are replacing. In seasonal situations, the temporary workers may be the bulk of the people being employed.

There is one more class of workers who appear as part-time workers to the scheduler but are in fact full-time workers. The name for this class differs from industry to industry and changes with time within an industry. ''Relief labor pool'' and ''float pool'' are names for a group of full-time people who are available to work in different jobs in the organization. The scheduler for one group of workers will use these people as needed. Essentially, the group scheduler is only assigning a job to the relief worker, who is already working to a predeveloped schedule. The person responsible for developing the relief pool schedule and for maintaining it for requests for changes for personal reasons, holidays, and vacation days is another issue. Someone must also be responsible for assigning jobs to the relief pool people who have not been requested to work by a specific unit.

4. CREATING SCHEDULES

Schedules have an important effect on employees' lives. A good schedule for one person may be considered a poor schedule by another person. There are no absolutes, but there are guidelines. If the schedules are good, worker performance and worker retention will improve, which is good for the organization as a whole. For example, in a major city hospital with 21 operating theaters, there was a 30% turnover of nurses. Changing the schedules made the turnover rate drop to the hospital average. A schedule can be considered optimal in the sense that it only reduces labor cost to the minimum, but an optimum schedule will minimize overall costs by reducing labor costs while improving performance and keeping workers satisfied with their schedules. As mentioned earlier, the goal of scheduling should be to develop optimum schedules, not just locally optimal schedules.

Much has been done in recent years to understand the personnel scheduling problem. As with most research, progress is made by making simplifying assumptions and solving the resulting problem. Insights are gained and the problem is then expanded slightly. By this iterative approach to research, problems resembling the real-world situation are finally attained. In personnel scheduling, many of the real-world problems have not yet been solved, but much of the insight obtained in the process of doing the research helps the scheduler to develop workable schedules. In the next few

TABLE 1 Week as Monday to Sunday

Line	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	X					X							X	X
2			X				X						X	X
3						X	X	X					X	
4						X	X			X				X

sections, some of the more useful facts will be presented, along with suggested ways of using them when scheduling.

4.1. Single Shift

Single-shift scheduling is still used in many situations. The shift is called a day shift, whether a person starts at 7 a.m. or 3 p.m. Schedules can be made for people and the time of starting can be determined later. The more interesting situation is a seven-day-a-week operation, with people being scheduled to work eight-hour shifts for an average of five shifts per week.

Cyclic scheduling was one of the first analytic approaches to single-shift scheduling. The situation can have a variable demand for the seven days and a requirement that all workers have two successive days off. This problem can be solved by mathematical programming, as shown in Baker (1974, 1976) and Tibrewala et al. (1972). The final schedules will have each person working five consecutive days followed by two days off. If a person has Tuesday and Wednesday off in one week, then he or she will always have the same two days off and will never have a weekend off. This type of schedule, while useful in some situations, is optimal from the short-term point of view of the employer but may in the long run cause a higher turnover of employees as they move in order to have some weekends off.

Burns (1978) introduced three very interesting facts to the process of scheduling. The first was that the definition of a work week should be Sunday to Saturday, not Monday to Sunday, as will be shown below. The second was that a simple analytic formula could be used to calculate how many workers were needed, and the third was that a simple polynomial time method could be used for developing optimal schedules without using mathematical programming. These three facts have formed the basis of many research papers and doctoral theses since that time, and they are useful in practice. Burns and Carter (1985) generalized the problem to allow for any frequency of weekends off.

Consider the problem of building a schedule that has one person working each day of the week, and the week is defined as Monday to Sunday. The schedule is to have each person working five days a week, with the maximum work stretch six days, and each person is to have every second weekend off. The difficulty arises in the week before a person has the weekend off. Once a person has Saturday and Sunday off in a week, he or she must work the five weekdays in order to have five working days per week. This means that in the week before, he or she must have either Sunday or Saturday off in order to restrict the work stretch to a maximum of six days. In Table 1, the schedule necessary to have at least one person working every day is given.

In order to have one person working each day, it is necessary to hire four people. To have n people each day, the requirement will be $4n$ workers. The schedule can be presented in a cyclic form, as shown in Table 2, with one person starting on each of the four lines and working that week before going to the next line. The person working the last line goes to the first line next. Notice that in order to have one person working on the weekend, there will be up to four people working on a weekday.

Changing the definition of the week to be from Sunday to Saturday means that the schedule can be constructed with only two people. Table 3 is an example of a cyclic schedule for a scheduling

TABLE 2 Week as Monday to Sunday, Cyclic Schedule

Line	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	X					X	
2						X	X
3			X				X
4						X	X
Totals	3	4	3	4	4	1	1

TABLE 3 Week as Sunday to Saturday, Cyclic Schedule

Line	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	X					X	
2			X				X
totals	1	2	1	2	2	1	1

problem with cyclic demand. Start one person on each line and have the person go to the next line after Saturday, where the first line is next after the second line.

In a cyclic scheduling problem, the demand on each day of the week is the same for each week. However, the demand may vary from day to day within a week. There are two basic types of schedules for the cyclic problem. The first is a cyclic schedule, such as the one in Table 3. In a cyclic schedule, the schedule itself repeats over some defined period, thereby giving a ''master rotation.'' The second type of schedule is an iterative schedule, where an algorithm is given that schedules the first week of the schedule, and then another part of the algorithm shows how to generate week $i + 1$, given week i of the schedule. An iterative schedule might never repeat, and no ''master rotation'' is available.

The following is the statement of the scheduling problem that can be solved using the methods of Burns and Carter (1985):

W is the minimum number of workers required to cover a seven-day-a-week operation with the following primary constraints satisfied:

1. The demand per day n_j, $j = 1, 2, \ldots, 7$; n_1 being Sunday's demand.
2. Each worker is given at least A out of B weekends off.
3. Each worker works exactly five out of seven days each week.
4. Each worker works no more than six consecutive days.

In Burns (1978), the requirement was for every second weekend off.

This last constraint is added because a person could work the last five days of one week, followed by up to five more consecutive days of the next week, unless the restriction were in place. In Brown et al. (1976) over 50% of the schedules produced had work stretches of 10 consecutive days. Even as late as 1994, papers have appeared where the work stretch is as long as 10 days. Such schedules are not of much practical use.

The algorithm of Burns and Carter (1985) is an iterative scheduling method. The basic approach of the algorithm for the problem in this section is simple.

Ignore some of the constraints to get an absolute lower bound on the number of people required to complete the schedule. It will be a lower bound because if more constraints were incorporated, some schedules might not be feasible and more people might be needed.

Build a schedule for this lower bound number of people by first assigning the weekends off so that the required frequency of weekends off will be met for everyone.

Then assign the remaining days off for each worker to satisfy the work stretch constraint and ensure that everyone has exactly two days off per week.

Note, as outlined in the previous section, that the method does not assign workers to work specific days, but does concentrate on the workers' time off, which will produce better schedules for the individual. Following the detailed steps of the algorithm ensures that the schedule for just the lower bound number of people will, when completed, satisfy the total requirements for workers in each time period. The schedule is then optimal because no other schedule could be produced using fewer people.

Lower bounds for the number of workers needed will now be given. There are three bounds that are used:

1. *The weekend constraint:* The total number of employees' shifts available for the B weekends must be sufficient to meet the maximum weekend demand for the B weekends. In B weeks, each employee is available for $(B - A)$ weekends. Hence, $(B - A)W \geq Bn$, where $n = \max(n_1, n_7)$ is the maximum weekend demand.

$$L_1\colon W \geq \left\lceil \frac{Bn}{(B - A)} \right\rceil, \text{ where } \lceil x \rceil \text{ is the upper integer part of } x.$$

2. *The total demand constraint:* The total number of employees' days worked per week must be sufficient to meet the total weekly demand for shifts. Since each employee works exactly five days per week:

$$L_2\text{: } 5W \geq \sum_{i=1}^{7} n_i \text{ or } W \geq \left[\frac{1}{5}\sum_{i=1}^{7} n_i\right]$$

3. *The maximum daily demand constraint:* The number of employees must be sufficient to meet the maximum demand on any day.

$$L_3\text{: } W \geq \max_i(n_i),\ i = 1, 2, \ldots, 7$$

The algorithm uses a workforce equal to the maximum of the three constraints. The paper proves that if the algorithm is used, then the W employees will be sufficient to satisfy all the problem constraints, including the ones not used in calculating W.

In scheduling the weekends off, the algorithm assigns a number from 1 to W to each employee. Since n people are required to work each weekend, $(W - n)$ can have the weekend off. Assign the first weekend off to the first $(W - n)$ people. Assign the next $(W - n)$ weekends off to the next $(W - n)$ people. This process is continued cyclically with employee 1 being next after employee W.

For each week of the schedule, every employee must have exactly two days off. In any given week, there are already $(W - n)$ Sundays off at the beginning of the week and $(W - n)$ Saturdays off at the end of the week. An additional $2n$ days must be given off so that all W people have two days off and the algorithm constructs n off-day pairs for this task.

Denote the surplus workers for day j by S_j.

For Monday to Friday $S_j = W - n_j$ for $(j = 2, 3, 4, 5, 6)$
For Sunday $S_j = n - n_1$
For Saturday $S_j = n - n_7$

Iteratively, construct a list of n pairs of off days, numbered from 1 to n, as follows:

1. Choose day k such that $S_k = \max(S_j)$.
2. Choose any $i \neq k$ such that $S_i > 0$. If $S_i = 0$ for all $i \neq k$, set $i = k$.
3. Add the pair (k, i) to the list and decrease S_k and S_i by 1.
4. Repeat this procedure n times.

Pairs of the form (k, k) are called nondistinct off-day pairs and will be at the end of the list.

The next step is to assign the off-day pairs in week 1. Employees will fall into one of four categories, depending on the weekends off at the beginning and end of the week. Table 4 illustrates the categories.

There are n people working on each weekend. Therefore, $|T3| + |T4| = n$ from weekend 1 and $|T2| + |T4| = n$ from weekend 2, for $|x|$ = the cardinality of the set x. Each employee of type $T2$ is paired with one of type $T3$.

Assign the n pairs of off-days from the top of the list, first to the employees of type $T4$ and then to employees of type $T3$. The $T3$ get the earliest day of the pair, and the latest day is assigned to their associated type $T2$ employee. The paper proves that nondistinct pairs of off-days always get assigned to type $T3$ workers and then split. Hence, the algorithm will not assign two days off to a worker where the two days are the same day.

The next step is to assign the off-day pairs in week i for $i > 1$. Assume that weeks 1 up to week $(i - 1)$ have been scheduled. In week i, categorize the workers into the four types $T1$, $T2$, $T3$, $T4$. The critical problem of this step is to ensure that no employee has a work stretch longer than six days. The assignment of off-days for week i falls into two cases:

TABLE 4 Worker Categories

Category	Weekend 1	Week 1	Weekend 2
Type $T1$	off	no off-days needed	off
Type $T2$	off	one off-day needed	on
Type $T3$	on	one off-day needed	off
Type $T4$	on	two off-days needed	on

Case 1: If the list of off-day pairs contains nondistinct pairs of the form (k, k), for some day k, then week i is scheduled in precisely the same way as week 1 and is independent of week $(i - 1)$.

Case 2: If all off-day pairs are distinct, of the form (j, k) where $j \neq k$, then all employees of type $T3$ and $T4$ in week i are associated with the same pair of off-days that they received in week $(i - 1)$. The type $T4$ are given both days off and the type $T3$ get the earliest day of the associated pair. The $T2$ employees get the remaining days of the $T3$ pairs. In this instance, the maximum possible work stretch will be five days.

Example 1:

day j	1	2	3	4	5	6	7
	Sun.	Mon.	Tue.	Wed.	Thu.	Fri.	Sat.
required:	4	4	5	5	7	7	5

where each person requires one out of three weekends off. $A = 1$; $B = 3$, and $n = 5$. Then:

$$L_1 = \left[\frac{3 * 5}{(3 - 1)}\right] = 8; L_2 = \left[\frac{37}{5}\right] = 8; L_3 = 7$$

The minimum workforce is $W = 8$. $(W - n) = 3$, which is how many workers are assigned off each weekend. The schedule after assigning the weekends off looks like Table 5.

After the weekends off are assigned, there are eight people available to work each weekday and five on the weekends. The surplus is:

day j	1	2	3	4	5	6	7
surplus S_j	1	4	3	3	1	1	0

Because of ties in the surplus, there are many different ways to make the pairs of off-days. The following is one list of pairs that would be generated using the algorithm as stated:

List 1: (Mon., Tue.), (Mon., Wed.), (Mon., Tue.), (Mon., Wed.), (Tue., Wed.)

Two features of the pair-choosing algorithm are incorporated in its design. The first is that the surplus workers on any one day, when the selection is completed, will be made as even as possible. The second feature is that, if at all possible, nondistinct pairs will be avoided. As far as the proofs of the optimality of the algorithm are concerned, only the second feature is necessary. This means that any other algorithm for choosing the pairs is acceptable as long as it does not unnecessarily create nondistinct pairs. The quality of the schedules can be improved by modifying the step of the algorithm that chooses the pairs of off-days. If list 1 above is used, the resulting schedule will have two features that are considered to be bad in schedules. The first is that there will be single-day work stretches where a person has a day off followed by a working day followed by a day off (on–off–on). Such a feature is not wanted by either management and the workers. The second poor feature is there are type 4 workers who work both weekends of a week and who will receive nonadjacent days off. If at all possible, this should be avoided.

TABLE 5 Example 1: Assigning the Weekends Off

Employee	S	S M T W T F S	S M T W T F S	S M T W _ _ _
1	X	X		X	X
2	X	X			
3	X	X			
4			X	X		
5			X	X		
6			X	X		
7					X	X
8					X	X

Some published research papers suggest that using heuristics after the schedule is built can minimize the single-day work stretch. It is often possible to choose a different set of pairs of days off, before creating the schedule, that will result in no single day work stretches or split days off for the type 4 workers. We will improve the schedule of this example, but first it should be clear that it is not always possible to remove all the single day work stretches.

Selkirk (1999), a doctoral student of Burns, published a thesis that shows that for some problems, a new bound that increases the size of the workforce needs to be introduced to ensure that the single-day work stretches are avoided. He gives very complicated, optimal polynomial time algorithms for different subsets of the single shift scheduling problems. These algorithms are beyond the scope of this article but are mentioned here to show that heuristics cannot always be used to eliminate single-day work stretches.

The quality of the schedules can be improved if the cause of the poor schedules can be avoided when creating the off-day pairs. By applying the method for choosing pairs by Burns and Carter (1985), it is ascertained that nondistinct pairs can be avoided. The example has two type 4 workers each week. A type 4 worker who just becomes a type 4 worker in week i must have worked the weekend in the previous week and had one other day off, and therefore was a type 2 worker in the previous week. In the algorithm, this person gets both days of the associated pair off in week i. If all the pairs were adjacent pairs, then there would be no problem. Therefore, one goal in picking the pairs is to have as many pairs of adjacent days as is possible when type 4 workers are required.

The single-day work stretches are caused by two types of pairs. The first are the pairs (Mon., Tue) and (Thu., Fri.). In the algorithm the person who has Sunday at the beginning week off gets the latest day of the pair. This means that for the pair (Mon., Tue.) they would receive the Tuesday off after having the weekend off, thus creating a single-day work stretch. Similarly, the person having the weekend off at the end of the week receives the earliest day of the pair off, which for the pair (Thu., Fri.) will give him or her Thursday and Saturday off, which creates a single-day work stretch. The algorithm used for choosing the pairs of days off should avoid creating these two troublesome pairs.

The second cause of single work stretches is having a type 4 worker receive both days of a pair off, where the days in the pair are separated by only one day. This pairing should be avoided when type 4 workers exist in the schedule.

There are other quality of work schedule considerations. For example, if the demand on Saturday is different from the demand on Sunday, then it is possible to give days off on one of the weekend days. Workers prefer to have weekend days off. The pair (Mon., Fri.) is also useful because it can be exchanged in the schedule for any pair and still maintain a maximum work stretch of six days.

A new list of off-day pairs can now be prepared for example 1, taking into consideration the quality considerations just discussed.

List 2: (Sun., Mon.), (Tue., Wed.), (Tue., Wed.), (Tue., Wed.), (Mon, Fri.)

In week 1, the algorithm will assign adjacent pairs from the beginning of the list. In subsequent weeks the type 4 workers will either receive a pair of adjacent days or will receive the pair (Mon., Fri.). With a little thought, it is easy to see that this pair can be exchanged with any other pair of type 3 and type 4 workers without violating the work stretch constraint of any of the three people involved in the switch.

The final schedule is shown in Table 6.

In week 2, worker 3, a type 4 person, would have received the pair of off-days (Mon., Fri.), but this can always be switched with any paired workers of type 2 and type 3 receiving days off from

TABLE 6 Example 1: Assigning the Weekdays Off

Employee	S	S M T W T F S	S M T W T F S	S M T W _ _ _
1	X	X X	X X	X
2	X	X X	X X	
3	X	X X	X X	
4		X X	X X	
5		X X	X X	
6		X X	X X	
7		X X	X	X
			X	
8		X X	X X	X
surplus		1 0 0 0 1 1 0	1 0 0 0 1 1 0	

the pair (Tue., Wed.), so the first and fourth workers were used for the switch. The final schedule has no single workday stretches, and all type 4 workers have pairs of adjacent days off.

The bounds for the minimum number of workers required can be used for other purposes. For example 1 above, if a part-time worker is hired for Saturday, the variable n is reduced to 4. Bound L_2 still requires that 8 workers be hired, but the frequency of weekends off can be increased to every second weekend off without hiring more workers. For $A = 1$, $B = 2$, $n = 4$, the bound L_1 will still be 8. An interesting result will be that in the final schedule, the surplus workers on the weekdays will be increased because more days off will be given on the weekend for the same 8 people. The employer will need to find work for these surplus people, if possible.

One way of eliminating surplus workers in the final schedule is to hire part-time workers for the weekend, and for the weekdays to have the bounds given by L_1 and L_2 equal. Extra part-time workers would be hired to ensure that the bound L_2 is divisible by five.

The bounds given are very general and can be used for much more complicated scheduling problems. For example, Burns (1981) showed that the same bounds are sufficient for the problem having multiple eight-hour shifts with different demands on the shifts, and with the restriction that there must be at least 24 hours off between shift changes. As mentioned above in the reference to Selkirk (1999), enforcing no single-day work stretches may necessitate increasing the size of W by using a new bound.

Adjacent days off is another restriction that has drastic implications when the maximum work stretch is restricted to six days. The six-day work stretch restriction is the key factor. Burns and Koop (1984) explored this case and found the following:

- There are very few possible schedules.
- It is no longer possible to have each worker work exactly five days per week as is shown in Table 7. Instead, the number of days worked averages five days per week over the B week period of the cycle.
- The number of workers will need to be increased drastically from the same problem where single days off are allowed.

Consider the lower bound L_1, based on the weekend constraint. In most practical problems, the weekend constraint will be tight because it is desirable to give as many weekends off as is possible with the current work force. If the work stretch is limited to six days and all days off must be adjacent, then for each string of weekends off, a worker must have either the Saturday off in the week preceding the string of weeks or the Sunday off after the string. Table 7 illustrates this by giving the only two possible schedules, each with two work stretches of six days, and one of three days, for the example of one weekend off in three weeks.

A new weekend bound needs to be calculated. The bound is constructed as before, using the obvious fact that the weekend shifts available from W workers over B weeks must be greater than or equal to the shifts needed on weekends in B weeks. If all the weekends off are given sequentially, followed by all the weekends worked, each worker will only need one extra weekend day off. If the weekends off are separated by worked weekends in the B week cycle, then more single weekend days off will be needed, which might require more workers.

$L_{1\,\text{paired}}$: Available weekend shifts $\geq$ required weekend shifts

$$[2(B - A) - 1]W \geq 2Bn$$

$$W \geq \left\lceil \frac{2Bn}{2(B - A) - 1} \right\rceil$$

Table 8 gives the values of the new and old bounds for some of the frequencies of weekends off found in practice.

The most startling result is for every second weekend off, where the number of workers needed in the schedule must have all paired days off doubled! Even for the other two cases, the number of

TABLE 7 Paired Days off, $A = 1$, $B = 3$

S	S M T W T F S	S M T W T F S	S M T W T F S	S
X	X X	6 days	X X 6 days	X X
X	X 6 days	X X 6 days	X X	X X

TABLE 8 All Paired Days Off Compared to Some Single Days off

Frequency of Weekends Off A	Cycle Length B	L_1 paired	L_1
1	3	$2n$	$3n/2$
2	5	$2n$	$5n/3$
1	2	$4n$	$2n$

workers needed increases by a large percentage. To reduce the surplus workers that would be scheduled for weekdays, the demand for workers on weekends would need to be reduced by hiring part-time workers for weekend shifts and hiring fewer full-time workers.

Shift scheduling is like job shop scheduling in that a small change in the problem statement results in an entirely new algorithm being needed. Since Burns and Carter (1985) solve the most general problem available with variable demand, why produce special algorithms for more restricted cases? The answer is that for more restricted cases, better schedules can be guaranteed. Several cases, such as constant demand each day, or a fixed demand of N for the weekdays, and a demand of n for both weekend days, have been addressed in the research literature. The algorithms presented for the case where $N \geq n$ are easier to implement and to prove that they are optimal. One the main differences is in picking the list of paired days off. However, the Burns Carter (1985) method, with the paired-off list modified, as suggested in this article, will solve all of these cases.

A second class of single-shift scheduling problems occurs when the demand pattern changes by shortened time intervals within a day. The demand for telephone operators and cashiers in retail stores may vary by the hour or half hour or by even less time. In some cases it is also necessary to build schedules that give the workers lunch breaks and coffee breaks while ensuring that the demands are met by the workers not on breaks. Most retail stores are closed for several hours a day. It is not necessary to consider shift rotation times from one day to the next, which makes it a single-shift problem, even though there will be many different shifts during the day. There are the same three steps with this type of scheduling as with the problems where the demand is constant over a shift: determining the demand; creating the schedules that meet the demand at the lowest cost possible; and meeting the scheduling needs of the people who work the schedule.

Many of the big chain stores have a database and computer software that can predict the requirements for the days to be scheduled, based on annual history and recent history that will account for trends. Exponential smoothing, running averages, or other methods can be used to predict the daily demand. Adjustments must be made based on weather conditions in the past, special promotions, or other predictable influences.

There are several methods reported in the literature dealing with creating the schedules, given the demand. One way is to use mathematical programming methods such as linear or integer programming. Other approaches such as Segal (1974), using network flows, Henderson and Berry (1976), using heuristics, and Glover and McMillan (1986), using a taboo search, have addressed the problem of determining the shifts to be worked on one day. But why are these methods not used extensively in practice? One of the reasons is that the demand is not cyclic. The daily demand continues to change each day of the year and from year to year. The scheduling problem needs to be solved every day, for every company location, and in some instances, for different groups of workers in each location. What is done in practice? Many retail stores are given the demand pattern expected for the next scheduling period by a central computer system, while others create the demand pattern locally. Some stores are given schedules from a central location, but in many cases these have not proven to be very useful. Management in the store usually create the schedules manually. The task is time consuming and unrewarding because workers are often unhappy with their schedules.

The methods currently used for this time-varying demand problem do not address the assigning of full-time workers to have a maximum work stretch of six days, or weekends off with a given frequency, or any constraints on the pairing of off-days, or, in fact, any conditions linking one day to the next. Research for the fixed-shift requirement, cyclic demand work, and research for the time-dependent, noncyclic demand case have been developed in parallel with little if any crossover. There is no need for this separation. The following is an outline of a method for creating the shifts needed and the schedules that will satisfy the full-time workers' requirements for good schedules, followed by a method of creating shifts for the part-time workers to complete the total schedule.

The suggested methods presented below are extracted from the retail shift scheduling algorithms used in Burns (1999). In a commercial computer package, there are many small details and specifics of the particular installation built into the algorithms. For example, the third step below would be modified based on how many hours the operation is open and on the number of peaks in the demand pattern. While important to the scheduling efficiency, these specific features are not necessary to the understanding of the method.

In any particular location, there will be a core of W full-time staff and many part-time staff. The full-time staff will have a set of conditions that need to be incorporated into their schedule, such as A out of B weekends off, a maximum work stretch of six days, and five days per week of work. Schedules for the full-time workers will be considered first.

Step 1. Calculate the demand pattern to be satisfied by the W full-time workers.

Shifts available on the weekend $= (B - A)W$

Shifts required $= Bn$, where n is the demand to be satisfied by full-time employees

Therefore, $Bn \leq (B - A)W$, or

Weekend shifts available from full time workers is given by

$$n = \left\lfloor \frac{(B - A)W}{B} \right\rfloor$$

where $\lfloor x \rfloor$ is the largest integer contained in x.

Assuming at least N shifts per weekday are to be satisfied by full-time people, the average number of shifts per weekday from full-time workers is given by

$$N = \left\lfloor \frac{5W - 2n}{5} \right\rfloor$$

If rounding down was done to create N, then let k be the number of eight-hour shifts rounded down, which is given by $k = (5W - 2n) \bmod (5)$. k weekday shifts will need to be created in the schedule, in addition to the N shifts each weekday.

The k shifts will be added to the N shifts on weekdays, requiring the most number of workers while ensuring that the maximum required per day does not exceed W, or will be distributed evenly so that the final demand pattern will be as follows:

Sunday $= d(1) = n$. Saturday $= d(7) = n$. The weekdays have demand of either $d(i) = N$ or $d(i) = N + a$, where $a =$ the number of extra shifts, from the k shifts, that have been allocated to day i, $i = 2, 3, 4, 5, 6$.

Step 2. Use the Burns and Carter (1985) method with the list of off-day pairs, as modified above, to schedule the W full-time workers to the next week. Since the method is iterative, the starting position, after the first time that the algorithm has been used, will be from the previous week, using the list of off-day pairs in existence. After step 1, the number of full-time shifts required per day is known, and after step 2, who will work the shifts is known; but when the full time shifts will actually start within a day has not been determined.

Step 3. Create the shifts to be worked on each day of the schedule, using the part-time workers as well as the full-time workers.

Let $s(j)$ be the number of people needed in each time period of the day being considered. The time period could be an hour, a half hour, or just 15 minutes. Let d be the requirement for eight-hour shifts on the day being considered. The algorithm will try to schedule the eight-hour shifts first and then five-hour part-time shifts. The selection of five hours is arbitrary and could be changed depending on the installation. The algorithm may need to complete the first pass at allocating part-time shifts by using shifts as short as three hours. Then the method will assign lunch breaks and coffee breaks. The final pass will be to add shifts where needed, and reduce shift lengths where needed.

Creating the eight-hour shifts: Start at period $k = 1$.

If $s(k) > 0$, let m be the minimum of $s(k)$ and d. Create m eight-hour shifts, starting at period k. Set $d = d - m$ and $s(j) = s(j) - m$ for the eight-hours, starting with period k.

If $s(k) \leq 0$, set $k = k + 1$. Repeat this step to create eight-hour shifts until $d = 0$.

Creating part-time shifts: Start with period k from the previous step.

If $s(k) > 0$, check to see if there is enough time left for a part-time shift. If the time left is at least five hours, assign the shift length L to be five hours. If the time left is greater than or equal to three hours but less than five hours, assign the shift length L to be the time left. If the time left is less than three hours, set L equal to three hours and set F equal to the period that starts three hours before the end of the day. If F was set in this step, start at period F and create $s(k)$ shifts of length L and set $s(j) = s(j) - s(k)$ for all periods from period F until the end of the day. If F was not set in this step, start at period k and create $s(k)$ shifts of length L. Set $s(j) = s(j) - s(k)$ for the L hours starting with period k. Set $k = k + 1$ and repeat this step until all $s(k) \leq 0$.

TABLE 9 Weekly Eight-Hour Shifts for Time-Dependent Workers

Worker	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
1	X						X	X			X						X				X
2	X			X						X				X	X						X
3			X				X	X						X	X		X				

The final steps are straightforward in principal but complicated in detail. Assign all the break times needed, using the surplus workers wherever possible, that is, where $s(k) < 0$. Then either add part-time shifts where $s(k) > 0$ or increase the length of a part time shift to reduce $s(k)$ to zero. Finally, try to reduce the length of part-time shifts to increase $s(k)$ to 0 wherever $s(k) < 0$.

The entire algorithm is repeated for all the days that are to be scheduled. An important output from the algorithm is the printed reports giving the schedules and who is to work each shift. The full-time people have been assigned using the method as outlined in this section. The part-time people will be assigned using the work of Section 3.4.

The algorithm is easy to computerize or to do manually and is very fast because the amount of work is polynomial.

Example 2: Time-variant, single-shift scheduling:

Consider a retail store that is open from 9 a.m. to 9 p.m. There are three full-time employees requiring every second weekend off, a maximum work stretch of six days, and five days of work per week. The required number of people is given in Table 10 in half-hour intervals. All eight-hour shifts must have a lunch period of a half hour; coffee breaks will not be scheduled but will be given during the day as time is available. The part-time shifts must be at least three hours long and can be up to five hours long.

From the facts given, $A = 1$, $B = 2$, $W = 3$ and from step 1 of the algorithm,

$$n = 1, N = \left[\frac{15 - 2}{5}\right] = 2, k = 3$$

There will be three weekdays with three workers required and two with two required. At this point, looking ahead to step 2 of the algorithm is useful. The modification to selecting the paired off-days above suggests that (Mon., Tue.) and (Thu., Fri.) pairs should be avoided to eliminate one-day work stretches for the full time workers, and we need $n = 1$ pairs of off-days. If the three extra working days are given to Wednesday, Thursday, and Friday, the only pair of off-days would be (Mon., Tue.), which is undesirable. To help make better schedules, the three extra workdays will be assigned to Monday, Thursday and Friday leaving (Tue., Wed.) as the pair of off-days for step 2 of the algorithm. Sunday $= d(1) = 1$, $d(2) = 3$, $d(3) = d(4) = 2$, $d(5) = d(6) = 3$, $d(7) = 1$.

In step 2 of the algorithm $(W - n) = 2$ people are given the weekend off each week. Table 9 gives the schedule for the eight-hour full-time workers.

Because there are no choices of days off when creating the schedule, the schedule is cyclic, with a period of three weeks; that is, the schedule is the first week repeated. For step 3 of the algorithm, completing one day will be sufficient to illustrate the algorithm. Consider a Thursday, with the requirement for workers given in Table 10.

Step 3 first assigns the following shifts:

Eight-hour shifts starting at periods 1, 3, and 4

Five-hour shift starting at period 5

Three-and-a-half hour shift starting at period 18

Two three-hour shifts starting at period 19

TABLE 10 Example 2: Required Workers for Time-Dependent Workers at the Start

Time	9		10		11		12		1		2		3		4		5		6		7		8	
Per	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$s(j)$	1	1	2	3	4	4	4	4	4	4	4	4	3	3	3	2	2	3	3	3	3	3	3	2

TABLE 11 Required Workers for Time-Dependent Workers on Thursday in the Middle

Time	9		10		11		12		1		2		3		4		5		6		7		8	
Per	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$s(j)$	0	0	0	0	0	0	0	0	0	−1	−1	−1	−1	−1	0	−1	0	0	−1	0	0	0	0	−1

Lunch breaks can be assigned to several places. Choose the half hour lunch break for the eight-hour worker number 1 to start at period 7, worker 2 at period 8, and worker 3 at period 9 and start a new three-hour shift to start at period 7 to cover for the lunch breaks. The current status of the worker requirement table is given in Table 11.

Using the last part of the step 3, the five-hour shift starting at period 5 would be reduced to a three-hour shift, and the three-and-a-half-hour shift starting at period 18 would be reduced to a three-hour shift. The final schedule would have the following shifts:

Eight-hour shifts starting at periods 1, 3, and 4

Three-hour shift starting at period 5

Three-hour shift starting at period 7

Three-hour shift starting at period 18

Two three-hour shifts starting at period 19

The final status of the worker requirement table is given in Table 12.

For the problem requirements and scheduling restrictions, the schedule is optimal. Two of the extra shifts could be removed if the minimum part time shift is reduced to two-and-a-half hours from three hours.

4.2. Multiple Eight-Hour Shift Crew Scheduling

As discussed in Section 3.2.1, a continuous operation working four complete crews has 2 hours of overtime per week per crew. Switching to 12-hour shifts or 8-hour shifts will not remove the problem. Later in this section, some special schedules for the mining industry are presented where the work to be done is reduced for short periods of time and no overtime occurs. In continuous operations, one industry standard is to have no single days off and to allow seven-day work stretches. Because the demand is constant, 24 hours a day for every day, it is possible to have as goals the following: a good schedule will have shift rotations always moving forward, that is, day to evening to night shift; a good schedule will also have three days off after the night shift, if possible, and will not start a weekend off after working the night shift on Friday, because the worker would need to sleep most of the Saturday. Tables 13 and 14 are two different schedules for four crews, each having the desired properties. In the schedules d, e, n are 8-hour shifts for the day, evening, and night respectively. As with the single-shift schedules, a crew (person) starts on each line and when a line is finished moves to the next line, with the first line being next after the last line.

Crew schedule 1 is the more standard type of shift rotation, whereas crew schedule 2 has the fast rotation of shifts that some places like to work. Note that, as expected, each crew only has seven days off in four weeks, which means they work eight hours overtime in four weeks.

If a maximum work stretch of six days is required, then a 24-week cyclic schedule can be made. An 8-week schedule forms the basis for the schedule and is repeated three times, with the starting shift changed each time. The starting shift will be determined by just continuing on the shift that was being worked at the end of the cycle. Table 15 gives the basic schedule that is worked by all four crews, starting the cycle with one crew on each of weeks 1, 7, 13, and 19.

Over the years, the author of this article has worked on numerous consulting projects for personnel scheduling. Repeatedly, the consulting has been initiated because a company signed a new contract with the union and then found that they either could not create any schedules to match the contract conditions or could not create schedules without a considerable increase in labor costs. The motive

TABLE 12 Required Workers for Time-Dependent Workers on Thursday when Finished

Time	9		10		11		12		1		2		3		4		5		6		7		8	
Per	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$s(j)$	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	−1	0	0	−1	0	0	0	0	0

TABLE 13 Crew Schedule 1

Crew	S	M	T	W	T	F	S
1	X	d	d	d	d	d	d
2	d	X	X	e	e	e	e
3	e	e	e	X	X	n	n
4	n	n	n	n	n	X	X

TABLE 14 Crew Schedule 2

Crew	S	M	T	W	T	F	S
1	X	d	d	e	e	n	n
2	n	X	X	d	d	e	e
3	e	n	n	X	X	d	d
4	d	e	e	n	n	X	X

TABLE 15 Crew Schedules with Six-Day Work Stretches

week 1 *	week 2	week 3 *	week 4
S M T W T F S	S M T W T F S	S M T W T F S	S M T W T F S
n n n X X d d	d d d d X X e	e e e e e X X	n n n n n n X

week 5 *	week 6	week 7 *	week 8
S M T W T F S	S M T W T F S	S M T W T F S	S M T W T F S
X d d d d d d	X X e e e e e	e X X n n n n	n n X X d d d

TABLE 16 Crew Scheduling with 12-Hour Shifts and Four 4-Hour Maintenance Shifts

Crew	S	M	T	W	T	F	S
1	X	[*d*/*D*]	[*D*/*d*]	X	X	D	D
2	D	X	X	D	D	X	X
3	X	N	N	X	X	[*n*/*N*]	[*N*/*n*]
4	N	X	X	N	N	X	X

TABLE 17 Crew Schedule with Eight-hour Shifts and One Eight-hour Maintenance Shift

Crew	S	M	T	W	T	F	S
1	X	X	d	d	d	d	d
2	d	X	X	e	e	e	e
3	e	e	e	X	X	n	n
4	n	n	n	n	n	X	X

for moving the start of the week in single-shift scheduling to be Sunday rather than Monday was one such instance, and the analysis of the cost of having paired off days, with a maximum work stretch of six days, was another. Multishift crew scheduling presented a third case of having an impossible contract signed. A mining company agreed that all overtime would be voluntary and all workers would work exactly 40 hours each week (lunch and other breaks occurred within the 40 hours). The company thought that they could change to 12-hour shifts to solve the problem. Although scheduling with extended length shift will be discussed in Section 4.4, for the sake of completeness, crew scheduling using 12-hour shifts will be included in this section. In particular, schedules for the mining industry will be discussed because it is often possible to work partial crews for a short period of time and still keep the above-ground processes working. The goal in doing this is to eliminate the overtime for the workers. When there is a partial crew working, other workers can do maintenance on the mine tunnels not in use. The methods presented are abstracted from Burns (1983).

With 12-hour schedules, the problem of working an average of 40 hours per week is difficult because 40 hours for one week, and even 80 hours for two weeks, are not divisible by 12. There are many ways of resolving this issue, and for the following examples the schedules will have six shifts of 12 hours, and one shift of 8 hours every two weeks, which has all workers average 40 hours per week every two weeks. As before, d and n are used for 8-hour day and night shifts. D and N are used for the 12-hour day and night shifts respectively. A new piece of notation is needed.

$[x/y]$ means that the crew is divided in half and the first half works shift x and the second half works shift y. If $x = d$ and $y = D$, then the first half of the crew will work an 8-hour shift and have 4 hours off while the other half of the crew is working a 12-hour shift. Table 16 is just one of many schedules that can be created for the mining industry. Each crew (person) works exactly 80 hours in each two-week schedule. On Monday, Tuesday, Friday, and Saturday there is a 4-hour period with only half a crew working.

The times of the 8-hour shifts can be moved to have two maintenance shifts of eight hours each by giving one day in every second week where there is an 8-hour shift for both halves of the crew at the same time, instead of where it is in Table 16.

The 12-hour schedule in Table 16 solves the overtime problem for the mine, but not the condition that each worker must work exactly 40 hours per week. It is possible to use 8-hour shifts and to have one full maintenance shift every Monday where no regular crew is working. In this case, everyone works exactly 40 hours per week and the overtime can be voluntary. The essence of the solution is that the reduced production happens on only one of the 21 shifts in a week. Table 17 illustrates this schedule, which is a slight modification to the schedule in Table 13. Note that on Monday there is no crew scheduled to work on the day shift. Voluntary overtime could happen on that shift.

4.3. Multiple Eight-Hour Shift Scheduling for Individuals

In addition to the requirements of the single-shift scheduling problem, there are the difficulties outlined in Sections 3.1 and 3.2.2. With the crew scheduling problems the goals included having shifts rotate forward and having three days off after completing night shifts. The methods of this section do not include these goals, because the need for workers is different for different shifts. In the research literature there are two approaches to the problem. The first, by Burns (1981), is an iterative algorithm for a very general shift scheduling problem involving eight-hour shifts. This paper presents an optimal, linear time algorithm to schedule the following problem:

- There are P shifts of length eight hours each day and the start times may overlap.
- The demand N_j per shift j, for weekdays Monday to Friday, is met and the demand n_j per shift j, for weekends Saturday and Sunday, is met, with $N_j \geq n_j \geq 0$ for all shifts $j = 1, 2, \ldots P$.
- Each employee is given at least A out of B weekends off.
- Each employee works exactly five out of seven days in each week, with the week defined as Sunday to Saturday.
- Each employee works no more than six consecutive days.
- Each employee who works two consecutive weekends has a pair of consecutive days off during the week.
- Each employee has at least one day off when changing shifts. Because shifts can have overlapping start times, the algorithm yields at least $(24 - t)$ hours off between shift changes, where, if the completion time of the last shift on one day is less than or equal to the start time of the first shift on the next day, t is the negative of the number of hours difference, otherwise t is just the difference in hours.

After the algorithm is described, ways of extending the use of the algorithm to more problems will be presented.

The first steps of the algorithm are almost identical to the modified Burns and Carter (1985) algorithm of Section 4.1. Since the demand pattern has been restricted to being the same for each weekday and the same, but of a different value, for each weekend day, rather than totally variable, the extra condition of having pairs of adjacent days off for the type $T4$ workers can be incorporated. Surprisingly, no change in the size of the workforce is required for the multiple shift problem, and the same bounds can be used.

Step 1. Compute the minimum workforce size. Set

$$n = \sum_{j=1}^{P} n_j, \text{ and } N = \sum_{j=1}^{P} N_j$$

and use n as the demand on weekends and N as the demand on weekdays to calculate the bounds on the workforce W as in the Burns and Carter (1985) algorithm.

Step 2. Schedule the weekends off. Schedule the weekends off as in the Burns and Carter (1985) algorithm.

Step 3. Determine the additional off-day pairs. Select n pairs of off-days cyclically from the list (Mon., Tue.), (Wed., Thu.), (Thu., Fri.), (Tue., Wed.), (Mon., Fri.).

Step 4. Assign off-day pairs in week 1. Use the categories of workers from the Burns and Carter algorithm. Assign $|T4|$ pairs of adjacent off-days to employees of type $T4$. Associate each employee of type $T3$ with an employee of type $T2$. From the list of pairs of off-days not yet assigned, assign the earliest day of each pair to a type $T3$ employee and the latest day to the associated type $T2$ employee.

Step 5. Assigning shifts in week i, ($i \geq 1$). For each week of the schedule, a total of n shifts are required on the weekends and N shifts on the weekdays. These requirements are met by forming n chains [Sun.–Sat.] and N–n chains [Mon.–Fri.] that are pair-wise distinct. A [$d1$–$d2$] chain, for $d1 < d2$, is a sequence of $d2 - d1 + 1$ workdays, taken from one or more employees. Two chains are pair-wise workday disjoint if no employee workday is a member of both chains. A [$d1$–$d2$] chain is formed by starting with an employee who works on day $d1$. This employee's workdays are assigned to the chain until either day $d2$ is reached or the employee has a day dk off, ($d1 < dk < d2$). In this latter case, the chain is transferred to a new employee who works on day dk but had day $dk - 1$ off. This new employee's workdays are then assigned to the chain. This process is continued until day $d2$ is reached.

Construct the (N–n) chains [Mon.–Fri.], first using employees of type $T1$ and pairs of type $T2$ and $T3$ employees who are associated with adjacent off-days. The type $T1$ employees form a [Mon.–Fri.] chain by themselves, while a [Mon.–Fri.] chain started with a type $T2$ employee can be completed by its associated $T3$ employee.

The n chains [Sun.–Sat.] are constructed once the current schedule has been temporarily modified so that exactly n workers are available each day, where ''available'' means an employee does not have the day off and has not already been assigned to a [Mon.–Fri.] chain. Sunday and Saturday already have exactly n workers available, but the weekdays may have more than n. Remove the excess workers for each weekday, starting with Monday, by temporarily giving an employee an extra day off as needed, if that employee would have started working that day. An employee would have started working on a particular day if he or she is not scheduled to have the day off and if, on the previous day, he/she had an actual day off or a designated extra day off. Then form the n chains [Sun.–Sat.], starting with any employee who is scheduled to work on Sunday, in the same way that [Mon.–Fri.] chains were formed.

The chains must now be assigned to shifts. Prior to this, however, it is necessary to associate the extra days off with chains so that they will also be assigned to shifts. Start with any extra days off that are immediately followed by a workday that is assigned to a chain. Associate this extra day off and any extra days off that precede it, back to an actual day off, with that chain. Once this is done, the only extra days off remaining are those that fall between two actual days off. Associate these extra days off with any [Sun.–Sat.] chain at random.

The (N–n) chains [Mon.–Fri.] and the n chains [Sun.–Sat.], with any extra associated workdays, must now be assigned to shifts. If it is week 1, assign the [Sun.–Sat.] chains to the P shift types according to n_j, the number of employees required for each shift of type j. If it is week i, ($i \neq 1$), assign the [Sun.–Sat.] chains to the same shift as the Saturday preceding the Sunday. In either case, assign the [Mon.–Fri.] chains to the shifts required to make the N_i–n_i weekday chains.

Step 6. Assigning off-day pairs in week i, ($i \neq 1$). Assume that weeks 1 up to ($i - 1$) have already been scheduled. As in step 4, each employee can be categorized into one of four types $T1$, $T2$, $T3$, and $T4$.

For each employee who is type $T4$ in week i, who was of type $T4$ in week ($i - 1$), assign the same pair of off-days that he or she received in week ($i - 1$).

For each employee who is of type $T4$ in week i, who was of type $T2$ in week ($i - 1$), and who was associated with a pair of adjacent off-days, assign both days of that pair in week i.

For each employee who is of type $T4$ in week i, who was of type $T2$ in week $(i - 1)$, and who was associated with a pair of nonadjacent off-days in week $(i - 1)$, trade pairs of off-days with an employee who was of type $T4$ in week $(i - 1)$ but is of type $T3$ in week i.

For each employee who is of type $T3$ in week i, assign the same pair of off-days that he or she received one off-day from in week $(i - 1)$, or the interchanged pair from above. The type $T3$ employees get the earliest day of the pair, while the type $T2$ employees get the remaining latest days off.

Go to step 5.

Example 3: Multiple-Shift Eight-Hour Schedules:

A company has a seven-day-a-week operation with four shifts working each day, with the following demand for workers.

Shift	Time	Weekdays	Weekends
1	Midnight to 8 a.m.	1	1
2	8 a.m. to 4 p.m.	2	2
3	1 p.m. to 9 p.m.	3	2
4	4 p.m. to midnight	1	1

Each worker is to receive at least one weekend off every three weeks; the maximum work stretch is to be six days; any worker working both weekends of a week will have a pair of adjacent days off in the week.

Step 1: $N = 7$ the daily demand on a weekday, and $n = 6$ the daily demand on the weekends.

$$L_1 = \left\lceil \frac{3 * 6}{2} \right\rceil = 9,\ L_2 = \left\lceil \frac{5 * 7 + 2 * 6}{5} \right\rceil = 10$$

Therefore, $W = 10$.

Step 2: Table 18 shows the results of step 2.

Step 3: Choosing six pairs of days from the list gives the following pairs: (Mon., Tue.), (Wed., Thu.), (Thu., Fri.), (Tue., Wed.), (Mon., Fri.), (Mon., Tue.).

Step 4: There are two type $T4$ workers every week (workers 9 and 10 in week 1). Assign the first two pairs from the list to workers 9 and 10. Associate workers 5&1, 6&2, 7&3, and 8&4, assigning the next four pairs from the list to get the weekday assignment for week i in Table 18.

Step 5: In this example, N is one bigger than n so one chain [Mon.–Fri.] needs to be constructed. Fill in this chain with a string of the letter A in Table 19 and then fill in the extra shifts with the letter Y to get the result in Table 19.

Completing step 5 using the letters B, C, D, E, F, G for the six chains, starting with workers 5–10, yields Table 20. The shifts assigned would be $B = 1$, $C = 2$, $D = 3$, $E = 4$, $F = 2$, $G = 3$, and $A = 3$, and the extra shifts would be removed to give the schedule after week 1.

In step 6 for week 2, the new type $T4$ workers are workers 3 and 4. Worker 3 is associated with worker 7 and assigned to the off-day pair (Mon., Fri.), so he or she must be interchanged with worker

TABLE 18 Multiple Eight-Hour Shifts: After Steps 1–4

	S	M	T	W	T	F	S		S	M	T	W	T	F	S
1	X					X									X
2	X			X											X
3	X					X									
4	X		X												
5					X		X		X						
6			X				X		X						
7		X					X		X						
8		X					X		X						
9		X	X												X
10				X	X										X

TABLE 19 Multiple Eight-Hour Shifts: After the First Part of Step 5

	S	M	T	W	T	F	S	S	M	T	W	T	F	S
1	X	A	A	A	A	A	X							X
2	X			X	Y	Y								X
3	X					X								
4	X		X	Y										
5					X	A	X	X						
6			X				X	X						
7		X					X	X						
8		X					X	X						
9		X	X											X
10				X	X									X

9, who became a type $T3$ worker. The new associations are: workers 9&7 assigned to pair (Mon., Fri.), worker 3 assigned to pair (Mon., Tue.), worker 4 assigned to pair (Mon., Tue.), and worker 8, who was associated with worker 4, joins the association 10&8 assigned to pair (Wed., Thu.). Table 21 gives the completed schedule for the first two weeks.

The algorithm works for situations where the shift demand on Monday to Friday is at least as big as on the weekends. For situations where the weekend demand is larger, the algorithm can still be used if part-time workers are hired to reduce the weekend demand to the required level. Any problem where the frequency of weekends off is greater than or equal to one off in two weeks will have no type $T4$ workers. Step 4 of the algorithm illustrates that clearly. Hence, for these cases the need for adjacent pairs disappears. One would think that the list could be modified, as was done above for the single-shift Burns and Carter (1985) algorithm, in order to reduce the number of single-day work stretches. Unfortunately, the proofs of optimality and feasibility require the existing list. All is not lost, though. Since the demand is constant for the week, it is relatively easy to modify the algorithm to ignore the list of off-day pairs and dynamically assign the days off to the type $T4$ workers as needed. Then assign the type $T3$ workers days off from Monday to Wednesday, and then assign days from Wednesday to Friday to the type $T2$ workers. A computer program can do this, along with the necessary checks to assure feasibility.

A second, completely different approach to the eight-hour scheduling problem was presented by Burns and Koop (1987). In this approach a master rotation is created rather than using an iterative algorithm. The method is optimal and works for the situation where there are three shifts a day, the weekday demand is at least as large as the weekend demand, the maximum length work stretch is six days, and each worker must have at least A out of B weekends off. A library of minischedules called modules is given, and the complete schedule is created by combining the modules. There are many situations where master shift rotations are either preferred or required; for example, in Canada, all nurse schedules and all prison guard schedules must be master rotations.

Rather than the methods of Burns and Koop (1987), a slightly different method that also uses modules to build optimal master schedules will be presented. In Burns (1985) the modules have two extra properties: the number of shift rotations is controlled and the workers receive exactly A out of B weekends off rather than the maximum number off that is possible. These modules have been used extensively to create schedules in hospitals and prisons.

TABLE 20 Multiple Eight-Hour Shifts: After Step 5

	S	M	T	W	T	F	S	S	M	T	W	T	F	S
1	X	3	3	3	3	X	5							X
2	X	2	2	X	3	3	3							X
3	X	4	4	4	4	X	2							
4	X	3	X	1	1	1	1							
5	1	1	1	1	X	3	X	X						
6	2	2	X	2	2	2	X	X						
7	3	X	2	2	2	2	X	X						
8	4	X	3	3	3	3	X	X						
9	2	X	X	3	3	3	3							X
10	3	3	3	X	X	4	4							X

TABLE 21 Multiple Eight-Hour Shifts: First Two Weeks

	S	M	T	W	T	F	S	S	M	T	W	T	F	S
1	X	3	3	3	3	X	2	2	2	2	2	X	3	X
2	X	2	2	X	3	3	3	3	3	X	4	4	4	X
3	X	4	4	4	4	X	2	2	X	X	2	2	2	2
4	X	3	X	1	1	1	1	1	X	X	2	2	2	2
5	1	1	1	1	X	3	X	X	3	3	3	3	X	4
6	2	2	X	2	2	2	X	X	2	2	X	3	3	3
7	3	X	2	2	2	2	X	X	1	1	1	1	X	3
8	4	X	3	3	3	3	X	X	3	3	3	X	1	1
9	2	X	X	3	3	3	3	3	X	3	3	3	3	X
10	3	3	3	X	X	4	4	4	4	4	X	3	3	X

The module library is separated into different categories, such as 8-hour vs. 12-hour shift. Even greater separation is created. Some modules are for people who rotate shifts, and some are for people who work a fixed shift. Within a workplace both types of schedules are often needed. In addition, there may be a group of people that want specific characteristics for their schedule, and these characteristics may differ for other groups.

The whole library of modules is too extensive to give in this work, but a sample will be given, along with enough information to show how to create modules. Having seen a sample, the reader should find it relatively easy to begin creating his or her own modules for different situations.

The reason that modules work so well in practice is that given a specific situation, there are often only a very small number of possible schedules. In addition, constructing the modules as outlined below means that the whole system works like building blocks for the complete schedule.

Consider situations such as nurse scheduling. The following is a set of basic requirements that apply in many hospitals: employees are to have one weekend off in three weeks; the maximum work stretch is six days; shift rotations to a shift that starts at an earlier time than the current one are to be done only after at least one day off; weekends off cannot start after an evening or night shift on Friday, where the night shift is the last shift of the day; employees are to work at least 50% of their time on day shift; there must be no single-day work stretches; there must be at least two days off after working a stretch of night shifts. The last restriction is necessary because if someone worked the night shift on a Tuesday, which ended, say, at 7:30 Wednesday morning, then had Wednesday off during which he or she spent most of the day sleeping, and then returned to work on the next day at 7:30 a.m. on Thursday, he or she would not really have had a very useful day off. As shown in Section 4.1, having all adjacent days off increases the number of nurses needed and reduces the possible schedules to two, which is not much choice.

The first example uses all the basic requirements, plus allowing only one single day off in the three-week cycle. The expectation is that there should be more feasible schedules than when the restriction is to have all paired off-days, but in fact there are only two possible ways of giving the days off. Table 22 shows the two basic schedules, and Table 23 shows the four possible schedules for the day (d) and evening (e) shift. Since there are only two possible ways of giving the days off, Table 22 also shows that it is not possible to give anyone a Wednesday off. This fact should be kept in mind when negotiating or promising schedules. Note that if module 1 is followed by module 2, it should be possible to have two people working on each shift every day of the week, with the extra two people on Wednesday as expected. Table 24 gives a six-week cycle with the shift assignments that give the even coverage for the two shifts.

TABLE 22 Basic Eight-Hour Shift Modules with One Single Day Off in Three Weeks

Basic Module 1							Basic Module 2						
S	M	T	W	T	F	S	S	M	T	W	T	F	S
X	X						X				X		
	X	X									X	X	
		X				X						X	X
2	1	1	3	3	3	2	2	3	3	3	1	1	2

TABLE 23 Day/Evening Modules, e Hour Shifts, One Single Day Off

d/e Module 1							
	S	M	T	W	T	F	S
1	X	X	e	e	e	e	e
2	e	X	X	d	d	d	d
3	d	d	X	d	d	d	X
e	1	0	1	1	1	1	1
d	1	1	0	2	2	2	1

d/e Module 2							
	S	M	T	W	T	F	S
1	X	X	d	d	d	d	d
2	d	X	X	e	e	e	e
3	e	e	X	d	d	d	X
e	1	1	0	1	1	1	1
d	1	0	1	2	2	2	1

d/e Module 3							
	S	M	T	W	T	F	S
1	X	d	d	d	X	e	e
2	e	e	e	3	X	X	d
3	d	d	d	d	d	X	X
e	1	1	1	1	0	1	1
d	1	2	2	2	2	0	1

d/e Module 4							
	S	M	T	W	T	F	S
1	X	d	d	d	X	d	d
2	d	d	d	d	X	X	e
3	e	e	e	e	e	X	X
e	1	1	1	1	1	0	1
d	1	2	2	2	0	1	1

For the modules with day and evening shifts, d/e modules 1, 3, and 4 can be converted to the d/n modules 1, 3, and 4, with the evening shift replaced with a night shift. Because of the one day off at a shift change, d/e module 2 cannot be converted to a d/n module. When a schedule is made, the different building blocks can be used. If the nursing unit required one nurse on night shift, and two on evening shift, with all nurses working all three shifts, then module d/e 5 would be followed by a d/n module. The nine nurses required would each start on a different line of the schedule and rotate in the usual way. Even if there were only seven or eight nurses available, the same schedule would be used because there is no other feasible schedule, and in this case, as the schedule was worked from week to week, part-time nurses would need to be used to fill in the missing shifts. Because all modules start with a Sunday off and end with a Saturday off, the modules can be put together in any order without violating the scheduling criteria. If there are more people required for the shifts, more modules will be used. If there are people who only work days and nights and others who only work days and evenings, the modules can be separated into two different master rotations for the unit.

A different module library can be built for the same basic requirements but allowing two single days off in a three-week schedule. The technique is the same as before, with the modules beginning with the Sunday off and ending with the Saturday off. In between, the four remaining off-days are placed as one adjacent pair of days off and two single days off. Each day off should be on a different day from the others to give even worker coverage, and no one-day work stretches should be allowed. As with building the chains in the iterative method, the last day of the adjacent pair off and one of the single days off should be in different weeks but adjacent days so that a chain of evening shifts is formed. There are only four basic modules that lead to only a few d/e and d/n modules for this case. A schedule can be built using modules from both the one-single-day-off library and the two-single-days-off library to give different schedules.

TABLE 24 Day/Evening Module for a Six-Week Schedule

d/e Module 5							
	S	M	T	W	T	F	S
1	X	e	e	e	e	e	e
2	e	X	X	d	e	e	e
3	e	e	X	d	d	d	X
4	X	e	e	e	X	d	d
5	d	d	d	d	X	X	d
6	d	d	d	d	d	X	X
e	2	2	2	2	2	2	2
d	2	2	2	4	2	2	2

TABLE 25 Twelve-Hour Modules

D/N Module 1	D/N Module 2	D/N Module 3
S M T W T F S	S M T W T F S	S M T W T F S
X X X	X X X X	X X X
X X X X	X X X	X X X X

The iterative schedules given before can be easily computerized. The modular scheduling is much more complicated and involves using a rule-based expert system in the computer program, but the modular system is easy to do manually.

4.4. Multiple Mixed Shift Length Scheduling For Individuals

In the 1990s there were several papers published that dealt with four day work weeks, and work weeks that had either three or four days. As is usual with research, the first set of papers gained some ground with the problem but were not of much practical use. The papers by Hung (1991, 1993, 1994a,b), ignored the length of work stretch and often had work stretches much longer than six days. In addition, some of the work by Hung applies only when the frequency of weekends off is greater than or equal to one out of two, which means that there are no workers of type $T4$ requiring pairs of adjacent days off. Burns et al. (1998) give an algorithm for the single-shift problem that restricts the work stretch to be at most five days and uses a mix of three- and four-day weeks. Burns and Narasimhan (1999) give an iterative method for optimally solving the multiple-shift scheduling problem with a restricted length of work stretch and with either four- or three-day work weeks. The algorithm is easy to understand and implement.

In this work, we will address the much more usual case of scheduling 12-hour shifts. In Burns (1985) there are many different modules given for the 12-hour scheduling problem. Only a few are needed to illustrate both the method of scheduling and the difficulties that can arise. The first difficulty is how to balance the number of hours worked. As explained in the crew section earlier, a common way of balancing the hours is to have one 8-hour shift every two weeks. Table 25 illustrates two basic modules giving every second weekend off, having a maximum work stretch of three days, having no single days off unless in a module having two weekends in a row off, where a single day off is necessary, and having a minimum work stretch of two days. The 8-hour shift replaces one of the 12-hour shifts in the module. By using two modules, with an 8-hour day shift in one and an 8-hour night shift in the other, it is possible to hire a part-time worker for an eight-hour evening shift. Many hospitals use this type of schedule for nurse schedules.

For D/N module 1, any shift can be the 12-hour N, but for D/N module 2, the Friday of the second week cannot be a N shift and still have a ''good'' weekend off. For this reason, many places only use the D/N module 1. Depending on the preferences of the employees, the schedule can have two weeks of nights whenever needed, followed by two or more weeks of days or the day and night shifts can be interspersed within the module.

There is an inherent problem if only module 1 is used to create a schedule. The employees working the schedule are essentially divided into two subgroups: the first working the odd-numbered weeks in the schedule and the second working the even-numbered weeks. These two groups will never work together, as they are never scheduled to work on the same day, let alone the same shift. To avoid this separation of the work force, introducing some of module 2, (or some of module 3 if needed) into the schedule will ensure that the groups see each other.

If fewer people are needed on the weekends than on the weekdays, then as many of the D/N module 3 as are required should be used.

Example 4:

The following are the requirements for workers: two people on night shifts and four people on day shift on weekdays, and two on night shift and two on day shift on the weekends. Because six people are needed each weekday, six modules will be needed. Choosing D/N1 with all the shifts D followed by D/N 1 with all the shifts N, followed by D/N module 3 on days, gives the complete schedule when two people are started on each line.

Carefully placing the 8-hour shifts allows the schedule to be created as in Table 26, with no part-time workers required. Two people are started on each line of the schedule, and each person works exactly 80 hours in every two-week period.

A second type of module can be used where no 8-hour shifts are used. In this type, an extra 12-hour shift is given off every six weeks, rather than 4 hours off every two weeks. Any of the modules can be used to do this, as long as the 12-hour shift given off is at the start or end of a three-day

TABLE 26 Example 4, Schedule 1

S	M	T	W	T	F	S
X	D	D	X	X	d	D
D	X	X	D	D	X	X
X	N	N	X	X	n	N
N	X	X	N	N	X	X
X	D	D	X	D	e	X
X	X	X	D	D	D	X

work stretch. For module 3 above, the single day off would be moved to Thursday and the Friday given as the extra day off, thus creating an off-day stretch of six days. Table 27 gives the schedule for example 4, using only 12-hour shifts. In practice, the hours worked balance over a six-week period but not over the two-week pay periods. Some agreement on overtime rules needs to be negotiated before this approach can be used.

4.5. Hierarchical Workforce Scheduling

Consider the situation where there are m different categories of workers with a definite ranking. Type i workers can do the work of any worker at a lower level j but the reverse is not true. It can be assumed that workers of type i are paid at least as much as, and probably more than, workers of type j for $i > j$. There is a demand for a minimum number of each type of worker on each shift and a cumulative demand for the number of workers of type 1 to type k for each shift. In the hierarchical problem, the requirement is to calculate the minimum number of workers in the most economical mix of types and to give each worker two days off per week and A out of B weekends off. A schedule is to be created with a maximum work stretch of six days for all values of A and B. Workers who must work two consecutive weekends should have two adjacent days off in the week.

Emmons and Burns (1991) solved the case for a single shift and a constant demand for each day of the week. Other papers were published trying extend the work, but they either did not adhere to the six-day maximum work stretch or only worked with the easier case where A of B was greater than or equal to one out of two.

Narasimhan (1993) solved several very interesting problems. He extended the work of Emmons and Burns (1987) to the case where the demand for each category of worker is constant for the weekdays and has a different constant value for the weekend. His work did not require a restriction that the weekday demand be greater than or equal to the weekend demand, and there was no restriction of the frequency of weekends off.

In the same work, Narasimhan also solved the multiple 8-hour, hierarchical workers shift problem for any number of shifts, with the weekday demand greater than or equal to the weekend demand, and he also solved the 10-hour, four-day-a-week problem for the same labor demands.

The algorithms and calculations are complicated but can be computerized. They will not be presented here because they would take too much space and because in practice they are often not applicable. For example, in the health care industry there are different categories of employees that can provide different patient needs. These categories, and the number of workers of each type required per shift, usually fall into a hierarchy that would fit the requirements of this section. However, in many cases the workers belong to different unions and have different scheduling rules in the hours of work clauses of their contracts. Since a single schedule, using one set of rules, could not be used in these instances, separate schedules are made for each category of worker. In some situations, the schedules are made sequentially, with the remaining needs for employees in the next category re-

TABLE 27 Example 4, Schedule 2

S	M	T	W	T	F	S
X	D	D	X	X	D	D
D	X	X	D	D	X	X
X	N	N	X	X	N	N
N	X	X	N	N	X	X
X	D	D	D	X	X	X
X	X	X	D	D	D	X

flecting the schedules already constructed. Generally, the schedules are done independently, using one of the methods described in this work for the single category of workers.

5. COMPUTER SYSTEMS

Despite the all-pervasive use of computers in society today, computers are not used frequently for shift-scheduling methods. There are notable exceptions in airline crew scheduling and some other industries. However, many health care companies and most retail companies still do manual scheduling. One of the reasons for this situation is the separation of institutional functions by the use of different third-party software packages, each custom-built for one function only. The merging of these companies to integrate such systems has not occurred as it has in the case of personnel records, payroll, and shift scheduling. All three systems must be integrated in a major company to have the shift scheduling work well.

5.1. Personnel Interface

Personnel systems have much of the key information about employees that is needed by a shift-scheduling computer package. The unique identifier to be used, such as a Social Security number or an employee number, is just one such item. Other needed information is a person's seniority and qualifications. Employees often take courses to upgrade their qualifications and are then eligible for a higher-paying position, and this information must be transmitted to the scheduling system on a timely basis. If someone leaves a scheduling group of employees and a new employee replaces that person, the seniority of members in the group changes and hence the schedules may need to be changed. For example, in a hospital, a change in staff may mean that a nurse may be able to move from a schedule of straight night shifts to a day shift schedule.

In industries such as pulp and paper, employees are offered chances to fill in for a higher-paying position whenever an open shift becomes available. If the person refuses the position more than once, he or she is no longer eligible for the position. The status of each employee regarding each position is required by the scheduling system.

5.2. Contract Regulations

When contracts are being negotiated, analysis using the methods of this chapter is very valuable. Calculating the required workforce size for increases in the weekend-off frequency and for any decrease in the work stretch can be done along with the related costs. If changes, such as having all pairs of days off with a work stretch maximum of six days, are contemplated, the increase in cost can be calculated and the paucity of possible scheduling modules can be illustrated.

The bounds given by Burns and Carter (1985) give the workforce size as the maximum of the bound on the total shifts required and the weekend-off bound. If a contract is being negotiated that will increase the frequency of weekends off and the new bound for this increase is higher than the bound on the total shifts required in the old contract, extra employees will be working on the weekdays. What may happen is that the employer will reduce the number of people required on the weekend until the two bounds are equal and then will hire part-time workers for the weekends. Each time the weekend requirements are reduced, the total shift requirements for full-time employees are reduced by two shifts. When the adjustments are done to balance the two bounds, the result may mean that fewer full-time employees are required under the proposed contract changes and more part-time employees will be used, which may not be what the employees want or expect.

5.3. Payroll Interface

Any computer scheduling package needs an extensive database to keep the necessary information to transmit to the payroll program. One copy of the schedule should be posted the required number of weeks in advance, usually six weeks. The employees can then trade shifts with other employees and request the scheduler to allow these changes. If the requested changes violate the contract, they may still be granted, but extra premiums, such as overtime, will not apply. The schedule, as modified with requested and granted changes, will then be saved. Up until a fixed time, say two weeks before the schedule is actually worked, management can make some types of changes without incurring a payable penalty. The final schedule, as it is to be worked, is then saved. The actual schedule worked will be saved, along with any last-minute changes made by management to satisfy unexpected needs or absenteeism. The information transferred to the payroll package will come from this final schedule, with the pay premiums calculated based on when, and by whom, the changes were made. All the information must be archived for later reference.

6. MANUAL SYSTEMS

Manual shift scheduling systems are widely used and for the most part are time consuming and frustrating. No matter how good the schedule, some employees will express discontent. This basically

arises from the fact that all employees want to be home with family and friends at the same time and from the fact that different people have different ideas as to what makes a good schedule. Since many of the people creating the schedules have other tasks to do, the scheduling often gets done at home, on personal time, which is not a good situation for creating good schedules. Computer scheduling would free up the time of the person doing the scheduling and would help divert any unavoidable discontent of personnel to a machine and away from the scheduler.

An interesting result of linking the payroll and the scheduling system arose when Burns first computerized nurse scheduling in the 1980s. The head nurses were able to receive reports on the total hours worked by their full-time and part-time employees. Several nurses were discovered to be working seven days a week by doing part-time shifts in different units. This was deemed to be an unsafe work situation and changes were made to put a stop to the practice. Other work patterns can be detected once the information is put into a general payroll system rather than in a unit-by-unit shift-scheduling system.

Manual systems have a manual entry of data from the schedules worked into the payroll system. Because of the large amount of information that needs to be processed and transferred, as outlined in Section 5.2, errors are inevitable. In addition, extra people are employed just to do the data entry, an expense that can be eliminated with a computer system that is integrated with the payroll system.

For companies that will continue to use a manual system, several things can be done to improve the situation. The people doing the scheduling can be educated and trained to understand the complications of scheduling. The methods outlined in this chapter can be introduced to improve the schedules and decrease the time required to create the schedules.

REFERENCES

Baker, K. R. (1974), ''Scheduling a Fulltime Workforce to Meet Cyclic Staff Requirements,'' *Management Science,* Vol. 20, pp. 1561–1568.

Baker, K. R. (1976), ''Workforce Allocation in Cyclical Scheduling Problems: A Survey,'' *Operations Research Quarterly,* Vol. 27, pp. 156–157.

Brownell, W. D., and Lowerre, J. M. (1976), ''Scheduling of Work Forces Required in Continuous Operations under Alternate Labor Policies,'' *Management Science*, Vol. 22, pp. 597–605.

Burns, R. N. (1978), ''Manpower Scheduling with Variable Demands and Alternate Weekends Off,'' *INFOR*, Vol. 16, pp. 101–111.

Burns, R. N. (1981), ''An Iterative Approach to Multiple Shift Scheduling,'' Manuscript, School of Business, Queen's University, Kingston, ON.

Burns, R. N. (1983), ''Shift Scheduling at Denison Mines,'' BCW Consulting Ltd., Kingston, ON.

Burns, R. N. (1985), ''Shift Scheduling Modules,'' BCW Consulting Ltd., Kingston, ON.

Burns, R. N. (1999), ''Algorithms for Time Dependent Schedules,'' BCW Consulting Ltd., Kingston, ON.

Burns, R. N., and Carter, M. W. (1985), ''Work Force Size and Schedules with Variable Demands,'' *Management Science,* Vol. 31, pp. 599–607.

Burns, R. N., and Koop, G. J. (1987), ''A Modular Approach to Optimal Multiple Manpower Scheduling,'' *Operations Research,* Vol. 35, No. 1, pp. 100–110.

Burns, R. N., and Narasimhan, R. (1999), ''Multiple Shift Scheduling of Workforce on Four-Day Workweeks,'' *Journal of Operational Research Society,* Vol. 50, pp. 979–981

Burns, R. N., Narasimhan., R., and Smith, L. D. (1998), ''A Set-Processing Algorithm for Scheduling Staff on 4-Day of 3-Day Work Weeks,'' *Naval Research Logistics,* Vol. 45, pp. 839–853.

Emmons, H., and Burns, R. N. (1991), ''Off-Day Scheduling with Hierarchical Worker Categories,'' *Operations Research,* Vol. 39, No. 3, pp. 484–495.

Glover, F., and McMillan, C. (1986), ''The General Employee Scheduling Problem: an Integration of MS and AI,'' *Computer and Operations Research,* Vol. 13, pp. 563–573.

Henderson, W. B., and Berry, W. L. (1976), ''Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis,'' *Management Science,* Vol. 22, pp. 1372–1380.

Hung, R. (1991), ''Single Shift Workforce Scheduling under a Compressed Workweek,'' *OMEGA,* Vol. 19, pp. 494–497.

Hung, R. (1993), ''Three-Day Workweek Multiple-Shift Scheduling Model,'' *Journal of Operational Research Society,* Vol. 44, pp. 141–146.

Hung, R. (1994a), ''Multiple-Shift Workforce Scheduling under the 3–4 Day Workweek,'' *Management Science,* Vol. 40, No. 2, pp. 280–284.

Hung, R. (1994b), ''A Multiple-Shift Workforce Scheduling Model under the 4-Day Workweek with Weekday and Weekend Labor Demand,'' *Journal of the Operational Research Society,* Vol. 45, No. 9, pp. 1088–1092.

Narasimhan, R. (1993), ''Optimal Workforce Shift Scheduling,'' Ph. D. thesis, School of Business, Queen's University, Kingston, ON.

Segal, M. (1974), ''The Operator Scheduling Problem: Network Flow Approach,'' *Operations Research,* Vol. 22, pp. 808–823.

Selkirk, C. G. (1999), ''Optimal Algorithms for Single Shift Workforce Scheduling to Avoid One Day Work Stretches in a Cyclic Schedule,'' Ph. D. thesis, School of Business, Queen's University, Kingston, ON.

Tiberwala, R., Phillippe, D., and Browne, J. (1972), ''Optimal Scheduling of Two Consecutive Idle Periods,'' *Management Science,* Vol. 19, pp. 71–75.