COMPRESSED WORKWEEK SCHEDULING WITH DAYS-OFF CONSECUTIVITY, WEEKEND-OFF FREQUENCY, AND WORK STRETCH CONSTRAINTS

HESHAM K. ALFARES

Systems Engineering Department
King Fahd University of Petroleum & Minerals
PO Box 5067, Dhahran 31261
Saudi Arabia
e-mail: hesham@ccse.kfupm.edu.sa

ABSTRACT

This paper presents a three-day workweek scheduling problem with four types of days-off scheduling constraints: (1) out of the four off days given to each employee per week, at least two must be consecutive, (2) two alternative types of constraints are imposed to ensure that employees get a sufficient proportion of weekends off, (3) the maximum work stretch is four consecutive workdays for each employee, and (4) the maximum number of successive weeks with weekend work is specified. A mathematical model of this problem is formulated and efficiently solved by introducing a bound on workforce size. First, the problem structure is used to determine the minimum workforce size. Subsequently, a workforce-size constraint is added to the integer programming model in order to facilitate solution. Finally, multiple-week rotation schedules are produced to satisfy all constraints and limit or minimize the number of successive weeks with weekend work.

Keywords: Employee scheduling, compressed workweek, integer programming

INTRODUCTION

Employee scheduling is an important and complex practical problem. This problem is especially significant for facilities that operate continuously, either 24 hours a day, or seven days a week, such as hospitals, restaurants, and train stations. If the organizations operate seven days a week, then different employees must be given different days-off, some of which do not necessarily correspond to the weekend. The problem in this case is called days-off scheduling, and its objective is to find the minimum size or cost of the workforce to satisfy the labor requirements for every day of the week.

In days-off scheduling literature, the (x, y) notation denotes the problem of assigning each employee x workdays out of a total period of y days. Initially, most of days-off scheduling literature was directed towards the (5, 7) problem, or the traditional 8-hour-per-day, 5-workday-per week schedule. Recently, there has been a worldwide trend towards compressed and flexible work schedules. This trend comes as a result of changing social values, demographic trends, and economic factors (McCampbell, 1996). McCampbell (1996) reports a phenomenal growth of alternative work schedules in the U.S, and provides a list of alternative work schedules suggested by the U.S. Office of Personnel Management. This list includes three compressed-schedule modules: the 3-day workweek, the 4-day workweek, and the 5-4/9 plan.

Numerous variations of the employee days-off scheduling problem have been considered, and several approaches have been proposed for classifying these variations. Narasimhan (2000) lists 6 factors that define the problem structure: (1) number of shifts, (2) number of employee categories, (3) pattern of labor demand, (4) existence of limits on the length of work stretches, (5)

Recd. Nov. 2005, Rev. Mar. 2006, Acc. May 2006

existence of requirements for weekend off frequency, and (6) number of workdays per week. To this list, we propose adding a seventh factor, which is the existence of requirements for consecutive work/off days.

The days-off scheduling problem considered in this paper involves a three-day workweek, or a (3, 7) problem. It is a practical problem focusing on an actual compressed work schedule suggested by the U.S. Office of Personnel Management. In terms of the 7-factor classification introduced above, the problem at hand is characterized by: (1) a single shift per day, (2) a single employee category, (3) a general pattern of labor demand, (4) a limit of 4 days on the length of work stretches, (5) two alternative requirements for weekend off frequency, (6) three workdays per week, and (7) at least two of the four off days must be consecutive.

Two types of weekend-off frequency constraints are considered in order to ensure that employees get a sufficient amount of weekend rest: (i) each employee must get a minimum proportion of *full weekends* off, or (ii) each employee must get a minimum proportion of *weekend days* off. An integer programming (IP) model will be formulated to represent the weekend-off frequency and work stretch length constraints. An efficient optimization algorithm will be developed to solve this model in order to minimize the number and cost of the workforce.

The algorithm starts by utilizing the problem structure to determine the minimum workforce size. This workforce size bound is then used in a workforce-size constraint, which is added to the IP formulation of the problem to make it possible to efficiently obtain optimum solutions. Finally, multiple-week rotation schedules are constructed to guarantee that work patterns in each week fit the work patterns in the following week while satisfying all the constraints. The computational efficiency of the new method is compared to direct integer programming solution.

Subsequent sections are organized as follows. First, a review of relevant literature is given. Then, the assumptions and the integer programming model of the given days-off scheduling problem are presented. Subsequently, the procedures for determining the minimum workforce size and assigning workers to days-off shifts are described. Next, computational comparisons are presented. Finally, a numerical example is solved and conclusions are given.

REVIEW OF RELATED RESEARCH

A recent review of employee scheduling literature is provided by Ernst et al. (2004). The focus here is on recent employee day-off scheduling models and algorithms, especially for compressed and flexible work schedules. Earlier approaches to compressed workweek scheduling are covered by Hung (1995), who surveys their applications for a police workforce.

Narasimhan (1997) considers days-off scheduling for a hierarchical workforce, assuming the following for each employee: (i) two days off per week, (ii) at least A out of B weekends off, and (iii) no more than 5 consecutive working days. Narasimhan (1997) aims to minimize the workforce cost, given that d_k category k workers are required on weekdays, and n_k workers on weekends. Billionnet (1999) uses integer programming to formulate and efficiently solve a hierarchical workforce scheduling problem under variable labor demand. The workweek length can be 3, 4, or 5 days, and the objective is to minimize the workforce cost.

Burns et al. (1998) present an algorithm for 3-day and 4-day workweeks, with variable daily demand and limits on work stretch lengths. Burns and Narasimhan (1999) present a multiple-shift algorithm for 3-day and 4-day workweek scheduling of a homogeneous workforce. The demand has two levels (D on weekdays and E on weekends, $D \geq E$), and constraints are imposed on weekend work frequency, work stretch length, and shift transition times. Narasimhan (2000) considers the same problem but for a hierarchical workforce consisting of several employee categories.

Lankford (1998) describes an actual pilot implementation of a compressed workweek schedule, consisting of four ten-hour work days (4×10) at the Analytical Central Call Management (CCM) group at Hewlett Packard. After careful evaluation, he found that the key to successful implementation of this schedule is cross-training. Alfares (2003) develops optimum solutions

for compressed workweek scheduling in which workers are given three or four consecutive workdays per week. The labor demand is assumed to have two different levels: D for weekdays and E for weekends.

Bard et al. (2003) use large-scale integer programming to formulate and solve a tour scheduling problem to minimize labor cost at the United States Postal Service. Azmat and Widmer (2004) develop a three-step procedure to assign days-off schedules to a minimum workforce using annualized hours. Jarray (2005) presents a three-step polynomial-time algorithm to allocate days-off to employees such that each employee gets two or three consecutive days off per week. Hung (2005) calculates cost savings obtained by 4-day and 3-day workweeks, provides formulae for calculating workforce sizes, and presents methods to construct days-off schedules.

Hung (1993) presents a multiple-shift algorithm for 3-day workweek scheduling, assuming: (i) D workers are required on weekdays and E workers on weekends, and (ii) each worker must receive at least A out of B weekends off, and (iii) cost is equal for all workdays/patterns. Alfares (2000) develops a single-shift manual optimization method for 3-day workweek scheduling, assuming: (i) varying labor demands for each day of the week, (ii) unequal costs for different work patterns, and (iii) consecutive work days. This paper presents a novel solution algorithm to a new 3-day workweek scheduling problem. The assumptions of this new problem are presented in the following section.

PROBLEM ASSUMPTIONS AND FORMULATION

Assumptions

The optimization algorithm presented in this paper applies to single-shift three-day workweek scheduling under the following assumptions:

(1) each employee is assigned 3 workdays and 4 off-days per week,
(2) at least 2 of the 4 weekly off-days must be consecutive,
(3) the maximum work stretch is 4 consecutive workdays, and
(4) two alternative weekend work frequency constraints are required:
    (I) a minimum proportion of *full weekends* off, or
    (II) a minimum proportion of *weekend days* off.
(5) the maximum number of successive weeks with weekend work is specified,
(6) the objective is to minimize the total cost of the workforce,
(7) the demand for employees may vary from day to day for the given week, but is constant for similar days of different weeks,
(8) a rotation schedule is used to ensure fairness among employees,
(9) the scheduling period is equal to the length of the rotation cycle, which is an integer number of weeks to be determined by the algorithm,
(10) weekend work is paid at overtime rate, thus the weekly cost of each days-off pattern depends on the number of weekend workdays it contains.

The Integer Programming Model

The integer programming model of the three-day scheduling problem at hand, with type I weekend work frequency constraints (minimum proportion P of *full weekends* off), is shown below.

$$\text{Minimize } Z = \sum_{j=1}^{35} c_j x_j \tag{1}$$

Subject to

$$\sum_{j=1}^{35} a_{ij} x_j \geq r_i, \qquad i = 1, 2, \ldots 7 \tag{2}$$

$$\frac{\sum_{j\in J_2} x_j}{\sum_{j=1}^{35} x_j} \geq P \tag{3}$$

$$x_1 \leq MQ_1 \tag{4a}$$

$$x_1 \geq Q_1 \tag{4b}$$

$$\sum_{j\in E} x_j \leq MQ_2 \tag{5a}$$

$$\sum_{j\in E} x_j \geq Q_2 \tag{5b}$$

$$Q_3 \leq \frac{1}{2}(Q_i + Q_2) \tag{6}$$

$$Q_3 \geq Q_1 + Q_2 - 1 \tag{7}$$

$$\sum_{j=1}^{35} x_j - \sum_{j\in E} x_j - x_1 \geq Q_3 \tag{8}$$

$$x_4 \leq MQ_4 \tag{9a}$$

$$x_4 \geq Q_4 \tag{9b}$$

$$\sum_{j\in J_0} x_j \leq MQ_5 \tag{10a}$$

$$\sum_{j\in j_0} x_j \geq Q_5 \tag{10b}$$

$$Q_6 \leq \frac{1}{2}(Q_4 + Q_5) \tag{11}$$

$$Q_6 \geq Q_4 + Q_5 - 1 \tag{12}$$

$$\sum_{j=1}^{35} x_j - \sum_{j\in J_0} x_j - x_4 \geq Q_6 \tag{13}$$

$$x_j \geq 0 \text{ and integer}, \qquad j = 1,2,\ldots,35 \tag{14}$$

$$Q_u = 0 \text{ or } 1, \qquad u = 1,2,\ldots,6 \tag{15}$$

where

W = workforce size, i.e., total number of employees assigned to all patterns
x_j = number of employees assigned to weekly days-off work pattern j
Q_u = binary logical variables used to represent work stretch constraints
$a_{ij} = 1$ if day i is a work day for pattern j, $a_{ij} = 0$ otherwise ($i = 1,2,\ldots,7$).
$i = 1,\ldots,5$ correspond to weekdays, $i = 6,7$ correspond to the weekend.
$i = 8$ corresponds to weekend-off frequency constraint (3).
Table 1 shows matrix $A = \{a_{ij}, i = 1,\ldots,8, j = 1,\ldots,35\}$
c_j = weekly cost of days-off pattern j shown in Table 1, assuming daily pay for regular workdays = 1, and daily pay for weekends = $1 + \beta$ ($\beta \geq 0$)
r_i = minimum number of employees required on day i, $i = 1,2,\ldots,7$, $r_8 = 0$
P = average proportion of full weekends off, $0 \leq P \leq 1$
M = a large number, $M \geq W$
E = set of days-off patterns in which both days 1 and 2 are workdays, $E = \{3,4,9,18,23,33\}$

Table 1: Days-off matrix $A = \{a_{ij}\}$ and cost vector $C = \{c_j\}$ for the 35 days-off work patterns

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
i																		
1	0	1	1	1	0	0	0	1	1	0	1	0	0	0	1	0	1	1
2	0	0	1	1	1	0	0	0	1	1	0	1	0	0	0	1	0	1
3	0	0	0	1	1	1	0	0	0	1	1	0	1	0	0	0	1	0
4	0	0	0	0	1	1	1	0	0	0	1	1	0	1	0	0	0	1
5	1	0	0	0	0	1	1	1	0	0	0	1	1	0	1	0	0	0
6	1	1	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0
7	1	1	1	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0
8(I)	$-P$	$-P$	$-P$	$1-P$	$1-P$	$1-P$	$-P$	$-P$	$-P$	$-P$	$1-P$	$1-P$	$-P$	$-P$	$-P$	$-P$	$-P$	$1-P$
8(II)	$-2P$	$-2P$	$1-2P$	$2-2P$	$2-2P$	$2-2P$	$1-2P$	$1-2P$	$1-2P$	$1-2P$	$2-2P$	$2-2P$	$1-2P$	$-2P$	$1-2P$	$-2P$	$1-2P$	$2-2P$
c_j	$3+2\beta$	$3+2\beta$	$3+\beta$	3	3	3	$3+\beta$	$3+\beta$	$3+\beta$	$3+\beta$	3	3	$3+\beta$	$3+2\beta$	$3+\beta$	$3+2\beta$	$3+\beta$	3

Table 1: (*Concluded*)

j	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
i																	
1	0	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0
2	1	0	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0
3	1	1	0	0	0	1	1	0	0	1	0	0	1	0	0	1	1
4	0	1	1	1	0	0	1	1	0	0	1	0	0	1	0	0	1
5	1	0	1	0	1	0	0	1	1	0	1	1	0	0	1	0	0
6	0	1	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0
7	0	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0	1
8(I)	$1-P$	$-P$	$-P$	$-P$	$1-P$	$-P$	$-P$	$1-P$	$-P$	$-P$	$1-P$	$-P$	$-P$	$-P$	$1-P$	$-P$	$-P$
8(II)	$2-2P$	$1-2P$	$1-2P$	$1-2P$	$2-2P$	$1-2P$	$1-2P$	$2-2P$	$1-2P$	$-2P$	$2-2P$	$1-2P$	$-2P$	$1-2P$	$2-2P$	$1-2P$	$1-2P$
c_j	3	$3+\beta$	$3+\beta$	$3+\beta$	3	$3+\beta$	$3+\beta$	3	$3+\beta$	$3+2\beta$	3	$3+\beta$	$3+2\beta$	$3+\beta$	3	$3+\beta$	$3+\beta$

J_k = set of days-off patterns with k weekend days off per week, $k = 0, 1, 2$
$J_0 = \{1, 2, 14, 16, 28, 31\}$
$J_1 = \{3, 7, 8, 9, 10, 13, 15, 17, 20, 21, 22, 24, 25, 27, 30, 32, 34, 35\}$
$J_2 = \{4, 5, 6, 11, 12, 18, 19, 23, 26, 29, 33\}$

The aim of the objective function (1) is to minimize the total cost of employees. Constraints (2) ensure that the all daily demands are satisfied by assigning at least the required number of employees on any given day i. Weekend work frequency constraint (3) guarantees that the total schedule contains a minimum proportion P of *full weekends* off. For Alternative I, constraint (3) can be expressed as follows:

$$-P\sum_{j\in J_0} x_j - P\sum_{j\in J_1} x_j + (1-P)\sum_{j\in J_2} x_j \geq 0$$

Thus

$$\begin{aligned} a_{8j} &= -P \quad \text{if } j \in J_0 \text{ or } j \in J_1, \\ &= 1-2P \quad \text{if } j \in J_2 \end{aligned}$$

Work stretch constraints, which ensure that no more than 4 successive workdays are assigned as work patterns are linked from one week to the next, come in two sets: (4)–(8) and (9)–(13). Set (4)–(8) ensures that work pattern 1, which has 3 consecutive workdays at the end of a given week, is not immediately followed by any work pattern of the set E, in which both days 1 and 2 are workdays. Thus, if both pattern 1 and some pattern(s) belonging to set E are assigned, then at least one of the remaining patterns must be also assigned to be scheduled in between during the weekly rotation.

Constraints (4a) and (4b) guarantee that $Q_1 = 0$ if $x_1 = 0$ and $Q_1 = 1$ if $x_1 \geq 1$. Likewise, constraint pair (5a) and (5b) guarantee that $Q_2 = 0$ if all x_j $(j \in E) = 0$ and $Q_2 = 1$ if some x_j $(j \in E) \geq 1$. Constraints (6) and (7) represent the logical relationship $Q_3 = Q_1 Q_2$, ensuring that $Q_3 = 1$ if Q_1 and Q_2 are both equal to 1, otherwise $Q_3 = 0$. The value of Q_3, the right-hand side of constraint (8), will be fixed only by (4)–(7).

Similarly, (9)–(13) ensure that any work pattern of the set J_0, in which both days 6 and 7 are workdays, is not immediately followed by work pattern 4, which has 3 consecutive workdays at the start of a given week. Thus, if both pattern 4 and some pattern(s) belonging to set J_0 are assigned, then some of the remaining patterns must be also assigned to act as a buffer. The purpose of (4)–(8) and (9)–(13) will become clearer in light of the subsequent description of the rotation algorithm. Although constraints on the number of successive weeks with weekend work are not explicitly expressed in the IP model, they are implemented directly in the rotation algorithm.

THE SOLUTION ALGORITHM

Minimum Workforce Size

In order to minimize the workforce size W, we temporarily replace the objective function (1) by:

$$\text{Minimize } W = \sum_{j=1}^{35} x_j \tag{16}$$

The minimum workforce size $W = \sum_{j=1}^{35} x_j$ is determined by searching through the feasible region defined by all the constraints (2)–(15). However, the presence of the binary variables $Q_1, \ldots, Q_6$ in constraints (4)–(13) makes it difficult to directly analyze their effect on the workforce size. Therefore, for the sake of facilitating the analysis, these constraints are temporarily replaced (only at this stage of the algorithm) by the following simplified combination of (8) and (13):

$$\sum_{j\in F} x_j \geq 1 \tag{17}$$

where

F = set of all days-off patterns excluding sets E and J_0, i.e.,

$F = \{5, 6, 7, 8, 10, 11, 12, 13, 15, 17, 19, 20, 21, 22, 24, 25, 26, 27, 29, 30, 32, 34, 35\}$

It must be noted that although constraint (17) is simpler than the set of constraints (4)–(13), it is actually stronger because it makes the assignment to a buffer days-off pattern unconditional. Constraint (17) simply guarantees that at least one days-off pattern from the set F is always available to act as a buffer, regardless of the values of patterns 1 and 4 and patterns from sets E and J_0.

In order to determine the minimum workforce size W, the dual solution and primal-dual relationships are used to determine the lower bounds on W. First, all cyclically distinct subsets of the dual variables are enumerated by the Alfares and Selim (2002) cyclic selection algorithm, in order to identify the dominant dual solutions. Subsequently, primal-dual complementary slackness relationships are used to find the corresponding primal solutions. For the days-off scheduling problem at hand, four dominant dual solutions are identified, leading to the following lower bounds on the workforce size:

1. First, the workforce size must be greater than the labor demand on any given day; thus:

$$W \geq r_i, \qquad i = 1, 2, \ldots, 7 \tag{18}$$

   Since constraint (17) leads to the trivial bound ($W \geq 1$), it is dominated by (18). Thus, (17), and subsequently (4)–(13), do not affect the workforce size W as long as all daily labor demands are strictly positive, i.e., $r_i \geq 1$, for $i = 1, \ldots, 7$.

2. Second, since each employee is assigned 3 workdays per week, the total man-days assignment is $3\sum x_j = 3W$. This must be greater than the total man-days requirement, which is $\sum r_i$. Thus, $3W \geq \sum r_i$, or

$$w \geq \frac{1}{3}\sum_{i=1}^{7} r_i \tag{19}$$

3. For the third bound, we must define seven sets of three daily demands (r_i, r_{i+2}, r_{i+4}), whose subscripts, denoted by s_i, are shown in Table 2. Each set $s_i = \{i, i+2, i+4\}$, $i = 1, \ldots, 7$, is a circular set which has a cycle of 7. Referring to matrix A in Table 1, each x_j variable has at most two non-zero coefficients in set s_i rows $\{i, i+2, i+4\}$ for any $i = 1, \ldots, 7$. Therefore, the sum of rows $\{i, i+2, i+4\}$ in constraints (2) gives:

$$2\sum x_j \geq r_i + r_{i+2} + r_{i+4}, \quad i = 1, 2, \ldots, 7$$

$$W \geq (r_i + r_{i+2} + r_{i+4})/2, \quad i = 1, 2, \ldots, 7$$

or

$$W \geq \frac{T_i}{2}, \quad i = 1, 2, \ldots, 7 \tag{20}$$

4. The fourth bound is derived by combining weekend staffing demands with the required frequency of weekends off. Starting with Saturday (day 6), adding rows 6 and 8(I) of Table 1 produces: $(1-P)W \geq r_6$, or $W \geq r_6/(1-P)$. Using the same procedure for Sunday (day 7) and adding rows 7 and 8(I) of Table 1, we obtain: $W \geq r_7/(1-P)$. Combining the two results in order to satisfy work and off demands in the weekend (both days 6 and 7), we obtain:

$$W \geq \frac{max(r_6, r_7)}{1-P} \tag{21}$$

Table 2: Sets of subscripts s_i defined by equation (22)

i	s_i
1	1, 3, 5
2	2, 4, 6
3	3, 5, 7
4	1, 4, 6
5	2, 5, 7
6	1, 3, 6
7	2, 4, 7

The minimum workforce size W is obtained from the maximum value of the four above bounds (18)–(21), rounded up to the nearest integer, as follows:

$$W = \max\left\{ r_{\max}, \lceil \frac{1}{3}\sum_{i=1}^{7} r_i \rceil, \lceil \frac{T_{max}}{2} \rceil, \lceil \frac{\max(r_6, r_7)}{1-P} \rceil \right\} \tag{22}$$

where

$$r_{\max} = \max\{r_1, r_2, \ldots, r_7\}$$
$$T_{\max} = \max\{T_1, T_2, \ldots, T_7\}$$
$$\lceil a \rceil = \text{smallest integer} \geq a,$$

and

$$T_i = \sum_{j \in s_i} r_j = r_i + r_{i+2} + r_{i+4}, \qquad i = 1, 2, \ldots, 7 \tag{23}$$

s_i is circular set with a cycle = 7 (see Table 2)

Equation (22) is comparable to formulas for the minimum workforce size developed by numerous authors such as Emmons and Burns (1991), Hung (1991), Burns and Narasimhan (1999), Burns et al. (1998), and Narasimhan (1997, 2000). The details of these formulas vary according to the given problem contexts. However, their general form is similar to (22), and they are developed using similar approaches. All of these formulas are based on finding the lower limits on workforce size to satisfy different subsets of constraints. Since the maximum of these bounds satisfies *all* applicable constraints, it is the minimum feasible workforce size.

Days-Off Assignments

Having determined the number of employees W by (22), we need to allocate them among the different days-off patterns in order to satisfy the daily labor demands at minimum cost. Reverting from (16) to the original objective function (1), the right-hand side of the dual constraints changes from $\mathbf{1}^T$ to $\{c_1, \ldots, c_{35}\}^T$. Naturally, the values of the basic dual variables, as well as the slacks of the dual constraints, do slightly change. However, no change affects which dual variables are basic/nonbasic, or which dual constraints are equations/inequalities. Thus, according to primal-dual relations, no change affects the four bounds (18)–(21) on workforce size W. Therefore, using the differential cost vector $(c_1, \ldots, c_{35})$ with any $\beta \geq 0$, the minimum labor cost is obtained with the minimum workforce size. In order to efficiently find the optimum days-off assignments $x_1, \ldots, x_{35}$, the following constraint is appended to the integer programming model defined by (1)–(15).

$$\sum_{j=1}^{35} x_j = W \tag{24}$$

Multiple-Week Rotation

The constraints included in the model specify that each employee must receive: (i) a proportion P of weekends off, (ii) at least two successive days off per week, and (iii) no more than four consecutive workdays. A multiple-week rotation scheme is needed to ensure that all these constraints are satisfied as employees switch from one days-off pattern to another in successive weeks. A rotation scheme is presented below, which is based on the value of workforce size W obtained by (22) and the values of days-off assignments $x_1,\ldots,x_{35}$ obtained by the optimum IP solution.

The rotation scheme has a rotation cycle with a length of W weeks. During this cycle, each employee is assigned to each days-off pattern j for a period of x_j weeks ($j = 1,\ldots,35$). To obtain a fair schedule, each employee is assigned to the same cyclic sequence of days-off patterns. Specifically, employee k starts the sequence at week k of the W-week rotation cycle ($k = 1,\ldots,W$). In order to satisfy work stretch constraints, assignments to pattern 1 cannot be followed by patterns belonging to set E, and assignments to patterns belonging to set J_0 cannot be followed by pattern 4.

The rotation algorithm is used to restrict or minimize the length of the weekend work stretch. If the maximum number of consecutive weeks that involves weekend work is given as L, then days-off work patterns in successive weeks must be sequenced in such an order that avoids weekend work stretches longer than L. Specifically, any sequence of assignments to J_0 and J_1 patterns cannot exceed L successive weeks, and then it must be followed by at least a one-week assignment to a J_2 pattern. To guarantee feasibility, a pre-specified limit on weekend work stretch must satisfy $L \geq \lceil (1-P)/P \rceil$. However, since the actual proportion of J_2 patterns in the final solution can be greater than P, the minimum value of L is given by:

$$L = \left\lceil \frac{\sum_{j\in J_0} x_j + \sum_{j\in J_1} x_j}{\sum_{j\in J_2} x_j} \right\rceil \tag{25}$$

HALF-WEEKEND-OFF FREQUENCY CONSTRAINTS

In the preceding analysis, a minimum proportion P of *full weekends* off was required for each employee. In several situations, however, weekend-off requirements may include half-weekends. Cezik et al. (2001), for example, require that at least one of the off days in any week must be on the weekend. If we modify the weekend-off frequency constraints to specify a proportion P of *weekend days off* (both *half weekends* off and full weekends off), then the model changes as follows.

First, constraint (3) must be replaced by the following constraint.

$$\frac{\sum_{j\in J_1} x_j + 2\sum_{j\in J_2} x_j}{2\sum_{j=1}^{35} x_j} \geq P \tag{26}$$

or

$$-2P\sum_{j\in J_0} x_j + (1-2P)\sum_{j\in J_1} x_j + (2-2P)\sum_{j\in J_2} x_j \geq 0 \tag{26}$$

For weekend-off frequency constraints alternative II, the coefficients a_{8j} of row 8(II) in Table 1 become applicable:

$$\begin{aligned} a_{8j} &= -2P && \text{if } j \in J_0, \\ &= 1-2P && \text{if } j \in J_1, \\ &= 2-2P && \text{if } j \in J_2, \end{aligned}$$

The numerator of (26) represents the number of weekend man-days assigned off, while the denominator represents the total number of possible weekend man-days. In order to determine

the minimum workforce size W, the first three bounds of (17)–(19) are still applicable. However, in order to satisfy work and off demands on days 6 and 7, we now use row 8(II) instead of row 8(I) in Table 1. Consequently, the bound expressed by (21) changes to:

$$W \geq \frac{r_6 + r_7}{2 - 2P} \tag{27}$$

Therefore, (22) is replaced by the following expression for the minimum workforce size W.

$$W = \max \left\{ r_{\max}, \lceil \frac{1}{3} \sum_{i=1}^{7} r_i \rceil, \lceil \frac{T_{max}}{2} \rceil, \lceil \frac{r_6 + r_7}{2 - 2P} \rceil \right\} \tag{28}$$

The solution algorithm for alternative II weekend-off frequency constraints involves the following steps. First, the workforce size W is calculated by (28). Next, constraint (24) is added to the IP model defined by (1)–(2), (4)–(15), and (26). Finally, the same rotation scheme described for weekend-off constraints alternative I, is similarly used to schedule the W employees.

COMPUTATIONAL EXPERIMENTS

Extensive computational experiments have been carried out to test the computational efficiency of the proposed algorithm. The proposed method, i.e., IP model (1)–(15) plus workforce-size constraint (24), was compared to the traditional IP solution, i.e., IP model (1)–(15) without (24). The comparisons have shown that the addition of (24) significantly reduces the computation time. In order to evaluate the impact on computational efficiency, 252 test problems were solved, with and without constraint (24). In all these problems, the value of β was set to 0.5 to indicate 50% higher pay for weekend work. Moreover, the value of P was set to 0.5 to indicate a requirement of every other weekend off. Both of these are typical values used in real life and in several research papers.

The 252 test problems are divided into 12 sets with different demand types, but all have an average demand of 50 employees per day. The first six sets involve 17 problems each, with a demand range of 34 to 64. Sets 1-6 have different specific labor demand patterns: level, trend, concave, convex, unimodal, and sinusoidal. These six sets were originally developed by Brusco and Jacobs (1993) for tour scheduling, but were adapted to the days-off problem. The last six sets involve 25 problems each. Sets 7-10 have randomly distributed labor demands over the intervals: [34, 66], [0, 100], [20, 80], and [45, 55], respectively. Sets 11-12 involve two constant levels of labor demand: workdays demand ($r_1 = r_2 \ldots = r_5 = D$) which is fixed at 50, and weekends demand ($r_6 = r_7 = E$). Set 11 has $E = 25, \ldots, 49$, while set 12 has $E = 51, \ldots, 75$.

Microsoft Excel Solver®, on a 450-MHz Pentium III PC, was used to solve the 252 test problems by integer programming. The default Solver options were used (Max Time = 100, Iterations = 100, Precision = 0.000001, Tolerance = 5%, Convergence = 0.001). The results of computational experiments using weekend-off constraints alternative I (full weekends off) are summarized in Table 3. Without constraint (24), 135 problems (54%) could not be solved within the 100-second time limit. For these problems, the solution time was simply assumed to be 100 seconds.

Even though the solution time without constraint (24) were underestimated for 54% of the problems, the advantages of adding constraint (24) are extraordinary. Including constraint (24) has decreased the mean, maximum, and variance of solution times for all 12 problem sets. Overall, average solution time dropped by 120 times, from 55.02 seconds to 0.46 seconds. Maximum solution time fell from more than 100 seconds to only 1.65 seconds, while the standard deviation decreased from 49.43 seconds to only 0.22 seconds.

Table 4 summarizes results of the experiments with weekend-off constraints alternative II using the same 252 test problems. Without constraint (24), 154 problems (61%) could not be solved within the 100-second time limit. Again, the solution times for these problems were

Table 3: Integer programming solution times in seconds (minimum, average, maximum, and standard deviations) with and without constraint (24), for weekend work frequency constraints alternative I.

Problem set	Number of problems	Without constraint (24)*					With constraint (24)			
		Min	Ave.*	Max*	Std dev*	> 100	Min	Ave.	Max	Std dev
1	17	0.25	64.41	100	48.54	10	0.26	0.46	1.16	0.25
2	17	0.28	68.82	100	46.12	11	0.28	0.55	1.43	0.34
3	17	0.25	63.31	100	48.21	10	0.28	0.45	0.85	0.20
4	17	0.26	64.90	100	48.99	11	0.27	0.42	0.65	0.13
5	17	0.26	70.74	100	46.73	12	0.28	0.42	0.90	0.20
6	17	0.25	64.98	100	48.87	11	0.26	0.50	1.09	0.30
7	25	0.25	56.13	100	50.52	14	0.26	0.41	0.98	0.23
8	25	0.25	20.24	100	40.70	5	0.23	0.31	0.47	0.06
9	25	0.25	32.19	100	47.47	8	0.25	0.32	0.57	0.08
10	25	0.27	76.12	100	43.37	19	0.26	0.43	0.90	0.19
11	25	0.27	68.10	100	47.47	17	0.28	0.41	0.94	0.18
12	25	0.27	31.80	100	46.93	7	0.28	0.45	1.65	0.29
Overall	252	**0.25**	**55.02**	**100**	**49.43**	**135**	**0.23**	**0.46**	**1.65**	**0.22**

*Solution could not be obtained in 100 seconds

Table 4: Integer programming solution times in seconds with and without constraint (24) for weekend work frequency constraints alternative II.

Problem set	Number of problems	Without constraint (24)*					With constraint (24)			
		Min	Ave.*	Max*	Std dev*	> 100	Min	Ave.	Max	Std dev
1	17	0.26	64.80	100	49.13	11	0.26	0.32	0.44	0.06
2	17	0.28	70.68	100	46.82	12	0.26	0.28	0.31	0.02
3	17	0.26	64.81	100	49.11	11	0.25	0.28	0.34	0.02
4	17	0.32	64.83	100	49.08	11	0.26	0.33	0.48	0.07
5	17	0.27	70.67	100	46.84	12	0.24	0.31	0.491	0.08
6	17	0.26	63.86	100	48.55	10	0.26	0.31	0.45	0.06
7	25	0.26	73.22	100	44.16	18	0.25	0.31	0.451	0.05
8	25	0.24	40.17	100	49.86	10	0.22	0.28	0.4	0.04
9	25	0.26	60.12	100	49.85	15	0.25	0.30	0.36	0.03
10	25	0.29	76.08	100	43.45	19	0.26	0.30	0.39	0.03
11	25	0.25	68.10	100	47.47	17	0.25	0.32	0.63	0.08
12	25	0.25	32.21	100	47.46	8	0.26	0.31	0.51	0.07
Overall	252	**0.24**	**61.67**	**100**	**48.46**	**154**	**0.22**	**0.30**	**0.63**	**0.05**

*Solution could not be obtained in 100 seconds

assumed to be 100 seconds. Overall, average solution time dropped by 200 times, from 61.67 seconds to 0.30 seconds. Maximum solution time dropped from more than 100 seconds to only 0.63 seconds, while the standard deviation decreased from 48.46 seconds to only 0.05 seconds. Clearly, adding constraint (24) has proved to be essential for efficient solution. Constraint (24) produces this remarkable improvement in computation time by acting as a very efficient cut, significantly reducing the feasible region (search space) and accordingly decreasing the solution time.

CONCLUSIONS

An efficient optimization algorithm has been developed for three-day workweek employee days-off scheduling with several realistic constraints. Employees are given four off days per week, out of which at least two days must be consecutive. Moreover, employees cannot be assigned to a work stretch of more than four consecutive workdays. Two alternative types of constraints on weekend work frequency are imposed: employees must be given either a certain proportion of full weekends off, or a certain proportion of weekend days off. Finally, the maximum weekend work stretch can be specified or minimized. An integer programming model has been formulated, and a simple expression has been derived to determine the minimum workforce size for each alternative. A feasible rotation scheme has been constructed for assigning employees to work patterns over several weeks.

In order to improve the computational efficiency, workforce-size constraint (24) is appended to the integer programming model. Extensive computational experiments with 252 test problems have shown that including this constraint is essential for efficient solution. Without this constraint, more than 57% of the problems could not be solved by Excel Solver with default settings. The addition of constraint (24) has made it possible to solve all test problems. Moreover, appending this constraint has increased the average solution speed by at least 150 times.

ACKNOWLEDGMENTS

This research was conducted at the University of Loughborough, under funding from both the British Council and King Fahd University of Petroleum and Minerals.

REFERENCES

1. Alfares, H.K. (2000). "Dual-based optimization of cyclic three-day workweek scheduling." *Asia-Pacific Journal of Operational Research* 17(2), 137–148.
2. H. Alfares, H.K. (2003) "Compressed workweek scheduling with differing weekdays and weekends labor demands," *Asia-Pacific Journal of Operational Research* 20(1), 1–20.
3. Alfares, H. and Selim, S. (2002). "A cyclic selection combinatorial algorithm with cyclic scheduling applications." *6th Saudi Engineering Conference*, Dhahran, Saudi Arabia, December 14–17, 2002, Vol. 4, pp. 513–520.
4. Azmat, C.S. and Widmer, M. (2004). "A case study of single shift planning and scheduling under annualized hours: A Simple three-step approach." *European Journal of Operational Research* 153 (1), 148–175.
5. Bard, J.F., Binici, C. and de Silva, A.H. (2003). "Staff scheduling at the United States Postal Service." *Computers & Operations Research* 30(5), 745–771.
6. Billionnet, A. (1999). "Integer programming to schedule a hierarchical workforce with variable demands." *European Journal of Operational Research* 114(1), 105–114.
7. Brusco, M. J., Jacobs, L. W. (1993). "A simulated annealing approach to the cyclic staff-scheduling problem." *Naval Research Logistics* 40(1), 69–84.
8. Burns, R.N. and Narasimhan, R. (1999). "Multiple shift scheduling of workforce on four-day workweeks." *The Journal of the Operational Research Society* 50(9), 979–981.
9. Burns, R.N., Narasimhan, R. and Smith, L.D. (1998). "A set processing algorithm for scheduling staff on 4-day or 3-day work weeks." *Naval Research Logistics* 45(8), 839–853.
10. Cezik, T., Gunluk, O. and Luss, H. (2001). "An integer programming model for the weekly tour scheduling problem." *Naval Research Logistics* 48(7), 607–624.
11. Emmons, H., Burns, R. (1991). "Off-day scheduling with hierarchical worker categories." *Operations Research* 39(3), 484–495.

12. Ernst, A.T., Jiang, H., Krishnamoorthy, M. and Sier, D. (2004). "Staff scheduling and rostering: a review of applications, methods, and models." *European Journal of Operational Research* 153(1), 3–27.
13. Hung, R. (1993). "A three-day workweek multiple-shift scheduling model." *The Journal of the Operational Research Society* 44(2), 141–146.
14. Hung, R. (1995). "Compressed work schedules in a police force: a survey of applications." *Optimum* 26(2), 32–36.
15. Hung, R. (1991). "Single-shift workforce scheduling under a compressed workweek." *Omega* 19(5), 494–497.
16. Hung, R. (2005). "Using compressed workweeks to save labour cost." *European Journal of Operational Research* 170(1), 319–322.
17. Jarray, F. (2005). "Solving problems of discrete tomography: Application in workforce scheduling, *4OR: A Quarterly Journal of Operations Research* 3(4), 337–340.
18. Lankford, W.M. (1998). "Changing schedules: a case for alternative work schedules." *Career Development International* 3(4), 161–163.
19. McCampbell, A.S. (1996). "Benefits achieved through alternative work schedules." *Human Resources Planning* 19(3), 30–37.
20. Narasimhan, R. (1997). "An algorithm for a single shift scheduling of hierarchical workforce." *European Journal of Operational Research* 96(1), 113–121.
21. Narasimhan R. (2000). "An algorithm for multiple shift scheduling of hierarchical workforce on four-day or three-day workweeks." *INFOR* 38(1), 14–32.

APPENDIX. A SOLVED EXAMPLE

Given that P (proportion of *full weekends* off) is 0.5, L (maximum weekend work stretch) is 2 weeks, and the following daily labor demands for a typical work week:

$$r_1, r_2, \ldots, r_7 = 2, 6, 2, 7, 2, 6, 2$$

Minimum workforce size
Using (23), we obtain:

$$T_1, T_2, \ldots, T_7 = 6, 19, 6, 15, 10, 10, 15$$

The arguments of (22) are calculated as follows:

$r_{\max}$		$= 7$
$\sum r_i/3$	$= 27/3$	$= 9$
$T_{\max}/2 = T_2/2$	$= 19/2$	$= 9.5$
$\max(r_6, r_7)/(1-P)$	$= 6/0.5$	$= 12$

Using (22), we obtain:

$$W = \max\{7, 9, \lceil 9.5 \rceil, 12\} = 12$$

Days-off assignments
To the IP model defined by (1)–(15), we add the workforce-size constraint defined by (24): $\sum_{j=1}^{35} x_j = 12$. Next, we use integer programming to obtain the days-off assignments $x_1, \ldots, x_{35}$. The optimal integer programming solution obtained by Excel Solver is given by:

J_0 and J_1 patterns:	$x_1 = 2, x_7 = 3, x_9 = 1,$
J_2 (full weekend off) patterns:	$x_5 = 3, x_{11} = 1, x_{19} = 2$

Multiple-week rotation
A 12-week rotation cycle must be used, during which each employee is assigned 2 weeks to pattern 1, 3 weeks to pattern 7, 1 week to pattern 9, and so on. We need to ensure that no more than 4 successive workdays are assigned as work patterns are linked from one week to the next during the rotation cycle. Therefore, pattern 1 assignments cannot be immediately followed by pattern 9 which belongs to set E. Using (25), the minimum value of L (maximum weekend work stretch) is: $L = \lceil 6/6 \rceil = 1$ week. A 12-week feasible rotation cycle, using the given value of $L = 2$ weeks, is shown in Figure 1.

Figure 1: A cyclic 12-week rotation schedule for the solved example, where shaded cells represent an assignment to a J_2 (full-weekend off pattern).

Week / Employee	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	5	5	7	7	5	11	7	9		
2	19	1	1	5	5	7	7	11		7	9	
3	19	19	1	1	5	5	7	7		11	7	9
4	9	19	19	1	1	5	5	7	7	11		7
5	7	9	19	19	1	1	5	5	7	7		
6	11	7	9	19	19	1	1	5	5	7	7	
7	5	11	7	9	19	19	1	1	5	5	7	7
8	7	5	11	7	9	19	19	1	1	5	5	7
9	7	7	5	11	7	9	19	19	1	1	5	5
10	5	7	7	5	11	7	9	19	19	1	1	5
11	5	5	7	7	5	11	7	9	19	19	1	1
12	1	5	5	7	7	5	11	7	9	19	19	1