J Sched
DOI 10.1007/s10951-007-0040-x

A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs

Moustafa Elshafei · Hesham K. Alfares

Received: 10 October 2006 / Accepted: 23 August 2007

Abstract This paper presents a dynamic programming (DP) algorithm for solving a labor scheduling problem with several realistic days-off scheduling constraints and a cost structure that depends on the work sequence for each employee. The days-off scheduling constraints include the following: (1) each employee is assigned no more than three workdays per week, (2) each employee is assigned at least two consecutive off days per week, and (3) any work stretch cannot exceed four consecutive workdays. The sequence-dependent cost structure assumes that the daily wage of each employee depends on two factors: (1) whether the given workday is weekend or a regular workday, and (2) the sequence of work patterns assigned in previous days. A DP algorithm suited to instances of moderate size is used to determine the optimum work assignments that minimize the total labor cost, while satisfying the work demand under the stated constraints.

Keywords Labor scheduling · Optimization · Dynamic programming · Compressed workweek

1 Introduction and overview

Employee scheduling is both an important and a challenging practical problem, especially for organizations that operate seven days a week or 24 hours a day. The problem is important because of its significant impact on labor cost, level of service, and employee productivity and morale. The problem is challenging because of the complexity associated with factors such as: varying customer demands, limited employee availability, strict labor rules and agreements, numerous scheduling alternatives, different employee skill levels, conflicting employee preferences, and employee seniority and fairness considerations.

Narasimhan (2000) classifies employee scheduling problems in terms of six factors: (1) number of shifts, (2) number of employee skills or categories, (3) pattern of labor demand, (4) limits on the length of work stretches, (5) limits on weekend work frequency, and (6) number of workdays per week. Accordingly, the days-off scheduling problem considered in this paper can be classified as follows: (1) single shift, (2) homogeneous workforce, (3) variable daily demands, (4) work stretch length limit of four days, (5) no weekend work frequency limits, and (6) three workdays per week. However, although there are no explicit limits on weekend work frequency, excessive weekend work is avoided by the sequence-dependent cost structure that penalizes such occurrences.

Azmat and Widmer (2004) classify the days-off scheduling problem as either single-shift or multiple-shift. For each type, they define four sub-categories: (1) regular 5 workdays a week work schedules, (2) compressed 3 or 4 workdays a week work schedules, (3) hierarchical schedules for a workforce with varying skill levels, and (4) annualized hours schedules. Following this classification, this days-off scheduling paper falls within the single-shift, compressed workweek category. The following survey of literature focuses on compressed workweek and dynamic programming approaches to employee days-off scheduling.

The interest in flexible and compressed work schedules has been steadily growing. According to McCamp-

M. Elshafei (✉) · H.K. Alfares
Systems Engineering Department, King Fahd University of Petroleum & Minerals, PO Box 5067, Dhahran 31261, Saudi Arabia
e-mail: elshafei@kfupm.edu.sa

H.K. Alfares
e-mail: alfares@kfupm.edu.sa

bell (1996), the US Office of Personnel Management suggests three compressed schedule modules: 3-day workweek, 4-day workweek, and the 5-4/9 plan. Management Services Journal (2003) reports that UK firms are required since April 2003 to provide their staff with flexible working options. This real-life interest in compressed schedules is reflected in many recent papers. For example, these include Browne and Nanda (1987), Gould (1988), Nanda and Browne (1992, pp. 245–249), Hung and Emmons (1993), Burns et al. (1998), Billionnet (1999), Alfares (2003, 2006), and Costa et al. (2006). Hung (1996) gives a literature review focused on compressed workweek scheduling. Ernst et al. (2004) provide the most recent and comprehensive survey of employee scheduling literature.

There have been several applications of dynamic programming (DP) to solve employee scheduling problems. Davis and Reutzel (1981) use DP to minimize the cost of scheduling full-time and part-time check processing operators at a commercial bank. This DP approach is similar to a production system model with backordering, in which the delivery of checks represents the demand. Vassilacopoulos (1985) uses DP to determine the optimal allocation of an emergency department doctors to each hour and each work shift in the week. The objective is to minimize the maximum absolute deviation from requirements for each hour. Jonsson (1987) employs DP to determine the number of standby bus drivers that minimizes the expected cost of unused drivers and canceled bus tours. While the demand varies considerably due to passengers commuting to and from work, the supply of drivers is affected by sudden absences from work.

Easton and Rossin (1991) use DP within a working subset tour scheduling heuristic, in which only a small subset of feasible tours are used. DP is used to decide which feasible tours should be included in the working subset in order to minimize the total labor cost. Alfares and Bailey (1997) use DP with heuristic bounds on workforce size to determine days-off assignments of employees within an integrated model for minimum-cost scheduling of project tasks and manpower.

Caprara et al. (2003) combine DP with heuristic procedures to schedule the minimum number of employees at an emergency call center. The definition of working patterns for each employee is formulated as a covering problem for which alternative ILP models and DP algorithms based on these models are presented. Mohamed et al. (2003) enhance DP with knowledge-based techniques, genetic algorithms, simulated annealing, and fuzzy logic to solve manpower allocation problems. Koole and Pot (2005) apply the policy iteration method in the context of approximate DP for scheduling multi-skilled call center agents. Considering a waiting queue for each skill type, they avoid the DP curse of dimensionality by using simulation to approximate value functions.

In light of the above literature review, previous DP-based employees scheduling approaches typically aim to minimize the total cost or number of employees. In general, the stages of these DP models are the different planning time periods, while the states are usually the different scheduling options (tours or days-off patterns). Evidently, all previous DP approaches to employee scheduling assume that the cost of any workday depends only on the given workday, regardless of the employee's previous work assignments.

This paper presents a new DP approach for a single-shift compressed workweek employee scheduling. In this DP approach, the objective is to minimize the total labor cost, the stages are the days in the planning horizon, and the states are the feasible days-off schedules. The novelty of this DP approach comes from the unique cost structure, in which an employee's daily wage depends on the sequence of work assignments in previous days. The remainder of this paper is organized as follows. Section 2 introduces the problem definition. In Sect. 3, the set of feasible days-off patterns is identified. Section 4 is devoted to problem formulation. The DP algorithm is described in Sect. 5. A real-life application is presented and computational results are reported in Sect. 6. Finally, a number of conclusions and suggestions are offered in Sect. 7.

2 Problem definition

Assume we have a vector of daily labor demands R:

$$R = \{r(1), r(2), \dots, r(T)\}, \tag{1}$$

where

$r(t)$ = the number of employees needed in the tth day.

The schedule of the workforce is subject to the following assumptions and restrictions:

1. Each employee works no more than 3 days per week.
2. At least 2 consecutive off days per week.
3. No more than 4 consecutive workdays per work stretch.
4. The number of employees working at any day t is $= r(t)$.
5. The worker's daily wage depends on the weekday, and the previous working pattern of the worker.

As highlighted in the previous section, assumption 1 is representative of the compressed workweek scheduling environment. Assumption 2 is a very common requirement in days-off scheduling, whose purpose is to give employees at least one meaningful extended break during each week. Assumption 3 is frequently encountered in multiple-week three-day workweek scheduling models (e.g., Burns et al. 1998; Narasimhan 2000). Assumption 4 takes advantage of the flexibility of work rules in order to eliminate overstaffing and understaffing. The daily labor demands, and

Table 1 Work sequence-dependent daily wage structure

Week	$j-1$				j		
Day	4	5	6	7	1	2	3
c_0		$\mathbf{c_0}$					
c_1			$\mathbf{c_1}$				
c_1				$\mathbf{c_1}$			
c_2			c_1	$\mathbf{c_2}$			
c_3			c_1	c_2	$\mathbf{c_3}$		
c_4		c_0	c_1	c_2	$\mathbf{c_4}$		
c_5	c_0	c_0	$\mathbf{c_5}$				
c_6			c_1	c_2	c_3	$\mathbf{c_6}$	
c_7				c_1	c_0	c_0	$\mathbf{c_7}$

consequently employee schedules, are not assumed to be cyclic, thus they may change from one week to another. Moreover, the number of weekly workdays per employee is not fixed as three, but it could range from one to three. This provides a lot of flexibility in the schedule that makes it always feasible and economical to satisfy the daily labor demands exactly.

To the best of our knowledge, the 5th condition above is unique to this paper. Usually, higher (premium) pay is given for work during certain time periods, such as overtime, weekend, and evening. Therefore, the pay for each period commonly depends only on the work period itself, regardless of the prior work/off sequence. According to the cost structure introduced in this paper, the pay for each period depends on the current work period and also the prior work/off sequence. There are several possibilities to implement this realistic condition in order to reflect the actual labor rules. In this paper, we use the following types of work sequence-dependent costs whose structure is illustrated in Table 1:

$c_0 =$ cost of a normal week day.

$c_1 =$ cost of a single weekend day.

$c_2 =$ cost of a second weekend day.

$c_3 =$ cost of first day in a week after working full weekend.

$c_4 =$ cost of the first day in a week following 3 workdays.

$c_5 =$ cost of the first day of a weekend after 2 consecutive workdays.

$c_6 =$ cost of the second weekday following 3 consecutive workdays.

$c_7 =$ cost of the third weekday following 3 consecutive workdays.

It is desired to develop the work assignment that minimizes the total labor cost, while satisfying the work demand under the stated constraints. Due to the unique cost structure, ILP models are generally not well suited to solve this problem, in which the labor cost depends on the sequence of work/break assignments. For example, the number of decision variables in the explicit ILP formulation must be equal to the number of work/off sequences, 2^T, where T is the number of days in the planning horizon. This exponentially growing problem size makes the problem NP-complete, whose optimum ILP solution is very difficult. Although implicit, more compact ILP models can be formulated, this kind of sequence-dependent cost structure is naturally best handled by DP.

Let us define a weekly assignment pattern as:

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}, \quad a_i \in \{0, 1\}. \tag{2}$$

Clearly the number of binary patterns is 128. Since not all patterns satisfy assumptions 1 to 3, a subset J is defined as follows $J = \{$all 7-bit binary numbers, whose bit patterns satisfy the assumptions 1 to 3$\}$. The size of set J is $N_J = 62$. These binary numbers and their equivalent binary patterns are given in Table 2. We refer to the feasible days-off assignments defined by the set J as the h-patterns (horizontal patterns). The use of the set J and Table 2 is in fact a general framework for representation of any restricted employee work patterns and can be applied to other restricted work assignment problems. For example, Table 2 excludes the all-zero pattern, which implies that "at least one shift per week" is also imposed. Removing this condition can simply be achieved by including the all-zero row in the table. The simulation examples consider the cases of one or more shift per week.

Next, we also need to define further subsets of J as follows:

$$J_{l,t} = \big\{\text{set of pattern numbers in } J \text{ such that their first } t \text{ bits } (b_1, b_2, \ldots, b_t)_2 = l\big\} \tag{3}$$

where l is a t-bit binary number.

We also define the size of such subsets to be $N_J(l, t)$.

If the weekend is to be undivided, we may impose also the condition

$$\big[(a_6 \oplus a_7) \text{ OR } (\bar{a}_6 \oplus \bar{a}_7)\big] = 1. \tag{4}$$

This case will not be considered in this paper. However, it is clear that this case could also be covered by a minor adjustment of the algorithm proposed in subsequent sections.

3 Problem formulation

Let $r_{\max}$ be the maximum number of employees needed in any day:

$$r_{\max} = \max\big\{r(1), r(2), \ldots, r(T)\big\}. \tag{5}$$

Table 2 Weekly patterns satisfying conditions 1–3

No.	1	2	3	4	5	6	7	No.	1	2	3	4	5	6	7
1	0	0	0	0	0	0	1	32	0	1	0	0	1	1	0
2	0	0	0	0	0	1	0	33	0	1	0	1	0	0	0
3	0	0	0	0	0	1	1	34	0	1	0	1	0	0	1
4	0	0	0	0	1	0	0	35	0	1	0	1	1	0	0
5	0	0	0	0	1	0	1	36	0	1	1	0	0	0	0
6	0	0	0	0	1	1	0	37	0	1	1	0	0	0	1
7	0	0	0	0	1	1	1	38	0	1	1	0	0	1	0
8	0	0	0	1	0	0	0	39	0	1	1	0	1	0	0
9	0	0	0	1	0	0	1	40	0	1	1	1	0	0	0
10	0	0	0	1	0	1	0	41	1	0	0	0	0	0	0
11	0	0	0	1	0	1	1	42	1	0	0	0	0	0	1
12	0	0	0	1	1	0	0	43	1	0	0	0	0	1	0
13	0	0	0	1	1	0	1	44	1	0	0	0	0	1	1
14	0	0	0	1	1	1	0	45	1	0	0	0	1	0	0
15	0	0	1	0	0	0	0	46	1	0	0	0	1	0	1
16	0	0	1	0	0	0	1	47	1	0	0	0	1	1	0
17	0	0	1	0	0	1	0	48	1	0	0	1	0	0	0
18	0	0	1	0	0	1	1	49	1	0	0	1	0	0	1
19	0	0	1	0	1	0	0	50	1	0	0	1	0	1	0
20	0	0	1	0	1	0	1	51	1	0	0	1	1	0	0
21	0	0	1	0	1	1	0	52	1	0	1	0	0	0	0
22	0	0	1	1	0	0	0	53	1	0	1	0	0	0	1
23	0	0	1	1	0	0	1	54	1	0	1	0	0	1	0
24	0	0	1	1	0	1	0	55	1	0	1	0	1	0	0
25	0	0	1	1	1	0	0	56	1	0	1	1	0	0	0
26	0	1	0	0	0	0	0	57	1	1	0	0	0	0	0
27	0	1	0	0	0	0	1	58	1	1	0	0	0	0	1
28	0	1	0	0	0	1	0	59	1	1	0	0	0	1	0
29	0	1	0	0	0	1	1	60	1	1	0	0	1	0	0
30	0	1	0	0	1	0	0	61	1	1	0	1	0	0	0
31	0	1	0	0	1	0	1	62	1	1	1	0	0	0	0

Define r_{ceil} as follows

$$r_{\text{ceil}} = \max_j \left\{ \left\lceil \frac{1}{3} \sum_{m=1+7j}^{\min(7+7j,T)} r(m) \right\rceil \right\}, \tag{6}$$

where $\lceil z \rceil$ is the ceiling function, j = week number = $0, 1, 2, \ldots, \lceil T/7 \rceil$.

Thus, the minimum number of employees to satisfy the daily labor demand is given by:

$$W = \max\{r_{\text{ceil}}, r_{\max}\}. \tag{7}$$

Define $x_{i,t}$ for $i = 1, 2, \ldots, W$ and $t = 1, 2, \ldots, T$ as follows

$$x_{i,t} = \begin{cases} 1, & \text{if day } t \text{ is a workday for employee } i, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The schedule of the ith employee may be expressed as:

$$X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,T}\}. \tag{9}$$

Assumption 1 may then be written as:

$$\sum_{m=1}^{7} x_{i,m+7j} \le 3,$$
$$m + 7j \le T, \ j = 0, 1, 2, \ldots, \lceil T/7 \rceil. \tag{10}$$

Assumption 4 can now be written as:

$$\sum_{i=1}^{W} x_{i,t} = r(t), \quad t = 1, 2, \ldots, T. \tag{11}$$

The work assignment of the W employees at day t is given by:

$$v_t = \{x_{1,t}, x_{2,t}, \ldots, x_{W,t}\}. \tag{12}$$

The cost associated with an employee i working on a day t is given by $c_{i,t}$. Such cost is assumed to depend on the particular weekday as well as the employee's previous working patterns, i.e., the consecutive working days prior to t. The objective of the scheduling problem is thus to minimize the total labor cost:

$$C = \sum_{i}^{W} \sum_{t=1}^{T} x_{i,t} c_{i,t}. \tag{13}$$

4 The DP algorithm

We define $J_v(t)$ to be the set of all vertical patterns at time t, and the size of this set is $N_v(t)$. On any given day t, the number of possible ways of selecting $r(t)$ employees from the total workforce of size W is equal to $N_v(t)$. These $J_v(t)$ selections are W-bit binary patterns, and we refer to these work assignment patterns as the v-patterns (vertical patterns). The kth vertical pattern at day t is represented by:

$$v_t^k = \{x_{1,t}^k, x_{2,t}^k, \ldots, x_{W,t}^k\}. \tag{14}$$

Let us define

$k =$ the current v-pattern number under consideration in day t;

$k_1 =$ the v-pattern number under consideration in day $t-1$;

$K(t) =$ the optimum v-pattern schedule (employee selection) in day t.

In dynamic programming (DP) terminology, the *stages* are the T days, the *states* are the current schedules (v-patterns) at day t, and the *alternatives* are the previous schedules at day $t-1$. The algorithm uses a forward recursion procedure, which proceeds from day 1 to day T. At any given day t, the algorithm computes the minimum cost over all patterns k_1 in $J_v(t-1)$, or path cost up to $t-1$ and k_1, plus the cost of the v-pattern k at time t as depicted in Fig. 1.

For each v-pattern in $J_v(t)$, the following DP recursive relationships are used to calculate the minimum path cost $C(t,k)$:

$$C(0,k) = 0, \tag{15}$$

$$C(t,k) = \min_{k_1 \in J_v(t-1)} \left\{ C(t-1,k_1) + \sum_{i=1}^{W} x_{i,t}^k c_{i,t} \right\}, \tag{16}$$

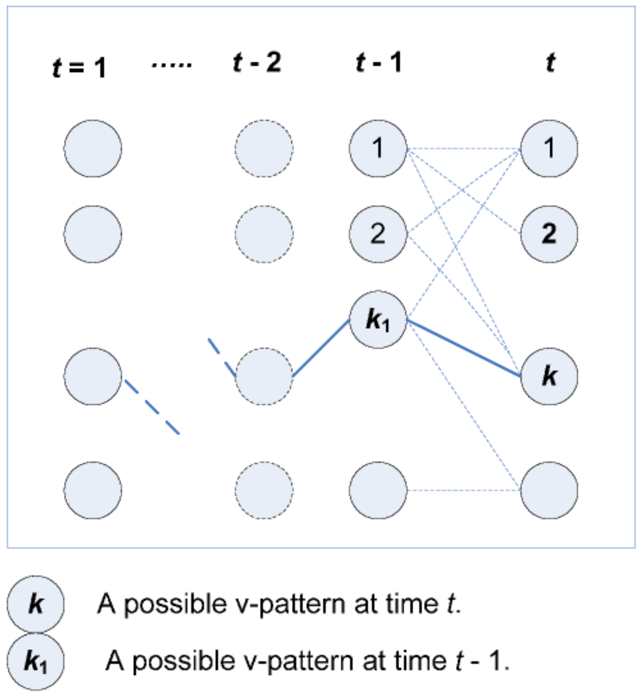
t = 1
.....
t - 2
t - 1
t
1
2
k1
1
2
k
k
A possible v-pattern at time t.
k1
A possible v-pattern at time t - 1.

Fig. 1 Finding the best path ending at $t-1$ for each possible v-pattern k at time t

where $\{x_{1,t}^k, x_{2,t}^k, \ldots, x_{W,t}^k\}$ is the work assignment at day t using the kth vertical pattern. $C(t,k)$ gives the optimal cost of the path ending at time t, when the v-pattern k is selected. $C(t,k)$ is calculated sequentially as t progresses. Once we reach $t = T$, the best path is the terminal path of minimum cost among all terminal paths. The selected best path can then be tracked backward to obtain the optimal v-pattern assignment for every t.

In order to track the optimal path backward, we need to store the path history. A back tracking matrix $P(k, W, t)$ stores the best previous v-patterns k_1 which minimizes (16). At any time t, the first index k points to a binary matrix of dimension $W \times t$, which stores the previous t optimal v-patterns.

The algorithm was coded in a MATLAB program. To improve the efficiency of the algorithm, several refinements can be considered. We notice in the minimization of (16), that not all v-patterns in $J_v(t)$ can precede a v-pattern k at time t. The sequence k_1–k of v-patterns must not violate the h-patterns of any of the employees. This can be easily tested with the help of the subsets $J_{l,t}$ defined earlier. If the k_1–k v-pattern sequence is selected, the sequence k_1–k is dismissed or assigned an arbitrarily high cost if such a sequence leads to an empty $J_{l,t}$ set for any employee. Similarly, we check if the sequence k_1–k leads to no more than 4 consecutive working days. The sequence is then assigned an arbitrarily high cost if it does not meet this condition.

The algorithm proceeds according to the following steps:

1. Compute and store the weekly h-patterns (see Table 2).

2. Compute the minimum number of workers W using (7).
3. Using the labor demand matrix R, compute the number of vertical patterns $N_v(t)$, at each time t. Compute and store the vertical patterns.
4. At time $t = 1$, initialize the back tracking matrix $P(k, W, t)$, and set the cost $C(t, k) = c_0 r(t)$.
5. For $t \geq 2$
   For $k = 1$ to $N_v(t)$
   For $k_1 = 1$ to $N_v(t-1)$
   (a) Check if the sequence k_1–k does not violate any of the workers' h-patterns.
   (b) Check if the sequence k_1–k does not lead to assignment of 4 or more workdays in sequence.
   (c) Compute the cost of k_1–k.
   Find the optimal k_1 which leads to the minimum cost of k_1–k sequence over all k_1 patterns.
6. Update the back tracking matrix $P(k, W, t)$.
7. At $t = T$, find the optimal terminal v-pattern $k^* = \arg[\min\{C(t, k)\}]$.
8. Use the back tracking matrix $P(k^*, W, t)$ to obtain the optimal work assignment.

Let $N_v(t)$ be the number of vertical patterns at time t, then the number of operations NOP is given by

$$NOP \cong O\left(\sum_{t=2}^{T} N_v(t) \cdot N_v(t-1)\right) \approx O\left(T \cdot \bar{N}_v^2\right); \qquad (17)$$

where $\bar{N}_v$ is the average number of v-patterns during the schedule period T.

5 An application and computational experiments

5.1 Assignment of security personnel

The DP algorithm has been applied to the problem of night shift assignment of security personnel in a university campus. The assignment includes the campus gate, a patrol car, and a call desk for receiving emergency calls. The demand for a typical three-week period is given below. During the first week, which has a normal workload, 5 security officers are needed on workday evenings and 3 security officers are needed on weekend evenings. The second week gives the manpower needed when night events are scheduled, while the third week represents typical night shift load during an exam week. Given the following labor demand and cost vectors, we would like to find the optimal employee days-off assignment subject to the conditions assumed in this paper.

$$R = \{r(1), \ldots, r(21)\}$$
$$= (5, 5, 5, 5, 5, 3, 3, 5, 5, 5, 5, 8, 7, 3, 6, 6, 7, 7, 5, 3, 3),$$
$$(c_0, c_1, \ldots, c_7) = (1, 1.5, 2, 1.25, 1.5, 2, 1.5, 1.5).$$

Using (5–7):

$$r_{\max} = \max\{5, 5, 5, 5, 5, 3, 3, 5, 5, 5, 5, 8, 7, 3, 6, 6, 7, 7, 5, 3, 3\} = 8,$$

$$r_{\text{ceil}} = \max\left\{\left\lceil \frac{31}{3} \right\rceil, \left\lceil \frac{38}{3} \right\rceil, \left\lceil \frac{37}{3} \right\rceil\right\} = 13.$$

The minimum number of workers is $W = \max\{8, 13\} = 13$.

Running the MATLAB program to apply the DP algorithm, we obtain the solution shown in Table 3, corresponding to

Table 3 Optimum days-off assignment for campus security personnel

Employee	Days of week 1							Days of week 2							Days of week 3						
i	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0
2	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0
3	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0
4	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0
5	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0
6	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	1	0	1	0	0	0
7	0	0	1	1	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0	1
8	0	0	1	1	0	0	1	0	0	1	1	0	1	0	0	0	1	1	0	0	1
9	0	0	1	1	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	0	1
10	0	0	1	1	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0
11	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1	0
12	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1	0
13	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0

Table 4 Summary of computational experiments with the DP algorithm

Problem number	Demand	Cost vector	Workforce size W	Additional conditions	Total cost (\$)	CPU time (sec)
1	$R1$	$S1$	8		126.5	16.7
2	$R1$	$S1$	9		141.5	61.5
3	$R1$	$S1$	10		156.5	183.7
4	$R1$	$S1$	8	$Q1$ or $Q2$	126.5	19
5	$R1$	$S1$	9	$Q1$	141.5	60.7
6	$R1$	$S1$	9	$Q2$	141.5	60.2
7	$R1$	$S1$	10	$Q1$	156.5	174.5
8	$R1$	$S2$	8		116.75	16.8
9	$R1$	$S3$	8		139	15.5
10	$R3$	$S1$	9		67.5	48
11	$R4$	$S1$	9		68	29
12	$R5$	$S1$	9		71	34
13	$R6$	$S1$	9		67.5	36
14	$R7$	$S1$	9		70	29
15	$R8$	$S1$	9		69	70

a minimum total cost $C = 311.5$. Note that the rows in the table represent the selected h-patterns, while the columns represent the v-patterns. The program was set to impose a minimum of one shift per week for every employee. The security department had a 15-employee workforce. An ad-hoc schedule was used, especially for special night events and exam weeks, causing conflicts and complaints by employees. The proposed schedule reduced the number of required employees to 13, improved employee satisfaction, and minimized the labor cost.

5.2 Computational experiments

Fifteen test problems, divided into three sets, were used to conduct extensive computational experiments. Using the MATLAB program to solve each test problem by the new DP algorithm, the results of these experiments are summarized in Table 4. All the test problems were run using MATLAB Version 6.5 on a notebook Pentium IV PC with 1.7 GHz CPU and 256 K memory. The first set, which contains 7 test problems (numbered 1–7), has the following demand pattern over a 14-day period and cost vector:

$$R1 = \{r(1), \dots, r(14)\}$$
$$= (3, 2, 1, 5, 3, 3, 1, 2, 6, 3, 2, 5, 4, 2),$$
$$S1 = (c_0, c_1, \dots, c_7) = (1, 1.5, 2, 1.25, 1.5, 2, 1.5, 1.5).$$

Using (5–7), the minimum number of workers to satisfy this demand subject to the conditions in this paper is $W = 8$. However, we investigate the solution under different combinations of the workforce size and the minimum number of workdays per week. In practical situations, employers tend to use some overstaffing, so that the work would not be affected in case of casual absence, vacations, or sick leaves of one or more of the employees. Therefore, for this problem we investigate the effect of imposing the condition $W = 9$ (one extra worker) or the condition $W = 10$ (two extra workers). Moreover, we also explore solutions obtained when constraints are imposed on the minimum number of workdays per week. Specifically, we investigate the effect of adding one of the two following constraints:

$Q1$: Each employee works at least one workday per week.

$Q2$: Each employee works more than one workday per week.

The second problem set, which contains 2 test problems (numbered 8–9), has the same labor demand pattern $R1$. We impose the minimum number of workers, $W = 8$, subject to two alternative cost vectors (compensation schedules):

$$S2 = (c_0, c_1, \dots, c_7) = (1, 1.25, 1.5, 1, 2, 1.5, 2, 2),$$
$$S3 = (c_0, c_1, \dots, c_7) = (1, 2, 2, 1, 1, 2, 1.25, 1.25).$$

The third set used to study the performance of the algorithm contains 6 test problems (numbered 10–15), corresponding to 6 different demand patterns. The total number of workers needed during a week was taken to be $\sum r = 27$ for all cases. Using (5–7), the minimum number of workers to meet the desired labor demand was also constant for all cases ($W = 9$). For these six problems, the minimum number of workers, $W = 9$, is imposed subject to the cost vector $S1$. The six labor demand patterns are specified as follows:

$$\begin{aligned}
&\text{Trend pattern:} && R3 = \{r(1), \dots, r(7)\} \\
& && = (2, 3, 3, 4, 4, 5, 6); \\
&\text{Concave pattern:} && R4 = \{r(1), \dots, r(7)\} \\
& && = (2, 3, 5, 7, 5, 3, 2); \\
&\text{Convex pattern:} && R5 = \{r(1), \dots, r(7)\} \\
& && = (6, 4, 3, 1, 3, 4, 6); \\
&\text{Triangular pattern:} && R6 = \{r(1), \dots, r(7)\} \\
& && = (3, 4, 7, 5, 4, 3, 1); \\
&\text{Binomial pattern:} && R7 = \{r(1), \dots, r(7)\} \\
& && = (4, 6, 3, 1, 3, 6, 4); \\
&\text{Level pattern:} && R8 = \{r(1), \dots, r(7)\} \\
& && = (4, 4, 4, 4, 4, 4, 3).
\end{aligned}$$

From the computational results shown in Table 4, we focus on the solution quality in terms of the total cost and the CPU execution time. Clearly, four factors affect the solution quality, namely: the demand pattern R, the cost vector S, specified workforce size W, and additional conditions Q. Obviously, the most important factor is specified workforce size W, which significantly affects both the total labor cost and the CPU time. The second most important factor is the given cost vector S, which considerably influences the total labor cost but has almost no impact on the CPU time. The next important factor is the particular demand pattern R, which substantially affects the CPU time but has almost no impact on the total labor cost. The least important factor is the additional conditions imposed on the minimum number of workdays per week Q, which seems to have no bearing on either the total labor cost or the CPU time.

6 Conclusions

A dynamic programming algorithm has been presented for solving a single-shift, compressed workweek employee days-off scheduling problem. Constraints are imposed on the maximum work stretch, maximum number of workdays per week, and minimum number of consecutive off days per week. The distinguishing feature of the problem is the unique work-sequence-dependent cost structure, in which the daily wage of each employee depends on the previous work assignments in the preceding days. This realistic assumption makes traditional integer programming models impractical for this problem. Thus, an efficient DP algorithm is developed to determine the optimum days-off assignments that minimize the total labor cost. This algorithm has been applied to a real-life scheduling employee problem, and has been tested using a set of computational experiments.

Several future research extensions of this research could be considered. Examples of these extensions include the following considerations: multiple shifts, multiple skill levels (i.e., hierarchical workforce), full weekends off or on, and weekend off frequency requirements. The algorithm can easily be modified to obtain the optimal schedule when the work demand changes on a daily basis, assuming a fixed pool of W employees is available. Another interesting question is the inclusions of off-day patterns that do not provide two consecutive off days in a single calendar week, but would provide two consecutive off days every seven days with a multi-week planning horizon.

Acknowledgements The authors are grateful to King Fahd University of Petroleum and Minerals for supporting this research effort.

References

Alfares, H. K. (2003). Flexible four-day workweek scheduling with weekend work frequency constraints. *Computers & Industrial Engineering*, *44*(3), 325–338.

Alfares, H. K. (2006). Compressed workweek scheduling with days-off consecutivity, weekend-off frequency, and work stretch constraints. *INFOR*, *44*(3), 175–189.

Alfares, H. K., & Bailey, J. E. (1997). Integrated project task and manpower scheduling. *IIE Transactions*, *29*(9), 711–718.

Anonymous (2003). UK business caught short on flexible working. *Management Services Journal*, *47*(3), 3.

Azmat, C. S., & 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.

Billionnet, A. (1999). Integer programming to schedule a hierarchical workforce with variable demands. *European Journal of Operational Research*, *114*(1), 105–114.

Browne, J., & Nanda, R. (1987). Scheduling efficiency of the four-day week at transportation facilities. In *Proceedings of the Institute of Transportation Engineers 57th annual meeting* (pp. 58–62), New York, 16–20 August 1987.

Burns, R. N., Narasimhan, R., & 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.

Caprara, A., Monaci, M., & Toth, P. (2003). Models and algorithms for a staff scheduling problem. *Mathematical Programming*, *98*(1–3), 445–476.

Costa, M.-C., Jarray, F., & Picouleau, C. (2006). An acyclic days-off scheduling problem. *4OR*, *4*(1), 73–85.

Davis, S. G., & Reutzel, E. T. (1981). A dynamic programming approach to work force scheduling with time-dependent performance measures. *Journal of Operations Management*, *1*(3), 165–171.

Easton, F. F., & Rossin, D. F. (1991). Sufficient working subsets for the tour scheduling problem. *Management Science*, *37*(11), 1441–1451.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: a review of applications, methods, and models. *European Journal of Operational Research*, *153*(1), 3–27.

Gould, C. H. (1988). Rolling fours: novel work schedule. *Journal of Construction Engineering and Management*, *114*(4), 577–593.

Hung, R. (1996). An annotated bibliography of compressed workweeks. *International Journal of Manpower*, *17*(6–7), 43–53.

Hung, R., & Emmons, H. (1993). Multiple-shift workforce scheduling under the 3–4 compressed workweek with a hierarchical workforce. *IIE Transactions*, *25*(2), 82–89.

Jonsson, H. (1987). Dimensioning of bus driver buffers subject to variations in the traffic load. *Engineering Costs and Production Economics*, *12*(1–4), 29–38.

Koole, G., & Pot, A. (2005). Approximate dynamic programming in multi-skill call centers. In *Proceedings of the 37th conference on winter simulation* (pp. 576–583), Orlando, FL, 4–7 December 2005.

McCampbell, A. S., (1996). Benefits achieved through alternative work schedules. *Human Resource Planning*, *19*(3), 30–37.

Mohamed, K. A., Datta, A., & Kozera, R. (2003). A knowledge-based technique for constraints satisfaction in manpower allocation. *Lecture Notes in Computer Science*, *2659*, 100–108.

Nanda, R., & Browne, J. (1992). *Introduction to employee scheduling*. New York: Van Nostrand Reinhold.

Narasimhan, R. (2000). An algorithm for multiple-shift scheduling of hierarchical workforce on four-day or three-day workweek. *INFOR*, *38*(1), 14–32.

Vassilacopoulos, G. (1985). Allocating doctors to shifts in an accident and emergency department. *Journal of the Operational Research Society*, *36*(6), 517–523.