Employee Timetabling

Contents
1. Workforce allocation and Employee Timetabling
2. An Algorithm for Single Shift Schedules

Literature:
Operations Scheduling with Applications in Manufacturing and Services, Michael Pinedo and Xiuli Chao, McGraw Hill, 2000.
Chapter 9
or

Workforce allocation and employee timetabling

- Workforce allocation and employee timetabling deals with the arrangement of work schedules and the assignment of personnel to shifts to cover the demand for resources that vary over time.

Terminology:
employee timetabling
personnel scheduling
workforce scheduling

Single Shift Schedules

Problem Statement
Find the minimum number of employees $W$, required to cover a 7-day-a-week operation so that the following constraints are satisfied:
- the demand per day $n_j, j=1,...,7$ is met ($n_1$ is Sunday, $n_7$ is Saturday)
- each employee is given $k_1$ out of every $k_2$ weekends off
- each employee works exactly 5 out of 7 days (from Sunday to Saturday)
- each employee works no more than 6 consecutive days

Constraints
1. Weekend constraints
   In $k_2$ weeks each employee is available for $k_2 - k_1$ weekends
   $n = \max(n_1, n_7)$
   $(k_2-k_1)W \geq k_2 n$
   $W \geq \frac{k_2 n}{k_2 - k_1}$

2. Total demand constraint
   $5W \geq \sum_{j=1}^{7} n_j$
   $W \geq \lceil \frac{1}{5} \sum_{j=1}^{7} n_j \rceil$

3. Maximum daily demand
   $W \geq \max(n_1,...,n_7)$

Algorithm and Example

<table>
<thead>
<tr>
<th>Day</th>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

- Employee requires 3 out of 5 weekends off: $k_1 = 3, k_2 = 5$

Step 1. Compute the minimum workforce.
$W$ is equal to the maximum of the three lower bounds given in constraints 1, 2 and 3.

$W \geq \frac{5 \cdot 3}{2} = 8$
$W \geq \frac{35}{5} = 7$
$W \geq ?$
$W = 8, n = 3$

Step 2. Schedule the weekends off
1. Assign the first weekend off to the first $(W-n)$ employees.
2. Assign the second weekend off to the next $(W-n)$ employees.
3. This process continues cyclically. Employee 1 is treated as the next employee after employee $W$.

$W - n = 8 - 3 = 5$

<table>
<thead>
<tr>
<th>S</th>
<th>M</th>
<th>T</th>
<th>W</th>
<th>F</th>
<th>S</th>
<th>M</th>
<th>T</th>
<th>W</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Step 3. Determine the additional off-day pairs
Total number of days off: 2W (everybody has 2 days off in a week)
W-n Saturdays and W-n Sundays are already assigned off (2W - 2n) therefore
2n days have to be assigned
\[ S_j = W - n_j \quad \text{surplus employees for day } j \]
\[ S_{i \neq j} = W - n_j \quad \text{for } j = 2, \ldots, 6 \]

Iteratively, construct a list of \( n \) pairs of off-days:
1. Choose day \( k \) such that \( S_k = \max (S_0, \ldots, S_7) \)
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 \( S_i \) and \( S_k \) by 1.
Repeat this procedure \( n \) times.

\[ \sum_{j=0}^{7} S_j = \sum_{j=0}^{7} (W - n_j) + n - n_k + n - n_i, \]
\[ = 5W - \sum_{j=1}^{7} n_j + 2n \]
(totals demand constraint)
\[ \geq 2n \]

Step 4. Assigning off-day pairs in week 1.
1. Categorise employees

<table>
<thead>
<tr>
<th>Category</th>
<th>Weekend 1</th>
<th>Weekend 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 off</td>
<td>no off days needed</td>
<td>off</td>
</tr>
<tr>
<td>T2 off</td>
<td>1 off day needed</td>
<td>on</td>
</tr>
<tr>
<td>T3 on</td>
<td>1 off day needed</td>
<td>off</td>
</tr>
<tr>
<td>T4 on</td>
<td>2 off days needed</td>
<td>on</td>
</tr>
</tbody>
</table>

\[ |T3| + |T4| = n \quad \text{(from weekend 1)} \]
\[ |T2| + |T4| = n \quad \text{(from weekend } i + 1) \]

⇒ \[ |T2| = |T3| \]
Pair T2 with T3.

2. Assigning off-day pairs
First assign pairs of days to the T4 employees
Assign to T3 and T2: T3 the earliest day of the pair, T2 the latest

Step 5. Assigning off-day pairs in week \( i \)
1. Categorise employees
2. Assigning off-day pairs
Case I. There is a nondistinct pair \((k, k)\) week \( i \) is scheduled in the same way as week 1 and is independent of week \((i-1)\).
Case II. All the pairs are distinct
All employees of type T3 and T4 in week \( i \) are associated with the same pair of off days which they received in week \((i-1)\).
T4 is given both days off
T3 gets the earliest day of the associated pair

Example

<table>
<thead>
<tr>
<th>Day</th>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ W \geq \left\lceil \frac{3 + 2}{3 - 2} \right\rceil = 3 \]
\[ W \geq \frac{16}{5} = 3 \]
\[ W \geq 3 \]
\[ W = 3, n = 2 \]

employee requires 1 out of 3 weekends off
\[ k_1 = 1, k_2 = 3 \]
Discussion

- There exist enough distinct off-days pairs to cover T4 employees.
- The algorithm will never assign a Saturday or Sunday surplus day off to an employee who already has that weekend off.
- Each employee works at least $k_1$ out of $k_2$ weekends off.
- Each employee works exactly 5 days per week from Sunday to Saturday.
- No employee works more than 5 consecutive days if all off-day pairs are distinct.
- No employee works more than 6 consecutive days when some off-day pairs are not distinct.

Summary

- This algorithm for single shift scheduling gives the bound for workforce size.
- The algorithm works by giving days off rather than scheduling to meet minimum daily needs.
- Single shift scheduling is a building block for the shift scheduling.