DM87
SCHEDULING, TIMETABLING AND ROUTING

Lecture 17

Workforce Scheduling

Marco Chiarandini

Outline

1. Workforce Scheduling
2. Crew Scheduling and Rostering
3. Employee Timetabling

Outline

1. Workforce Scheduling
2. Crew Scheduling and Rostering
3. Employee Timetabling

Workforce Scheduling

Workforce Scheduling:

- Crew Scheduling and Rostering
- Employee Timetabling

Shift: consecutive working hours
Roster: shift and rest day patterns over a fixed period of time (a week or a month)

Two main approaches:

- coordinate the design of the rosters and the assignment of the shifts to the employees, and solve it as a single problem.
- consider the scheduling of the actual employees only after the rosters are designed, solve two problems in series.

Features to consider: rest periods, days off, preferences, availabilities, skills.

Crew Scheduling and Rostering

Workforce scheduling applied in the transportation and logistics sector for enterprises such as airlines, railways, mass transit companies and bus companies (pilots, attendants, ground staff, guards, drivers, etc.)

Employee Timetabling

Employee timetabling (aka labor scheduling) is the operation of assigning employees to tasks in a set of shifts during a fixed period of time, typically a week.
Days off, shifts, tours (set of shifts)

Examples of employee timetabling problems include:

- assignment of nurses to shifts in a hospital,
- assignment of workers to cash registers in a large store
- assignment of phone operators to shifts and stations in a service-oriented call-center

Differences with Crew scheduling:

- no need to travel to perform tasks in locations
- start and finish time not predetermined

Outline

1. Workforce Scheduling

2. Crew Scheduling and Rostering

3. Employee Timetabling

Crew Scheduling

Input:

- Flight leg (departure, arrival, duration)
- A set of feasible combinations of flights for a crew

Output: A subset of flights feasible for a crew

Set partitioning problem!

Often treated as set covering because:

- its linear programming relaxation is numerically more stable and thus easier to solve
- it is trivial to construct a feasible integer solution from a solution to the linear programming relaxation
- it makes possible to restrict to only rosters of maximal length

Extension: a set of crews

Subgradient Optimization Lagrange Multipliers

$$\begin{aligned} \max \quad & c^T x \\ \text{st} \quad & Ax \le b \\ & Dx \le d \\ & x_j \in \mathbf{Z}^+, \quad j = 1, \ldots, n \end{aligned}$$

Lagrange Relaxation, multipliers $\lambda \ge 0$

$$\begin{aligned} \max \quad & z_{LR}(\lambda) = c^T x - \lambda(Dx - d) \\ \text{st} \quad & Ax \le b \\ & x_j \in \mathbf{Z}^+, \quad j = 1, \ldots, n \end{aligned}$$

Lagrange Dual Problem

$$z_{LD} = \min_{\lambda \ge 0} z_{LR}(\lambda)$$

Lagrangian dual solved by Subgradient optimization

- Works well due to convexity
- Roots in nonlinear programming

Outline

1. Workforce Scheduling
2. Crew Scheduling and Rostering
3. Employee Timetabling

Shift Scheduling

Creating daily shifts:

- cycle made of m time intervals not necessarily identical
- during each period, b_i personnel is required
- n different shift patterns (columns of matrix A)

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax \ge b \\ & x \ge 0 \text{ and integer} \end{aligned}$$

(k, m)-cyclic Staffing Problem

Assign persons to an m-period cyclic schedule so that:

- requirements b_i are met
- each person works a shift of k consecutive periods and is free for the other $m - k$ periods. (periods 1 and m are consecutive)

and the cost of the assignment is minimized.

$$\begin{aligned} \min \quad & cx \\ \text{st} \quad & \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} x \geq b \\ & x \geq 0 \text{ and integer} \end{aligned} \tag{P}$$

Recall: Totally Unimodular Matrices

Definition: A matrix A is totally unimodular (TU) if every square submatrix of A has determinant +1, -1 or 0.

Proposition 1: The linear program $\max\{cx : Ax \leq b, x \in \mathbf{R}^m_+\}$ has an integral optimal solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular

Efficient algorithms to recognize if a matrix is totally unimodular are nontrivial.

Proposition 2: A matrix A is TU if

(i) $a_{ij} \in \{+1, -1, 0\}$ for all i, j

(ii) each column contains at most two nonzero coefficients ($\sum_{i=1}^{m} |a_{ij}| \leq 2$)

(iii) there exists a partition (M_1, M_2) of the set M of rows such that each column j containing two nonzero coefficients satisfies $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$

Proposition 3: A matrix is TU if

(i) $a_{ij} \in \{+1, -1, 0\}$ for all i, j

(ii) for any subset M of the rows, there exists a partition (M_1, M_2) of M such that each column j satisfies

$$\left| \sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} \right| \leq 1$$

Definition: A $(0, 1)$–matrix B has the consecutive 1's property if for any column j, $b_{ij} = b_{i'j} = 1$ with $i < i'$ implies $b_{lj} = 1$ for $i < l < i'$. That is, if there is a permutation of the rows such that the 1's in each column appear consecutively.

Whether a matrix has the consecutive 1's property can be determined in polynomial time [D. R. Fulkerson and O. A. Gross; Incidence matrices and interval graphs. 1965 Pacific J. Math. 15(3) 835-855.]

A matrix with consecutive 1's property satisfies Proposition 3 and is therefore TU.

What about this matrix?

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Definition A $(0, 1)$-matrix B has the circular 1's property for rows (resp. for columns) if the columns of B can be permuted so that the 1's in each row are circular, that is, appear in a circularly consecutive fashion

The circular 1's property for columns does not imply circular 1's property for rows.

Whether a matrix has the circular 1's property for rows (resp. columns) can be determined in $O(m^2n)$ time [A. Tucker, Matrix characterizations of circular-arc graphs. (1971) Pacific J. Math. 39(2) 535-545]

Integer programs where the constraint matrix A have the circular 1's property for **rows** can be solved efficiently as follows:

Step 1 Solve the linear relaxation of (P) to obtain $x'_1, \ldots, x'_n$. If $x'_1, \ldots, x'_n$ are integer, then it is optimal for (P) and STOP. Otherwise go to Step 2.

Step 2 Form two linear programs LP1 and LP2 from the relaxation of the original problem by adding respectively the constraints

$$x_1 + \ldots + x_n = \lfloor x'_1 + \ldots + x'_n \rfloor \qquad \text{(LP1)}$$

and

$$x_1 + \ldots + x_n = \lceil x'_1 + \ldots + x'_n \rceil \qquad \text{(LP2)}$$

The solutions to LP1 and LP2 can be taken to be integral and the best of the two solutions is an optimal solution to the staffing problem (P)

Cyclic Staffing with Overtime

- Hourly requirements b_i
- Basic work shift 8 hours
- Overtime of up to additional 8 hours possible

minimize $\quad cx$

subject to

$$\begin{array}{c} 07\\08\\09\\10\\11\\12\\13\\14\\15\\16\\17\\18\\19\\20\\21\\22\\23\\24\\01\\02\\03\\04\\05\\06 \end{array}
\left[\begin{array}{ccccccccccccccccccccccccccc}
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1
\end{array}\right] \quad x \geq b$$

$x \geq 0$ and integer.

Days-Off Scheduling

- Guarantee two days-off each week, including every other weekend.

IP with matrix A:

$$\begin{array}{l} \\ \\ \\ \text{first week} \\ \\ \\ \\ \\ \\ \\ \text{second week} \\ \\ \\ \\ \end{array}
\left[\begin{array}{cccccc|cccccc}
1&1&1&1&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&0&0\\
1&1&1&1&1&1&1&1&1&0&0&1\\
1&1&1&1&1&1&1&1&0&0&1&1\\
1&1&1&1&1&1&1&0&0&1&1&1\\
0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&1&1&1&1&1\\
\hline
0&1&1&1&1&1&1&1&1&1&1&1\\
0&0&1&1&1&1&1&1&1&1&1&1\\
1&0&0&1&1&1&1&1&1&1&1&1\\
1&1&0&0&1&1&1&1&1&1&1&1\\
1&1&1&0&0&1&1&1&1&1&1&1\\
1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&1&0&0&0&0&0&0&0
\end{array}\right]$$

Cyclic Staffing with Part-Time Workers

- Columns of A describe the work-shifts
- Part-time employees can be hired for each time period i at cost c'_i per worker

$$\begin{aligned} \min \quad & cx + c'x' \\ \text{st} \quad & Ax + Ix' \geq b \\ & x, x' \geq 0 \text{ and integer} \end{aligned}$$

Cyclic Staffing with Linear Penalties for Understaffing and Overstaffing

- demands are not rigid
- a cost c'_i for understaffing and a cost c''_i for overstaffing

$$\begin{aligned} \min \quad & cx + c'x' + c''(b - Ax - x') \\ \text{st} \quad & Ax + Ix' \geq b \\ & x, x' \geq 0 \text{ and integer} \end{aligned}$$

Once rosters (set of shifts) are designed, people can be assigned to them according to availabilities, preferences, skills.

Alternatively one can take care of these two phases at the same time:

Nurse Scheduling

- Hospital: head nurses on duty seven days a week 24 hours a day
- Three 8 hours shifts per day (1: daytime, 2: evening, 3: night)
- In a day each shift must be staffed by a different nurse
- The schedule must be the same every week
- Four nurses are available (A,B,C,D) and must work at least 5 days a week.
- No shift should be staffed by more than two different nurses during the week
- No employee is asked to work different shifts on two consecutive days
- An employee that works shifts 2 and 3 must do so at least two days in a row.

Mainly a feasibility problem

A CP approach

Two solution representations

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Shift 1	A	B	A	A	A	A	A
Shift 2	C	C	C	B	B	B	B
Shift 3	D	D	D	D	C	C	D

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Worker A	1	0	1	1	1	1	1
Worker B	0	1	0	2	2	2	2
Worker C	2	2	2	0	3	3	0
Worker D	3	3	3	3	0	0	3

Variables w_{sd} nurse assigned to shift s on day d and y_{id} the shift assigned for each day

$$w_{sd} \in \{A, B, C, D\} \qquad y_{id} \in \{0, 1, 2, 3\}$$

Three different nurses are scheduled each day

$$\texttt{alldiff}(w_{\cdot d}) \qquad \forall d$$

Every nurse is assigned to at least 5 days of work

$$\texttt{cardinality}(w_{\cdot\cdot} \mid (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6))$$

At most two nurses work any given shift

$$\texttt{nvalues}(w_{s\cdot} \mid 1, 2) \qquad \forall s$$

All shifts assigned for each day

$$\texttt{alldiff}(y_{\cdot d}) \qquad \forall d$$

Maximal sequence of consecutive variables that take the same values

$$\texttt{stretch-cycle}(y_{i\cdot} | (2, 3), (2, 2), (6, 6), P) \qquad \forall i, P = \{(s, 0), (0, s) | s = 1, 2, 3$$

Channeling constraints between the two representations:
on any day, the nurse assigned to the shift to which nurse i is assigned must be nurse i

$$w_{y_{id}, d} = i \qquad \forall i, d$$

$$y_{w_{sd}, d} = s \qquad \forall s, d$$

Global Constraint Catalog
http://www.emn.fr/x-info/sdemasse/gccat/

Solved by

- Constraint Propagation (Edge filtering)
- Search: branch on domains (first fail)
- Symmetry breaking

Local search methods and metaheuristics are used if the problem has large scale. Procedures very similar to what we saw for timetabling.