DMP204
SCHEDULING,
TIMETABLING AND ROUTING

Lecture 23
Workforce Scheduling

Marco Chiarandini

Outline

Outline

1. Transportation Timetabling

2. Workforce Scheduling
   Crew Scheduling and Rostering
   Employee Timetabling
      Shift Scheduling
      Nurse Scheduling

Periodic Event Scheduling Problem

Blackboard

Outline

1. Transportation Timetabling

2. Workforce Scheduling
   - Crew Scheduling and Rostering
   - Employee Timetabling
     - Shift Scheduling
     - Nurse Scheduling

Workforce Scheduling

Overview

A note on terminology
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.

Workforce Scheduling

Overview

Workforce Scheduling:

1. Crew Scheduling and Rostering
2. Employee Timetabling

1. Crew Scheduling and Rostering is 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.)

   The peculiarity is finding logistically feasible assignments.

Workforce Scheduling

Overview

2. 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.

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

Crew Scheduling

Input:

- A set of flight legs (departure, arrival, duration)
- A set of crews

Output: A subset of flights feasible for each crew

How do we solve it?

Set partitioning or set covering??

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

Shift Scheduling

Creating daily shifts:

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

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

(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.

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

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

Recognizing total unimodularity can be done in polynomial time (see [Schrijver, 1986])

Total Unimodular Matrices

Resume'

Basic examples:

Theorem

The $V \times E$-incidence matrix of a graph $G = (V, E)$ is totally unimodular if and only if G is bipartite

Theorem

The $V \times A$-incidence matrix of a directed graph $D = (V, A)$ is totally unimodular

Theorem

Let $D = (V, A)$ be a directed graph and let $T = (V, A_0)$ be a directed tree on V. Let M be the $A_0 \times A$ matrix defined by, for $a = (v, w) \in A$ and $a' \in A_0$

$$M_{a',a} := \begin{cases} +1 & \text{if the unique } v - w\text{-path in } T \text{ passes through } a' \text{ forwardly;} \\ -1 & \text{if the unique } v - w\text{-path in } T \text{ passes through } a' \text{ backwardly;} \\ 0 & \text{if the unique } v - w\text{-path in } T \text{ does not pass through } a' \end{cases}$$

M is called network matrix and is totally unimodular.

Total Unimodular Matrices

Resume'

All totally unimodular matrices arise by certain compositions from network matrices and from certain 5×5 matrices [Seymour, 1980]. This decomposition can be tested in polynomial time.

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 is called an interval matrix and they can be shown to be network matrices by taking a directed path for the directed tree T

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

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

From LP1 and LP2 an integral solution certainly arises (P)

Cyclic Staffing with Overtime

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

minimize cx

subject to

07	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
08	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
09	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
10	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
11	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
12	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
13	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
14	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
15	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
16	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
17	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
18	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
19	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
20	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
21	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
22	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
23	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
24	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
01	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
02	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
03	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
04	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
05	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
06	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

x ≥ b

x ≥ 0 and integer.

Days-Off Scheduling

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

IP with matrix A:

first week	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
second week	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

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

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

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

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

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

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

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

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

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

The complete CP model

Alldiff: $\left\{ \begin{array}{c} (w_{\cdot d}) \\ (y_{\cdot d}) \end{array} \right\}$, all d

Cardinality: $(w_{\cdot\cdot} \,|\, (A, B, C, D), (5, 5, 5, 5), (6, 6, 6, 6))$

Nvalues: $(w_{s\cdot} \,|\, 1, 2)$, all s

Stretch-cycle: $(y_{i\cdot} \,|\, (2, 3), (2, 2), (6, 6), P)$, all i

Linear: $\left\{ \begin{array}{l} w_{y_{id}d} = i, \text{ all } i \\ y_{w_{sd}d} = s, \text{ all } s \end{array} \right\}$, all d

Domains: $\left\{ \begin{array}{l} w_{sd} \in \{A, B, C, D\}, \; s = 1, 2, 3 \\ y_{id} \in \{0, 1, 2, 3\}, \; i = A, B, C, D \end{array} \right\}$, all d

Constraint Propagation:

- alldiff: matching
- nvalues: max flow
- stretch: poly-time dynamic programming
- index expressions $w_{y_{id}d}$ replaced by z and constraint: `element`(y, x, z): z be equal to y-th variable in list $x_1, \ldots, x_m$

Search:

- branching by splitting domanins with more than one element
- first fail branching
- symmetry breaking:
  - employees are indistinguishable
  - shifts 2 and 3 are indistingushable
  - days can be rotated

  Eg: fix A, B, C to work $1, 2, 3$ resp. on sunday

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