MONASH UNIVERSITY
FACULTY OF BUSINESS & ECONOMICS

USING MIXED INTEGER PROGRAMMING TO DESIGN EMPLOYEE ROSTERS

Nicholas Beaumont

Working Paper 40/97
April 1997

ABSTRACT

This paper describes the problem of rostering a workforce so as to optimise a weighted sum of three criteria while satisfying several constraints. The rostering entailed deciding on a pattern of working days and breaks over a period of (typically) one year. Demand had to meet 24 hours each day and 356 days each year.

It was possible to formulate this problem as a mixed integer program and, with some experimentation, solve it using an 'off the shelf' linear programming package. The results obtained are compared with rosters the client now uses.

Keywords: manpower planning, mixed integer programming, workforce scheduling

Introduction

The client for which this research was done employed staff who drove to and serviced customers. The demand varied with day of week and (especially) time of day, and had to be met 24 hours each day and 365 days each year. Weekends and public holidays had no special status and there was no discernible seasonal variation in demand. Queuing was undesirable albeit inevitable but its cost had not been quantified by the client. The identity of the client and the kind of work done is confidential but analogous activities are repairpeople travelling between and servicing faulty lifts and police travelling between and dealing with incidents.

The workforce comprised the client's own employees (paid by the week) and self employed contractors who were available on call and paid a fixed amount per call. Only the former were rostered. Employees got no extra pay for working on weekends or public holidays.

The 'cycle' problem (as this rostering problem was called in by client) required finding a pattern of days off (known as rostered days off or RDO's) and working days (days on) which satisfied criteria agreed between the employees and management and had desirable features (see page 4).

The system had to be able to design cycles of any length. The most commonly used cycle comprised 48 working weeks and 4 weeks leave. It was assumed that one four week holiday would be taken but the cycles generated allowed this to be split into separate 2 week holidays thus making leave arrangements more flexible. Staff cycle through the cycle, moving from week 48 to week 1 (usually after annual leave). For example, if 104 people were employed, then two staff would be working each week of a 48 week cycle so 96 would be working and 8 would be on leave. It was possible to use different cycles simultaneously. At one time 48, 20 and 1 week cycles were being used (the last means that the employee works the same days each week) but management wanted to move to a single cycle. It is possible for two different cycles of the same length to be used simultaneously. Negotiations with the union meant that client was now had to design 47 week cycles.

It should be emphasised that constructing cycles to meet various objectives was an aspect of long term planning unaffected by short-term variation in demand (affected by weather) or supply (an absent worker). Daily variation was met by changing the number of contractors used or (when possible) calling on staff prepared to work an extra day.

That designing cycles by hand was a difficult and tedious process was manifest in the client not being able to design cycles which obeyed rules agreed by the client and the labour union. The client had a small repertoire of cycles and it was anxious to know whether these were optimal or nearly optimal and whether cycles meeting the requirements could be generated. If the agreement between with the labour union was renegotiated, it wanted

to be able to produce new cycles quickly, partly so as to help estimate how much any change in the agreement might cost it. This was exemplified by the need to change from a 48 to a 47 week cycle.

The problem of designing a cycle was fairly simply expressed as a mixed integer program (MIP). There were two contradictory objectives — workers preferred long workstretches (a set of consecutive working days is called a workstretch) and hence long breaks but management preferred that the number of people on duty on any day of the week be proportional to that day's mean demand.

Once a cycle (or cycles) had been agreed on, a subsequent problem was deciding the times at which employees should start each day's work. This problem could be expressed as another, large and computationally difficult, mixed integer program. This problem (which is of great practical importance because demand varies markedly with the time of day) will be discussed in a separate paper. It was computationally impractical to combine the two optimisations.

Past Work

The rostering problem (the problem of designing patterns of days on and days off) is a fairly old problem with many variants. The workforce may or may not be homogenous— some workers may be more efficient than others, paid more or able to undertake a wider range of duties. The work done may be more or less homogenous, may vary markedly with time of day or day of week and may be deferable (e.g. by making clients queue). The scheduling constraints vary: a certain proportion of days off may have to fall on weekends, there may be minimum or maximum limits on the length of breaks and workstretches.

Depending on the problem, either finding a feasible solution or finding the best of many feasible solutions may be emphasised. Solution methods can be broadly classified as based on heuristics or combinatorial optimisation.

Heuristics

The algorithm given by Burns and Carter[1] yields rosters which use the minimum possible number of workers. For the class of problems considered,

simply obtained lower bounds on the number of workers required are shown to be sufficient. The algorithm would not be useful in the current problem because it allows single days off and has a single objective function.

Khoong[2] uses, as part of the *ROMAN* rostering package, a heuristic which yields a feasible solution and bound for a subsequent branch and bound process. This heuristic, not relevant for this problem, implements the desirable idea that shift start times for an individual should move forward on the clock.

Bechtold, Brusco & Michael[3] exemplify the approach of generating numerous feasible rosters and finding the few which maximise desirable criteria. Easton & Rossin[4] generate a set of feasible rosters and use a heuristic to try and find improvements to them.

Combinatorial Optimisation

Early examples of the use of integer programming in rostering problems is given by Baker & Magazine[5] and Bartholdi, Orlin & Ratcliff [6]. The former considers the maximum workstretch constraint and gives limited recognition of demand varying with the day of the week.

Ferreira and Guimarães[7] discuss the problem of allocating staff to non-homogenous duties prior to creating a feasible roster. In their case, staff were paid the same salary regardless of the duties assigned, the problem would become more complicated if one had to consider the possibilty of using a skilled (and more highly paid) worker to do unskilled work. Ferreira and Guimarães use travelling salesman techniques to minimise the variance of rest periods' lengths. Bianco et al[8] use integer programming to create rosters which allocate work fairly amongst staff.

The solution of a problem with some similarities to ours is given by Townsend[9]. An aspect of this problem was there being several different duties which had to be distributed fairly amongst crews. The rules governing the pattern of days on and days off was simpler than those described in this paper. The objectives were different e.g. it was important to spread weekend work evenly through the roster.

Set covering techniques have been used in the more complex aircrew scheduling by, for example, Ryan[10]. This technique uses rosters as an input, one aspect of the technique is allocating rosters to staff.

Hung[11–13] considers problems complicated by factors such as a non- homogenous labour force (one kind of worker can replace another, but not vice

versa). These papers tend to emphasise problems in which the same number of days are worked each week. Some attention is given to the important practical requirement that the days worked each week be contiguous.

Packages

At least two computer packages which purport to solve rostering problems are described in the literature. Dawson[14] describes a heuristic package called *WORKPLAN* and rightly alludes to the sometimes considerable difficulties in implementing a new roster. Renegotiation of industrial agreements and persuading staff to change their work patterns and undertake different kinds of work are all possible obstacles. Khoong[2] describes the *ROMAN* package.

Expressing the Cycle Problem as an MIP

Preliminary

A cycle is defined as a sequence of L days each of which are marked as either a day on (work day or WD) or a day off (rostered day off or RDO). The problem is to find a cycle of nominated length which satisfies the constraints and provides the desirable features listed below.

Constraints The agreement between the client and the relevant union dictated that:

1. No worker shall be on duty for fewer than four or more than seven consecutive days.
2. No worker shall be on duty for more than five days in a calendar week (which starts on Sunday).
3. A break must comprise two to four consecutive days.
4. The mean number of hours worked per week must be very close to 38. This, and an 8 hour shift implies that $47 * 38/8 = 223.25$ so 223 or 224 days must be worked in a 47 week cycle.

Desired Features

1. Long workstretches and long breaks (sequences of RDO's) are preferred by employees.

2. The numbers of days worked in each 30 day month should be approximately equal.

3. The numbers of workers on duty on any given day of the week should be roughly proportional to the demand on that day (For example, Monday is traditionally busy, and there should be relatively few Monday RDO's). The mean demands on each day of the week in 1993 are given in the first line of table 1.

Formulation

The cycle problem can be formulated as follows:

Parameters and Subscripts

L The length of a cycle (in days). L must be a multiple of seven.

$i \in I = \{1, 2 \ldots, L\}$ indexes the days in the cycle.

$m \in M = \{1, 2, \ldots, \lfloor L/30 \rfloor\}$ indexes the complete 30 day months of the cycle ($\lfloor x \rfloor$ means the integer part of x).

$w \in W = \{1, 2, \ldots, 7\}$ indexes the days of the week (which start on Sunday).

Constants

D_w $w \in W$ The mean demand on each day of the week.

$D = \sum_{w \in W} D_w$ The total mean weekly demand.

W_1, W_2, W_3 Weights given to objective function terms.

S_l, S_h The minimum and maximum number of consecutive days allowed on duty (now four and seven respectively).

$DMAX$ The maximum number of days that can be worked in a calendar week (now five).

F_l, F_h The minimum and maximum number of consecutive days allowed off duty (now two and four respectively).

C_l, C_h The minimum and maximum number of days an employee must work in a cycle of length L to very nearly average the required number (now 38) of hours/week. These are now 223 and 224 respectively for 47 week cycle.

Variables

$\delta_i, i \in I$ (binary) is 1 if i is a workday, 0 if it is a rostered day off.

$y_i, i \in I$ is 1 if i is the first day of a workstretch, otherwise 0.

$u_w, w \in W$ The difference between the mean demand on a given day of the week and the number of employees on duty on that day both expressed as fraction of the mean total weekly demand and the total number of working days in the cycle respectively (see constraints (16) and (17)).

$z_m, m \in M$ The number of days worked in each 30 day period.

$v_m, m \in M$ The absolute deviation of z_m from its mean.

Constraints No more than $DMAX$ days may be worked in any calendar week.

$$\sum_{k=j-6}^{j} \delta_k \leq DMAX \quad j = 7, 14, \ldots, L. \tag{1}$$

Employees cannot work more than S_h consecutive days. In this and constraints (3)–(6), (8)–(11), because day $L+1$ is identical to day 1, the subscript must be interpreted as $((j-1)|L)+1$ where $i|j$ denotes the remainder on dividing integer i by integer j.

$$\sum_{j=i}^{i+S_h} \delta_j \leq S_h \quad i \in I. \tag{2}$$

Employees cannot have more than F_h consecutive days off.

$$\sum_{j=i}^{i+F_h} \delta_j \geq 1 \quad i \in I. \tag{3}$$

At least S_l consecutive days must be worked. Thus a day off must be followed by another day off or at least S_l days on duty. We require that

$$(\delta_i = 0 \wedge \delta_{i+1} = 1) \Rightarrow \delta_{i+k} = 1,\ k = 2, 3, \ldots, S_l \quad i \in I. \tag{4}$$

This can be expressed as:

$$\delta_i + (1 - \delta_{i+1}) + \delta_{i+k} \geq 1 \quad k = 2, 3, \ldots, S_l \quad i \in I \text{ or} \tag{5}$$

$$\delta_i - \delta_{i+1} + \delta_{i+k} \geq 0 \quad k = 2, 3, \ldots, S_l \quad i \in I. \tag{6}$$

When the δ_i take integer values, these constraints are always slack except when $\delta_i = 0$ and $\delta_{i+1} = 1$.

The total number of days worked is bounded

$$C_l \leq \sum_{i \in I} \delta_i \leq C_h. \tag{7}$$

There must be at least F_l consecutive days off, i.e.

$$(\delta_i = 1 \wedge \delta_{i+1} = 0) \Rightarrow \delta_{i+k} = 0 \quad k = 2, 3, \ldots, F_l \quad i \in I, \tag{8}$$

which can be expressed as:

$$(1 - \delta_i) + \delta_{i+1} + (1 - \delta_{i+k}) \geq 1, \quad k = 2, 3, \ldots, F_l \quad i \in I \text{ or} \tag{9}$$

$$\delta_i - \delta_{i+1} + \delta_{i+k} \leq 1 \quad k = 2, 3, \ldots, F_l \quad i \in I. \tag{10}$$

When the δ_i take integer values, these constraints are always slack except when $\delta_i = 1$ and $\delta_{i+1} = 0$.

It is generally desirable to have long workstretches and long breaks and accordingly y_i is defined for use in the objective function as:

$$y_i \geq \delta_i - \delta_{i-1} \quad i \in I \tag{11}$$

$$y_i \leq 1 \quad i \in I, \tag{12}$$

Although these bounds are ostensibly redundant (the direction of optimisation forces each y_i to 0 or 1) they hasten solution.

It is desirable that the number of days worked be spread evenly through the eleven 30 day months of the working year. The differences in each month from the average number of work days per month should be as small as possible. Accordingly v_m is defined for use in the objective function as:

$$z_m = \sum_{i=30m-29}^{30m} \delta_i \quad m \in M \tag{13}$$

$$v_m \geq z_m - C_l/30 \quad m \in M \tag{14}$$

$$v_m \geq -z_m + C_l/30 \quad m \in M. \tag{15}$$

It is desirable that the ratios of the total workforce available on any given day of the week should, as nearly as possible, match the ratios of the mean demand for that day of the week. We define:

$$u_w \geq (D_w/D - (\sum_{i=w,w+7,\ldots}^{L} \delta_i)/C_l \quad w \in W \tag{16}$$

$$u_w \geq (-D_w/D + (\sum_{i=w,w+7,\ldots}^{L} \delta_i)/C_l \quad w \in W \tag{17}$$

The direction of optimisation will result in each $u_w, w \in W$ being made as small as possible.

The Objective Function The objective function comprises weighted terms each of which represents a desirable feature of solutions to the problem. These features and their weights are:

- Employees prefer long workstretches and long breaks (W_1).
- The number of days worked in all 30 day months should be approximately equal (W_2).
- The number of staff on duty on each day of the week should be proportional to the demand (W_3).

The objective function is:

$$\min Z = W_1 \sum_{i \in I} y_i + W_2 \sum_{m \in M} v_m + W_3 \sum_{w \in W} u_w \tag{18}$$

Solution and Results

Initial runs prompted a search for ways of more tightly constraining the problem. It was noted that many of the binary variables had a value of 228/328 in the solution of the 48 week relaxation. Although constraint (3) express the fact that the breaks cannot be longer than F_h; we tried adding

$$\delta_i - \delta_{i+1} \leq \delta_{i+F_h+k} \quad k = 1, 2, \; i \in I. \tag{19}$$

This expresses the implication $\delta_i = 1 \wedge \delta_{i+1} = 0 \Rightarrow \delta_{i+F_h+k} = 1 \quad k = 1, 2, \; i \in I$. Although these constraints (or cutting planes) can be inferred from other constraints expressing logical conditions, including them very slightly lessened the integrality gap and markedly decreased run times. In the relaxation these constraints forced more binary variables away from 228/328 to values at or closer to zero or 1.

Finding efficient cutting planes is a problem with a very large literature (see Balas[16] for an overview). An elegant method of generating a hierarchy (which includes the conventional relaxation and the convex hull) of sets of cutting planes for MIP problems is given by Sherali and Adams[17].

Multiplying constraints 16 and 17 by $D*C_l$ reduced the time to find a MIP solution even though some coefficients thus introduced were anomalously large. For each solution of the 48 week problem there are 47 other, equivalent, solutions obtainable by moving the cycle forward 1, 2, ...47 weeks. This can impede the search for genuinely new solutions. Cyclicity was partly eliminated by setting $\delta_i, i = 335, 336, 1, 2$ to $1, 0, 0, 1$ respectively (to avoid single work days adjacent to holidays).

Despite these measures, the integrality gap was large for both problems (the optimum of the relaxed problem was nearly zero).

The problem of evening staffing over months was considered comparatively unimportant by the client and not considered, constraints (13), (14) and (15) and some variables were dropped from the model. The problem statistics for the 48 week problem were then: 3025 constraints, 678 variables (329 binary) and 12831 non-zero coefficients.

The package CPLEX 3.0[15] was used on a CRAY computer. The package's default strategy was used. In the Branch and Bound tree, the node with the best objective function value is processed next. The rule used to select a variable for arbitration and the first used direction of arbitration were

decided by the program itself[15] (p 137). A parameter was set to ensure that successive MIP solutions had appreciably better solutions.

Each problem was run for 1000 'seconds' (this would have cost a commercial organisation $A33.00). For most problems, a solution was found early (after less than 100 'seconds') and not improved on.

Comparison with Existing 48 Week Cycles

The client provided two 48 week cycles (identified as A and B). These had the same number of RDO's on each day of the week. Neither met the client's own requirements (each had a one day and one (A) or two (B) five day breaks). Three 48 week cycles were computed (C, D, E): C attempted to match supply and demand, E to give long breaks and the E gave equal weight to both objectives. The five cycles' characteristics are compared in table 1.

47 Week Cycles

The client required a sample of 47 week cycles; Three were generated (F, G, H). F attempted to match supply and demand, G to give long breaks and the H gave equal weight to both objectives. The characteristics of F and H are given in table 2. G is omitted because it was dominated by H.

Implementation

The problem requires little input data and it was felt appropriate to store this in a spreadsheet. Some of the users were used to spreadsheets and the MIP package could read data from them. A considerable advantage is that it is possible to copiously annotate (in neighbouring cells) the contents of cells containing input data. The annotations can include instructions, warnings and advice. It is also possible to incorporate tests for the validity of input data (this has not yet been done).

The LP package wrote the results to a binary file (the standard output file would have been neither compact nor comprehensible to the users) and a program was written to read the output and produce a report detailing a cycle and giving summary statistics.

Conclusion

At present, the client finds it difficult and laborious to generate cycles by hand (in fact some cycles include ostensibly forbidden single days off). The method allows a variety of cycles to be generated fairly conveniently. This would be especially important if arrangements were being negotiated with the union.

This problem generated a moderately difficult combinatorial problem with a large integrality gap. It exemplifies the value of adding ostensibly redundant constraints and the need to modify the formulation so as to make the problem more computationally tractable. This exercise demonstrates that moderately complicated rostering problems can be solved by expressing them as MIP problems which can to be routinely solved using modern hardware and software.

Acknowledgment The author would like to thank Mr Jim Youngman of the client for introducing us to this problem and his help, especially in obtaining data and software. He would also like to thank Professor R. D. Snyder and anonymous referees for incisive comments which have greatly improved this paper.

References

1. BURNS R. N. & CARTER M. W. (1985) Work force size and single shift schedules with variable demands. *Mgmt Sci.* **31**, 599–607.

2. C. M. KHOONG (1993) A simple but effective heuristic for workshift assignment. *Omega* **21**, 393–395.

3. S. E. BECHTOLD & M. J. BRUSCO (1994) Working set generation methods for labor tour scheduling. *Eur. J. Opl. Res.* **74**, 540–551.

4. F. F. EASTON & D. F. ROSSIN (1991) Equivalent alternate solutions for the tour scheduling problem. *Decision Sciences* **22**, 985-1007. .

5. K. R. BAKER & M. J. MAGAZINE (1977) Workforce scheduling with cyclic demands and day-off constraints. *Mgmt Sci.* **24**, 161–170.

6. J. J. BARTHOLDI, J. B. ORLIN & H. D. RATCLIFF (1980) Cyclic scheduling via integer programs with circular ones. *Opns. Res.* **28**, 1075- -1085.

7. J. V. FERREIRA & R. C. GUIMARÃES (415– 426) A travelling salesman model for the sequencing of duties in bus crew rotas. *J. Opl Res. Soc.* **46**, 1995.

8. L. BIANCO, M. BIELLI, A. MINGOZZI, S. RICCIARDELLI & M. SPADONI (1992) A heuristic procedure for the crew rostering problem. *Eur. J. Opl. Res.* **58**, 272–283.

9. W. TOWNSEND (1988) An approach to bus-crew roster design in London Regional Transport. *J. Opl Res. Soc.* **39**, 543–550.

10. D. M. RYAN (1992) The solution of massive generalized set partitioning problems in aircrew rostering. *J. Opl Res. Soc.* **43**, 459–467.

11. R. HUNG (1994) Single-shift off-day scheduling of a hierarchical workforce with variable demands. *Eur. J. Opl. Res.* **78**, 49–57.

12. R. HUNG (1994) A multiple-shift workforce scheduling model under the 4-day workweek and weekend labour demands. *Journal of the Opns. Res. Society* **45**, 1088–1092.

13. R. HUNG (1994) Multiple-shift workforce scheduling under the 3-4 workweek with different weekday and weekend labor requirements. *Mgmt Sci.* **40**, 280–284.

14. DAWSON K. (1987) Rostering in the Process Industries. *Mangement Services* **31**, 12–17.

15. CPLEX OPTIMIZATION, INC (1995) Using the CPLEX callable library and CPLEX mixed integer optimizer. CPLEX Optimization, Inc.

16. BALAS E. (1985) Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems. *SIAM J. Alg. Disc Meth.* **6**, 466– 486. .

17. SHERALI, H. D. ADAMS, W. P. (1994) A hierarchy of relaxations and convex hull characterisations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics* **52**, 83–106.

	Roster	Characteristic	Sun	Mon	Tue	Wed	Thu	Fri	Sat	TOTAL	Number of Shifts
1		Mean demand (calls/day)	2349	3212	2827	2929	2939	2954	2504	19714	
2		As a fraction of the total	0.119	0.163	0.143	0.149	0.149	0.150	0.127		
3	A and B	Total days worked	33	33	32	31	31	35	33	228	35, 36
4		As a fraction of the total	0.145	0.145	0.140	0.136	0.136	0.156	0.145		
5		|row2- row4|	0.005	0.008	0.045	0.009	0.026	0.005	0.026	0.125	
6	C	Total days worked	26	39	43	36	28	33	23	228	45
7		As a fraction of the total	0.114	0.171	0.189	0.158	0.123	0.145	0.101		
8		|row2- row7|	0.026	0.018	0.003	0.013	0.013	0.004	0.018	0.094	
9	D	Total days worked	16	39	42	41	37	37	16	228	39
10		As a fraction of the total	.070	.171	.184	.180	.162	.162	.070		
11		|row2- row10|	0.049	0.008	0.041	0.031	0.013	0.012	0.057	0.212	
12	E	Days worked	24	25	36	39	38	36	30	228	43
13		As a fraction of the total	0.105	0.110	0.158	0.171	0.167	0.158	0.132	1.0	
14		|row2- row13|	0.014	0.053	0.014	0.022	0.018	0.008	0.005	00.134	

Table 1: Comparison of 48 week cycles

	Roster		Sun	Mon	Tue	Wed	Thu	Fri	Sat	TOTAL	Number of shifts
1		Mean demand (calls/day)	2349	3212	2827	2929	2939	2954	2504	19714	
2		As a fraction of the total	0.119	0.163	0.143	0.149	0.149	0.150	0.127		
3	F	Total days worked	29	35	33	32	33	31	30	223	45
4		As a fraction of the total	0.130	0.157	0.148	0.143	0.148	0.139	0.135	1.0	
5		|row2- row4|	0.011	0.006	0.005	0.005	0.001	0.011	0.008	0.046	
6	H	Days worked	28	31	32	32	36	34	30	223	40
7		As a fraction of the total	0.126	0.139	0.143	0.143	0.161	0.152	0.135		
8		|row2- row7|	0.006	0.024	0.0	0.005	0.012	0.003	0.008	0.058	

Table 2: Comparison of 47 week cycles