TECHNICAL REPORT

Computational results on new staff scheduling benchmark instances

Tim Curtois, Rong Qu

ASAP Research Group, School of Computer Science, University of Nottingham, NG8 1BB, Nottingham, UK

First published online: 19-Sep-2014, last updated: 06-Oct-2014.

This report lists results of applying the algorithms presented in [2] to the staff scheduling problem benchmark instances 1..24 [3]. The algorithms are an ejection chain metaheuristic and a branch and price method. The branch and price method was shown to be very effective on smaller and medium sized instances, often finding the optimal solution. Its weakness is on the larger instances on which it may run out of memory trying to solve a sub-problem. The metaheuristic is a more robust and practical method. Although it is outperformed on the smaller instances, it will still find good solutions on the larger instances if given sufficient time. For additional comparisons we have also included the results of applying Gurobi 5.6.3 [1] to an integer programming formulation.

Instances

Many of the original benchmark instances available at [3] can now be easily solved [2]. Most of the original instances are also quite difficult to use due to their real world nature. They contain many different types of constraints and objectives which are complicated to model and implement whatever type of solving approach is being used (integer programming, metaheuristic, etc). For these reasons the collection of instances has been recently supplemented with a new set of instances. The new instances are designed to reflect real world requirements and scheduling scenarios yet still be easy to use. They are also designed to represent a range of difficulty: from very easy to very challenging. To make them easier to use and test, the number of constraint and objective types has been reduced to a core of constraints found commonly in staff rostering problems. The new instances are also given in a plain text format which is a lot simpler to parse and use. This allows researchers to spend less time writing code for parsing the instances and more time on developing the algorithms and producing results. Table 1 lists the instances and their dimensions. They range from very small (8 staff, 2 weeks, 1 shift type) to very large (150 staff, 52 weeks, 32 shift types).

Instance	Planning horizon (weeks)	Staff	Shift types
Instance1	2	8	1
Instance2	2	14	2
Instance3	2	20	3
Instance4	4	10	2
Instance5	4	16	2
Instance6	4	18	3
Instance7	4	20	3
Instance8	4	30	4
Instance9	4	36	4
Instance10	4	40	5
Instance11	4	50	6
Instance12	4	60	10

Instance13	4	120	18
Instance14	6	32	4
Instance15	6	45	6
Instance16	8	20	3
Instance17	8	32	4
Instance18	12	22	3
Instance19	12	40	5
Instance20	26	50	6
Instance21	26	100	8
Instance22	52	50	10
Instance23	52	100	16
Instance24	52	150	32

Table 1 Benchmark instances

Integer Programming Formulation

An integer programming model for the problem is given below. All instances start on a Monday and the planning horizon h is always a whole number of weeks (h mod 7 = 0).

Parameters:

I set of employees.

h number of days in the planning horizon.

D set of days in the planning horizon = {1…h}.

W set of weekends in the planning horizon = {1...h/7}.

T set of shift types.

R_t set of shift types that cannot be assigned immediately after shift type t.

N_i set of days that employee i cannot be assigned a shift on.

l_t length of shift type t in minutes.

$m_{it}^{\max}$ maximum number of shifts of type t that can be assigned to employee i.

$b_i^{\min}$ minimum number of minutes that employee i must be assigned.

$b_i^{\max}$ maximum number of minutes that employee i can be assigned.

$c_i^{\min}$ minimum number of consecutive shifts that employee i must work.

$c_i^{\min}$ maximum number of consecutive shifts that employee i can work.

$o_i^{\min}$ minimum number of consecutive days off that employee i can be assigned.

$a_i^{\max}$ maximum number of weekends that employee i can work.

q_{idt} penalty if shift type t is not assigned to employee i on day d.

p_{idt} penalty if shift type t is assigned to employee i on day d.

u_{dt} preferred total number of employees assigned shift type t on day d.

$v_{dt}^{\min}$ weight if below the preferred cover for shift type t on day d.

$v_{dt}^{\max}$ weight if exceeding the preferred cover for shift type *t* on day *d*.

Decision variables:

x_{idt} 1 if employee *i* is assigned shift type *t* on day *d*, 0 otherwise

k_{iw} 1 if employee *i* works on weekend *w*, 0 otherwise

y_{dt} total below the preferred cover for shift type *t* on day *d*.

z_{dt} total above the preferred cover for shift type *t* on day *d*.

Constraints:

1. An employee cannot be assigned more than one shift on a single day.

$$\sum_{t \in T} x_{idt} \leq 1, \quad \forall i \in I, d \in D$$

2. Shift rotation. A minimum amount of rest is required after each shift. Therefore certain shifts cannot follow others. For example, an early shift cannot follow a late shift.

$$x_{idt} + x_{i(d+1)u} \leq 1, \quad \forall i \in I, d \in \{1...h-1\}, t \in T, u \in R_t$$

3. The maximum numbers of shifts of each type that can be assigned to employees. For example, some employees will have contracts which do not allow them to work night shifts or only a maximum number of night shifts.

$$\sum_{d \in D} x_{idt} \leq m_{it}^{\max}, \quad \forall i \in I, t \in T$$

4. Minimum and maximum work time. The total minutes worked by each employee must be between a minimum and maximum. These limits can vary depending on whether the employee is full-time or part-time.

$$b_i^{\min} \leq \sum_{d \in D} \sum_{t \in T} l_t x_{idt} \leq b_i^{\max}, \quad \forall i \in I$$

5. Maximum consecutive shifts. The maximum number of shifts an employee can work without a day off. For example, part-time employees sometimes do not work as many consecutive shifts as full-time staff.

$$\sum_{j=d}^{d+c_i^{\max}} \sum_{t \in T} x_{ijt} \leq c_i^{\max}, \quad \forall i \in I, d \in \{1...h - c_i^{\max}\}$$

6. Minimum consecutive shifts. This can be modelled by preventing every sequence of consecutive shifts below the minimum. For example, if the minimum number of consecutive shifts is four then we must not allow any of the sequences: {*off-on-off, off-on-on-off, off-on-on-on-off*} where *off* is a day without a shift and *on* is a day with a shift assigned.

$$\sum_{t \in T} x_{idt} + \left(s - \sum_{j=d+1}^{d+s} \sum_{t \in T} x_{ijt} \right) + \sum_{t \in T} x_{i(d+s+1)t} > 0, \quad \forall i \in I, s \in \{1...c_i^{\min} - 1\}, d \in \{1...h - (s+1)\}$$

7. Minimum consecutive days off. This can be modelled in a similar way to the minimum consecutive shifts constraint. For example, if the minimum number of consecutive days off is three then we must not allow any of the sequences: {*on-off-on, on-off-off-on*}.

$$\left(1-\sum_{t\in T} x_{idt}\right)+\sum_{j=d+1}^{d+s}\sum_{t\in T} x_{ijt}+\left(1-\sum_{t\in T} x_{i(d+s+1)t}\right)>0, \qquad \forall i\in I, s\in\{1...o_i^{\min}-1\}, d\in\{1...h-(s+1)\}$$

8. Maximum number of weekends. A weekend is considered as being worked if the employee has a shift on the Saturday *or* the Sunday.

$$k_{iw}\le\sum_{t\in T} x_{i(7w-1)t}+\sum_{t\in T} x_{i(7w)t}\le 2k_{iw}, \qquad \forall i\in I, w\in W$$

$$\sum_{w\in W} k_{iw}\le a_i^{\max}, \qquad \forall i\in I$$

9. Days off. These are days that employees cannot work because, for example, they are on vacation.

$$x_{idt}=0, \qquad \forall i\in I, d\in N_i, t\in T$$

10. Cover requirements.

$$\sum_{i\in I} x_{idt}-z_{dt}+y_{dt}=u_{dt}, \qquad \forall d\in D, t\in T$$

Objective function:

$$Minimise\sum_{i\in I}\sum_{d\in D}\sum_{t\in T} q_{idt}(1-x_{idt})+\sum_{i\in I}\sum_{d\in D}\sum_{t\in T} p_{idt}x_{idt}+\sum_{d\in D}\sum_{t\in T} y_{dt}v_{dt}^{\min}+\sum_{d\in D}\sum_{t\in T} z_{dt}v_{dt}^{\max}$$

The objective function models the requirement to maximise the allocation of employee shift requests and minimise under and over staffing. The parameters q_{idt} and p_{idt} are the weights for shift on and shift off requests respectively. For example, an employee may request to work a certain shift type on a particular day. The higher the weight, the more important the request is to the employee. If there is no request then the parameter has the value zero.

The variables y_{dt} and z_{dt} are the total numbers of staff below and above the preferred cover level for each shift type *t* on each day *d*. The parameters $v_{dt}^{\min}$ and $v_{dt}^{\max}$ are weights to represent the importance of minimising under and over coverage.

Results

To provide other researchers with results to compare against, we have used the two existing algorithms presented in [2] and Gurobi 5.6.3 [1] and applied them to the new instances.

All the experiments were performed on Intel Core 2 Duo 3.16GHz, 8GB ram. The Gurobi solver was limited to a single thread and a maximum time of 1 hour. Table 2. lists the results. Known optimal solutions are in **bold.**

				Ejection chain		Branch and Price			Gurobi 5.6.3		
Instance	Weeks	Staff	Shifts	10 min	60 min	LB	Sol.	Time (s)	LB	Sol.	Time (s)
Instance1	2	8	1	**607**	**607**	558	**607**	0.27	607	**607**	1.62
Instance2	2	14	2	923	837	828	**828**	0.13	828	**828**	5.22
Instance3	2	20	3	1003	1003	1001	**1001**	0.45	1001	**1001**	13.54
Instance4	4	10	2	1719	1718	1716	**1716**	1.50	1716	**1716**	158.99
Instance5	4	16	2	1439	1358	1141	1160	25.61	1143	**1143**	1520.24
Instance6	4	18	3	2344	2258	1949	1952	10.46	1950	**1950**	440.93
Instance7	4	20	3	1284	1269	1055	1058	93.73	1056	**1056**	2152.48
Instance8	4	30	4	2529	2260	1297	1308	11831.06	1281	1323	3599.83
Instance9	4	36	4	474	463	406	439	76.99	247	439	3599.85
Instance10	4	40	5	4999	4797	4631	**4631**	113.44	4631	**4631**	244.20
Instance11	4	50	6	3967	3661	3443	**3443**	19.11	3443	**3443**	109.92
Instance12	4	60	10	5611	5211	4040	4046	1336.4	4040	**4040**	2303.84
Instance13	4	120	18	8707	3037	Out of memory	-	-	1346	3109	3600.55
Instance14	6	32	4	2542	1847	Out of memory	-	-	1277	1280	3600.13
Instance15	6	45	6	6049	5935	Out of memory	-	-	3806	4964	3600.00
Instance16	8	20	3	4343	4048	3224	3323	265.02	3211	3233	3599.99
Instance17	8	32	4	7835	7835	Out of memory	-	-	5726	5851	3600.00
Instance18	12	22	3	6404	6404	Out of memory	-	-	4351	4760	3599.99
Instance19	12	40	5	6522	5531	Out of memory	-	-	2945	5420	3605.90
Instance20	26	50	6	23531	9750	Out of memory	-	-	4743	-	3600.05
Instance21	26	100	8	38294	36688	Out of memory	-	-	20868	-	3600.21
Instance22	52	50	10	-	516686	Out of memory	-	-	-	-	3600.19
Instance23	52	100	16	-	54384	Out of memory	-	-	-	-	3600.43
Instance24	52	150	32	-	156858	Out of memory	-	-	Out of memory	-	-

Table 2. Results

References

1. *Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual*. 2014; Available from: http://www.gurobi.com.
2. E.K. Burke and T. Curtois, New Approaches to Nurse Rostering Benchmark Instances, European Journal of Operational Research 237(1) (2014) 71–81.
3. T. Curtois. *Employee Shift Scheduling Benchmark Data Sets*. 2014; Available from: www.cs.nott.ac.uk/~tec/NRP.