Modelling History in Nurse Rostering

Jeffrey H. Kingston (University of Sydney)

The history issue

- Create nurses' timetables one week (say) at a time
- But constraints cover more than one week e.g.
  - At least 15 and at most 20 shifts per month
  - At most 5 consecutive busy days

Known for decades, made visible and concrete by INRC2

Arguably the biggest remaining roadblock to modelling nurse rostering

An example

Define a *global instance I* and *local instances* or *projections* I_i:

- Global instance *I* has 4 weeks, each with 7 days, each with 3 shifts
- Timetables needed week by week: I_1, I_2, I_3, I_4
- Global constraint *C*: a resource may work at most 20 days
- *Nurse1* worked 7 days during Week 1, and 7 days during Week 2
- How should *Nurse1*'s workload be constrained during I_3?

Counter start values and counter remainder values

Local instances must supply *counter start values* x_i and *counter remainder values* c_i.

Previous weeks (known)	Current week (known)	Future weeks (unknown)
x_i	y_i	c_i

x_i is the number of 'things' that happened before the current week.

y_i is the number of 'things' that are happening during the current week.

c_i is the number of 'things' that could possibly happen after the current week.

Compare $x_i + y_i$ with the global maximum limit.

Compare $x_i + y_i + c_i$ with the global minimum limit.

Nothing new so far, but …

- The ‘things’ could be busy days, free days, busy weekends, …, so counter start values and counter remainder values are a *general* and *uniform* way to handle history.
- So previous implementations seem daunting mainly because they apply the method to a daunting list of constraint types.

Counter start values and counter remainder values in XESTT

- XESTT nurse rostering instances use only 9 types of constraints
- Most constraints project trivially e.g. avoid unavailable times
- This leaves just cluster busy times constraints and limit active intervals constraints
- Handle them and you have a uniform and complete solution to the history issue

(The ‘things’ are busy time groups.)

Double counting of penalties

- Same example, suppose *Nurse1* works on all 28 days.
- C_3 has cost $3 \times 7 - 20 = 1$
- C_4 has cost $4 \times 7 - 20 = 8$

The total cost is 9, but it should be 8.

INRC2 handles this, but in an ad-hoc manner.

The paper shows how C_i can calculate the cost of C_{i-1} and so subtract it away.

Limit active intervals constraints

As before except that the limits apply to the number of *consecutive* things, not the *total* number of things.

Paper gives formulas for incorporating history and avoiding double counting.

x_i is the number of 'things' that happened *immediately preceding* the current week.

c_i is the number of 'things' that could possibly happen after the current week, as before.

Conclusion

- Fully defined and documented in XESTT
- Fully implemented in free public software (HSEval and KHE)
- User supplies x_i and c_i in local instance files, everything just happens
- Implementation based on complete formulas in the paper
- Arguably a complete solution to the history problem