Specifying Timetabling Problems:

An Ongoing Story

Jeffrey H. Kingston
University of Sydney

Timetable Models and Formats

- A model is a set of concepts defining instances and solutions of timetabling problems
- A format is a model expressed in a concrete syntax
- Needed for data exchange and verifying solutions
- But timetabling is a complicated real-world problem

Some History

1995 Seminar at ICPATAT'95 led by Cumming

1996 'Toronto' examination instances assembled by Carter, Laporte and Lee

2002 First International Timetabling Competition led by Paechter

2005/6 Nurse rostering instances assembled by Curtois

2006 Plenary talk on 'measurability and reproducibility' by Schaerf

2008 First paper on XHSTT, by Post, Kingston et al.

Many close associations with the PATAT conferences.

Issue 1: Real-World Detail

One view: real-world details need to be simplified

- Because there are too many to analyse
- Because implementing them all is not worth the cost

Another view: over time, we should aim for complete detail

- We want our field to progress, why not towards more detail?
- Leaving out some features excludes some instances and researchers
- If we never try, we will never know

Issue 2: Generality (or Abstraction)

Too little: many specific features, no analysis

- Implementing all the features is a heavy software burden
- Virtually no chance of handling new instances

Too much: a general language (integer programming, functional language, etc.)

- No hooks for a specifically timetabling-oriented solver
- Might cross the NP-completeness boundary unnecessarily

Good Generalizations

Good generalizations unify related concepts without creating new problems.

- Make all constraints visible, and make all constraint types subtypes of a common supertype which has *required*, *weight*, and *cost_function* attributes.
- Have *resources*: things that attend meetings and want no clashes, e.g. students, student groups, rooms, teachers, nurses, sports teams, sports fields, …
- Allow arbitrary sets of times and resources.

How to Model an Instance

Events (lectures, exams, lessons, meetings, shifts, games, …), each with:

- A *starting time* (preassigned or open for assignment)
- A *duration* (usually fixed)
- Any number of *resources* (preassigned or open for assignment)

Constraints, each with:

- Common attributes: *required*, *weight*, *cost_function*
- *Type*: prefer resources, unavailable times, limit workload, …
- Type-specific attributes: sets of times, upper and lower limits, …
- Semantics are *implicit*: no general expressions or functions

The XHSTT High School Timetabling Data Format

```
Instance
   Times
      Time
      TimeGroup
   Resources
      ResourceType
      Resource
      ResourceGroup
   Events
      Event
      EventGroup
   Constraints
      Constraint
```

XHSTT Constraint Example: the Limit Busy Times Constraint

- Teacher Smith may be busy for at most 7 out of the 8 times on Mondays.
- Nurse Jones wants at least 5 night shifts.

```
LimitBusyTimesConstraint Id
   Name
   Required
   Weight
   CostFunction
   AppliesTo
   TimeGroups
   Minimum
   Maximum
```

Archive files

One archive file holds:

- A set of instances
- A set of sets of solutions
- Metadata

Allows automatic generation of the usual tables comparing solvers.

Software Support

The more widely accepted the format, the more these are worth doing well:

- Precise documentation
- Solution evaluator on the web
- Solve platform
  - Reading and writing of archive files
  - Basic operations on solutions (assign and unassign time or resource)
  - Incremental evaluation as the solution changes

The Story Goes On

- More sub-disciplines with good, real-world, widely accepted models
- More detail—which basically means more constraints
- Time models: discrete times, … , sets of intervals of real time
- Segmented instances
- Unification across sub-disciplines – do we need it?

Proceed cautiously, based on real instances