UBC Exam Timetabling

Chris Fawcett

Presented for EARG

2010/11/24
Outline - background

- The general timetabling problem
- Post-enrollment course and exam timetabling
- History of timetabling in our group
- Our problem model
- Our solver structure
Outline - collaboration with UBC

- UBC’s current timetabling process
- Elicitation of constraints
- Necessary data cleaning and pre- and post-processing
- Ideal winter 2010 process schedule
- What actually happened
- Necessary changes and next steps
The general timetabling problem

- Given:
  - A set $A$ of activities
The general timetabling problem

- **Given:**
  - A set $A$ of activities
  - A set $T$ of available timeslots
  - A set $C$ of hard constraints

Produce a schedule (or timetable) $S$ assigning each $a \in A$ to a timeslot $t \in T$, such that:

- All constraints $c \in C$ are satisfied.

Easily extended to finding $S$ such that an objective function $f$ is optimized.
The general timetabling problem

Given:
- A set \( A \) of activities
- A set \( T \) of available timeslots
- A set \( C \) of hard constraints

Produce a schedule (or timetable) \( S \) assigning each \( a \in A \) to a \( t \in T \), such that:
- All constraints \( c \in C \) are satisfied.

Easily extended to finding \( S \) such that an objective function \( f \) is optimized.
The general timetabling problem

- **Given:**
  - A set $A$ of activities
  - A set $T$ of available timeslots
  - A set $C$ of hard constraints

- **Produce a schedule (or timetable) $S$ assigning each $a \in A$ to a timeslot $t \in T$, such that:**
  - All constraints $c \in C$ are satisfied.
The general timetabling problem

Given:
- A set $A$ of activities
- A set $T$ of available timeslots
- A set $C$ of hard constraints

Produce a schedule (or timetable) $S$ assigning each $a \in A$ to a timeslot $t \in T$, such that:
- All constraints $c \in C$ are satisfied.

Easily extended to finding $S$ such that an objective function $f$ is optimized.
Post-enrollment course and exam timetabling

- Activities are *courses* or *exams*. 
Activities are courses or exams.
Additionally given a set $S_t$ of students and a set $R$ of rooms.
Activities are *courses* or *exams*.

- Additionally given a set $St$ of students and a set $R$ of rooms.
- Each activity is attended by a subset of the students.
Activities are *courses* or *exams*.

Additionally given a set $S_t$ of students and a set $R$ of rooms.

Each activity is attended by a subset of the students.

Each activity must be assigned to both a timeslot $t \in T$ and a room $r \in R$. 
Activities are *courses* or *exams*.

Additionally given a set $S_t$ of students and a set $R$ of rooms.

Each activity is attended by a subset of the students.

Each activity must be assigned to both a timeslot $t \in T$ and a room $r \in R$.

An objective function $f$ based around the idea of soft constraints.
Example hard constraints

- Students shouldn’t have to attend two activities in the same timeslot.
Example hard constraints

- Students shouldn’t have to attend two activities in the same timeslot.
- Only one activity should be scheduled into each room.
Example hard constraints

- Students shouldn’t have to attend two activities in the same timeslot.
- Only one activity should be scheduled into each room.
- Room capacities should be respected.
Example hard constraints

- Students shouldn’t have to attend two activities in the same timeslot.
- Only one activity should be scheduled into each room.
- Room capacities should be respected.
- Groups of activities may need to be scheduled into the same timeslot or room.
Example hard constraints

- Students shouldn’t have to attend two activities in the same timeslot.
- Only one activity should be scheduled into each room.
- Room capacities should be respected.
- Groups of activities may need to be scheduled into the same timeslot or room.
- Each activity may need to be scheduled into only a subset of the timeslots.
Example hard constraints

- Students shouldn’t have to attend two activities in the same timeslot.
- Only one activity should be scheduled into each room.
- Room capacities should be respected.
- Groups of activities may need to be scheduled into the same timeslot or room.
- Each activity may need to be scheduled into only a subset of the timeslots.
- Each activity may need to be scheduled into a subset of the rooms, satisfying any additional features the activity requires (building, tables, projector, etc.).
Example soft constraints

- Students should not have two activities in the same day.
Example soft constraints

- Students should not have two activities in the same day.
- Students should not have three (or more) activities in four consecutive timeslots.
Example soft constraints

- Students should not have two activities in the same day.
- Students should not have three (or more) activities in four consecutive timeslots.
- Some activities should not be placed into certain timeslots (e.g., first-year courses in the last two days for December schedules at UBC).
Solver history

- In development beginning in late 2007.
- Third place in the post-enrollment track of the Second International Timetabling Competition (January 2008).
- Subsequently improved, achieving substantially better performance than the competition winner.
- Currently the state-of-the-art solver for this problem.
Collaboration with UBC classroom services beginning in winter 2009.

Problem size is 100x larger than seen in the competition.

Solver extension and improvements to support solving the UBC problem.

Dry run in winter 2010, parts of resulting schedule were used.

Hopefully full schedule used in Spring 2010.
Our problem model

- Six primitive objects:
  - Course
  - Room
  - Student
  - Timeslot
  - Feature
  - CourseGroup
Our problem model

- Courses have students, feature requirements, timeslot restrictions and a single group id.
- Rooms have capacities and features.
- Students have courses.
- Timeslots are ordered.
- CourseGroups have courses.
- Solver deals strictly with assigning groups to timeslots and rooms.
Our solver

- Randomised solver, designed to find good (but not necessarily optimal) solutions quickly.
- Extremely modular, with a general and flexible problem specification format.
- Leverages automated design and configuration techniques to tailor performance specifically to a given problem, in this case UBC’s exam scheduling.
- Partially automated pre-processing stage to convert UBC data into our format, including merging or splitting course sections as necessary.
Our solver

UBC DATA

Preprocessing
- Exam and Group Merges/Splits
- Room Region Specification
- Manual Changes
- Constraint/Objective Weights

General Problem Instance

Randomised Solver

General Solution

Translated Solution

Published Schedule

Human Expert
Solver algorithm specifics

Our solver

Chris Fawcett (EARG)  UBC Exam Timetabling  2010/11/24  14 / 26
The current UBC process

- Course DB
- Room DB
- Requests
- Text Files
- Schedule
- Expert
- Manual
- Changes
- Manual
- Room Assignment
- Final
- Schedule

Diagram:

- Course DB
- Room DB
- Requests
- Text Files
- Schedule
- Expert
- Manual
- Changes
- Manual
- Room Assignment
- Final
- Schedule
Elicitation of constraints

- Users have great difficulty expressing the constraints they care about.
- They often give misleading or incorrect information.
- Best progress achieved from asking hypothetical questions about groups of exams.
- UBC problem is inherently multi-objective, with opposing objective components.
- Some constraints will likely never exist explicitly in our solver.
Discovered constraints

- "Student conflicts": Two exams in the same timeslot.
- "Student hardships":
  - Two exams in the same day.
  - Three or more exams in the same period.
  - Two exams within 8 consecutive timeslots.

- Is this exhaustive? Definitely not, as we found out.
Data cleaning and preprocessing

- Convert UBC database reports to standard csv format.
- Each course has a “type”, with a timeslot restriction template defined for each type.
  - Template can be overridden for each exam section.
- Construct room region(s) and features for each room, and initial feature requirements for each exam section based on course code.
- Merge exam sections in the same exam group into a single exam.
- Split sections as required or requested in order to have valid room assignments.
- Formalize special requests as feature requirements, timeslot restrictions, or merges and splits.
Room regions

- Impossible to determine allowable rooms for each course based on rules.
- Retrieved 20 previous schedules. For each exam section, stored the buildings used.
- The reverse mapping of course code to building corresponds to a first draft of “room regions”.
A generalised reporting tool has been implemented.

Currently outputs text files, but could be quite easily modified to support other formats.

Six report types are currently available:

- Seat report - A high level summary of the number of students and sections in each exam period.
- Summary report - A brief summary of objective values for a given schedule.
- Student report - For each student, their classes and constraint/preference violations are shown.
- Period report - For each exam period, the scheduled exams, their exam groups, and their scheduled rooms are shown.
- Room report - For each exam period, the rooms used are shown along with the exams scheduled in them and the enrollment/capacity.
Ideal winter 2010 process
What actually happened

- **Sep 24**: Drop date
- **Sep 27**: All Data
- **Oct 7**: First usable data
  - First schedule sent for feedback
- **Oct 14**: Schedule released
  - New constraint, "back to back"
- **Oct 21**: All cancellations
  - All merges need to be done only if the instructor is the same
- **Oct 29**: First 3 days using new merges
- **Nov 4**: Final schedule
  - LSC 1,2,3 only available by request, HEBB TH overcapacity forced for distance classes.

Iterate with Feedback

Tweaks

Iterate

Best to CS

Best to CS

Chris Fawcett (EARG)
## Results

<table>
<thead>
<tr>
<th>Constraint</th>
<th>UBC</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Conflicts</td>
<td>27</td>
<td>6</td>
</tr>
<tr>
<td>Three Exams in Four Periods</td>
<td>139</td>
<td>202</td>
</tr>
<tr>
<td>Two Exams in the Same Day</td>
<td>4437</td>
<td>3615</td>
</tr>
<tr>
<td>Less than 8 Periods Between Exams</td>
<td>77970</td>
<td>93544</td>
</tr>
</tbody>
</table>
Results

- Competitive schedules produced in 2-3 hours, based on our previous model of the constraints.
- No timeslot assignments used.
  - Could not produce a new schedule quickly enough after “back-to-back” concern was raised on October 7.
- Approximately half of generated room assignments were used.
Necessary changes and next steps

- A clear, objective measure of schedule quality agreed upon in advance.
- Small modifications to room regions based on feedback.
- Implement the “back-to-back” constraint based on feedback.
- Ability to support room disjunctions.
- Dynamic section splitting, inside the solver.