Solving Large-scale Timetabling Problems

Chris Fawcett
fawcettc@cs.ubc.ca

CPSC322 Guest lecture

October 23, 2015
Today

- Scheduling and timetabling problems
  - Post-enrollment exam timetabling
  - Constraints
  - Data cleaning
  - Local search for timetabling
  - Efficiency + incremental data structures
Scheduling problems

- Given some *activities* (shifts, meetings, exams),
- some *resources* (people, equipment, rooms),
- and a *start* and *end* time,
- assign a start time between *start* and *end* to every activity, along with its required resources, making sure any additional conditions are satisfied.

Many variations of this problem:
  - Ordering tasks and assigning them to machines in a large machine shop.
  - Assigning crews to maintenance tasks for machinery on a production line.
  - Scheduling work and material purchases on a building construction site.
  - Exam times and rooms at a university.
# Timetables

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>MT2202 Ordinary Differential Equations FTh1</td>
<td>LABC S2072 Computer Graphics (G) Dual</td>
<td>MT2282 Numerical Analysis II Williamson, G03</td>
<td>XMT2292 Ordinary Differential Equations</td>
<td>XMT2292 Ordinary Differential Equations</td>
</tr>
<tr>
<td>10:00</td>
<td>XMT2292 Ordinary Differential Equations H15 / Roscoe, 2.3</td>
<td>LABC S2072 Computer Graphics (G) Dual</td>
<td>XMT2292 Ordinary Differential Equations</td>
<td>XMT2292 Ordinary Differential Equations H15</td>
<td>XMT2292 Ordinary Differential Equations</td>
</tr>
<tr>
<td>11:00</td>
<td>C52012 Algorithms and Data Structures 1.1</td>
<td>XMT2212 Further Linear Algebra 1.1</td>
<td>MT2202 Ordinary Differential Equations</td>
<td>Stopford, Theatre 1</td>
<td>MT2202 Ordinary Differential Equations</td>
</tr>
<tr>
<td>12:00</td>
<td>MT2212 Further Linear Algebra Roscoe, Theatre A</td>
<td>MT2282 Numerical Analysis II Williamson, G03</td>
<td>C52072 Computer Graphics 1.1</td>
<td>MT2212 Further Linear Algebra Stopford, Theatre 1</td>
<td>MT2212 Further Linear Algebra</td>
</tr>
<tr>
<td>1:00</td>
<td>C52072 Computer Graphics 1.1</td>
<td>C62072 Computer Graphics 1.1</td>
<td>XMT2212 Further Linear Algebra</td>
<td>XMT2212 Further Linear Algebra Simon Engineering, Basement Theatre 4A</td>
<td></td>
</tr>
<tr>
<td>2:00</td>
<td>C52072 Computer Graphics 1.1</td>
<td>C62072 Computer Graphics 1.1</td>
<td>XMT2212 Further Linear Algebra</td>
<td>XMT2212 Further Linear Algebra Simon Engineering, Basement Theatre 4A</td>
<td></td>
</tr>
<tr>
<td>3:00</td>
<td>C52072 Computer Graphics 1.1</td>
<td>C62072 Computer Graphics 1.1</td>
<td>XMT2212 Further Linear Algebra</td>
<td>XMT2212 Further Linear Algebra Simon Engineering, Basement Theatre 4A</td>
<td></td>
</tr>
<tr>
<td>4:00</td>
<td>C52072 Computer Graphics 1.1</td>
<td>C62072 Computer Graphics 1.1</td>
<td>XMT2212 Further Linear Algebra</td>
<td>XMT2212 Further Linear Algebra Simon Engineering, Basement Theatre 4A</td>
<td></td>
</tr>
</tbody>
</table>
Scheduling can be complicated!

- Human attendees, each with their own constraints.
- Time restrictions for activities and resources.
- Activity precedences (A starts/finishes before B starts/finishes).
- Competition for resources (meeting rooms, machines, work crews).
- Most of the time, you want a *good* schedule.
  - How to define “good”?
  - This is an *optimization problem*.

Chris Fawcett (fawcettc@cs.ubc.ca)  CPSC322 - Scheduling and timetabling
Timetabling

- Scheduling problems where time is divided into *timeslots*:
  - Non-overlapping
  - Fixed duration
  - Large enough so that activities “fit” inside

- Assign a timeslot to each activity.
  - Shifts (noon-midnight, midnight-noon)
  - Days (Monday Oct 26, Tuesday Oct 27, Wednesday Oct 28, ...)
  - Every hour on the hour, one-hour duration (8:00, 9:00, 10:00, ...)
Post-enrollment exam timetabling

- Activities are exams.
  - Attended by students and instructors.
  - Assigned to both a timeslot and a room (resource).
- Which students are attending every exam is known at scheduling time.
- Want to minimize the violation of the problem constraints.
Constraints: student preferences

- CPSC101 asked: “What properties would your perfect exam schedule have?”
  - 1-2 days off between each exam.
  - No 8:00am exams, ever.
  - No evening exams, ever.
  - First exam is early in the exam period.
  - Last exam is as early in the exam period as possible.
UBC solver constraints

- **Hard Constraints** *(must be satisfied)*:
  - Students with exams at the same time (if more than 30)
  - Allowable times for each exam
  - Allowable rooms for each exam
  - Requested room features for each exam
  - Unrelated exams cannot share a room
  - Cross-listed courses must have the same exam time
  - Evening courses must have evening exams
UBC solver constraints

- **Soft Constraints** *(should be satisfied)*:
  - Students with exams at the same time (fewer than 30)
  - Students with 2+ exams on the same day
  - Students with 3+ exams in 4 consecutive timeslots
  - Students with back-to-back exams
  - Students with fewer than 8 timeslots between exams
  - Preferred times for each exam
  - Preferred rooms for each exam
  - Room capacities
  - No first-year exams on the last two days (Fall exams)
  - No fourth-year exams on the last two days (Spring exams)
UBC problem scale

- 30,000 unique students
- 1,700 sections with exams
- 105,000 student-exam pairs
- 274 rooms across 38 buildings
- 13 exam days, 52 timeslots
Data cleaning

- Academic problems tend to have nice, clean data formats.
- Real-world data defies any formatting rules you come up with.
- Data for exam timetabling at UBC comes from five separate data stores, each having slightly different data formats.
- The data itself has many edge cases!
- Needs to be standardized and sanitized before any solving can be done.
Data cleaning examples

- Course section id, depending on data source:
  - CPSC101 001
  - CPSC101.001
  - cpsc101 001
  - 2012WCPSC101.001
  - “CPSC”, “101”, “ ”, “001”
  - “PSYC308AGIS”
Data cleaning examples

- Student names:
  - Some students have only a first name, or only a last name
  - Some students have two or more first names, or two or more last names
  - Some UBC students have digits in their names in the DB (like “Alice4”)
  - Some names start or end with a hyphen (“-Bob”), have apostrophes in them, etc.
Further processing

- Sanitize and convert database reports to a 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 regions” 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 room feature requirements, timeslot restrictions, or merges and splits.
UBC process

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
Local search for timetabling

- Start with an empty timetable
- Greedily assign exams to the best available timeslots, starting with those having the most students.
- Multiphase local search strategy:
  - Optimize to reduce hard constraint violations as much as possible (ideally zero),
  - then soft constraint violations, while not letting hard constraint violations increase.
- Schedule quality based on a *linear combination* of the hard and soft constraint violations.
- This tries to balance the trade-offs associated with each of the constraints.
Search neighbourhoods

- Need to provide structure to the search space, only consider local “neighbourhoods” around each timetable.
- Search strategies consider the “moves” from a given timetable to another in its local neighbourhood.
- We consider the following neighbourhoods:
  - Choose an exam and move it to a different timeslot.
  - Choose two exams and swap their timeslots.
  - Choose two timeslots and swap all of their exams.
Room assignment

- Every time the exams in a timeslot change, we ensure there is a room assignment for that timeslot.
- Every room-exam pair has a numerical score, encoding a preference.
- Every assignment of exams to rooms is a *matching* in a weighted bipartite graph.
- We choose the assignment with the highest possible sum of scores over all exams.
Weighted bipartite matching

In our case the blue nodes are exams, and the green nodes are rooms.

Weight of matching = 9 + 2.1 + 1 = 12.1

Credit: Rahul Simha
Greedy initialization

- Sort the exams by the number of students registered, most to least
- For each exam in order, place it in the best possible timeslot given the exams that have already been assigned.
- If all timeslots cause hard constraint violations, don’t schedule that exam yet.
- After trying to insert all exams, place the leftovers into their “least-worst” timeslot.
- Can generate 1000s of these schedules extremely quickly.
Solving the hard constraints (Tabu Search)

- Starting from a fully-assigned schedule (thanks to the greedy initialization)
- Using the union of all three search neighbourhoods
- First improvement with ties: “accept” the first move that doesn’t make the schedule worse. Ties allowed!
- Many ties: can lead to cycles
- Tabu search: keep a record of the exams involved in the $k$ previous moves, don’t make a move that touches an exam in the “tabu list”.

Chris Fawcett (fawcettc@cs.ubc.ca)  CPSC322 - Scheduling and timetabling
Solving the soft constraints (Iterated Local Search)

- Still get to assume a fully-assigned schedule.
- Only consider moves which move one exam or swap two exams.
- Still first improvement, ties are much less likely because of the soft constraints.
- When no moves are improving (local minimum), perform a series of random moves, or remove and reschedule a set of exams.
- This is to “tunnel” the search into a different part of the search space.
- Repeat first improvement from new position, remember best solution(s) ever found.
Diversifying from local minima

Credit: Roberto Battiti

Credit: Roberto Battiti
Problem representation

- Timetables need to be very lightweight data structures, we’re creating a lot of them.
- Number exams from 0 to $n$, rooms from 0 to $r$, timeslots from 0 to $t$.
- Timetable is just two integer arrays storing 1) the timeslot index and 2) the room index for each exam.
Efficiency

- Very important to create new timetables in each neighbourhood quickly, and evaluate them quickly.
- Data locality matters a lot. Processor cache misses, when to reuse data vs. making a copy.
- Incremental computation: moves are only making small changes, so don’t recompute constraint violations from scratch.
- Keep track of which exams are “most responsible” for violations of each constraint.
Results

- Usable schedules produced after 10-15 minutes of solver runtime.
- Schedules as good or better than the official ones produced in 2-3 hours.