Scheduling and Timetabling

or

“Where do my exams come from?”

Chris Fawcett
fawcettc@cs.ubc.ca

CPSC101 Guest lecture

January 8, 2013

Today

- Scheduling and exam timetabling
  - Scheduling problems in general
  - Timetabling
  - Post-enrollment exam timetabling
  - Constraints
  - Reading questions
  - Exam timetabling at UBC

Scheduling problems

- Given some activities,
- some resources,
- 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.

- This is a very general problem, with many variations
  - 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.
  - Can you think of more?

About me

- Ph.D. student in UBC BETA lab
- Research: applying automated algorithm design techniques to hard real-world problems.
Scheduling complications

- Activities might have participants, each of whom have conditions to satisfy.
- Activities and resources might have restrictions on what times they can be scheduled.
- There may be precedences between activities (one must be scheduled before another).
- There may be competition for resources (meeting rooms, machines, or work crews).
- Furthermore, you may want the best schedule according to some criteria.
  - Known as an optimization problem.

Gantt charts

Timetabling

- Class of scheduling problems where time is divided into timeslots:
  - Non-overlapping
  - Fixed duration
  - Large enough so that activities “fit” inside
- Assigning a start time to each activity is reduced to assigning a timeslot.
- Examples:
  - Every hour on the hour, one-hour duration (8:00, 9:00, 10:00, ...)
  - Shifts (noon-midnight, midnight-noon)
  - Days (Monday Jan 7, Tuesday Jan 8, Wednesday Jan 9, ...)
  - ?
Post-enrollment exam timetabling

- Activities are exams.
  - Attended by students and instructors.
  - Must be assigned to both a timeslot and a room (resource).
- Which students are attending every exam is known at scheduling time.
  - This is true for UBC, but not for every university!
- The best schedule is the one minimizing the violation of many conditions known as constraints.

Constraints

- **Hard constraints** are conditions that must be satisfied for a schedule to make sense.
  - Two unrelated exams can't be in the same room at the same time.
  - Allowable start times or timeslots for activities / resources must be respected.
  - Provincial/federal shift requirements in hospitals.
  - Often precedences (A must be scheduled before B).

- **Soft constraints** are conditions that should be satisfied, but the schedule still makes sense if they aren’t.
  - Activities shouldn’t be too close together (or maybe not too far apart!)
  - Activities should be scheduled at a preferred time.
  - Activities should be assigned to a preferred resource (a room, perhaps)

Reading questions

- What did you like and dislike about your Fall 2012 exam schedule?
- What properties would your “perfect” exam schedule have?
- What do you think the process is to generate the exam schedule for every course on campus?
- Discuss your answers with your neighbours: 5 minutes!

Likes:
- ‘My exams were scheduled with gaps in between.”
- “My exams were scheduled early in the exam period.”

Dislikes:
- “One week+ gap between exams”
- “Evening/Saturday exams”
- “8:00am exams”
- “Two exams on the same day”
- “Three successive days of exams”
- “I got my schedule so late!”
- “All of my hard exams were first, and all of my easy exams came after!”

Any others?
Reading questions

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.

Any others?

Reading questions

What do you think the process is to generate the exam schedule for every course on campus?

- Use “teacher schedules”, “room schedules”, brute force search for schedule with fewest conflicts/hardships.
- Schedule the huge exams first, fill in the gaps with the rest of the exams by guessing and checking.
- Faculties schedule separately, schedules are combined afterward.
- “No idea.”
- “Completely at random.”
- “Must be a computer, since it would be utter chaos if a person had to do it.”
- “I assume a magical algorithm that makes everything work.”
- “... complex algorithms that shift the predetermined exams slots around to ensure that all students can be accommodated in accordance with their timetables ... without locking itself into an endless loop (error) of continuous rescheduling to accommodate everyone.”

UBC collaboration history

- Collaboration with UBC classroom services beginning in winter 2009.
- Problem size is 100x larger than seen in typical academic timetabling problems.
- Solver extension and improvements to support solving the UBC problem.
- Dry run in winter 2010, parts of resulting schedule were used.
- Full schedule, with minor manual tweaks, has been used since Spring 2011.
**UBC solver constraints**

- **Hard Constraints:**
  - Conflicts (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

- **Soft Constraints:**
  - Conflicts
  - Students with 2+ exams on the same day
  - Students with 3+ exams in 4 consecutive timeslots
  - Students with back-to-back exams
  - Students with less 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

**Current UBC process**

1. **UBC DATA**
   - Preprocessing
     - Exam and Group Requirements
     - Time and Room Selection
     - Conflicts
   - General Problem Instance
   - Human Expert
2. **Randomised Solver**
3. **General Solution**
4. **Translated Solution**
5. **Published Schedule**
Data cleaning

- Academic problems tend to have nice, clean data formats.
- Real-world data is disgusting.
- Data for UBC timetabling comes from five separate data stores, each having slightly different data formats.
- And sometimes the data itself is strange!
- Needs to be standardized and sanitized before hitting the preprocessing stage.

Possible course section id, depending on data source:
- CPSC101 001
- CPSC101.001
- cpsc101 001
- 2012WCPSC101.001
- “CPSC”, “101”, “”, “001”
- Oh, and a few courses are like “PSYC308AGIS”

Student name permutations:
- 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 degenerate names start or end with a hyphen (“-Ann”)

Preprocessing

- Sanitize and 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 room feature requirements, timeslot restrictions, or merges and splits.

Our solver

- Randomised solver, designed to find good solutions quickly.
- Extremely modular, with a general and flexible problem specification format.
- Leverages automated algorithm design and configuration techniques.
Solver architecture

- Start from an empty timetable
- Greedily assign exams to the best available timeslots, starting with those having the most students.
- Using local search techniques,
  - Optimize first for hard constraint violations (conflicts) alone,
  - then soft constraint violations, while not letting hard constraint violations increase

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, leave that exam out
- After trying to insert all exams, place the leftovers into the “least-worst” timeslot

Local search

- Starting from a fully assigned schedule (thanks to the greedy initialization)
- Iteratively perform one of the following “moves”:
  - 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
- Only “accept” a move if it improves the schedule, otherwise reverse the move and continue

Room assignment

- Every time the exams in a timeslot changes, we recalculate a room assignment for that timeslot
- Use a sophisticated algorithm for the Weighted bipartite matching problem
- Every room-exam pair has a numerical preference score
- We select the assignment with the highest possible sum of scores.
Results

- Usable schedules produced after 10-15 minutes of solver runtime.
- "Good" schedules produced in 2-3 hours.
- Both the time and room of your final exams were (and will be) produced by this solver!