University Course Timetabling Solver Evolution

PATAT 2016
• University Course Timetabling: Solver Evolution
  • Short introduction to UniTime
  • Course timetabling problem
  • CPSolver and its improvements since our last publication (2008)
  • Data sets from Purdue University
  • Conclusions
Introducing UniTime

What is UniTime?

• Comprehensive academic scheduling solution
• Four components: course timetabling, examination timetabling, student scheduling and event management
• Open source, web-based, written in Java using modern technologies
• Distributed data entry and timetabling in multi-user environments
• Started as a research project back in 2001
• Became an enterprise system for many timetabling needs of a university
  • USA, Czech Republic, Pakistan, Croatia, Poland, Turkey, Peru, Kuwait, Canada, Malaysia, Spain, UAE, Palestine, Zambia, Kenya,…
• Apereo Foundation project since March 2015
Course Timetabling

What is Course Timetabling?
• The process of assigning times and rooms to classes

Constraints
• Rooms of various sizes, equipment and availability
• Faculty with requirements and preferences
• Courses that are to be offered, organized in a structure
• Students with their course demands (curricula, pre-registration, etc.)

Goal
• Assign classes in both time and space in a way that
  • All hard constraints and other requirements are met
  • All the desirable objectives are satisfied as much as possible
• Objectives: student conflicts, time and room preferences, class distributions, fairness, travel, etc.
Classes are organized in a course structure

- Intuitive data entry and display of classes and their requirements
- Helps to define a way how students can enroll into the course
- Additional relations can be derived from the structure
Date Patterns

- Weeks of instructions (All weeks, Even/Odd weeks, Week 5, ...)

Time Patterns

- Possible time slots within a week
Rooms

- Each department may have a different set of rooms
- Some times may be unavailable or given to a different department

Room coordinates, travel times

Minimal Room Size

- Calculated from class limit and room ratio

Room Preferences

- Particular room or building
- Room group
- Room feature
Each student has a list of courses he/she wants to attend

- Using pre-registrations, curricula, last-like enrollments, or a combination

Conflict: A student cannot take a combination of courses

- Because there is a (time) conflict
  - Classes are offered at overlapping times or one after the other in rooms that are too far apart
  - Or, there is not enough space in a non-conflicting combination of classes

Biology and chemistry lectures are in a time conflicts

Math students can choose, unless they need statistics as well

Chemistry students need a lecture and one of the two labs
Distribution Constraints

- Relationship between two or more classes
  - Precedence
  - Back-To-Back
  - Same Room
  - Same Days
  - Meet Together
  - Spread in Time
  - At Most 6 Hours A Day
  - Can Share Room
- ...
Problem Formulation

Model
- Variable: class
- Value: time and room placement

Hard Constraints
- Room size, sharing, availability
- No instructor / room can have two classes at the same time
- Required or prohibited preferences

Soft Constraints (Objectives)
- Time, room, and distribution preferences
- Student conflicts
- Additional criteria (too big rooms, back-to-back instructors, …)
Local-search based, however
- Operates over feasible, though not necessarily complete, solutions
- Feasibility is ensured automatically

Iterative Forward Search

```java
while (termination.canContinue(solution)) {
  Neighbour n = neighbourSelection.select(solution);
  if (n != null) n.assign(solution);
  if (solutionComparator.isBetterThanBest(solution)) solution.saveBest();
}
```

- Guided by neighborhood selection, termination, and solution comparison heuristics
- Select variable and its value, unassigns conflicting variables with the new assignment
- Conflict-based Statistics
  - If $A=a$ is unassigned because of a $B=c$, a counter $CBS[A\neq a, B=c]$ is incremented
  - Conflicts are weighted by their past occurrences
- Additional Variants
  - MPP: original solution, modified problem $\rightarrow$ minimize differences
  - Interactive: branch and bound of limited depth proposing schedule changes
Solver Evolution

Benchmark Data Sets

- From Purdue University, Fall 2007 and Spring 2007
- The results are presented on the combined problem (of 8 departmental problems)
- Over 2,400 classes, around 30k students and 200 rooms
- Available at http://www.unitime.org/uct_datasets.php in XML format
- Complete real world instances in an anonymized form (no names, etc.)

Experiment

- 10 independent runs for each solver build and (combined) instance
  - since the paper (March 2008) till the one released with UniTime 4.1 (Dec 2015)
- Same configuration, solution evaluated using the latest solver
  - Except of the last two data points where a different algorithm was used

Solver Evolution

2007 course timetabling problems from Purdue

More details are available in the paper.
Results

• There was 50% improvement in the solution quality since UniTime 3.1
  • 33% less student conflicts
  • 15% improvement in time preferences
  • 40% in room preferences
  • 80% in distribution preferences
• Besides of these, there have been a lot of new constraints and other features added into the solver over the years.
Solver Improvements

A lot comes from many small changes here and there

- There have been two major releases since 2008 and most of the solver code has been rewritten at least once
  - Making use of Java 5 generics and the ability to split the objective into individual criteria in CPSolver 1.2
  - More versatile assignment model and the ability to use multiple solver threads in CPSolver 1.3

Distribution Preferences

- Partial satisfiability of soft distribution preferences
- Imagine a different room constraint between four classes
- Not satisfied (full penalty) → 83.3% satisfied
- Forward checking along hard constraints
Solver Improvements

Student Scheduling

- Initial sectioning using aka Carter’s homogenous sectioning (students with similar course selection are kept together)
- During or after the search: swap students between alternative classes

Improvements

- Move a single student into an alternative class (if there is space in it)
- Swap student between classes with different parents
Plug-in different algorithms and search heuristics

- Additional algorithms and heuristics available out of the box
- IFS, Great Deluge, Simulate Annealing
- For course timetabling: using GD after a complete solution is found, never leaving the space of complete feasible solutions
- Besides of the usual neighborhoods, we also use a branch & bound of a limited depth
  *(same that is used by the interactive solver to propose changes)*

Ability to use parallel solver threads

- Two models
  - Parallel threads share a common solution (proposing changes to it)
  - Each thread works with its own solution (assignment)
- For course timetabling: second model is used, sharing properties of the best solution ever found
Solver Evolution

2007 course timetabling problems from Purdue

More details are available in the paper.
Conclusion

• **Key points**
  - Open-source university timetabling system UniTime
  - A very general course timetabling problem that fits many institutions
  - (Large) benchmark data sets from Purdue University
    - With the potential to have more data sets in the future
  - A lot of work has been done on the solver since our last publication
  - Solver framework can also be used to test new algorithms and heuristics
    - Or on different timetabling problems

• **For more details, please see me at the conference**
  - Or visit www.unitime.org

An online demo is available at [http://demo.unitime.org](http://demo.unitime.org)