

UniTime

Using UniTime to build optimised academic schedules

Tomáš Müller

June 2025



Agenda

What is UniTime

- A tool for building academic timetables/timetables
- Main component is a solver

Artificial Intelligence

- Timetabling is a hard problem, with a lot of competing objectives
- Using a hybrid approach based on constraint programming (CP)
 - CP is a subfield of AI focusing on problem-solving

UniTime Solver

- Constraint Programming
- Algorithm basics
- A few details, pro & cons, etc.



What is UniTime?

- Comprehensive academic scheduling solution

The collage displays several key features of the UniTime system:

- Rooms:** A list of rooms with filters for department, capacity, and availability. A map view shows the physical layout of the campus.
- Class Detail:** A detailed view of a specific class (ALG 101 - Algebra I: Lec 1) showing enrollment, room, instructor, and timetable.
- Instructional Offering Detail:** A view for a specific instructional offering (C S 101 - Introductory Computing) showing course offerings, enrollment, and preferences.
- Log In:** A login screen for users, including a username and password field, and a 'Log In' button.
- Personal Timetable:** A view of a user's personal timetable, showing the schedule for a specific user (Root, Abraham).
- Examinations:** A view for managing examinations, including a search for final examinations and a list of examination details.
- Configuration:** A view for managing system configuration, including a table of external IDs, enrollment limits, and room ratios.
- Enrollments:** A view for managing student enrollments, including a table of enrollment details.

Version 3.5.89 built on Wed, 7 May 2014
© 2008 - 2014 UniTime LLC, distributed under GNU General Public License.



What is UniTime?

- Comprehensive academic scheduling solution
- Five components

Courses: time & rooms

- time preferences
- room requirements
- availability
- course structure
- avoid time clashes
- student course demand
- distributions

Events: room bookings

- course-related events
- personal schedule
- approval process
- unavailabilities

Exams: period & rooms

- minimise student conflicts
- period & room prefs
- distributions
- room sharing or splitting

Instructors: teaching assignments

- teaching load
- qualifications / skills
- preferences
- course spreading

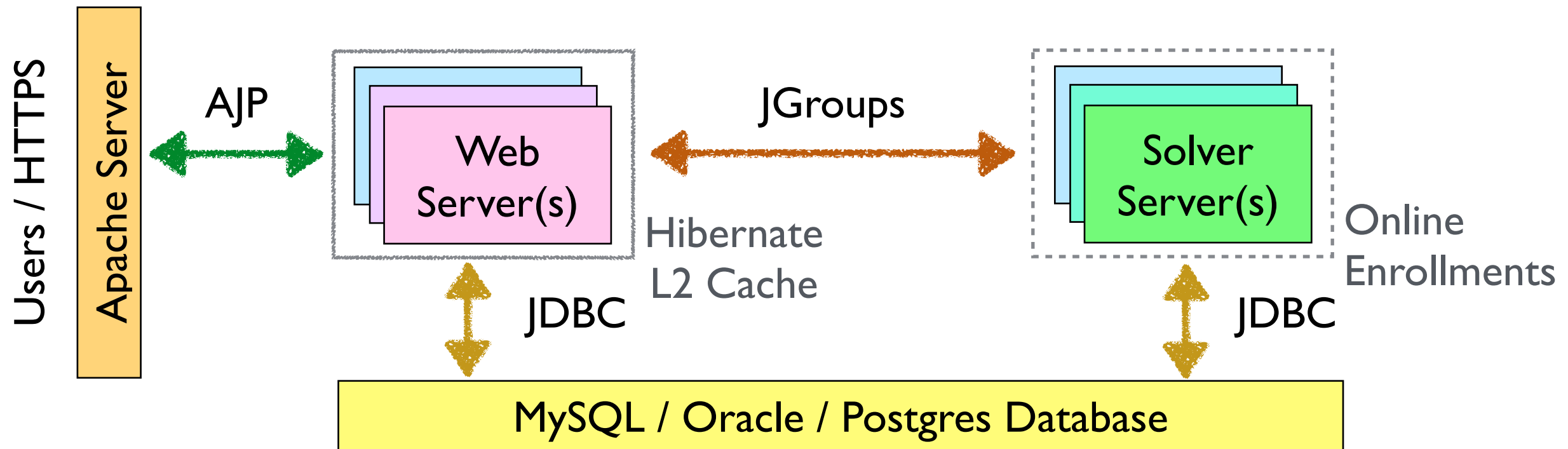
Students: class schedule

- course requirements
- alternatives / free times
- reservations
- section balancing
- schedule quality



What is UniTime?

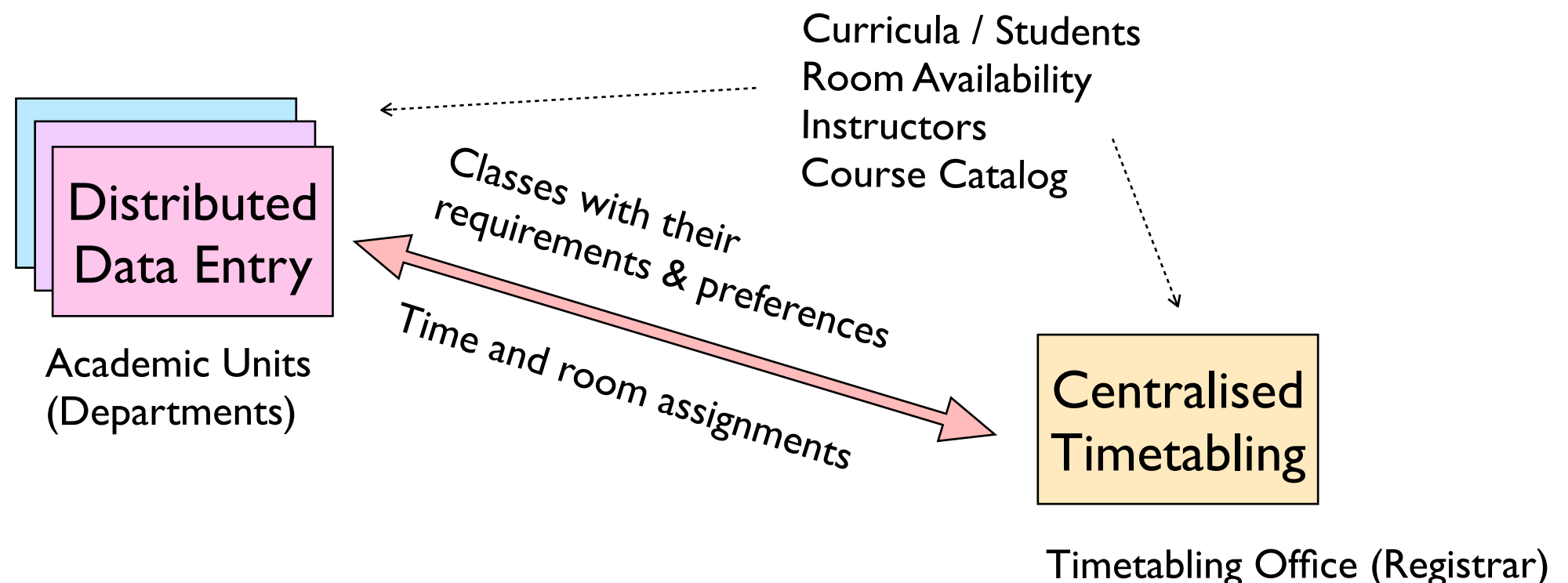
- Comprehensive academic scheduling solution
- Five components
- Open Source, web-based, written in Java using modern technologies





What is UniTime?

- Comprehensive academic scheduling solution
- Five components
- Open Source, web-based, written in Java using modern technologies
- Distributed data entry and timetabling in a multi-user environment





What is UniTime?

- Comprehensive academic scheduling solution
- Five components
- Open Source, web-based, written in Java using modern technologies
- Distributed data entry and timetabling in a multi-user environment
- Using state of the art optimisation algorithms

PURDUE
UNIVERSITY



International
Timetabling
Competition 2019



ITC 2019

- course timetabling with students
- 3 sets of problems (10 each)
- 600+ registrations
- 20+ solvers
- 20+ publications

<https://www.itc2019.org>



Constraint Programming

Constraint Satisfaction Problem $\Theta = (V, D, C)$

- $V = \{v_1, v_2, \dots, v_n\}$ is a finite set of variables
- $D = \{D_{v_1}, D_{v_2}, \dots, D_{v_n}\}$ is a set of domains
 - Domain is a finite set of values
- $C = \{c_1, c_2, \dots, c_m\}$ is a set of constraints
 - A constraint limits the combination of values that can variables simultaneously take
- Solution is an assignment of all variables $\eta: V \rightarrow D$
 - That satisfy all the constraints from C

Optimisation Problem $\Theta' = (V, D, C, f)$

- f is an objective function
 - That maps every partial feasible assignment to a number
 - Usually expressed by *soft* constraints and penalties

Examinations Example

Example: Examination Timetabling

- Variables: individual examinations
- Domains: period and room for each exam
- Hard Constraints:
 - Examination size \leq room capacity
 - Period and room requirements
 - No two exams in the same room at the same period
 - No student taking two exams during the same period
 - ...
- Soft constraints:
 - Period and room preferences
 - Back-to-backs
 - Not more than 2 exams on a day
 - Large exams first
 - ...

	from:	8:00a	10:30a	1:00p	3:30p	7:00p
	to:	10:00a	12:30p	3:00p	5:30p	9:00p
Mon 12/09						
Tue 12/10						
Wed 12/11						
Thu 12/12						
Fri 12/13						
Sat 12/14						

	Required
	Strongly Preferred
	Preferred
	Neutral
	Discouraged
	Strongly Discouraged
	Prohibited



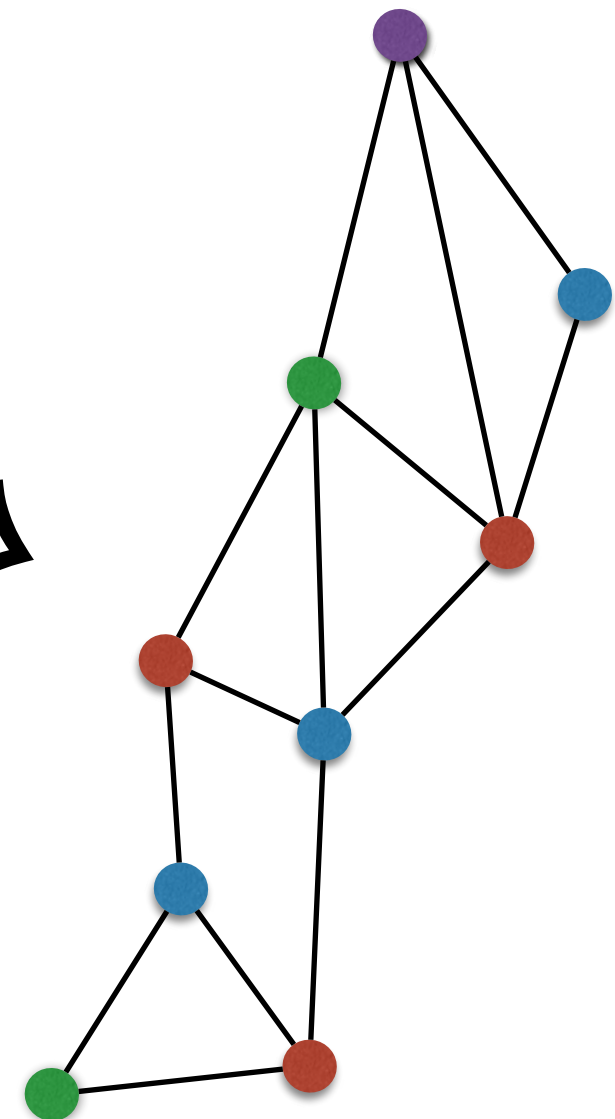
Timetabling is hard

Example: Examination Timetabling

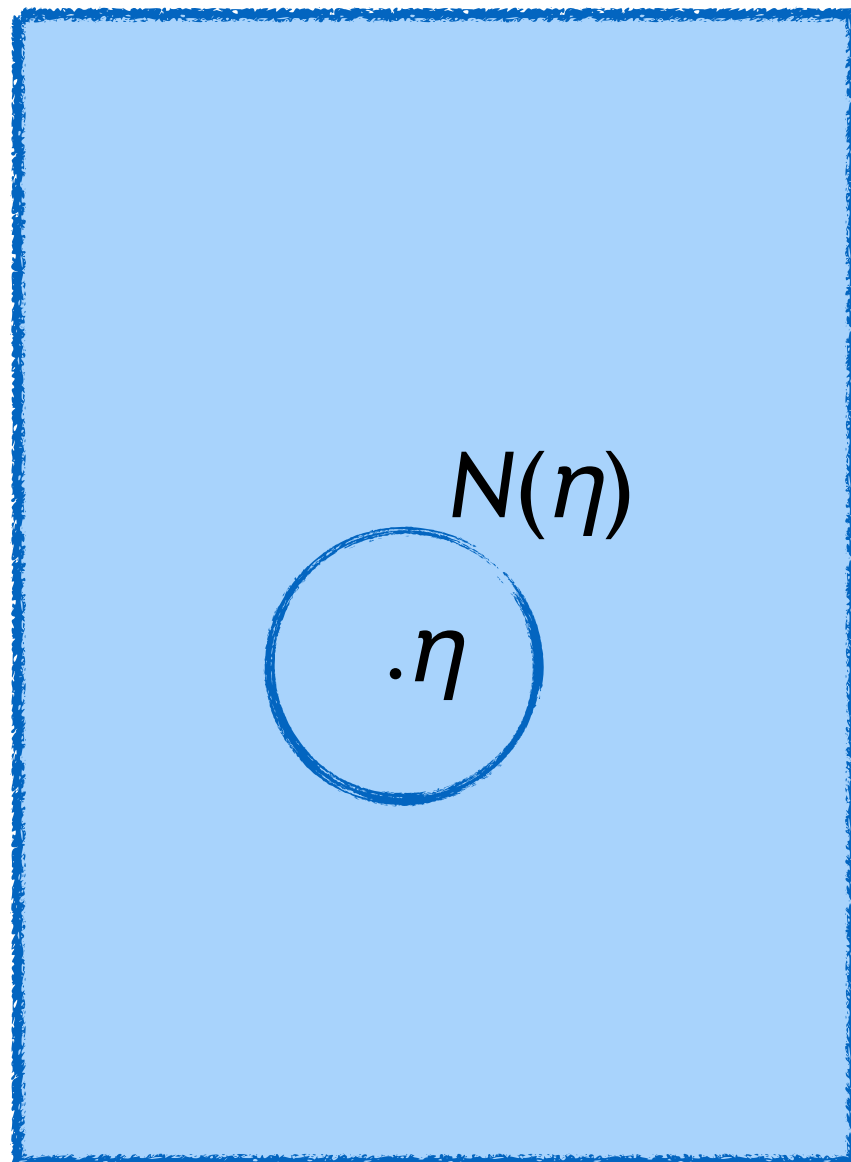
- Variables: individual examinations
- Domains: period and room for each exam
- Hard Constraints:
 - Examination size \leq room capacity
 - Period and room requirements
 - No two exams in the same room at the same period
 - No student taking two exams during the same period
 - ...
- Soft constraints:
 - Period and room preferences
 - Back-to-backs
 - Not more than 2 exams on a day
 - Large exams first
 - ...

Graph colouring

Exams: vertices
Students: edges
Colours: periods



Search Space



In each iteration

- Generate a candidate solution $y \in N(x)$
- Evaluate a solution
 - Discard when conflicting
 - When accepted assign $x = y$
- Check if termination conditions are met

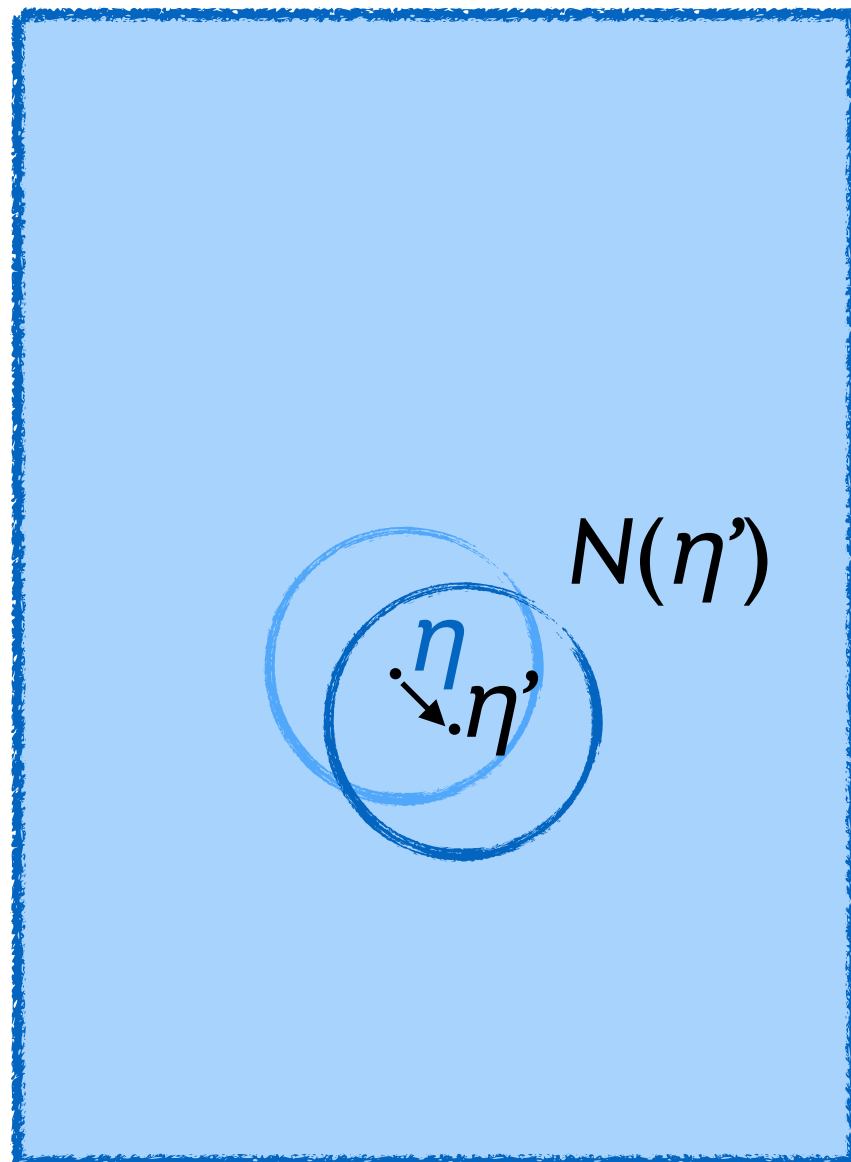
Neighbours

- Swap period/room
- Swap exams
- Assign exam, resolve/unassign conflicts
- Limited-depth search
- Kempe chain, MIP, ...

Acceptance criteria

- Hill Climbing $F(y) \leq F(x)$
- Simulated Annealing $P(\text{accept}) = e^{-\Delta E/T}$
- Great Deluge $F(y) \leq \text{Bound}$

Search Space



In each iteration

- Generate a candidate solution $y \in N(x)$
- Evaluate a solution
 - Discard when conflicting
 - When accepted assign $x = y$
- Check if termination conditions are met

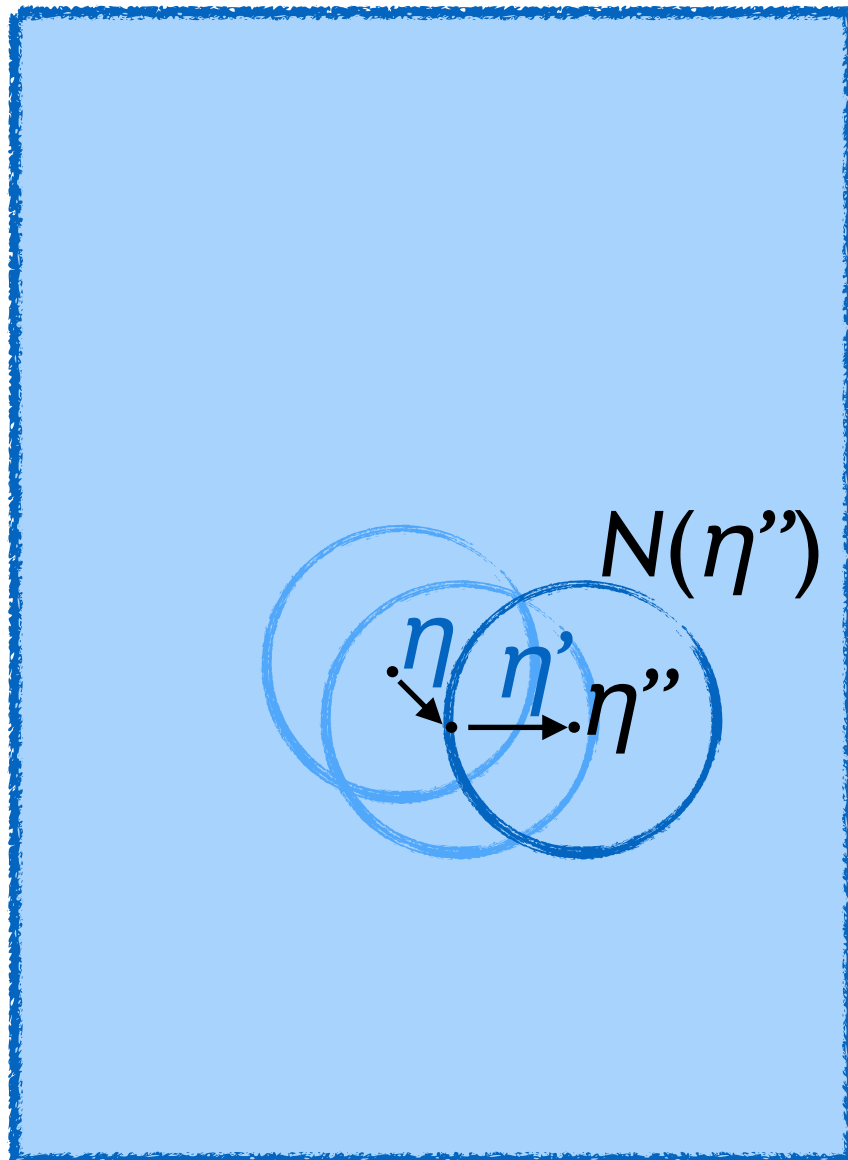
Neighbours

- Swap period/room
- Swap exams
- Assign exam, resolve/unassign conflicts
- Limited-depth search
- Kempe chain, MIP, ...

Acceptance criteria

- Hill Climbing $F(y) \leq F(x)$
- Simulated Annealing $P(\text{accept}) = e^{-\Delta E/T}$
- Great Deluge $F(y) \leq \text{Bound}$

Search Space



In each iteration

- Generate a candidate solution $y \in N(x)$
- Evaluate a solution
 - Discard when conflicting
 - When accepted assign $x = y$
- Check if termination conditions are met

Neighbours

- Swap period/room
- Swap exams
- Assign exam, resolve/unassign conflicts
- Limited-depth search
- Kempe chain, MIP, ...

Acceptance criteria

- Hill Climbing $F(y) \leq F(x)$
- Simulated Annealing $P(\text{accept}) = e^{-\Delta E/T}$
- Great Deluge $F(y) \leq \text{Bound}$

Solver Framework

Problems

Course Timetabling

Examination Timetabling

Instructor Scheduling

Student Scheduling

Constraint-based Model

Variables

Values

Constraints

Criteria

Solver

Configuration

Parameters

General

Problem Specific

Algorithms

IFS

HC

SA

GD

Neighbourhoods



UniTime Solver

Construction Phase

- Incomplete but feasible
- Pick an unassigned variable
- Find the best assignment
 - Unassign conflicts
- Conflict statistics
 - To prevent cycling
 - Provides feedback when over-constrained

Improvement Phase

- Complete and feasible
- Random selection of a candidate change
- Various acceptance criteria
 - HC, SA or GD
- Reheating
- Parameter adjustments
- Until time runs out

$$A \neq a \Leftarrow \begin{cases} 3 \times B = a \\ 4 \times B = c \\ 2 \times C = a \\ 120 \times D = a \end{cases}$$



Pros and Cons

Pros

- Usually very fast and scalable (compared to other methods)
- Flexible
 - Easy to extend
 - Exceptions are fairly easy to model
- Can start from an existing solution
 - Distance from previous solution can be also minimised
 - Can be used to provide suggestions (*interactive timetabling*)

Cons

- Cannot guarantee optimality
- Small changes can have large consequences
- Large changes may be expensive to make
- Often requires implementation of the model and at least a few problem specific heuristics/neighbourhoods



Other Approaches

Mixed Integer Programming (*Operations Research*)

- Using boolean or integer variables
- All constraints must be linear
- Existing (commercial) solvers that are getting very good
- Can guarantee optimality or provide upper bounds
- Heuristic approaches needed for large problems
 - Large neighbourhood search, fix & optimise, ruin & recreate

Comparison

- The first two places at ITC 2019 were using MIP solvers
 - Both using some version of fix & optimise
- Problem had to be segmented into smaller sub-problems
- The winner solver can provide better results than UniTime
 - But require lots of time and computational resources
- See ITC2019.org for more details



Conclusion

Thank you!

See www.unitime.org for more details about UniTime!

An online demo is available at <https://demo.unitime.org>